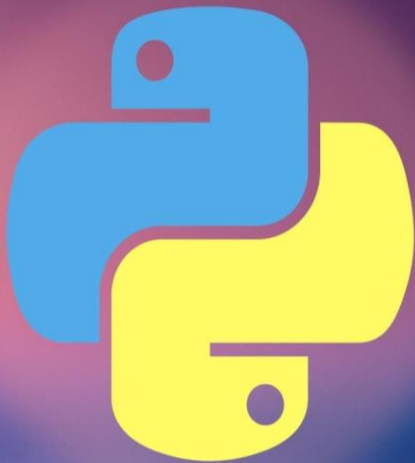


21 day challenge with Python

Part 0 out of 21



```
Print("what do we need ?".title())
```

1-Python History

2-Python source

3-Python IDE

4-Python text editor

5-Python courses

6-Python documents

1-Python History?

It was created by **Guido van Rossum**, and first released on February 20, 1991.

Zen of python :

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one – and preferably only one – obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
- Although never is often better than right now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea – let's do more of those!

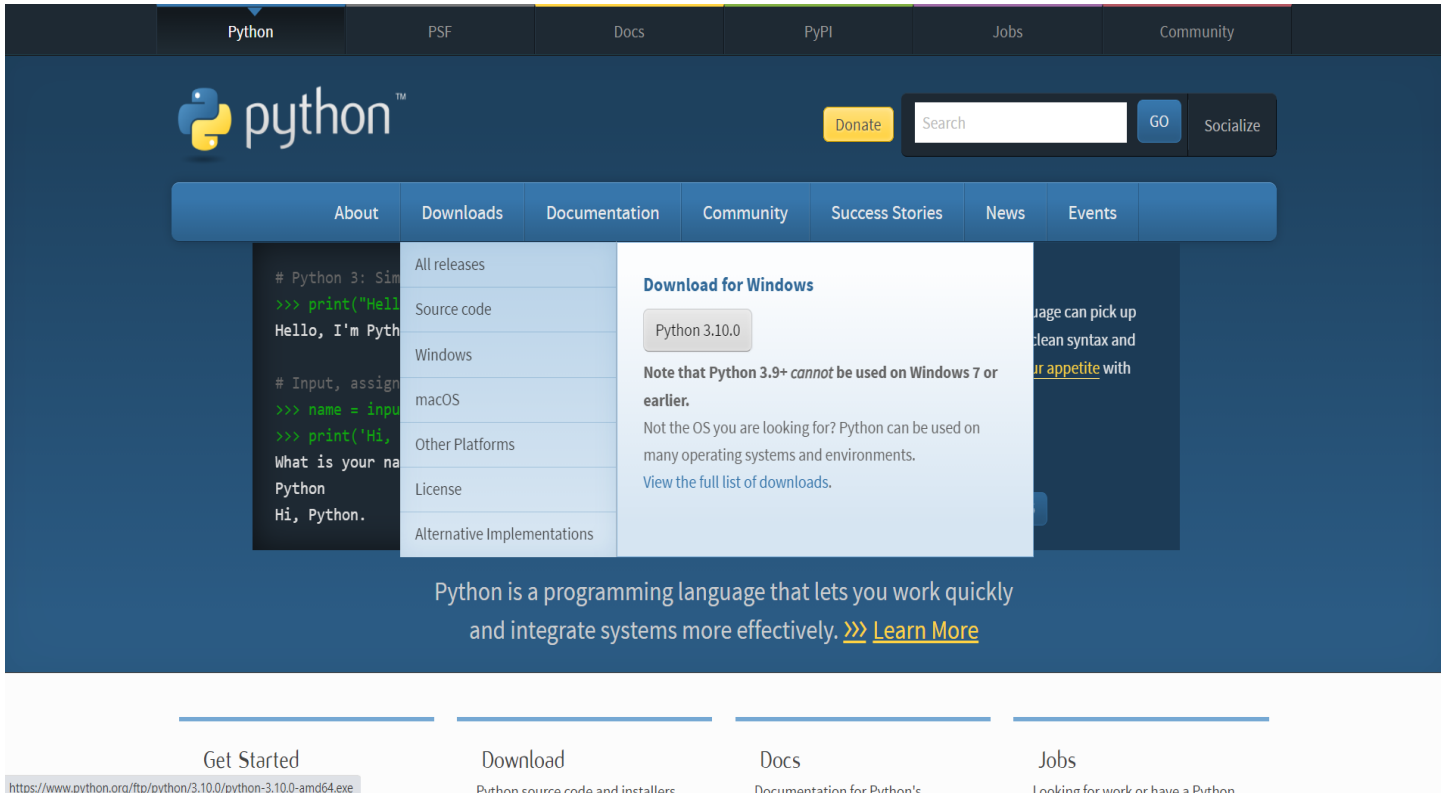
LETS GO TO THE NEXT PART, IF YOU WANT TO KNOW MORE ABOUT PYTHON HISTORY, SEARCH IT.



2-Python Source?

You should install python from: python.org

Last version, released about a week ago.



The screenshot shows the Python.org website. At the top, there is a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below this is the Python logo and a search bar. A main navigation bar contains links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The 'Downloads' menu is open, showing options for All releases, Source code, Windows, macOS, Other Platforms, License, and Alternative Implementations. The 'Download for Windows' section is highlighted, showing a button for Python 3.10.0. A note states: 'Note that Python 3.9+ cannot be used on Windows 7 or earlier. Not the OS you are looking for? Python can be used on many operating systems and environments. View the full list of downloads.' Below this, a text block says: 'Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)'

Get Started <https://www.python.org/ftp/python/3.10.0/python-3.10.0-amd64.exe>

Download Python source code and installers

Docs Documentation for Python's

Jobs Looking for work or have a Python

You should install the file.

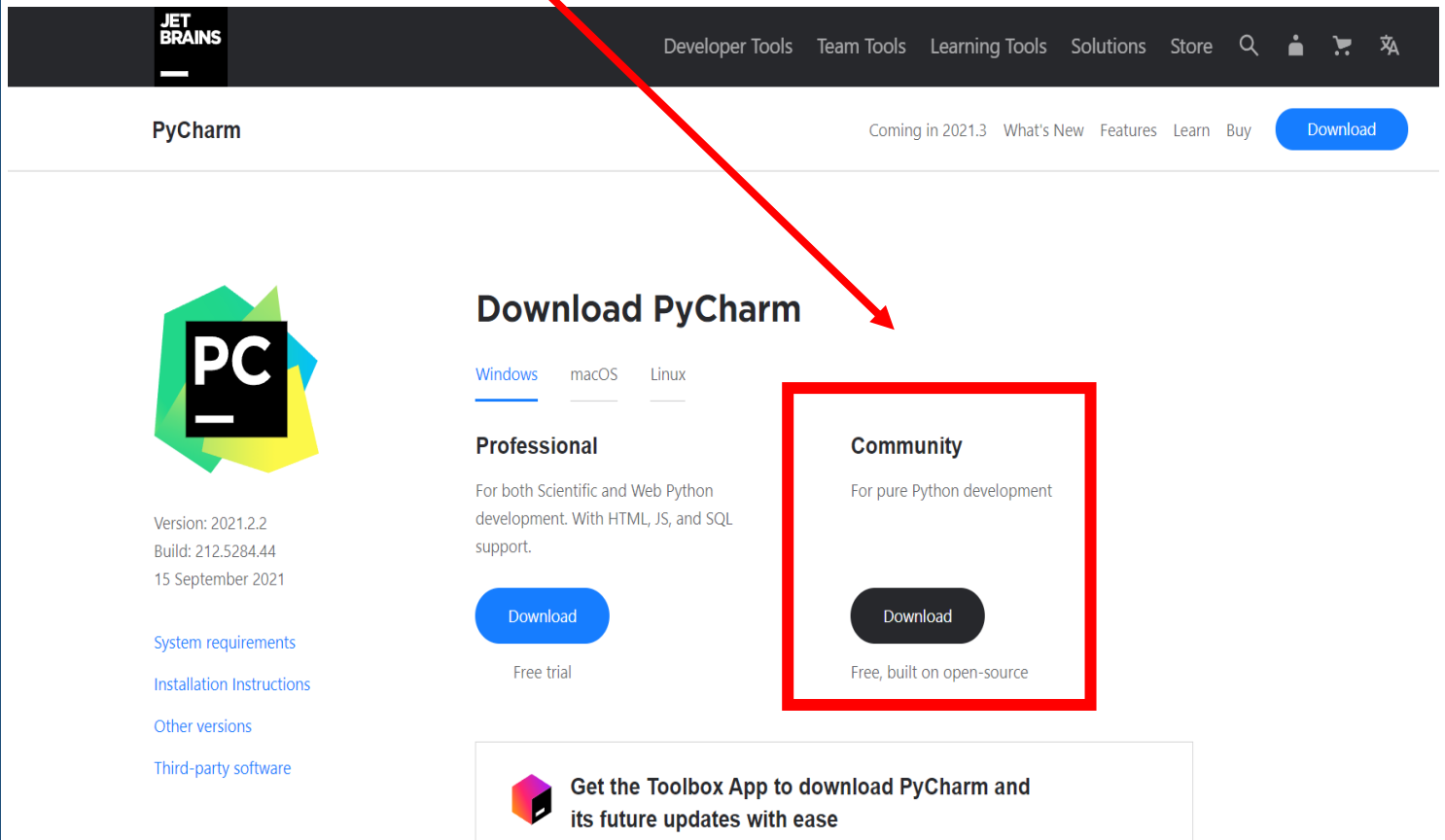
THAT'S ALL.

3-Python IDE?





There is always an argue between developers over which IDE is better.

I won't get into this fight because neither of us has the time, but I think the best Python IDE is PyCharm.

You can download the free version of PyCharm from : jetbrains.com



JET BRAINS

Developer Tools Team Tools Learning Tools Solutions Store    

PyCharm Coming in 2021.3 [What's New](#) [Features](#) [Learn](#) [Buy](#) [Download](#)

Download PyCharm

[Windows](#) [macOS](#) [Linux](#)

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[Download](#)

Free trial

Community


For pure Python development

[Download](#)

Free, built on open-source

Version: 2021.2.2
Build: 212.5284.44
15 September 2021

[System requirements](#)
[Installation Instructions](#)
[Other versions](#)
[Third-party software](#)

 **Get the Toolbox App to download PyCharm and its future updates with ease**

4-Python Text Editor?

Do not be too sensitive about what text editor you want to use.

They all do the same thing, but my favorite one is : VS code

You can use whatever you like.



5-Python Courses?

Best python courses which are very useful on the Internet :

1-Youtube



Python Tutorial - Python for Beginners [Full Course]

21M views • 2 years ago

Programming with Mosh

Want to learn more from me? Courses: <https://codewithmosh.com> Twitter: <https://twitter.com/moshamedani> Facebook: ...

CC

2-Maktab Khoone (jadi)



آموزش پایتون مقدماتی

مکتب‌خونه

جادی میرمیرانی



آموزش برنامه‌نویسی با پایتون (پیشرفته)

مکتب‌خونه

جادی میرمیرانی

3-Coursera

Browse > Computer Science > Software Development

Offered By

M UNIVERSITY OF MICHIGAN

Python for Everybody Specialization

Learn to Program and Analyze Data with Python. Develop programs to gather, clean, analyze, and visualize data.

★★★★★ 4.8 178,215 ratings



Charles Russell Severance

Python



Python is a general-purpose, versatile, and powerful programming language. It's a great first language because it's concise and easy to read. Whatever you want to do, Python can do it. From web development to machine learning to data science, Python is the language for you.

Why we love it:

- Great first language
- Large programming community
- Excellent online documentation
- Endless libraries and packages
- World-wide popularity
- Powerful and flexible

Recommended

Course
Learn Python 3
Completed

Featured resources



FORUM
Python 3 Codecademy Forums



CHART/GRID
Python 3: Syntax Cheatsheet



ARTICLE
Installing Python 3 Locally

Beginner friendly courses

Skill Path Analyze Data with Python • Beginner-friendly, 28 Lessons With Final Project	Skill Path Analyze Financial Data with Python • Beginner-friendly, 28 Lessons With Final Project	Skill Path Build Chatbots with Python • Beginner-friendly, 28 Lessons With Final Project
Skill Path Fundamental Math for Data Science • Beginner-friendly, 14 Lessons With Final Project	Skill Path Visualize Data with Python • Beginner-friendly, 12 Lessons With Final Project	Course Learn Python 3 Completed
Course Learn Python 2 • Beginner-friendly, 20 Lessons Language Fluency	Course Probability Enroll Now... View Syllabus	Course Learn the Basics of Blockchain with Python • Beginner-friendly, 8 Lessons
Course Build Connect Four Using Python • Beginner-friendly	Course Exploratory Data Analysis in Python • Beginner-friendly, 8 Lessons	Course Learn How to Get Started with Natural Language Processing • Beginner-friendly, 1 Lesson
Course Differential Calculus • Beginner-friendly, 11 Lessons	Course Linear Algebra • Beginner-friendly, 2 Lessons	Course Learn Statistics with Python • Beginner-friendly, 11 Lessons
Course Learn Hardware Programming with CircuitPython • Beginner-friendly, 21 Lessons	Course CS101 Livestream Series • Beginner-friendly, 18 Lessons	Course Getting Started On Platform for Data Science • Beginner-friendly, 1 Lesson

Intermediate courses

Skill Path Apply Natural Language Processing with Python • Intermediate, 8 Lessons With Final Project	Skill Path Build Deep Learning Models with TensorFlow • Intermediate, 6 Lessons With Final Project	Skill Path Build Python Web Apps with Django • Intermediate, 8 Lessons With Final Project
--	---	--

6-Python Documents?

Best python documents (we will get to know more about this part soon) :

1-W3 Schools

The screenshot shows the W3Schools Python Tutorial page. The top navigation bar includes links for HTML, CSS, JAVASCRIPT, SQL, PYTHON (highlighted), PHP, BOOTSTRAP, HOW TO, W3.CSS, JAVA, JQUERY, C++, C#, R, and React. The left sidebar lists various Python topics, with 'Python HOME' selected. The main content area features the title 'Python Tutorial', a '< Home' button, and a 'Next >' button. Below this is a green banner with the heading 'Learn Python' and the text 'Python is a popular programming language. Python can be used on a server to create web applications.' A 'Start learning Python now »' button is also present. The next section is 'Learning by Examples', which includes a 'Try it Yourself' editor and an example code block:

```
print("Hello, World!")
```

2-Codes Dope

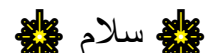
The screenshot shows the Codes Dope Python Tutorial page. The top navigation bar includes the Codes Dope logo and links for About, Courses, Discussion, Practice, Blog, and PRO. The left sidebar lists various Python topics, with 'Python Introduction' selected. The main content area features the title 'Python Tutorial - Introduction' and a 'Python Video Lectures' button. Below this is a paragraph of text: 'Python is a widely used high-level dynamic programming language. It is a very simple, friendly and easy to learn programming language. It is the best choice for a beginner programmer. Python source code is also available under GNU General Public License (GPL).' The next section is 'Who use Python?', which includes a paragraph of text: 'As mentioned above, it is a widely used programming language. Some of the places where Python is used are :'. A list of examples follows:

- **Google** - Python is one of the key language used in google.
- **Philips** - Philips uses Python for the sequencing language (language that tells what steps each robot should take).

I tried to say everything you need to start learning Python without going into details.

-By : Ali Moeinian

You can find me easily on LinkedIn app.



سلام

اول از همه، خیلی ممنونم ازتون که چالش 20 روز با پایتون رو دنبال میکنید.

راستش رو بگم، من اصلا درباره ی این چالش فکر نکرده بودم و خیلی یهویی تصمیم گرفتم این کار رو انجام بدم.

هدف من از این چالش، دوره ی سریع مطالبی هست که طی 7 ماه گذشته از پایتون یادگرفتم.

نه ادعای آموزش دارم و نه ادعای حرفه ای بودن، از هر جونیوری، جونیور تر هستم.

اما این کار رو دارم انجام میدم تا بتونم هم خودم پیشرفت کنم و شایااااید یکی هم خوشش اومد و علاقمند شد.

سعی میکنم در این 20 روز، با آماده کردن فایل هایی مثل همین فایل، اصل مطالبی که نیاز هست دوره بشه رو مطرح کنم.

و خیلی خوشحال میشم نظر دوستان رو در ارتباط با محتوای هر فایل بدونم.

اگر مطلبی ناقص یا اشتباه بیان شد، ممنون میشم اصلاح بفرمایید تا بتونم یاد بگیرم.

سعی میکنم لحن فایل ها خیلی خودمونی و راحت باشه، عین سینتکس پایتون.

توی قسمت صفر، سعی کردم خیلی سریع و خلاصه و بدون توجه به حواشی، هر آن چیزی که نیاز داریم برای یادگیری و کار با پایتون رو معرفی کنم.

فایل های بعدی حتما به زبان فارسی منتشر خواهد شد.



تشکر

کاری از : علی معینیان

21 day challenge with Python

Part 1 out of 21

```
Print("Let's Start ?".title())
```

1-Syntax

2-Arithmetic operations

3-Variables

4-Int,Str,Complex,Float

5-Input,Output

6-Comment



بریم سراغ قسمت اول، سینتکس پایتون.

اولا باید بدونیم سینتکس (Syntax) یعنی نوع نوشتار یک زبان، نوع دستورات و قوانین نوشتاری.

سینتکس پایتون : همونطور که حرف میزنی، بنویس، تمام.

وقتی که به متغیر ها و تعریفشون رسیدیم، خیلی بیشتر با سینتکس آشنا میشیم.

از اونجایی که اولین برنامه هایی که خیلی از ماها مینویسیم، بیشتر چیزاییه که با ریاضیات درگیره، پس بریم سراغ عملگر های ریاضی توی پایتون :

مثال	نام عملگر	نماد عملگر
$x + y$	جمع	+
$x - y$	تفریق	-
$x * y$	ضرب	*
x / y	تقسیم	/
$x \% y$	درصد	%
$x ** y$	توان	**
$x // y$	تقسیم باقی مانده	//

باید حواستون به این دو تا عملگر آخری خیلی باشه، یکم متفاوت هستند با عملگر های دیگه توی زبان های مختلف.

اما یکم بریم کد بزنیم، بفهمیم متغیر ها چجوری تعریف میشن توی پایتون و چند تا مثال هم با تعریف متغیر ها و استفاده از عملگر های ریاضی با هم ببینیم.

قبل از شروع این قسمت، باید بگم کتاب عمو باب رو حتما مطالعه کنید، واقعا تغییر بزرگی توی نوع کد زدنتون ایجاد میکنه. من هم سعی میکنم در حد یک جمله ی کلیدی توی هر بحث از کتاب عمو باب استفاده کنم.

متغیر چیه ؟؟؟؟

توی برنامه نویسی ما همیشه سعی میکنیم کار هامون دنیای واقعی رو پوشش بده. اسم شما، یک متغیره...

به جای اینکه بیان و شما رو با صفت های قشنگتون صدا بزنند، یک اسم براتون گذاشتند که با اون اسم، شما میفهمید باهاتون کار دارند.

مثلا به جای اینکه بگن : آقای قد بلند مو مشکی که قشنگ حرف میزنی، بیا کمک کن.

میگن : علی، پاشو بیا 😊

توی برنامه نویسی هم دقیقیقا همینه.

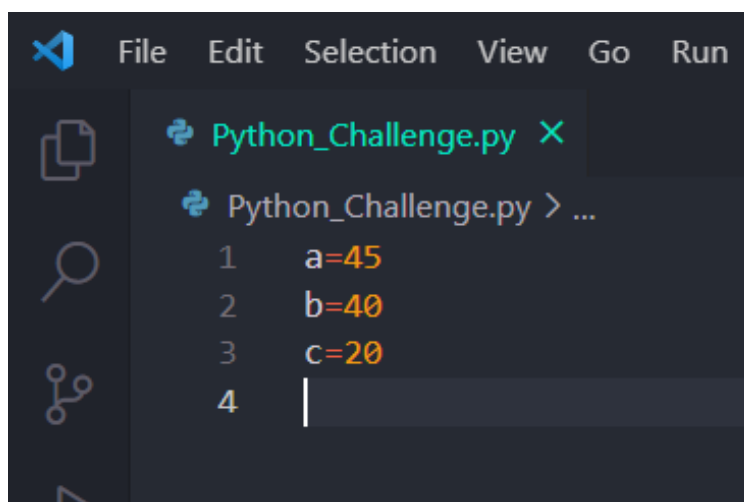
عمو باب : توروخداااا اسم متغیر ها رو با معنی بردارید.

راستی، یادت نره گفتم سینتکس پایتون همونطوریه که حرف میزنی !

بریم سراغ چند تا مثال جالب :

میخوام یک برنامه بنویسم که حاوی سن مادر و پدر و فرزند خانواده است.

کد زشت



```
File Edit Selection View Go Run
Python_Challenge.py X
Python_Challenge.py > ...
1 a=45
2 b=40
3 c=20
4 |
```

کد خوشگل

```
Python_Challenge.py
Python_Challenge.py > [F] Farzand
1 Pedar=45
2 Madar=40
3 Farzand=20
4
```

دیگه با همین یک مثال منظورمو فهمیدی از اینکه باید متغیر هامون رو قشنگ نام گذاری کنیم.

و دیدی که سینتکس اولیه ی پایتون چجوریه 😊

یادتون باشه که ما انواع روش های نام گذاری متغیر ها رو داریم. یکم که بیشتر تمرین کردیم همشو میگم براتون.

انواع متغیر ها توی پایتون :

یادت باشه این متغیر ها کم کم کامل میشن، ولی فعلا بدونید ما با اینا کار داریم :

Int: عدد صحیح توی دبستان ← 1 و 2 و 3 و

Number types

Float : عدد اعشاری توی دبستان ← 1.5 و 25.36522365 و

Complex : اعداد مختلط توی دانشگاه (خیلی کاری باهش نداریم)

String : متن ← Hello, bye, python و

یادتونه گفتم دنیای کامپیوتر مثل دنیای ما آدماس ؟

یادته داستان اسم و انتخاب نام متغیر رو گفتم ???

یادت باشه عین دنیای ما آدما، نمیتونی هر اسمی رو برای متغیرت استفاده کنی.

بعضی اسما هستند که توسط خود پایتون رزور شده اند.

مثلا شما هیچوقت اسم بچتو نمیزای یخچال ، چرا؟ چون یخچال یک اسم رزور شده است.
گرفتی چی میگم؟

اینا حرف های کلیدی رزور شده توی پایتون هستند :

البته match رو هم بهش اضافه کنید، توی نسخه ی 3.10 پایتون تازه اومده، بهش میرسیم.

False	def	if	raise
None	del	import	return
True	elif	in	try
and	else	is	while
as	except	lambda	with
assert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	

چند تا مثال ببینیم از چیزایی که تاحالا گفتم :

```
1 Adade_sahih=128569946
2 Adade_Ashar=12236.56632
3 Adad_mokhtalet=15j
4 matn ='Salam rofagha 😊'
5 if=1256
6 else=12.256
7 def = 'These are wrong !'
```

اینجا متغیرم یک عدد صحیح داره، از اسمش هم معلومه

این یکی هم که عدد اعشاریه

این یکی هم عدد مختلط هست که کاری باهاش نداریم

این متغیرم حاوی یک متن هست که بیشتر قراره درباره اش بخونی.

این چند تا هم همششش غلطه ! آگه گفتمی چرا؟

یادت باشه، توی هر زبان برنامه نویسی برای اینکه متن هایی که توی برنامه برای دادن پیغام یا هشدار استفاده میشه، با سینتکس برنامه قاطی نشه، همیشه پیام ها بین دو تا " " یا ' ' قرار میگیره.

توی سی پلاس پلاس یکم متفاوته ولی بدون توی پایتون هیچ فرقی بین این دوتا نیست.

یعنی اینکه :

```
Python_Challenge.py ●
Python_Challenge.py > [🔍] String_Holder2
1   # C++
2   character_Holder = 'A';
3   String_Holder = "Ali";
4
5   #Python
6   character_Holder1='a'
7   character_Holder2="a"
8
9   String_Holder1='ali'
10  String_Holder2="ali"
```

راستی اون خط کم رنگ ها که قبلش # داره چیه؟؟

توی برنامه نویسی، خیلی وقتا لازمه برای خودمون یا کسی که قراره بعد از ما با این کد کار کنه، یه سری توضیحات تکمیلی بنویسیم که باعث افزایش خوانایی کد ما میشه.

مثلا من اگه بالاش ننوشته بودم سی پلاس پلاس شاید از من غلط میگرفتید که اصلا کل سینتکس اشتباهه! پس من با اضافه کردن یک کامنت، باعث شدم شما درک کنید که این قطعه کد مربوط به سی پلاس پلاس و بعدیش مربوط به پایتون هست و قصدم مقایسه ی دو تا string هست.

عمو باب : تا وقتی احتیاجی نیست نباید خیلی از کامنت استفاده کنی! اگر کدت تمیز باشه، اصلا شاید هیچ کامنتی نخواد. کامنت خیلی زیاد توی کد نشون میده که یکم کارت ایراد داره!

دستور های ورودی و خروجی در پایتون :

یادته گفتم همونطور که میخونی، بنویس ؟

دستور ورودی : input

دستور خروجی : print

چند تا مثال ببینیم، میفهمی چی میگم :

```
Python_Challenge.py > ...
1 Adade_sahih=128569946
2 Adade_Ashar=12236.56632
3 Adad_mokhtalet=15j
4 matn ='Salam rofagha 😊'
5
6 # اینجا میخوام دستور چاپ رو بهش توجه کنی
7 print(Adade_sahih)
8 print('-----')
9 print(Adade_Ashar)
10 print('-----')
11 print(Adad_mokhtalet)
12 print('-----')
13 print(matn)
14 print('-----')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/AppData/
128569946
-----
12236.56632
-----
15j
-----
Salam rofagha 😊
-----
PS C:\Users\User\Desktop\Python\python projects> █
```

*کامنت فارسی رو

خواهشا ایراد

نگیرید فعلا.

پس دستور چاپ اینطوری کار میکنه :

Print(نام متغیر مورد نظر)

میتونی به جای 3 تا دستور از یک دستور استفاده کنی اما یادت باشه دستور print نهایتاً سه تا آرگومان میگیره :

```
Python_Challenge.py > ...
1 Adade_sahih=128569946
2 Adade_Ashar=12236.56632
3 Adad_mokhtalet=15j
4 matn ='Salam rofagha 😊'
5
6 اینجا میخوام دستور چاپ رو بهش توجه کنی#
7 print(Adad_mokhtalet, Adade_Ashar , Adade_sahih)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/A
15j 12236.56632 128569946
PS C:\Users\User\Desktop\Python\python projects> |
```

یه کار جالب رو بزار نشونت بدم (مقایسه کن و نتیجه بگیر !) :

```
Python_Challenge.py > [ⓧ] Adade_sahih
1 Adade_sahih,Adade_Ashar,Adad_mokhtalet=128569946,12236.56632,15j
2 print(Adad_mokhtalet, Adade_Ashar , Adade_sahih)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/AppData/Local/Programs,
15j 12236.56632 128569946
PS C:\Users\User\Desktop\Python\python projects> |
```

اما بعضی وقتا لازمه برنامه ی ما ورودی داشته باشه، یعنی کاربر ما اطلاعاتی رو به برنامه وارد کنه :

مثلا من از کاربرم میخوام اسم و فامیلشو وارد کنه و نهایتا اسم کاملش رو تحویل بگیره.

```
Python_Challenge.py > ...
1 Esm=input('Lotfan esmeto begoo :')
2 Famil=input('Lotfan famileto begoo :')
3 print(Esm + Famil)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/Python/Python_Challenge.py
Lotfan esmeto begoo :Ali
Lotfan famileto begoo :Moeinian
AliMoeinian
PS C:\Users\User\Desktop\Python\python projects>
```

دایره نارنجی ها توسط خودم وارد شده، یعنی همون ورودی. اما یکم خروجی زشت شده، نه ؟ بزارید یکم خوشگلش کنم.

```
Python_Challenge.py > ...
1 Esm=input('Lotfan esmeto begoo :')
2 Famil=input('Lotfan famileto begoo :')
3 print(Esm + ' ' + Famil)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/Python/Python_Challenge.py
Lotfan esmeto begoo :Ali
Lotfan famileto begoo :Moeinian
Ali Moeinian
PS C:\Users\User\Desktop\Python\python projects>
```


چند تا تمرین ساده با اعداد و استفاده از عملگر های ریاضی :

```
Python_Challenge.py
1 print(2+2)
2 print(2*3)
3 print(2**4)
4 print(24/2)
5 print(256//6)
6 print(15%3)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/U
4
6
16
12.0
42
0
PS C:\Users\User\Desktop\Python\python projects> |
```

فعلا برای روز دوم این چالش، تا همین حد کافیه.

بعدا باز هم برمیگردیم سراغ این مطالب و بیشتر باهاشون آشنا میشیم.

تمام مطالب رو من قطعا نمیتونم توی این فایل ها بیارم و خودتون با نوشتن برنامه های زیاد میتونید به جواب خیلی از سوالاتون برسید.

ولی به نظرم تا همینجا کافیه ! قسمت بعدی یکم بیشتر وارد برنامه نویسی میشیم تا بتونیم بیشتر آشنا بشیم با فضای برنامه نویسی و خیلی سریع بریم سراغ اصل کار.

شروع هر زبانی یکم مشکله، با تمرین همه چی حل میشه 😊

روی استفاده از دستور پرینت و استفاده از عملگر ها توی این دستور خیلی کار کنید.

من قصدم اصلا آموزش نیست و هیچ ادعایی هم در بلد بودن پایتون ندارم و صرفا هدفم دوره کردن مطالبی هست که تا حالا یاد گرفتم.

قطعا نمیتونم همشو توی این پی دی اف ها جا بدم چون وسعت مطالب خیلی زیاده ، در حدی که من سه یا چهار جلسه میتونم درباره ی دستورات ورودی و خروجی و تعریف متغیر ها مطلب بهترن ارائه بدم ولی اصل یادگیری هم وقتی هست که خودتون کد میزنید و به مشکل بر خورد میکنید و داکيومنت مطالعه میفرمایید.

ممنونم که من رو دنبال میکنید 😊

خوشحال میشم ایراداتم رو بهم بگید تا بتونم پیشرفت کنم.

کاری از : علی معینیان

21 day challenge with Python

Part 2 out of 21



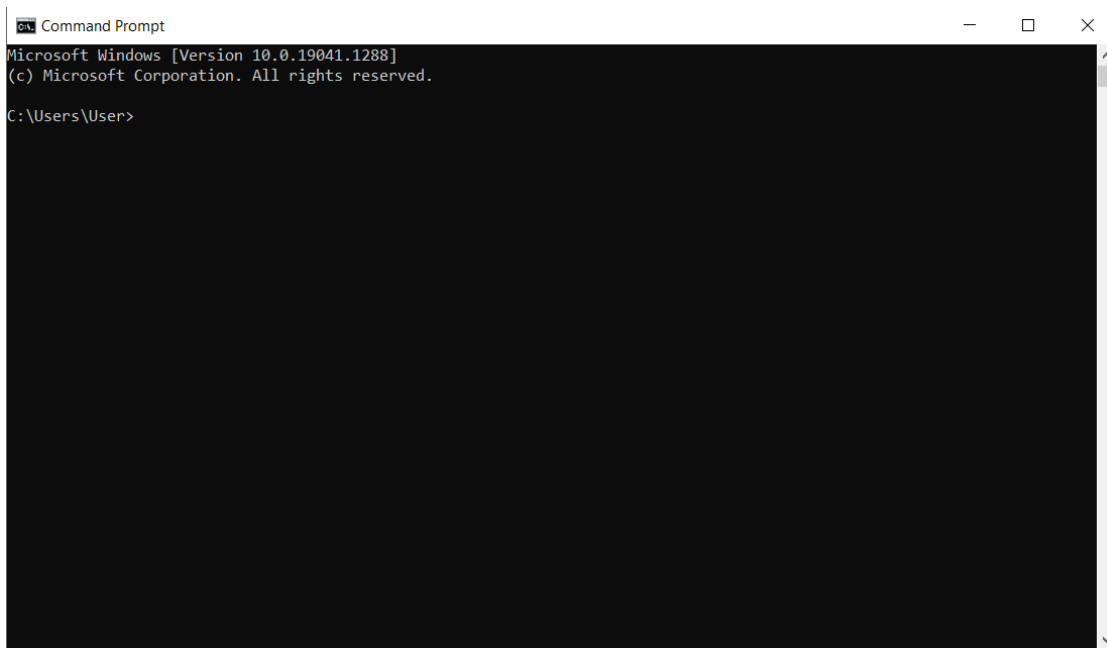
```
Print("Last part of simple subjects?".title())
```

- 1- More rules of python syntax
- 2- Built in functions
- 3- Key words
- 4- Conditions
- 5- Logical operators
- 6- Arguments vs parameters
- 7- Standard python documents
- 8- Type conversion

دیگه بابت این طرح های فوق حرفه ای گرافیکی منو ببخشید 😁 😊

امیدوارم که حالتون خوب باشه.

امروز تصمیم گرفتم کد ها رو توی یک محیز متفاوت تر بزنم، محیط cmd. اگر لینوکس دارید که قطعا میدونید باید چطوری با محیط کامندش کار کنید. اگر هم با ویندوز کار میکنید کافیه سرچ کنید تا محیط cmd براتون بیاد. یادتون نره حتما پایتون رو باید نصب بکنید روی سیستمتون.



```
Command Prompt
Microsoft Windows [Version 10.0.19041.1288]
(c) Microsoft Corporation. All rights reserved.
C:\Users\User>
```

این صفحه ی سیاه خوشگل محیط cmd هست که توش میتونیم کد بزنیم. کافیه کلمه ی python رو تایپ بکنید تا اجرا بشه، حتما امتحان کنید.

1 – توضیحات بیشتر درباره ی سینتکس پایتون :

پایتون مثل اکثر زبان ها روی کوچک و بزرگ بودن حروف متغیر ها حساسه، یعنی ALI با ali با ALi با aLi و... همشون با هم فرق دارن.

و اینکه پایتون روی space ها هم حساسه و حتما باید رعایت کنید.

اگر از ide ها یا text editor ها استفاده میکنید، خودشون اتوماتیک این کار رو براتون میکنند، نگراناش نباشید.

اما اگر خواستید خودتون انجام بدید، طبق استاندارد w3 ، هر تو رفتی یا indent باید معادل 4 تا space یا 2 تا tab باشه.

داخل پایتون میتونید به یک متغیر چندین دیتا رو اختصاص بدید ولی بدونید خود برنامه، آخرین دیتا رو محاسبه میکنه، مثال رو ببین :

```
Command Prompt - python
Microsoft Windows [Version 10.0.19041.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>python
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> ali=23
>>> ali=45
>>> ali=235.5
>>> ali=15226.32585
>>> print(ali)
15226.32585
>>> _
```

راستی، دوست دارم خودت بری و درباره ی نحوه ی کار کامپایلر پایتون و مقایسه اش با کامپایلر زبان C تحقیق کنی. چه فرقی با هم دارند ???

راستی یادت باشه به طور دیفالت، ورودی برنامه با استفاده از input ، متن یا همون string حساب میشن..... اینو داشته باش تا توی قسمت های بعدی بهش برسیم.

: Built in functions – 2

هنوز وارد بحث تابع ها نشدیم ولی لازم دونستم درباره ی built in func ها خیلی کم توضیح بدم.

به جای اینکه 100 خط کد بنویسی و بخوای یه کار مشخصی رو انجام بدی، یه بنده خدایی قبل از تو نوشته این مد ها رو، و توی غالب یک کلمه توی پایتون برایت تعریف کرده.

مثلا توی میخوای اعداد 1 تا 20 رو چاپ کنی، بجای اینکه حلقه ی for . تعریف کردن متغیر های مختلف استفاده کنی، میای و از یک فانکشنی استفاده میکنی که قبلا نوشته شده :

راستی یادت باشه for توی پایتون یکم داستانش با بقیه ی زبان ها فرق داره که فردا میریم سراغش.

```
>>> for numbers in range(0,21):
...     print(numbers)
...
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
>>>
```

یا مثلا توی سر تیترا همه ی فایل هام از title. توی دستور پرینت استفاده میکنم.

```
>>>
>>> print('deghat kon be built in fucntions'.upper())
DEGHAT KON BE BUILT IN FUCNTIONS
>>>
>>> print('deghat kon be built in fucntions'.title())
Deghat Kon Be Built In Fucntions
>>>
```

از این دست فانکشن ها خیلی زیاده که جلوتر میایم سراغش.

اما چرا الان گفتیم؟؟

میخوام نحوه ی استفاده از این built in func ها رو با key word ها مقایسه کنی.

3 – key words :

کلمات کلیدی در پایتون رو قبلا توی فایل قبلی هم گفتیم.

از کلمات کلیدی خیلییی اسفاده میکنیم. مثل del ، if ، else ، و...

من قصدم معرفی کلمات کلیدی نیست فعلا

یادت باشه کلمات کلیدی رو قبل از متغیر یا هر چیزی میاریم اما اون built in function رو همیشه با یک نقطه بعد از متغیر یا متن خودمون میاریم.

بعدا خیلییی باهاشون کار داریم.

4 - شرط ها :

خب دیگه خیلی عادیه پایتون هم مثل بقیه ی زبان ها دارای ساختاری های شرطیه (چشم بسته غیب گفتم) 😊

بریم یه سری ساختاراشو ببینیم (برگشتم توی vs code یکم خوشگلتره) :

```
Python_Challenge.py > ...
1   age=25
2   if age<10:
3       print('salam bache')
4   elif age>10 and age<40:
5       print('salam agha')
6   else:
7       print('salam marde ziba') |
```

اماااa

```
Python 3.10 Features | Ramin F.
# Switch Case, Finally!
def error(code):
    match code:
        case 404:
            return 'not found'
        case 401 | 403:
            return 'not allowed' # we can use logical statements also
        case 418:
            return "i'm teapot!" # :D
        case _:
            return 'Siyanat is on! RIP Internet'

>>> Code Snippet By Ramin F.
```

هنوز به توابع نرسیدیم پس فقط به خود سوئیچ کیس توجه کنید.

به نظرم بهترین مثال برای سوئیچ کیس ها، چیزی هست که مشاهده میکنید.

این سورس کد مربوط به من نیست و توسط Ramin F نوشته شده که حتما

پیجشون رو تگ میکنم.

5 - عملگر ها منطقی :

یادتونه گفتم سینتکس پایتون همونطوریه که حرف میزنی ؟

Python_Challenge.py X

Python_Challenge.py > ...

```
1 age=20
2 if age>10 and age<23:
3     print('ye chizi masalan inja print mishe.')
4 #-----
5 age_ali=23
6 age_mohammad=35
7 if age_ali>20 or age_mohammad<40:
8     print('ye chize alaki inja print mishe')
9 #-----
10 print(not True)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/Desktop/Python/python projects/Python_Challenge.py"
```

```
ye chizi masalan inja print mishe.
```

```
ye chize alaki inja print mishe
```

```
False
```

```
PS C:\Users\User\Desktop\Python\python projects> █
```

ببخشید دیگه مثال ها خیلی ساده است، صرفا مرور کلی مطالب هدفمه.

6 – آرگومان ها در برابر پارامتر ها :

پارامتر ها متغیر هایی هستند که هنگام ساخت توابع از اونها استفاده میکنیم.
آرگومان ها متغیر هایی هستند که پاس میشن به توابع و کلاس ها.

7 – داکيومنت استاندارد پایتون :

خیلییییییی خیلییییی مهمه که همه ی این داکيومنت ها رو حداقل یک نگاه کوچیکی بهش بندازید.

به درد من که خیلی خورده، گاهی اوقات که گیر افتاده بودم واقعا راه گشا بودند.

این هم لینکش : <https://docs.python.org/3/library>

8 – تبدیل انواع داده ها در پایتون :

با اضافه کردن اون وعی که میخواید دیتای شما بهش تبدیل بشه، این کار رو انجام بدید.

خیلی طبیعی که شما نمیتونید یک متن رو به عدد طبیعی یا عدد مختلط تبدیل کنید.

یک دستور جدید : `type()`

با استفاده از این دستور میتونید نوع دیتای خودتون رو در خروجی مشاهده کنید.

بريم سراغ مثال ها :

Python_Challenge.py > ...

```
1 number1=25365
2 print(number1)
3 print(type(number1))
4 print('-----')
5 number2=float(number1)
6 print(number2)
7 print(type(number2))
8 print('-----')
9 string_number=str(number2)
10 print(string_number)
11 print(type(string_number))
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/Desktop/Python/python projects/Python_Challenge.py"
```

```
25365
```

```
<class 'int'>
```

```
-----
```

```
25365.0
```

```
<class 'float'>
```

```
-----
```

```
25365.0
```

```
<class 'str'>
```

```
PS C:\Users\User\Desktop\Python\python projects> █
```

یادت باشه دوباره میگم که هر چیزی رو با input دریافت کنی به عنوان متن یا string در نظر گرفته میشه، پس اگه مثلا میخوای عدد دریافت کنی باید همونجا تبدیل نوع رو انجام بدی، به کد زیر دقت کن :

Python_Challenge.py X

Python_Challenge.py > ...

```
1 number1=input('adad vared konid : ')
2 print(number1)
3 print(type(number1))
4 print('-----')
5 number2=int(input('adad vared konid : '))
6 print(number2)
7 print(type(number2))
8 print('-----')
9 number3=float(input('adad vared konid : '))
10 print(number3)
11 print(type(number3))
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/AppData/Local/Programs/Python/Python39-64/Scripts/python /python projects/Python_Challenge.py"
```

```
adad vared konid : 4562
```

```
4562
```

```
<class 'str'>
```

```
-----
```

```
adad vared konid : 4562
```

```
4562
```

```
<class 'int'>
```

```
-----
```

```
adad vared konid : 4562
```

```
4562.0
```

```
<class 'float'>
```

```
PS C:\Users\User\Desktop\Python\python projects> █
```

حرف نهایی :

مثل همیشه میگم که من اصلا قصدم آموزش نیست و کاملا مبتدی هستم.
این چالش 21 روزه رو صرفا گذاشتم تا محرکی باشه برای من تا مطالب رو دوره کنم.

مثال ها بسیار ساده است، که در چالش بعدی میریم سراغ مثال های سنگین تر و خیلی جذاب تر و کاربردی تر.

قطعا شما همه بلدید با پایتون کار کنید و این مطالب براتون بسیار ساده است.
قطعا مطالب گفته شده کامل نیست، چون حجم مطالب خیلییی زیاده و همیشه همشو در قالب این فایل ها بیان کرد.

خودتون با نوشتن پروژه به ایرادات خودتون برسید و با حل اون ایرادات پیشرفت کنید.

خیلی ممنون از همراهی شما 

کاری از : علی معینیان

Windows PowerShell

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\User> python

Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

```
>>>
>>> # 21 Day Challeneg With Python
>>>
>>> # Part 3 out of 21
>>>
>>> print( 'things that all python developer know !' .title())
Things That All Python Developer Know !
>>>
>>> # 1 - f string and all kind of using print
>>> # 2 - Loops in python
>>> # 3 - important modules
>>> # Next day : we will learn about functional programming and finally we start python data structures
>>>
>>> # By : Ali Moeinian
```

🎄 بریم سراغ پارت 3 از چالش 21 روزه ی من 🎄

سلام!!!!!!!!!!!!!! 😊

روز اول با PyCharm آشنا شدیم.

بعدش با vc code کار کردیم.

دیروز توی محیط cmd کد زدیم.

و امروز هم توی windows PowerShell.

1 – استفاده از f string و انواع روش های چاپ پیام در دستور پرینت :

```
# 1 - Simple way
name = 'ali'
last_name='Moeinian'
age=20
# If you want to print string, all of your data must be string, like here that i changed age into a string variable
print('Hello my full name is : ' + name + ' ' + last_name + ' and im ' + str(age) + ' years old.')
```

```
# 2 - format string, mood 1
name = 'ali'
last_name='Moeinian'
age=20
print(f'Hello my full name is : {name} {last_name} and im {age} years old.')
```

```
# 3 - format string, mood 2 --> Standard and best way ✓
name = 'ali'
last_name='Moeinian'
age=20
print('Hello my full name is : {} {} and im {} years old.'.format(name,last_name, age))
```

```
# 4 - % foramt
name = 'ali'
last_name='Moeinian'
age=20
print('Hello my full name is : %s %s and im %i years old.'%(name,last_name, age))
```

```
/python projects/Python_Challenge.py"
```

```
Hello my full name is : ali Moeinian and im 20 years old.
```

```
Hello my full name is : ali Moeinian and im 20 years old.
```

```
Hello my full name is : ali Moeinian and im 20 years old.
```

```
Hello my full name is : ali Moeinian and im 20 years old.
```

```
PS C:\Users\User\Desktop\Python\python projects> █
```

دوست دارم خودتون برید و در ارتباط با روش % تحقیق کنید که اصلا چطوری باید باهاش کار کرد، ولی حتما Format string رو یاد بگیرید.

2 – حلقه های تکرار در پایتون :

عین فایل قبلی براتون چشم بسته غیب بگم که پایتون هم for و while داره 😁

اما یکم اینجا داستان فرق داره !

قبل از شروع باید بگم شاید مثال ها جلوتر از مبحث باشه، ولی مهم فهم نوع کاربرد حلقه هاست، بزن بریم !

اول بریم سراغ while :

```
Python_Challenge.py > ...
1 # while loop
2 counter = 0
3 age = 23
4 while age<30:
5     print('YOU ARE MORE {} YEARS OLD.'.format(age))
6     counter +=1
7     print(counter)
8     break
9 print('-----')
10 # else statement in while loop
11 counter = 0
12 age = 35
13 while age<30:
14     print('YOU ARE MORE {} YEARS OLD.'.format(age))
15     counter +=1
16     print(counter)
17     break
18 else:
19     print('you are more than 30')
```

یکم ارجاع بزنی به عمو باب؛ تورو خدا!!!
اسم شمارنده هاتون رو i و j نزارید،
بزارید counter اینجوری قشنگ تره.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/AppData/Local/Programs/Python/Python38-64/Python.exe C:/Users/User/Desktop/Python/python projects/Python_Challenge.py
YOU ARE MORE 23 YEARS OLD.
1
-----
you are more than 30
PS C:\Users\User\Desktop\Python\python projects> █
```

قطعا با دستور های break و continue آشنا هستید و نیازی به توضیحات من نیست.

و مطمئن می‌دونید که چه زمانی از حلقه ی while و چه زمانی از for استفاده میکنیم؛ اگر هم اطلاعی ندارید عیب نداره منم از اول نمیدونستم، برید خودتون دنبالش و سرچ کنید و به جواب برسید.

نکته ی انحرافی : در برنامه نویسی علاوه بر مسلط بودن به الفبای کار و زبان برنامه نویسی مد نظرتون، دو تا چیز دیگه خیلییی مهمه :

1 – توانایی سرچ کردن در گوگل

2 – زبان انگلیسی

اما قبل از اینکه بریم سراغ حلقه ی for لازمه که دو تا مفهوم بسیااااار مهم رو براتون توضیح بدم :

دو تا کلمه ی مهم به نام iterable و iterator .

Iterable ها، آبجکت هایی هستند که ما به وسیله ی iterator ها میتونم توی اونها، iterate کنیم 😊

نفهمیدی؟

منم اولاش خیلی سختم بود تا مفاهیمشو درک کنم، ولی عیب نداره الان با هم یاد میگیریم.

مثال رو ببین (هنوز به لیست ها نرسیدیم ولی میدونم بلدی) :

Python_Challenge.py > [🔍] items

```
1 iterable_list=[2,3,4,5,12,14,16, 'ALI', 'IRAN']
2 for items in iterable_list:
3     print(items)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2
3
4
5
12
14
16
ALI
IRAN

PS C:\Users\User\Desktop\Python\python projects> █

این لیست **iterable** هست، یعنی قابلیت اینو داره که ما با حلقه ی **for** داخل این لیست گردش کنیم.
یعنی به تمام اعضای داخلی لیست دسترسی داشته باشی.

حلقه ی **for**، ابزار **iterate** کردن است.

یعنی حلقه ی **for**، **iterator** است که با استفاده از متغیر **items**، اعضای داخل لیست را **iterate** میکند.

خب این مفاهیم رو که خوب فهمیدیم، بریم سراغ حلقه ی **for** .

حلقه ی for کارش یکم متفاوته توی پایتون و در 90 درصد مواقع برای دسترسی داشتن به محتوای هرررررر چیزی استفاده میشه.

مثلا توی سی پلاس پلاس از حلقه ی for برای ساخت اعداد 1 تا 10 استفاده میکردی و خودش برات اعداد رو میساخت :

```
main.cpp
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i;
6      for(i=0 ; i<11 ; i++){
7
8          cout<<i<<" ";
9      }
10     return 0;
11 }
12
```

0 1 2 3 4 5 6 7 8 9 10

...Program finished with exit code 0
Press ENTER to exit console.

اما توی پایتون، از حلقه ی for استفاده میکنیم تا بیایم داخل تابعی که اعداد 0 تا 10 رو میسازه، گردش کنیم تا اون اعداد رو نمایش بده، یعنی حلقه ی for خودش واسه خودش چیزی رو تولید نمیکنه. مثال صفحه ی بعد رو ببین :

```
Python_Challenge.py ×
Python_Challenge.py > [🔗] numbers
1  for numbers in range(0,11):
2  print(numbers)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\User\Desktop\Python\python projects> & C:/Users/
/python projects/Python_Challenge.py"
0
1
2
3
4
5
6
7
8
9
10
PS C:\Users\User\Desktop\Python\python projects> █
```

اصولا وقتی از حلقه ی for استفاده میکنیم، یک متغیر برای گردش در جایی که میخواهیم تعریف میکنیم، مثلا من متغیرم رو توی مثال بالا، اسمش رو گذاشتم numbers.

با حلقه ی for خیلی باید کار کنید تا بفهمید ماهیتش چطوریه و چه کار هایی ارزش بر میاد.

راستی اون () range چیه من استفاده کردم؟

یه تابع خیلییییی پر کاربرد توی پایتون برای ساخت اعداد هست.

(گام پیشرفت یا پسرفت ، انتهای بازه ، شروع بازه) range

مثال ها رو ببین و به یک نکته ی خیلییی مهم توی قسمت " انتهای بازه " رو خودت بفهم.

```
1 # simple range example
2 for numbers in range(0,11):
3     print (numbers)
4 print('-----')
5 #range with positive step
6 for numbers in range(0,11,2):
7     print (numbers)
8 print('-----')
9 #range with negative step
10 for numbers in range(11,0,-2):
11     print (numbers)
12 print('-----')
13
```

```
0
2
4
6
8
10
-----
```

```
11
9
7
5
3
1
-----
```

```
0
1
2
3
4
5
6
7
8
9
10
-----
```

3 – ماژول های مهم پایتون :

این قسمت صرفاً معرفی هست، در چالش پروژه ها از این ماژول ها بیشتر استفاده میکنیم.

ماژول های سری کد های بسیار به درد بخور و آماده هتند که یک سری انسان زیبا اونها رو نوشتن و در اختیار ما قرار دادن تا زمانمون حفظ بشه.

ماژول ها رو باید توی برنامه ات بیاری به طور دستی با دستور `import`.

ماژول های زمان و اعداد تصادفی رو اصولاً بیشتر باهاشون کار داریم.

این قسمت رو خودتون برید بخونید تا آشنا شید، قابلیت سرچ کردن خودتون

رو محک بزنید ببینم چیکار میکنید ! 😊

ممنونم که منو دنبال میکنید ❤️

کاری از : علی معینیان


```
Python_Challenge.py > ...
1 # 21 day challenge with python
2
3 # زبان آقای هاشمی
4 bechap('python data structures (pt 1) and functional programming !')
5
6
7 one = 'Functional programming and the reason why is that good ?'
8 two = 'Recursive fuinctions in python'
9 Three = 'Introduction to python data structure 😊'
10 Four = 'Lists and all useful methodes'
11 Five = 'Tuples VS Lists'
12
13 # By : Ali Moeinian
```

سلام

قبل از شروع دوره ی مباحث امروز، یه نگاهی بنداز به عکس بالایی!

زبان آقای هاشمی رو حتما بخون 😊

داکیومننت هاش و نحوه ی نصبش داخل گیت هاب هست!

البته جنبه ی فان داره 🌻🌻

1 – توابع در برنامه نویسی پایتون :

توابع در برنامه نویسی چیا هستن ؟؟؟؟

ببین بعضی وقت ها تو یک تکه کدی داری که ممکنه در طول برنامه چندین بار ازش استفاده بکنی.

حالا میایم و میگیریم به جای اینکه هییییی بیای و توی بدنه ی اصلی برنامه ات این تکه کد مشخص رو تکرار کنی، یه جایی مینویسیش و هر جا دوست داشتی اسمشو صدا میزنی.

نحوه ی استفاده از توابع در پایتون :

این برنامه اسمتو از ورودی میگیره و 4 بار پشت سر هم چاپش میکنه (خیلی برنامه ی سنگینی هست.)

```
def exapmle_of_function(Input_arg):  
    print(Input_arg * 4)  
  
input_arg=input('Please write your name : ')  
print('-----')  
exapmle_of_function(input_arg)
```

در کل نحوه ی کار با توابع به همین شکل هست.

اما سورس کد بالا رو اگه بزاری جلوی عمو باب، فکر کنم یه نگاه سنگینی بهمون میکنه !

عمو باب : بدنه ی تابع شما هیچ چیزی رو نباید چاپ کنه و صرفا باید موارد نهایی که به عنوان جواب میخواهید رو به بدنه ی اصلی برنامه برگردونه.
پس کد استاندارد تر، این خواهد بود :

```
def exapmle_of_function(Input_arg):  
    return Input_arg * 4  
  
input_arg=input('Please write your name : ')  
print('-----')  
print(exapmle_of_function(input_arg))
```

حالا یه دعوایی هم هست بین طرفدارای Functional Programming و طرفدارای object oriented programming که همیشه هم بازارش داغ بوده؛ به موقع یکم وارد این دعوایا هم میشیم.

من خودم عاشق برنامه نویسی و استفاده از توابع توی برنامه هام هستم و اکثر برنامه هایی هم که مینویسم سعی میکنم توش از functional programming استفاده کنم.

برنامه نویسی با استفاده از توابع، خوبی های خیلی زیادی داره :

1 – بدنه ی اصلی برنامتون به شدت تمیز تر و خلوت تر خواهد شد.

2 – خوانایی کد شما توسط یک فرد دیگه خیلی افزایش پیدا میکنه.

3 – کد شما بسیار منظم تر خواهد شد.

4 – این نوع برنامه نویسی به شما کمک میکنه که مسائل سخت و پیچیده رو به راحتی بتونید به راه حلش نظم بدید و همین باعث میشه خیلی راحت تر از قبل مسائل رو حل کنید.

2 – توابع بازگشتی در پایتون :

تابع بازگشتی : اگر در بدنه ی یک تابع، از اسم خود اون تابع استفاده بکنیم، اسمش همیشه تابع بازگشتی !

شما بری تو کورس فوق تخصصی پایتون در دانشگاه MIT بشینی، به این بحث که میرسه، قطعاً مثال فاکتوریل و فیبوناچی رو میزنه و چون این چالش ما در بالاترین سطح دنیا قرار داره (😄) ما هم همین مثال ها رو اینجا میزنیم :

1 – فاکتوریل

```
def factorial(number):  
  
    if number == 1:  
        return 1  
    else:  
        return (number * factorial(number-1))  
  
numer=int(input('Please Enter a number : '))  
print(factorial(numer))
```

```

def recur_fibo(n):
    if n <= 1:
        return n
    else:
        return(recur_fibo(n-1) + recur_fibo(n-2))

nterms=int(input('Enter a Number : '))
if nterms <= 0:
    print("Plese enter a positive integer")
else:
    print("Fibonacci sequence:")
    for i in range(nterms):
        print(recur_fibo(i))

```

3 - ساختمان های داده در پایتون :

رسیدیم به نقطه قوت و یکی از مهم ترین مباحث در پایتون 🤩

ساختمان داده های پایتون :

1 - لیست (مهم ترین - رتبه 1)

2 - تاپل

3 - دیکشنری (رتبه 2)

4 - ست

توی این فایل در ارتباط با لیست ها و تاپل ها توضیح میدم.

لیست ها خیلییییییییی مهم اند، خیلییییی زیاد.

من تا حالا نشده پروژه ای بنویسم و توی اون از لیست ها استفاده نکنم.

لیست ها یکی از ساختمان های داده در پایتون اند که همه چی توشون هست، اعداد صحیح، اعداد اعشاری، اعداد مختلط، متن ها و حتی انواع دیگر ساختمان داده ها !

بی مقدمه بریم سراغ مهم ترین کار هایی که میتونیم با لیست ها انجام بدیم.

```
Python_Challenge.py > ...
1 numbers=[1,2,12.5,3565,225.256,25j]
2 print(numbers)
3
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/
/python projects/Python_Challenge.py"
[1, 2, 12.5, 3565, 225.256, 25j]
PS C:\Users\User\Desktop\Python\python projects> |
```

```
Python_Challenge.py > [🔍] items
1 numbers=[1,2,12.5,3565,225.256,25j]
2 for items in numbers:
3     print(items)
4
5
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/U
/python projects/Python_Challenge.py"
1
2
12.5
3565
225.256
25j
PS C:\Users\User\Desktop\Python\python projects> |
```

```
Python_Challenge.py > [🔍] items
1  names=['Ali' , 'mohammad' , 'maryam' , 'neda']
2  for items in names:
3      print(len(items))
4
5

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\User\Desktop\Python\python projects> & C:/Users/
/python projects/Python_Challenge.py"
3
8
6
4
PS C:\Users\User\Desktop\Python\python projects> [ ]
```

```
Python_Challenge.py > [🔍] items
1  names=['Ali' , 'mohammad' , 'maryam' , 'neda']
2  int_len_names=[]
3  for items in names:
4      str_to_int=int(len(items))
5      int_len_names.append(str_to_int)
6  print(max(int_len_names))
7

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\User\Desktop\Python\python projects> & C:/Users/
/python projects/Python_Challenge.py"
8
PS C:\Users\User\Desktop\Python\python projects> [ ]
```

یکی از نقاط قوت لیست ها اینه که یک سری متد دارند که خیلییییی به دردمون میخوره و عملا هر وقت از لیست ها استفاده میکنیم، مجبوریم از این متد ها هم استفاده بکنیم.

اما نکته ی مهمه اینه که خودتون باید برید و روی این متد ها مسلط بشید چرا که خیلییی مهم هستند و صرفا نباید حفظشون بکنید.

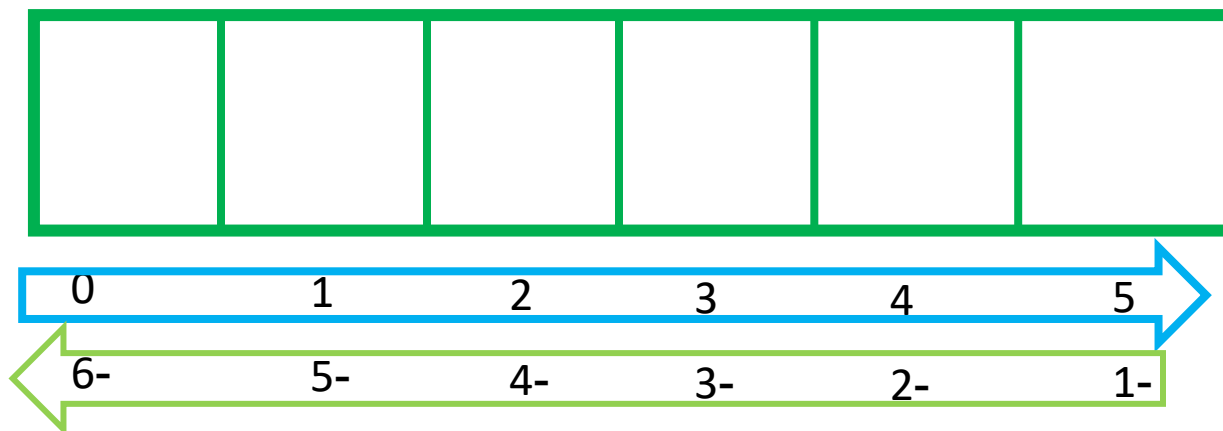
از داکيومنت هايي كه توي پارت صفر اين چالش معرفي كردم استفاده كنيد تا
تمامي متد ها و كارهايي كه ميتونيد با ليست ها انجام بديد رو يادبگيريد .

```
Python_Challenge.py > ...
1 numbers=[int(input('enter numbers : ')) for nums in range(int(input('how many numbers you want to enter?')))]
2 even_nums=[]
3 odd_nums=[]
4 for items in numbers:
5     if items % 2 == 0:
6         even_nums.append(items)
7     if items % 2 != 0:
8         odd_nums.append(items)
9 print(even_nums)
10 print(odd_nums)
11
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/
/python projects/Python_Challenge.py"
how many numbers you want to enter?6
enter numbers : 12
enter numbers : 10
enter numbers : 11
enter numbers : 30
enter numbers : 33
enter numbers : 69
[12, 10, 30]
[11, 33, 69]
PS C:\Users\User\Desktop\Python\python projects> |
```

راستي يادتون نره ايندكس هاي ليست ها از 0 به بعد شروع ميشه :



هر چقدر بگم که لیست ها و متد های مربوطه بهشون مهم هستن، کم گفتم!

Method	Description
<u>append()</u>	Adds an element at the end of the list
<u>clear()</u>	Removes all the elements from the list
<u>copy()</u>	Returns a copy of the list
<u>count()</u>	Returns the number of elements with the specified value
<u>extend()</u>	Add the elements of a list (or any iterable), to the end of the current list
<u>index()</u>	Returns the index of the first element with the specified value
<u>insert()</u>	Adds an element at the specified position
<u>pop()</u>	Removes the element at the specified position
<u>remove()</u>	Removes the item with the specified value
<u>reverse()</u>	Reverses the order of the list
<u>sort()</u>	Sorts the list

اونایی که دورشون رو مستطیل کشیدیم از بقیه پر کاربرد تر و مهم تر هستند.

4 - تاپل ها :

آقا تاپل ها از دو لحاظ با لیست ها فرق میکنند :

1 - ظاهر : tuple=() list=[]

2 - برخی از ویژگی ها :

List	Tuple
It is mutable	It is immutable
The implication of iterations is time-consuming in the list.	Implications of iterations are much faster in tuples.
Operations like insertion and deletion are better performed.	Elements can be accessed better.
Consumes more memory.	Consumes less memory.
Many built-in methods are available.	Does not have many built-in methods.
Unexpected errors and changes can easily occur in lists.	Unexpected errors and changes rarely occur in tuples.

با توجه به تفاوتی که دارند، اما متد هایی که استفاده میشه برای هر دو یکسان هست.

نکته ی درگوشی : من تاحالا خیلی با تاپل ها کار نکردم و اکثر نیاز هام رو

لیست ها و متد های مربوط به لیست ها برطرف میکنه 😊

اوایلش من هم توی حفظ کردن متد ها مشکل داشتم، ولی وقتی توی برنامه نویسی و نوشتن پروژه های خودم ازشون استفاده کردم، کم کم همشون رو حفظ شدم. پس نگران نباشید، به مرور زمان همشون رو یاد میگیرید.

**** یادتون نره که هدفمون صرفا دوره کردن مطالب کلی هست ****

ارادت 

کاری از : علی معینیان

21 day challenge with python ❤️

Part 5 out of 21

CA. Command Prompt - python

```
C:\Users\User>python
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> print(' Data structures in python - last part ')
Data structures in python - last part
>>>
>>> # 1 - all kinds of variables in python
>>>
>>> # 2 - all ways of set a name for variables
>>>
>>> # 3 - Dictionaries
>>>
>>> 3 4 - Sets_
```

سلاااااام و صد درووووود 🎄

حالتون چطورہ ؟

1 - انواع متغیرها در پایتون :

متغیر سراسری : متغیری هست که در دسترس تمام توابع موجود در برنامه خواهد بود.

```
Python_Challenge.py > ...
1 # Global variable
2 my_age=20
3 my_name='ali'
4 my_country='iran'
5 def show_age():
6     print('My first name is {} and im {} years old and im from {} '.format(my_name , my_age , my_country))
7
8 show_age()
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/
My first name is ali and im 20 years old and im from iran .
PS C:\Users\User\Desktop\Python\python projects> █
```

در مثال بالا، کدام متغیرها سراسری هستند؟

تعریف خارجی متغیر سراسری :

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

متغیر محلی : متغیر هایی هستند که صرفا داخل توابع تعریف میشوند.

```
Python_Challenge.py > [🔍] age
1 # A function with local variable
2 def double_age (age):
3     make_age_double = age*2
4     return make_age_double
5
6 age = int(input(' Please eneter your age : '))
7 print(double_age(age))
8
```

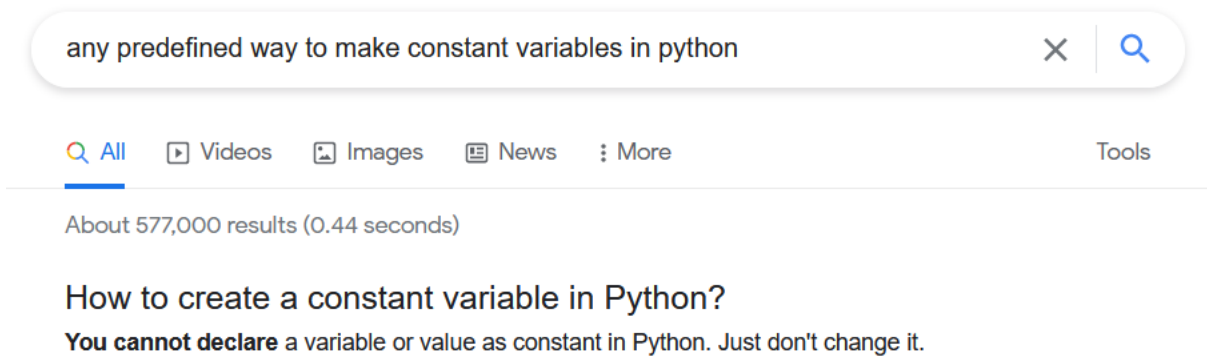
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User
Please eneter your age : 20
40
PS C:\Users\User\Desktop\Python\python projects> █
```

ثابت های سراسری : متغیر هایی هستند که در طول برنامه مقدار آنها ثابت خواهد بود.

نکته ی مهم : تا حد امکان از تعریف ثابت های سراسری دوری کنید.

اما آیا ممکنه ما بتونی متغیر های ثابت بسازیم ???
بزارید از گوگل پرسیم :



خب ظاهرا به مشکل خوریدم !

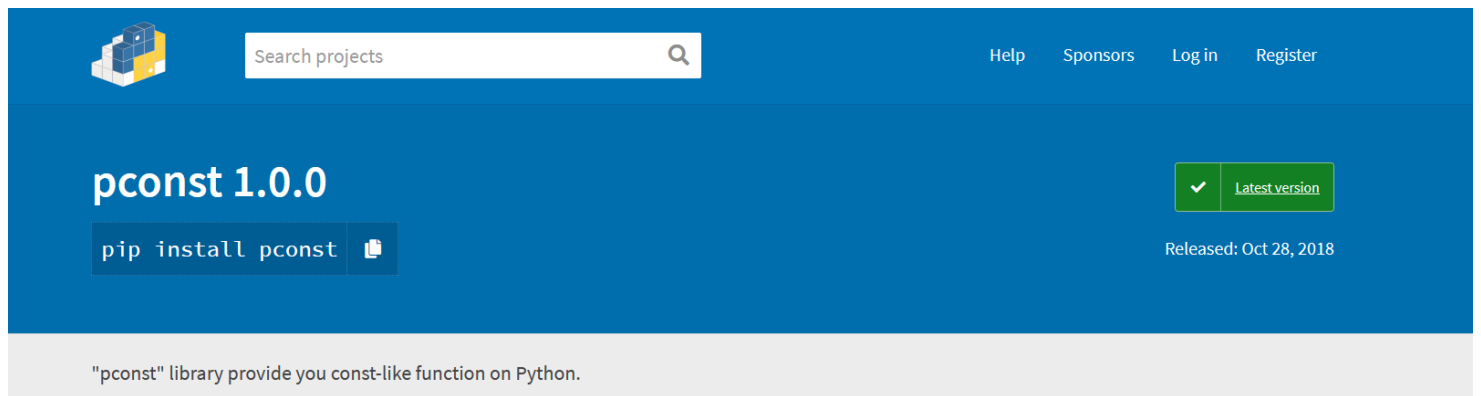
اما 📺 یادتونه گفتم برید خودتون در ارتباط با ماژول های به درد بخور پایتون بخونید؟؟

امروز میخوام یکی از ماژول ها و کتابخونه های به درد نخور پایتون برای تعریف ثابت های سراسری رو براتون معرفی کنم.

اما ثبل از هر چیزی سایت بسیار مهم pypi.org رو میخوام بهتون معرفی بکنم . جایی که هر کتابخونه یا ماژولی برای پایتون رو داره و به راحتی میتونید برید سراغش و نصبشون کنید.

نحوه ی نصب رو هم کاملا توضیح داده.

اما کتابخونه ای که ما برای تعریف ثابت های سراسری ازش استفاده خواهیم کرد :



The screenshot shows the PyPI page for the 'pconst' library. At the top, there is a search bar and navigation links for 'Help', 'Sponsors', 'Log in', and 'Register'. The main content area displays 'pconst 1.0.0' with a 'Latest version' badge. Below the version, there is a code block containing the command 'pip install pconst' and a 'Released: Oct 28, 2018' date. At the bottom, a description reads: '"pconst" library provide you const-like function on Python.'

ولی خب ... کامل نمیگم باید چیکار کرد ...

دوست دارم خودتون برید دنبالش؛ اما یک راهنمایی کوچولو هم میزارم براتون :



Constant variable the name itself says that it is constant. We have to define a constant variable at the time of declaration. After that, we will not be able to change the value of a constant variable. In some cases, constant variables are very useful.

Creating constant variables, functions, objects is allowed in languages like c++, Java. But in python creating constant variable, it is not allowed. There is no predefined type for a constant variable in Python. But we can use **pconst** library for that.

Installation:

```
pip install pconst
```

Below are some examples which depict how to use constants in python

Example 1:

اگر برنامه نویس هستی و از سایت Geeks for Geeks خبری نداری، خسته نباشی ! 😞 😞

2 – انواع نحوه ی نام گذاری برای متغیر ها :

```
1 # All cases in set a name for varibale
2
3 myfavoriatename = 'ali' # Lazy case
4 MYFAVORITENAME = 'ali' # Upper flat case
5 myFavoriteName = 'ali' # Lower camel case
6 MyFavoriteName = 'ali' # Upper camel case *
7 my_favorite_name = 'ali' # Snake case *
8 My_Favorite_Name = 'ali' # Snake camel case *
9 MY_FAVORITE_NAME = 'ali' # Screaming snake case
10 _myfavoriatename = 'ali' # Undercore notion
11
```

اون کیس هایی که براشون ستاره گذاشتم، استاندارد ترین حالت هایی هستند که میتونید برای نام گذاری متغیر ها، ازشون استفاده کنید.

این نامگذاری زمانی ارزش پیدا میکنه که کد شما رو یک شخص حرفه ای دیگر ببینه؛ چرا؟

چون باعث تمیزی کد شما و خوانا تر بودن کد شما خواهد شد و این یعنی شما آدمی هستی که خودت فهمیدی چیکار کردی و این یعنی تو عالی هستی



3 - دیکشنری ها :

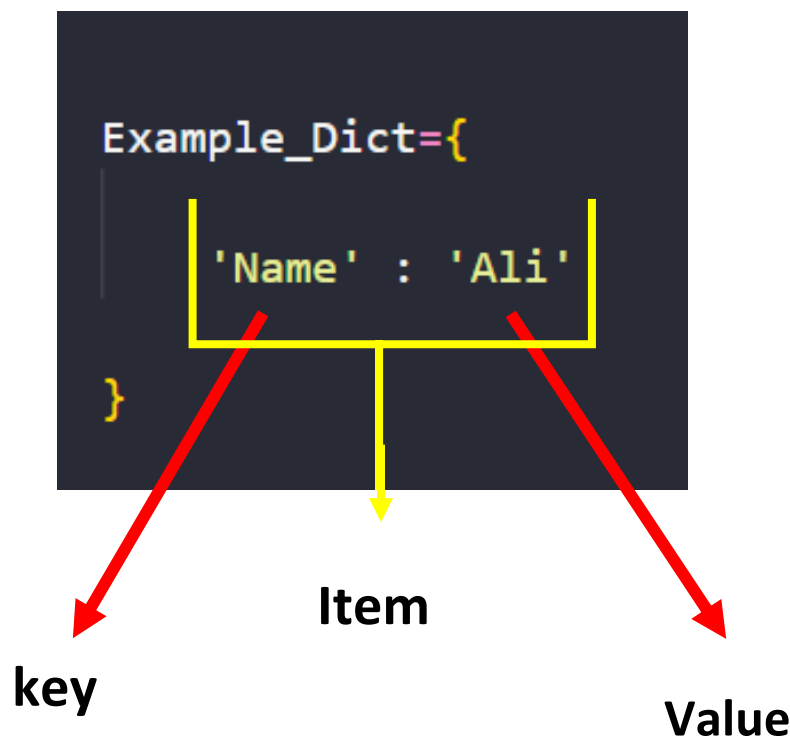
خب من خیلی نمیخوام از دیکشنری ها بگم ! چرا ؟

چون خیلییی مهمه 😊

عین لیست ها و تاپل ها، باید خودتون برید دنبالش تا مسلط بشید.

```
1 Example_Dict={
2     'name' : 'ali',
3     'age' : 21,
4     'country' : 'iran',
5     'programmin language' : 'python , c++'
6 }
```

دو تا اصطلاح مهم توی دیکشنری ها :



و اما مهم ترین متد های دیکشنری ها :

Method	Description
<code>clear()</code>	Removes all the elements from the dictionary
<code>copy()</code>	Returns a copy of the dictionary
<code>fromkeys()</code>	Returns a dictionary with the specified keys and value
<code>get()</code>	Returns the value of the specified key
<code>items()</code>	Returns a list containing a tuple for each key value pair
<code>keys()</code>	Returns a list containing the dictionary's keys
<code>pop()</code>	Removes the element with the specified key
<code>popitem()</code>	Removes the last inserted key-value pair
<code>setdefault()</code>	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
<code>update()</code>	Updates the dictionary with the specified key-value pairs
<code>values()</code>	Returns a list of all the values in the dictionary

دیگه تکرار نمیکنم که باااااید با این متد ها در غالب مینی برنامه هاتون کار کنید تا بتونید روی همشون مسلط بشید 😊

و برای آخرین نکته ی این مبحث میتونم بگم که دیکشنری های تو در تو را خیلییی باهاش کار خواهید کرد .

راستی یادتون باشه ! حتما دیکشنری های داخل لیست ها و نحوه ی دسترسی به key ها و value ها در دیکشنری رو تمرین کنید!

```
myfamily = {  
    "child1" : {  
        "name" : "Emil",  
        "year" : 2004  
    },  
    "child2" : {  
        "name" : "Tobias",  
        "year" : 2007  
    },  
    "child3" : {  
        "name" : "Linus",  
        "year" : 2011  
    }  
}
```

4 - ست ها :

ست ها هم از ديگر ساختمان داده ها در پايتون هستند، اما من خيلي باهاشون کار نکردم تا حالا.

من هميشه هر نيازي که داشته ام رو با ليست ها و ديکشنري ها يا ترکيب اين دو ساختمان داده، حل کرده ام و خيلي کاري به ايل ها و ست ها نداشتم.

اما صرفا متد هاشو ميزارم تا مطالعه کنيد :

Method	Description
<u>add()</u>	Adds an element to the set
<u>clear()</u>	Removes all the elements from the set
<u>copy()</u>	Returns a copy of the set
<u>difference()</u>	Returns a set containing the difference between two or more sets
<u>difference_update()</u>	Removes the items in this set that are also included in another, specified set
<u>discard()</u>	Remove the specified item
<u>intersection()</u>	Returns a set, that is the intersection of two other sets
<u>intersection_update()</u>	Removes the items in this set that are not present in other, specified set(s)
<u>isdisjoint()</u>	Returns whether two sets have a intersection or not
<u>issubset()</u>	Returns whether another set contains this set or not
<u>issuperset()</u>	Returns whether this set contains another set or not
<u>pop()</u>	Removes an element from the set
<u>remove()</u>	Removes the specified element
<u>symmetric_difference()</u>	Returns a set with the symmetric differences of two sets
<u>symmetric_difference_update()</u>	inserts the symmetric differences from this set and another
<u>union()</u>	Return a set containing the union of sets
<u>update()</u>	Update the set with the union of this set and others

اما حرفام به اين معنی نيست که اصلا ست ها به درد نميخوره !

به مثال های زیر درباره ی کاربرد ست ها توجه کنید :

```
A = {1, 2, 3}
B = {1, 4, 5}

print(A.difference(B))
{2, 3}

print(B.difference(A))
{4, 5}

print(A.intersection(B))
{1}

print(A.union(B))
{1, 2, 3, 4, 5}
```

```
A = {1, 2, 3, 4}
B = {1, 2}
C = {1, 2, 8}

print(A.issuperset(B))
True

print(B.issubset(A))
True

print(A.issuperset(C))
False
```

```
>>> a = {1, 2, 3, 4}
>>> b = {2, 3, 4, 5}
>>> c = {3, 4, 5, 6}
>>> d = {4, 5, 6, 7}

>>> a.union(b, c, d)
{1, 2, 3, 4, 5, 6, 7}

>>> a | b | c | d
{1, 2, 3, 4, 5, 6, 7}
```

به خاطر سفری که برام پیش اومده، شاید فردا و پس فردا کمی روند دوره ی مطالبمون کند تر بشه و مطالب کمتری رو بیان کنم اما بعد از این سفر، با قدرت تر پیش خواهیم رفت 😊

ارادات ❤️

کاری از : علی معینیان

21 day challenge with python

Part 6 out of 21

- 1 – Python Standard Document
- 2 – another famous site for python
- 3 – Useful Built in functions in python

 سلام دوستان

به علت اینکه سفر فوری برام پیش اومد و بسیاااار خسته هستم، سعی کردم فایل امروز رو خیلی خلاصه بکنم و کمی از سرعتمون کم میشه و قطعاً جبران میکنم.

در ارتباط با داکيومنت استاندارد پایتون، قبلا هم خیلی کم توضیح دادم اما الان دوست دارم کمی مفصل تر توضیح بدم !

- Introduction
 - Notes on availability
 - Built-in Functions
 - Built-in Constants
 - Constants added by the `site` module
 - Built-in Types
 - Truth Value Testing
 - Boolean Operations — `and`, `or`, `not`
 - Comparisons
 - Numeric Types — `int`, `float`, `complex`
 - Iterator Types
 - Sequence Types — `list`, `tuple`, `range`
 - Text Sequence Type — `str`
 - Binary Sequence Types — `bytes`, `bytearray`, `memoryview`
 - Set Types — `set`, `frozenset`
 - Mapping Types — `dict`
 - Context Manager Types
 - Type Annotation Types — `Generic Alias`, `Union`
 - Other Built-in Types
 - Special Attributes
 - Built-in Exceptions
 - Exception context
 - Inheriting from built-in exceptions
 - Base classes
 - Concrete exceptions
 - Warnings
 - Exception hierarchy
 - Text Processing Services
 - `string` — Common string operations
-

این ها همش یکسری موضوع های خیلی کوچک از داکيومنت های استاندارد پایتون هست که واقعا هر پایتون دولوپری باید این ها رو حداقل یک بار مطالعه کنه.

قصد بر حفظ کردن این موارد نداشته باشید چون قطعاً نشدنی هستند اما اینکه یک بار از روی هر کدوم بخونید و بتونید درک کنید که هر کتابخونه کجا و چطوری میتونه بهترتون کمک کنه.

خیلی از مسائلی که با چند خط میتونید حل کنید، با استفاده از توابع همین کتابخونه های استاندارد پایتون، در یک خط حلشون کنید؛ مثل میانگین گیری از اعداد.

در ارتباط با نحوه ی استفاده از این فایل ها داخل برنامه و نحوه ی استفاده از سینتکس اونها، میتونید مثال های داخل همین داکيومنت رو مطالعه کنید؛ قطعاً مثال هاش خیلی کارگشا هستند براتون.

2 – یک سایت جالب و باحال برای پایتون :

توی دوره ای که از سایت کورسرا دارم میگذروم برای پایتون، سایت زیر رو به عنوان مرجع و نقشه ی راه یادگیری معرفی کردند.

جادى هم همینطور 😊

به نظرم توصیه هایی که این سایت به یک پایتون دولوپر میکنه میتونه خیلی بهش کمک کنه و واقعا کاربردی هستند.

Python for Everybody

The goal of this book is to provide an Informatics-oriented introduction to programming. The primary difference between a computer science approach and the Informatics approach taken in this book is a greater focus on using Python to solve data analysis problems common in the world of Informatics.

The sample code and data files for the book is here: [Code Samples](#).

Other courses / web sites using this book

Book translations:

- English - Python for Everybody: Exploring Data in Python3
 - Printed book on Amazon India (low-cost shipping within India thanks to Shroff Publishing)
 - Kindle edition of the book
 - Free: PDF, HTML, EPUB
 - HTML with examples in Jupyter Notebooks from LibreTexts.org
 - Interactive HTML from Trinket.io
- Spanish - Python para todos: Explorando la información con Python 3
 - Translated book, autograders, resources, and web site at <https://es.py4e.com>
 - Github repo - Contributors: Juan Carlos Perez Castellanos, Juan Dougnac, Daniel Merino Echeverría, Jaime Bermeo Ramirez and Fernando Tardio.
- Italian Python per tutti: Esplorare dati con Python3
 - Free PDF, EPUB
 - Book source on github, (Thanks to Alessandro Rossetti and Vittore Zen)
- Portuguese - Python Para Todos: Explorando Dados com Python 3
 - Free: PDF, EPUB
 - Book source on github and translation team (Thanks to Yuri Loia de Medeiros).
- Polish - Python dla wszystkich: Odkrywanie danych z Python 3 (from Amazon.pl and from Amazon.de)

Select Language

این سایت در قسمت های مختلف بهتون کتاب های مرجع، نکات بسیار کاربردی و هر آنچیزی که نیاز دارید رو معرفی میکنه .

یه دونه پایتون کامپایلر آنلاین هم داره که خیلی تعریفی نداره 😊

بهترین پایتون کامپایلر که میتونید به طور آنلاین ازش استفاده بکنید رو میتونید توی سایت زیر پیدا کنید (خیلی عالیه - یه جورایی یک ide آنلاین هست) :

Replit.com

3 – برخی از معروف ترین Built in Functions ها که خیلی مورد استفاده

قرار میگیره :

```
Python_Challenge.py > ...
1   for numbers in range(10,20,3):
2       print(numbers)
3   print('-----')
4   for numbers in range(30,10,-4):
5       print(numbers)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Us
/python projects/Python_Challenge.py"
10
13
16
19
-----
30
26
22
18
14
PS C:\Users\User\Desktop\Python\python projects> █
```

مثال بالا از range() است که قبلا هم در ارتباط باش حرف زده بودم.

بریم سراغ یک دستور جدید : `ord()`

با استفاده از این دستور میتونید کد اسکی کاراکتر مورد نظر خودتون رو پیدا کنید.


اگر نمیدونید کد ASCII چیه، یکم باید بیشتر در ارتباط با انواع کد های حرفی - عددی بخونید.

```
Python_Challenge.py
1 print(ord('a'))
2 print('-----')
3 print(ord('A'))
4 print('-----')
5 print(ord('b'))
6 print('-----')
7 print(ord('J'))
8 print('-----')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/User
/python projects/Python_Challenge.py"
97
-----
65
-----
98
-----
74
-----
PS C:\Users\User\Desktop\Python\python projects> █
```

راستی برعکس این دستور، `char()` هست !

 Python_Challenge.py

```
1 print(len("My Name Is Ali Moeinian"))
2 print('-----')
3 print(len('Python'))
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/U
/python projects/Python_Challenge.py"
```

```
23
```

```
-----
```

```
6
```

```
PS C:\Users\User\Desktop\Python\python projects> |
```

دستورات مربوط به رشته ها :

```
Python_Challenge.py
1 print('hello' .upper())
2 print('HELLO' .lower())
3 print('hello im ali.' .title())
4 print(List(reversed('Hello Howe are you ?')))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Users/User/AppData/Local/Programs/Python/Python31
/python projects/Python_Challenge.py"
HELLO
hello
Hello Im Ali.
['?', ' ', 'u', 'o', 'y', ' ', 'e', 'r', 'a', ' ', 'e', 'w', 'o', 'H', ' ', 'o', 'l', 'l', 'e', 'H']
PS C:\Users\User\Desktop\Python\python projects> |
```

برخی از توابع برای اعداد :

```
Python_Challenge.py
1
2 print(round(1.2332664))
3 print(round(1.53365))
4 print(abs( -12.322 ))
5 print(abs( -6 ))
6 print(min(1,2,3,0,12,356))
7 print(max(124,123,6625,120,124))
8
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/Use
/python projects/Python_Challenge.py"
1
2
12.322
6
0
6625
PS C:\Users\User\Desktop\Python\python projects> |
```

کمی از مبانی اعداد بگم براتون :

```
Python_Challenge.py  
1 print(bin(8))  
2 print(hex(325))  
3
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python\python projects> & C:/U  
/python projects/Python_Challenge.py"  
0b1000  
0x145  
PS C:\Users\User\Desktop\Python\python projects> □
```

ممنون از همراهی شما ❤️

در قسمت های بعدی بیشتر از قبل پیشرفت خواهیم کرد.

کاری از : علی معینیان

21 day challenge with python ❤️

Part 8 out of 21



1 – More tips on past episodes

2 – Be friend with errors

3 – try, except, finally

4 – map

5 – math

6 – more about python standard document

(lambda and filter are for the next part)

سلااااا ، امیدوارم حالتون خوب باشه 🎄

تا قبل اینکه این فایل رو آماده کنم، خیلی دغدغه ی فکری داشتم و یکم حرفام رو هم پست کردم توی لینکدین، شایدتونستید کمک کنید! مرسی ازتون📍

1 – نگاهی به مباحث قبلی و تکمیل نکات تا Control Flow Tools

در ارتباط با استفاده از cmd برای پایتون، لازمه یک دستور خیلی کوچیک و ابتدایی رو برای خروج از برنامه یادآوری کنم.

```
Command Prompt
Microsoft Windows [Version 10.0.19041.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>python
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> print("Aim that you do all your works and you want to quit from python ")
Aim that you do all your works and you want to quit from python
>>>
>>> # Remember :
>>> quit()
```


در ارتباط با کامنت گذاری در برنامه های پایتون قبلا حرف زدیم، اما گاهی کامنت گذاشتن ما خیلی طولانی میشه ، بریم مثال پایین رو ببینیم در ارتباط با کدهایی هست که خودم زدم:

```
python projects > 4-class.py > ...
1
2 # class variable : property of the class instance variable : unique property of the object of the class lets work on it and learn more about
3
4 # Question : we want to add 10% to payment of each employee
5 class employee:
6     pay_rising=0.1
7
8     def __init__(self,FirstName,LastName,payment):
9         self.FirstName=FirstName
10        self.LastName=LastName
11        self.payment=payment
12        self.mail=str(self.FirstName) + "_" + str(self.LastName) + "@gmail.com"
13
14    def fullname(self):
15        return 'Full Name : %s %s ' %(self.FirstName,self.LastName)
16
17    def payment(self):
18        return 'Last Payment : %s' %(self.payment)
19
20    def payment_increase(self):
21        self.payment += float(self.payment * self.pay_rising)
22        return 'New Payment : %s' %(self.payment)
```

میدونم خیلی واضح نیست ! اما به خط اول توجه کنید.

یک کامنت بسیار طولانی !

برای اینکه کامنت های چند خطی بزارید داخل پایتون از روش زیر استفاده کنید :

```
python projects > Python_Challenge.py
1 # One line comment in python
2
3
4 Multiple lines for
5 commenting
6 in python
7
```

یک نکته ی خیلی باحال از پایتون :

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

2 - قدرت خطا خوانی :

این تیتر به شدت مهمه !

اگر برنامه نویسی انجام میدی و دوست داری واقعا توی کارت خوب باشی، به نظر من باید بتونی خطاهایی که سیستم بهت میده رو خوب بخونی و بفهمی چرا همچین خطایی رو بهت داده .

اگر که با سیستم خطایابی هر زبانی که کار میکنید دوست بشید، دیگه از خطا کرد و باگ های مختلف نمیترسید.

مثلا توی مثال پایین، دقیقا بهت میگه خطا کجا بوده، چو کم گذاشتی و باید چیکار کنی :

```
python projects > Python_Challenge.py > ...
1 list=['mohammad',1,2,3,15j]
2 for items in list
3     print(items)
4
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users
.py"
File "c:\Users\User\Desktop\Python\python
for items in list
^
SyntaxError: expected ':'
PS C:\Users\User\Desktop\Python> |
```

در تایپ متغیر ها ممکنه بعضی وقت ها غلط تایپی داشته باشید، برای من که خیلیییی پیش اومده، پس بهتره این خطا رو ببینید و چقدر قشنگ داره راهنمایمون میکنه :

```
python projects > Python_Challenge.py > ...
1 age_ali=20
2 age_mohammad=35
3 if ag_ali > age_mohammad:
4     print('ali is older .')
5 else:
6     print('Mohammad is older.')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe .py"
Traceback (most recent call last):
  File "c:\Users\User\Desktop\Python\python projects\Python_Challenge.py", line 3, in <module>
    if ag_ali > age_mohammad:
NameError: name 'ag_ali' is not defined. Did you mean: 'age_ali'?
PS C:\Users\User\Desktop\Python> █
```

ترکیب قوانین ایندکس ها، لیست ها، و رشته ها :

```
python projects > Python_Challenge.py > ...
1 example_text = 'This is an example text for list slicing'
2 print(example_text)
3 print('-----')
4 print(example_text[:])
5 print('-----')
6 print(example_text[0:12])
7 print('-----')
8 print(example_text[2:20:2])
9 print('-----')
10 print(example_text[20:2:-2])
11 print('-----')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe .py"
This is an example text for list slicing
-----
This is an example text for list slicing
-----
This is an e
-----
i sa xml
-----
e lmx as
-----
PS C:\Users\User\Desktop\Python> █
```

اگر با رشته ها کار میکنید، یکی از مهم ترین و پر کاربرد ترین متد ها برای شما، اسپلیت خواهد بود، بیشتر نمیگم که خودتون برید دنبالش !

3 – ساختار try, except, finally

توی پایتون میتونید خطاهاتون رو کنترل کنید و طبق میل خودتون، پیام خطا چاپ بشه ! یا اصلا میتونید کل حلقه های for یا کل موارد شرطیتون رو کنترل کنید.

اما نیاز دارید که انواع خطاهای پایتون رو بشناسید ، که بهترین منبع شناخت این نوع خطا ها، داکيومنت های استاندارد پایتون هست :

8. Errors and Exceptions

Until now error messages haven't been more than mentioned, but if you have tried out the examples you have probably seen some. There are (at least) two distinguishable kinds of errors: *syntax errors* and *exceptions*.

8.1. Syntax Errors

Syntax errors, also known as parsing errors, are perhaps the most common kind of complaint you get while you are still learning Python:

```
>>> while True print('Hello world')
File "<stdin>", line 1
  while True print('Hello world')
                ^
SyntaxError: invalid syntax
```

The parser repeats the offending line and displays a little 'arrow' pointing at the earliest point in the line where the error was detected. The error is caused by (or at least detected at) the token *preceding* the arrow: in the example, the error is detected at the function `print()`, since a colon (':') is missing before it. File name and line number are printed so you know where to look in case the input came from a script.

این هم لینک این داکيومنت :

<https://docs.python.org/3/tutorial/errors.html>

یک مثال کوچک در ارتباط با کنترل خطا در قسمت تقسیم اعداد بر صفر :

1 - خود پایتون

```
python projects > Python_Challenge.py
1 print(2/0)
2
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/powershell>

PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe /Python_Challenge.py

Traceback (most recent call last):
File "c:\Users\User\Desktop\Python\python projects\Python_Challenge.py", line 1, in <module>
print(2/0)
ZeroDivisionError: division by zero ←

PS C:\Users\User\Desktop\Python>

2 - بریم از این قابلیت استفاده کنیم

```
python projects > Python_Challenge.py > ...
1 number1=int(input('Adade aval:'))
2 number2=int(input('Adade dovom:'))
3 try:
4     print(number1/number2)
5 except ZeroDivisionError:
6     print("Adad ra nemishe bar 0 taghsim kard")
7 finally:
8     print("Good LUCK !" .swapcase())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe /Python_Challenge.py

Adade aval:2
Adade dovom:0
Adad ra nemishe bar 0 taghsim kard
gOOD Luck !

PS C:\Users\User\Desktop\Python>

به متد جدیدی که توی قسمت finally استفاده کردم دقت کنید.

راستی، یادت باشه قسمت finally در هر صورت اجرا میشه 😊

مثال پایین رو ببین، به نظرت اگه عدد اول 5 و عدد دوم رو 3 وارد کنیم، جواب نهایی چه پیام یا پیام هایی خواهد بود؟

```
python projects > Python_Challenge.py > ...
1  number1=int(input('Adade aval:'))
2  number2=int(input('Adade dovom:'))
3
4  if number1/number2 > 0:
5      try:
6          print("Something to print")
7      except ZeroDivisionError:
8          print("Oooopa! zero division error")
9      else:
10         print("Something to print in this part")
11     finally:
12         print('Thank you for your attention')
13
```

4 - مپ

از قابلیت های جذاب پایتون برای بحث iterable و iterator ها که قبلا بیان کردم، همین map هست.

بریم سراغ سایت جذاب [geeks for geeks](https://www.geeksforgeeks.org/) ببینیم چی میگه :

function to each item of a given iterable (list, tuple etc.)

Syntax :

```
map(fun, iter)
```

Parameters :

fun : It is a function to which map passes each element of given iterable.

iter : It is a iterable which is to be mapped.

NOTE : You can pass one or more iterable to the map() function.

Returns :

Returns a list of the results after applying the given function to each item of a given iterable (list, tuple etc.)

پس فهمیدیم مپ دو تا آرگومان رو میگیره، اولیس تابع مد نظر هست که نیاز داریم توی اون تابع گردش کنیم و دومین آرگومان هم چیه؟؟
خودت بگو (:)

راستی! کار map چقدر شبیه کار حلقه ی for هست !

```
python projects > Python_Challenge.py > ...
1 list_numbers=[1,14,25,30,236]
2 print(list_numbers)
3 print('-----')
4 # Using for loop
5 for numbers in list_numbers:
6     print(numbers)
7 print('-----')
8 # Using map
9 print(list(map(lambda x: x , list_numbers)))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
/Python_Challenge.py"
[1, 14, 25, 30, 236]
-----
1
14
25
30
236
-----
[1, 14, 25, 30, 236]
PS C:\Users\User\Desktop\Python> █
```


راستی اون lambda چیه؟؟ (فردا بهش میرسیم و فردا هم مپ را کامل میکنم.)

یه اشاره ی کوچک کردم به مپ فقط! برمیگردیم سراغش 😊

math – 5

دوباره هم فلش بک میزنم به دو سه تا فایل قبل! جایی که کتابخونه ها و ماژول ها رو معرفی کردم.

از (مهم ترین * 100) کتابخانه های پایتون، کتابخونه ی ریاضیش هست. اما صرفا هدفم معرفی نیست الان.

بریم ببینیم کلا با کتابخونه های چطوری باید کار کنیم و مبحث قبلی رو هم تکمیل کنیم.

اول میریم سراغ داکيومنت جذابش :

math — Mathematical functions

This module provides access to the mathematical functions defined by the C standard.

These functions cannot be used with complex numbers; use the functions of the same name from the `cmath` module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don't is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place.

The following functions are provided by this module. Except when explicitly noted otherwise, all return values are floats.

Number-theoretic and representation functions

`math.ceil(x)`

Return the ceiling of x , the smallest integer greater than or equal to x . If x is not a float, delegates to `x. ceil__()`, which should return an `Integral` value.

`math.comb(n, k)`

Return the number of ways to choose k items from n items without repetition and without order.

Evaluates to $n! / (k! * (n - k)!)$ when $k \leq n$ and evaluates to zero when $k > n$.

Also called the binomial coefficient because it is equivalent to the coefficient of k -th term in polynomial expansion of the expression $(1 + x) ** n$.

همونطور که از قبل هم گفتیم، توی هر قسمت داکيومنت مثال هایی هست که خیلی میتونه کمک کنه بهمون.

Power and logarithmic functions

`math.exp(x)`

Return e raised to the power x , where $e = 2.718281\dots$ is the base of natural logarithms. This is usually more accurate than `math.e ** x` or `pow(math.e, x)`.

`math.expm1(x)`

Return e raised to the power x , minus 1. Here e is the base of natural logarithms. For small floats x , the subtraction in `exp(x) - 1` can result in a **significant loss of precision**; the `expm1()` function provides a way to compute this quantity to full precision:

```
>>> from math import exp, expm1
>>> exp(1e-5) - 1 # gives result accurate to 11 places
1.0000050000069649e-05
>>> expm1(1e-5) # result accurate to full precision
1.0000050000166668e-05
```

New in version 3.2.

`math.log(x[, base])`

With one argument, return the natural logarithm of x (to base e).

With two arguments, return the logarithm of x to the given *base*, calculated as `log(x)/log(base)`.

خودمون هم بریم یکم تمرین کنیم :

```
python projects > Python_Challenge.py > ...
1 import math
2
3 list_numbers=[12,14.96,236.5,256]
4 print(math.ceil(235.256))
5 print(math.fsum(list_numbers))
6 print(math.log10(152))
_

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Python/Python38-64/Python_Challenge.py"
236
519.46
2.1818435879447726
PS C:\Users\User\Desktop\Python> □
```

خیلییی میتونید از کتابخونه ی ریاضی پایتون استفاده کنید اما هدفم بیشتر این بود که بفهمیم چطور باید از کتابخونه ها استفاده کرد، راستی میتونی اسم کتابخونه ها رو تغییر بدی و با هر اسمی که دوست داری کد بزنی، اصلا بیاید اسم کتابخونه ی ریاضی رو بزاریم اکبر :

```
python projects > Python_Challenge.py > ...
1 import math as akbar
2
3 list_numbers=[12,14.96,236.5,256]
4 print(akbar.ceil(235.256))
5 print(akbar.fsum(list_numbers))
6 print(akbar.log10(152))
7
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/Python_Challenge.py
236
519.46
2.1818435879447726
PS C:\Users\User\Desktop\Python> |
```

میتونید یک قسمت خاص از کتابخونه را هم وارد برنامه کنید، که باید خودتون برید بخونید درباره اش.

6 – یک قسمت بسیار مهم از داکيومنت استاندارد پایتون

اگر قسمت های قبلی را خوانده باشید، میدونید که من خیلی تاکید دارم روی داکيومنت های استاندارد پایتون و امشب هم میخوام یک قسمت جالب و جدید این داکيومنت ها رو بهتون معرفی کنم :

Previous topic

Changelog

Next topic

1. Whetting Your Appetite

This Page

Report a Bug
Show Source

The Python Tutorial

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

« The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

For a description of standard objects and modules, see [The Python Standard Library](#). [The Python Language Reference](#) gives a more formal definition of the language. To write extensions in C or C++, read [Extending and Embedding the Python Interpreter](#) and [Python/C API Reference Manual](#). There are also several books covering Python in depth.

This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in [The Python Standard Library](#).

توی این قسمت میتونید پایتون رو یاد بگیرید و به ترتیب و خیلی منظم موضوعات بیان شده اند. من هم توی توضیحات تکمیلی ام, سعی میکنم از نکات خیلی باحال این داکيومنت استفاده کنم.

همونطور که میبینید، موضوعات بسیار دسته بندی شده و طبق به روز ترین نسخه ی پایتون هستند :

The Glossary is also worth going through.

- 1. Whetting Your Appetite
- 2. Using the Python Interpreter
 - 2.1. Invoking the Interpreter
 - 2.1.1. Argument Passing
 - 2.1.2. Interactive Mode
 - 2.2. The Interpreter and Its Environment
 - 2.2.1. Source Code Encoding
- 3. An Informal Introduction to Python
 - 3.1. Using Python as a Calculator
 - 3.1.1. Numbers
 - 3.1.2. Strings
 - 3.1.3. Lists
 - 3.2. First Steps Towards Programming
- 4. More Control Flow Tools
 - 4.1. if Statements
 - 4.2. for Statements
 - 4.3. The range() Function
 - 4.4. break and continue Statements, and else Clauses on Loops
 - 4.5. pass Statements
 - 4.6. match Statements
 - 4.7. Defining Functions
 - 4.8. More on Defining Functions
 - 4.8.1. Default Argument Values
 - 4.8.2. Keyword Arguments
 - 4.8.3. Special parameters
 - 4.8.3.1. Positional-or-Keyword Arguments
 - 4.8.3.2. Positional-Only Parameters
 - 4.8.3.3. Keyword-Only Arguments
 - 4.8.3.4. Function Examples

این هم لینک پیج مربوطه :

<https://docs.python.org/3/tutorial/index.html>

قسمت 8 تمام 😊

سعی کردم توی این قسمت کم کاری دو روز قبل را جبران کنم.

من علاقه دارم خیلی از مطالب را تک اشاره ای بهش بکنم و کم کم در فایل های بعدی کاملش کنم که مثل پازل باشه و خواننده اونها رو به هم ربط بده و به نظرم روش خیلی جذابه، عین فیلم های نتفلیکس!

خیلی ممنون که دنبال میکنید – ارادت ❤️

کاری از : علی معینیان

21 day challenge with python ❤️



Part 9 out of 21

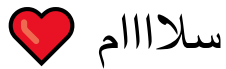
1 – more tips on last episodes

2 – lambda

3 – map

4 – filter

5 – in , not in



1 – ادامه ی تکمیل مباحث قبل از control flow تا Data structure

بریم سراغ مثال های بیشتر در ارتباط با ترکیب چیز هایی که تا حالا یاد گرفتیم و مطلبی که امروز میریم سراغش :

```
python projects > Python_Challenge.py > ...
1 # Find that if your favorite fruits is between our fruits or not ?
2 fruits=['banana','strawberry','apple','coconat','orange']
3
4 fav_fruit = input('enter your favorite fruit : ')
5 if fav_fruit in fruits:
6     print('we have ' + fav_fruit + 'and its on ' + str(fruits.index(fav_fruit)) + ' index')
7 else:
8     print('Sorry, we dont have ' + fav_fruit)
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe "c:/U
/Python_Challenge.py"
enter your favorite fruit : apple
we have appleand its on 2 index
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe "c:/U
/Python_Challenge.py"
enter your favorite fruit : kdgiuc
Sorry, we dont have kdgiuc
PS C:\Users\User\Desktop\Python> █
```

بریم اسامی نفرات برتر یک آزمونی رو با رتبهشون ببینیم چطور باید چاپ کرد :

```
python projects > Python_Challenge.py > [e] persons
1 students = ['ali' , 'maryam' , 'mohammad' , 'zahra' , 'elham']
2 print('Final resylt will be :')
3 for persons in range(len(students)):
4     print(persons , students[persons])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Pyt
/Python_Challenge.py"
Final resylt will be :
0 ali
1 maryam
2 mohammad
3 zahra
4 elham
PS C:\Users\User\Desktop\Python> █
```

مثال های بیشتر در ارتباط با سوئیچ کیس جذاب و تازه نفس در پایتون :

```
# point is an (x, y) tuple
match point:
    case (0, 0):
        print("Origin")
    case (0, y):
        print(f"Y={y}")
    case (x, 0):
        print(f"X={x}")
    case (x, y):
        print(f"X={x}, Y={y}")
    case _:
        raise ValueError("Not a point")
```

در ارتباط با مبحث توابع و آرگومان های ورودی توابع باید خیلی مطالعه کنید ولی فقط یک مثال کوچک ازش ببینیم از داکيومنت استاندارد پایتون :

4.8.1. Default Argument Values

The most useful form is to specify a default value for one or more arguments. This creates a function that can be called with fewer arguments than it is defined to allow. For example:

```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True
        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise ValueError('invalid user response')
        print(reminder)
```

This function can be called in several ways:

- giving only the mandatory argument: `ask_ok('Do you really want to quit?')`
- giving one of the optional arguments: `ask_ok('OK to overwrite the file?', 2)`
- or even giving all arguments: `ask_ok('OK to overwrite the file?', 2, 'Come on, only yes or no!')`

This example also introduces the `in` keyword. This tests whether or not a sequence contains a certain value.

The default values are evaluated at the point of function definition in the *defining* scope, so that

تعریف خارجی لامبدا :

Python Lambda

[← Previous](#)

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

لامبدا یه دونه بچه تابع است!

یعنی فعالیت هایی شبیه توابع انجام میده.

اصوولا از لامبدا به عنوان تابع ورودی برای جاهای مختلف استفاده میکنیم.

نحوه ی تعریف و استفاده از لامبدا :

lambda (variable) : (what do you want to be done?)

بریم سراغ چند تا مثال مقایسه ای :

```
python projects > Python_Challenge.py > ...
1 #Functional programming
2 def sum_func(number1,number2):
3     return number1 + number2
4 number1=int(input('adade aval : '))
5 number2=int(input('adade dovom : '))
6 print('Answer is :'+ str(sum_func(number1,number2)))
7
8 print('----- ')
9
10 # usingg lambda
11 sum_func_lambda=lambda x,y : x + y
12 print('Answer is :'+ str(sum_func_lambda(number1,number2)))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python_Challenge.py"
adade aval : 2
adade dovom : 2
Answer is :4
-----
Answer is :4
```

Example

Multiply argument **a** with argument **b** and return the result:

```
x = lambda a, b : a * b
print(x(5, 6))
```

Try it Yourself »

Example

Summarize argument **a**, **b**, and **c** and return the result:

```
x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```

Try it Yourself »

3 – مپ (map)

مپ رو دیروز براتون تعریف کردم، و اونجا اشاره ی کوچکی به لامبدا داشتیم.

امشب فقط مثال های مپ رو خواهیم دید :

```
python projects > Python_Challenge.py > ...
1 list_numbers=[1,2,5,6,9,6]
2 print(list(map(lambda x: x*x , list_numbers)))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python38-64/Python_Challenge.py"
[2, 4, 10, 12, 18, 12]
PS C:\Users\User\Desktop\Python> █
```

Syntax:

```
map(function, iterator1,iterator2 ...iteratorN)
```

Parameters

Here are two important

- **function:** A mandatory function to be given to map, that will be applied to all the items available in the iterator.
- **iterator:** An iterable compulsory object. It can be a list, a tuple, etc. You can pass multiple iterator objects to map() function.

Return Value

The map() function is going to apply the given function on all the items inside the iterator and return an iterable map object i.e a tuple, a list, etc.

How map() function works?

The map() function takes two inputs as a function and an iterable object. The function that is given to map() is a normal function, and it will iterate over all the values present in the iterable object given.

```
def square(n):  
    return n*n  
my_list = [2,3,4,5,6,7,8,9]  
updated_list = map(square, my_list)  
print(updated_list)  
print(list(updated_list))
```

Output:

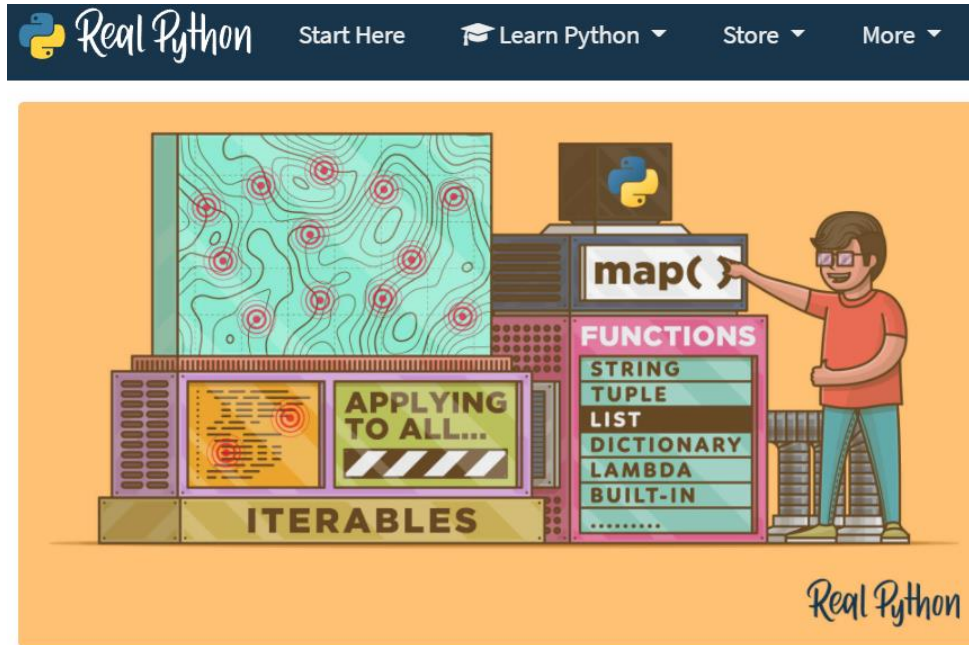
```
<map object at 0x0000002C59601748>  
[4, 9, 16, 25, 36, 49, 64, 81]
```

The output of the map() function, as seen in the output, is a map object displayed on the screen as <map object at 0x0000002C59601748>.

You will have to iterate the output from the map using a for-loop or using list() method to get the final output. I have used list() in the code that displays the values inside the list given.

برای مثال های بیشتر لینک زیر را مطالعه کنید :

[/https://realpython.com/python-map-function](https://realpython.com/python-map-function)



Python's map(): Processing Iterables Without a Loop

by Leodanis Pozo Ramos · Sep 30, 2020 · 19 Comments · basics best-practices python

Mark as Completed

Tweet Share Email

4 – فیلتر

Definition and Usage

The `filter()` function returns an iterator where the items are filtered through a function to test if the item is accepted or not.

Syntax

```
filter(function, iterable)
```

Parameter Values

Parameter	Description
<code>function</code>	A Function to be run for each item in the Iterable
<code>iterable</code>	The iterable to be filtered

Example

Filter the array, and return a new array with only the values equal to or above 18:

```
ages = [5, 12, 17, 18, 24, 32]

def myFunc(x):
    if x < 18:
        return False
    else:
        return True

adults = filter(myFunc, ages)

for x in adults:
    print(x)
```

[Try it Yourself »](#)

The `filter()` function extracts elements from an iterable (list, tuple etc.) for which a function returns `True`.

Example

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# returns True if number is even
def check_even(number):
    if number % 2 == 0:
        return True

    return False

# Extract elements from the numbers list for which check_even() returns True
even_numbers_iterator = filter(check_even, numbers)

# converting to list
even_numbers = list(even_numbers_iterator)

print(even_numbers)

# Output: [2, 4, 6, 8, 10]
```

Example 1: Working of filter()

```
letters = ['a', 'b', 'd', 'e', 'i', 'j', 'o']

# a function that returns True if letter is vowel
def filter_vowels(letter):
    vowels = ['a', 'e', 'i', 'o', 'u']
    return True if letter in vowels else False

filtered_vowels = filter(filter_vowels, letters)

# converting to tuple
vowels = tuple(filtered_vowels)
print(vowels)
```

Output

```
('a', 'e', 'i', 'o', 'u')
```

Example 3: Using None as a Function Inside filter()

```
# random list
random_list = [1, 'a', 0, False, True, '0']

filtered_iterator = filter(None, random_list)

#converting to list
filtered_list = list(filtered_iterator)

print(filtered_list)
```

Output

```
[1, 'a', True, '0']
```

When `None` is used as the first argument to the `filter()` function, all elements that are truthy values (gives `True` if converted to boolean) are extracted.

Application:

It is normally used with [Lambda functions](#) to separate list, tuple, or sets.

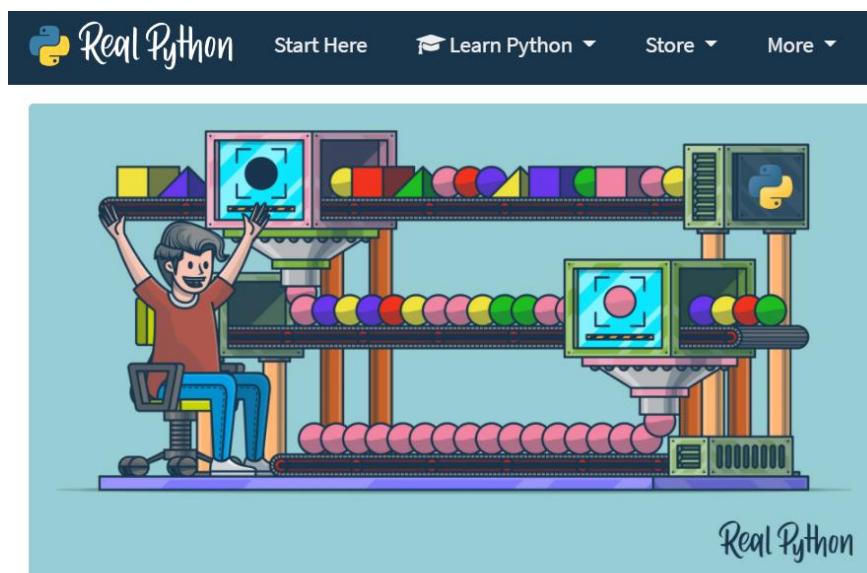
```
# a list contains both even and odd numbers.  
seq = [0, 1, 2, 3, 5, 8, 13]  
  
# result contains odd numbers of the list  
result = filter(lambda x: x % 2 != 0, seq)  
print(list(result))  
  
# result contains even numbers of the list  
result = filter(lambda x: x % 2 == 0, seq)  
print(list(result))
```

Output:

```
[1, 3, 5, 13]  
[0, 2, 8]
```

لینک برای مطالعه ی بیشتر در ارتباط با فیلتر ها :

[/https://realpython.com/python-filter-function](https://realpython.com/python-filter-function)



Python's filter(): Extract Values From Iterables

by Leodanis Pozo Ramos · Jun 09, 2021 · 6 Comments · best-practices · intermediate

python

سینتکس پایتون رو یادته ؟ هر طور حرف میزنی بنویس.

در مثال های قبلی خیلی از این دو تا **key word** استفاده کردم ولی لان هم بیشتر بهشون میپردازم :

Basically, the `in` operator in Python checks whether a specified value is a constituent element of a sequence like `string`, `array`, `list`, or `tuple` etc.

When used in a condition, the statement returns a Boolean result evaluating into either `True` or `False`. When the specified value is **found** inside the sequence, the statement returns `True`. Whereas when it is **not found**, we get a `False`.

Not let us take an example to get a better understanding of the `in` operator working.

```
#in operator working

list1= [1,2,3,4,5]
string1= "My name is AskPython"
tuple1=(11,22,33,44)

print(5 in list1) #True
print("is" in string1) #True
print(88 in tuple1) #False
```


The `not in` operator in Python works exactly the opposite way as the `in` operator works. It also checks the presence of a specified value inside a given sequence but its return values are totally opposite to that of the `in` operator.

When used in a condition with the specified value present inside the sequence, the statement returns `False`. Whereas when it is not, we get a `True`.

Let us take the previous example, just replacing `in` operator with the `not in` one.

```
#not in operator working

list1= [1,2,3,4,5]
string1= "My name is AskPython"
tuple1=(11,22,33,44)

print(5 not in list1) #False
print("is" not in string1) #False
print(88 not in tuple1) #True
```

```
#in and not in operator working on Dictionary

dict1 = {1: "one", 2: "two", 3: "three", 4: "four"}

print("one" in dict1)
print("one" not in dict1)

print(3 in dict1)
print(3 not in dict1)

print(5 in dict1)
print(5 not in dict1)
```

python projects > Python_Challenge.py > overlapping

```
1 def overlapping(list1,list2):
2
3     c=0
4     d=0
5     for i in list1:
6         c+=1
7     for i in list2:
8         d+=1
9     for i in range(0,c):
10        for j in range(0,d):
11            if(list1[i]==list2[j]):
12                return 1
13        return 0
14 list1=[1,2,3,4,5]
15 list2=[6,7,8,9]
16 if(overlapping(list1,list2)):
17     print("overlapping")
18 else:
19     print("not overlapping")
```

```
>>> (2, 3) not in [(2, 3), (5, 6), (9, 1)]
```

```
False
```

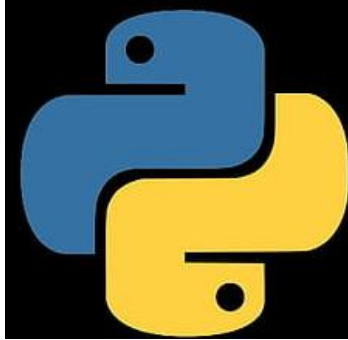
```
>>> (2, 3) not in [(2, 7), (7, 3), "hi"]
```

```
True
```

این پازل، کامل خواهد شد 

کاری از : علی معینیان

21 day challenge with python ❤️



Part 10 out of 21

Part 10 and 11 will be related to the previous topics and the completion of the previous puzzle

سلامم ✨

توی این فایل و فایلی که فردا داریم، سعی دارم مباحث قبلی رو کامل کنم و پازلی که ساخته بودیم رو کامل کنم.

تا توی پارت 12 بریم سراغ فایل ها توی پایتون و کم کم وارد شی گرای بشیم.

1 – برگردیم سراغ cmd و بیشتر بدونیم درباره اش :

برای چک کردن ورژن پایتونی که استفاده میکنید، از دستور زیر میتونید استفاده کنید :

```
C:\> Command Prompt
Microsoft Windows [Version 10.0.19041.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>python --version
Python 3.10.0

C:\Users\User>_
```

توی دوره های مختلفی که گذروندم، همیشه یکی از مواردی که روش تاکید میشه، اجرای یک فایل پایتون از طریق cmd هست.

خب اخیه وقتی شما توی ide کد میزنید، همونجا میتونید اجرا بگیرید از کد و نتیجه رو ببینید ! نمیفهمم چرا باید لقمه رو دور سر خودمون بچرخونیم.

در نام گذاری متغیر ها، علاوه بر اینکه نمیتونید از کلمات از پیش رزرو شده استفاده کنید، یک سری قواعد دیگر رو هم باید رعایت کنید :

Example

Illegal variable names:

```
2myvar = "John"
```

```
my-var = "John"
```

```
my var = "John"
```

- 1 – شروع متغیر با عدد ممنوع است.
- 2 – استفاده از (-) در نام متغیر ممنوع است.
- 3 – استفاده از space بین حروف نام متغیر ممنوع است.

کمی بیشتر درباره ی data type های پایتون :

Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type:	<code>str</code>
Numeric Types:	<code>int</code> , <code>float</code> , <code>complex</code>
Sequence Types:	<code>list</code> , <code>tuple</code> , <code>range</code>
Mapping Type:	<code>dict</code>
Set Types:	<code>set</code> , <code>frozenset</code>
Boolean Type:	<code>bool</code>
Binary Types:	<code>bytes</code> , <code>bytearray</code> , <code>memoryview</code>

مثال های کوچولو :

x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

گاهی پیام هایی که میخواهید چاپ بکنید بیش از یک خط هستند و بسیار طولانی؛ چه کنیم؟ مثال پایینی رو ببین :

```
python projects > Python_Challenge.py > ...
1  alert='''This is example of a long
2  message to show in my program.
3  its going to be an alert message'''
4  💡
5  print(alert)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\User\Desktop\Python> & C:/Users/User/Desktop/Python/Python_Challenge.py
This is example of a long
message to show in my program.
its going to be an alert message
PS C:\Users\User\Desktop\Python> |
```

پیدا کردن یک کلمه ی خاص در متن :

```
python projects > Python_Challenge.py > ...
1  alert=''This is example of a long
2  message to show in my program.
3  its going to be an alert message''
4
5  word=input('what word do you like to find ?')
6
7  if word in alert:
8      print(word + ' is in text.')
9  else:
10     print(word + ' not found')
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData
/Python_Challenge.py"
what word do you like to find ?message
message is in text.
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData
/Python_Challenge.py"
what word do you like to find ?apple
apple not found
PS C:\Users\User\Desktop\Python> |
```

مهم ترین متد ها در ارتباط با رشته ها :

Example

The `replace()` method replaces a string with another string:

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

[Try it Yourself »](#)

Split String

The `split()` method returns a list where the text between the specified separator becomes the list items.

Example

The `split()` method splits the string into substrings if it finds instances of the separator:

```
a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']
```


خیلی اوقات ما از یکسری کاراکتر ها باحال توی دستورات استفاده میکنیم (بیشتر دستور print) که خیلییی هم به درد بخور هستن :

Escape Characters

Other escape characters used in Python:

Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

جمع بندی انواع عملگر های پایتون :

Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
*	Multiplication	$x * y$
/	Division	x / y
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$

Python Assignment Operators

Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	$x = 5$	$x = 5$
+=	$x += 3$	$x = x + 3$
-=	$x -= 3$	$x = x - 3$
*=	$x *= 3$	$x = x * 3$
/=	$x /= 3$	$x = x / 3$
%=	$x \% = 3$	$x = x \% 3$
//=	$x //= 3$	$x = x // 3$
**=	$x ** = 3$	$x = x ** 3$
&=	$x \& = 3$	$x = x \& 3$
=	$x = 3$	$x = x 3$
^=	$x \wedge = 3$	$x = x \wedge 3$
>>=	$x >> = 3$	$x = x >> 3$
<<=	$x << = 3$	$x = x << 3$

Python Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Python Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Python Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

اگر دوست دارید داکيومنتی که مطالعه میکنید، بیشتر مثال داشته باشد تا توضیحات حتما از سایت زیر استفاده کنید :

Python Introduction

- Python Print
- Python Basics
- Python Variables
- Python Data Types
- Python Input
- Python Operators
- Python If and Else
- Python While Loop
- Python For Loop
- Python Break and Continue
- Python Lists

Python Tutorial - Introduction

Python Video Lectures

Python is a widely used high-level dynamic programming language. It is a very simple, friendly and easy to learn programming language. It is the best choice for a beginner programmer. Python source code is also available under GNU General Public License (GPL).

Who use Python?

As mentioned above, it is a widely used programming language. Some of the places where Python is used are :

- Google - Python is one of the key language used in google.
- Philips - Philips uses Python for the sequencing language (language that tells what steps each robot should take).

CODESDOPE PRO

It's **Simple and Conceptual** with **Chance for Internship***

Pro Course Features

- Simple Videos
- Questions to Practice
- Solved Examples
- Internship Chance*
- Certificate of Completion
- Discussion with Experts

Learn for **FREE**

مثال های خیلی خوبی داره به همراه یک سری انیمیشن های کوچیک برای فهم بهتر مبحث :

```
i = 1
while i<=2:
    print(i*14)
    i=i+1
```

← False

Value of i = 3 LOOP STOPS

Output

```
i = 1
while i<=10:
    print(i*14)
    i=i+1
```

Output

توی مبحث حلقه ها، یک قسمتی هست به نام حلقه های بی نهایت. توی اینطور حلقه ها ما هیچ دستوری برای توقف برنامه نداریم. و همینطور برنامه تکرار میشه!

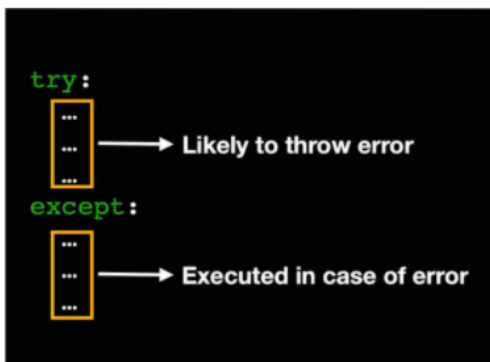
```
cubes = {num = num**3 for num in range(1, 11)}
```

```
for num in range(1, 11):
```

```
    cubes[num] = (num**3)
```

برای مبحث کنترل خطاها که دیروز مطرح کردم ببینیم چیا داره این سایت :

`try` and `except` clauses are used to handle exceptions. The piece of code which is likely to throw an exception is placed inside the `try` clause. The statements which we want to execute if the code inside the `try` clause throws an exception are placed inside the `except` clause.



Let's see how to use `try` and `except` clauses for handling the exception we were getting in the first example of this chapter.

```
num1 = input("Please enter a number")  
try:  
    num2 = int(num1)  
    print(num2)  
except:  
    print("Please enter the correct input")  
print("Outside of try-except clauses. This statement is always executed.")
```

اما بریم سراغ لیست ها و مثال هایی از استفاده ی متد ها :

```
python projects > Python_Challenge.py > ...
```

```
1  fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
2  print(fruits.count('apple'))
3  print(fruits.count('tangerine'))
4  print(fruits.index('banana'))
5  print(fruits.index('banana', 4))
6  print(fruits.reverse())
7  print(fruits.append('grape'))
8  print(fruits.sort())
9  print(fruits.pop())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python3
/Python_Challenge.py"
```

```
2
0
3
6
None
None
None
pear
PS C:\Users\User\Desktop\Python> █
```

خب این مثال رو هم دیدیم ولی اون none ها چی هستن توی جواب !!؟

این ارور یا جواب رو من خیلیییییییی تجربش کردم.

بریم درستش کنیم (خودتون دوتا کد رو مقایسه کنید) و من واقعا خیلییی

وقت ها با این none روبرو شدم و برام عجیب بوده :

```
python projects > Python_Challenge.py > ...
```

```
1 fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
2 print(fruits.count('apple'))
3 print(fruits.count('tangerine'))
4 print(fruits.index('banana'))
5 print(fruits.index('banana', 4))
6 (fruits.reverse())
7 print(fruits)
8 (fruits.append('grape'))
9 print(fruits)
10 (fruits.sort())
11 print(fruits)
12 print(fruits.pop())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/Python_Challenge.py"
```

```
2
0
3
6
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']
['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear']
pear
PS C:\Users\User\Desktop\Python> |
```

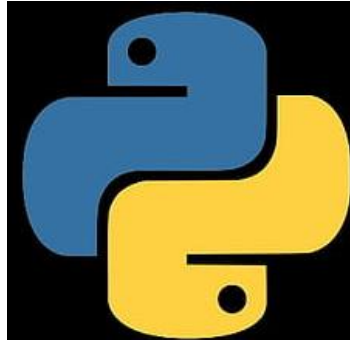
حالا درست شد 😊

خب از اونجایی که نمیخوام طولانی تر بشه، ادامه ی دوره ها میشه برای فردا

موفق باشید - ارادات ❤️🎈

کاری از : علی معینیان

21 day challenge with python + corona ❤️



Part 11 out of 21

Part 10 and 11 will be related to the previous topics and the completion of the previous puzzle

سلام!!! 😊

من بعد 6 روز برگشتم!

دوست نداشتم تاخیری داخل چالشم بیوفته ولی متاسفانه کرونا گرفتم و عملا

مردم و زنده شدم و از امروز با قدرت پیش میریم 😊

این پارت اخرین پارت در ارتباط با تکمیل مطالب قبلی خواهد بود.

خب همیشه یکی از مسائلی که درباره ی پایتون مهم بوده، فرق ساختمان های داده ای بوده که با هم کار کردیم. جدول زیر خیلی بهمون کمک میکنه تا بتونیم فرق لیست و تاپل و دیکشنری و ست رو درک کنیم و باعث میشه بتونیم تصمیم بگیریم از کدام یک در کجا استفاده کنیم :

Parameters	List	Tuple	Set	Dictionary
Basics	A list is basically like a dynamically sized array that gets declared in other languages (Arraylist in the case of Java, vector in the case of C++).	The tuples refer to the collections of various objects of Python separated by commas between them.	The sets are an unordered collection of data types. These are mutable, iterable, and do not consist of any duplicate elements.	In Python, the dictionary refers to a collection (unordered) of various data types. We use these for storing data values such as maps, and unlike other data types capable of holding only one value in the form of an element, a dictionary can hold the key: value pair.
Homogeneity	A list refers to a data structure of a non-homogenous type that functions to store various elements in columns, multiple rows, and single rows.	A tuple also refers to a data structure of the non-homogenous type that functions to store various elements in columns, multiple rows, and single rows.	A set also refers to a data structure of the non-homogenous type, but it stores various elements in a single row.	A dictionary also refers to a data structure of the non-homogenous type that functions to store key-value pairs.
Representation	We can represent a List by []	We can represent a Tuple by ()	We can represent a Set by {}	We can represent a Dictionary by {}
Duplicate elements	It allows various duplicate elements.	It allows various duplicate elements.	It does not allow any duplicate elements.	The keys are not at all duplicated.

Representation	We can represent a List by []	We can represent a Tuple by ()	We can represent a Set by {}	We can represent a Dictionary by {}
Duplicate elements	It allows various duplicate elements.	It allows various duplicate elements.	It does not allow any duplicate elements.	The keys are not at all duplicated.
Nested Among All	It can be utilized in a List.	It can be utilized in a Tuple.	It can be utilized in a Set.	It can be utilized in a Dictionary.
Example	[6, 7, 8, 9, 10]	(6, 7, 8, 9, 10)	{6, 7, 8, 9, 10}	{6, 7, 8, 9, 10}
Function for Creation	We can create a list using the list() function.	We can create a tuple using the tuple() function.	We can create a set using the set() function.	We can create a dictionary using the dict() function.
Mutation	It is mutable. It means that a user can make any changes to a list.	It is immutable. It means that a user can't make any changes to a tuple.	It is mutable. It means that a user can make any changes to a set.	It is mutable, but the keys are not at all duplicated.
Order	It is ordered in nature.	It is ordered in nature.	It is unordered in nature.	It is ordered in nature.
Empty Elements	If we want to create an empty list, we use: l=[]	If we want to create an empty tuple, we use: t=()	If we want to create an empty set, we use: a=set() b=set(a)	If we want to create an empty dictionary, we use: d={}

و اما يك تغيير مهم :

As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

داخل پایتون همانند همه ی زبان های دیگه میتونید از if یک خطی استفاده کنید، بریم مثال ببینیم ازش :

```
python projects > Python_Challenge.py > ...
1 #Simple if
2 age_ali=25
3 if age_ali > 18:
4     print('You have reached the legal age')
5 else:
6     print('You have not reached the legal age')
7 print('-----')
8 #One line if
9 result = 'You have reached the legal age' if age_ali>18 else 'You have not reached the legal age'
10 print(result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/User/Desktop/Python_Challenge.py"
You have reached the legal age
-----
You have reached the legal age
PS C:\Users\User\Desktop\Python> □
```

تعریف پارامتر و ارگومان رو یادتونه ؟ اینجا تعریف انگلیسیشو ببینیم :

Parameters or Arguments?

The terms *parameter* and *argument* can be used for the same thing: information that are passed into a function.

From a function's perspective:

A parameter is the variable listed inside the parentheses in the function definition.

An argument is the value that is sent to the function when it is called.

در ارتباط با متن ها و قوانین انها یک مثال خیلی عالی دیدم که گفتم شما هم بخونیدش واقعا کمک میکنه :

Python Accessing Characters from String

A particular character from a string can be accessed using its index.

Just like lists, every character in a string has a unique index based on its position. Indices start from 0. Therefore, the first character of a string has index 0, the second character has index 1, and so on.

For example, consider the following string.

```
mystring = "Hey!"
```

In this string, the index of 'H' is 0, 'e' is 1, 'y' is 2 and '!' is 3. Thus, the index started from 0 and went up to 3.

character	'H'	'e'	'y'	'!'
index	0	1	2	3

A character of a string can be accessed by writing `name_of_string[index]`, where index is the index of the character in the string.

Therefore, the 2nd character of the string `mystring` can be accessed by `writing mystring[1]` (because the index of the 2nd character is 1).

و اما برنامه ی به درد نخور برعکس کردن یک جمله 🙄 :

```
python projects > Python_Challenge.py > ...
1 sentence=input('Please enter your sentence : ')
2 print(sentence[::-1])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Python_Challenge.py
Please enter your sentence : this is how it works
skrow ti woh si siht
PS C:\Users\User\Desktop\Python> █
```

توی بحث دیکشنری ها، بحث `get` پیش میاد که خیلی مهمه !
یه گوشه ای ازش رو بهترتون نشون میدم تا خودتون برید دنبالش :

```
python projects > Python_Challenge.py > ...
1 fruit = {'mango':40,'banana':10}
2 print(fruit.get('mango'))
3 print(fruit.get('banana'))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/Desktop/Python_Challenge.py"
40
10
PS C:\Users\User\Desktop\Python> |
```

گاهی نیاز داریم در دیکشنری خودمون تغییراتی ایجاد کنیم :

```
python projects > Python_Challenge.py > ...
1 mydict = {'name': 'John', 'age': 45}
2 mydict.update(age = 50) # changing value
3 print(mydict) # printing changed dictionary
```

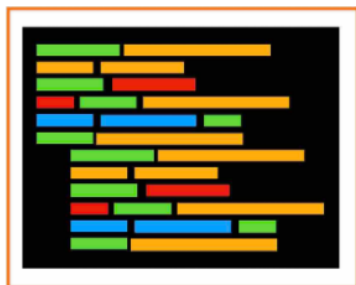
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Python_Challenge.py"
{'name': 'John', 'age': 50}
PS C:\Users\User\Desktop\Python> |
```

توی سایتی که قبلا معرفی کردم، در ارتباط با فانکشن ها توضیح جالبی نوشته
بیاید و با هم بخونیم :

Why do we need functions?

Function is a set of statements written together and given a name. We can call that set of statements at any place in our program by just calling its name and without writing the whole set of statements again. Quite convenient, right?



Need the entire code again?
Just call the function.

Suppose we need to check where a number is even or not multiple times in our program. Instead of writing the same login again and again, we can write a function and call it whenever we need to check whether a number is even or not.

Let's take an example.

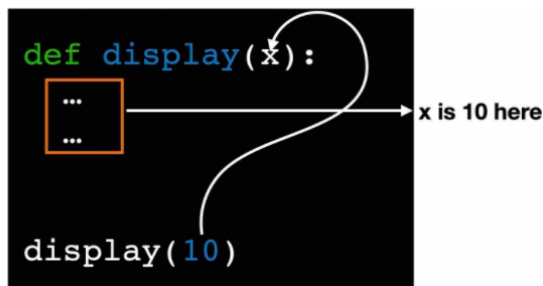
```
# function definition
def display(x):
    print("Hello")
    print("You passed:", x)

# calling function
display(10)
```

`def` is a keyword used for making functions. To make a function, we first write `def` and then the name of the function.

In the above example, `display` is the name of the function.

If we want to pass some value to a function, we give a parameter in the function definition. In our example, 10 is the value passed to the function and `x` is the name of the parameter. So, whenever a value is passed to the function, the parameter takes that value. Therefore, `x` takes the value 10 in our example.



The **body of the function** consists of the statements `print("Hello")` and `print("You passed:", x)`.

So this was the description of the function. Now let's call this function. A function is called by writing the name of the function and passing a value (if required) to it. In the above example the function is called by writing `display(10)`.

Once a function is called, the statements in the body of the function are executed. So, when `display(10)` was called, then the statements inside the function `display` got executed with `x = 10`.

و باز هم بحث پارامتر و آرگومان :

Python Parameters and Arguments

Parameters and arguments are just the different terms used for the variables/values you are already aware of. Parameters are the variables we use in the function definition whereas arguments are the values we pass in the function call.

```
def display(x):  
    ...  
    ...  
display(10)
```

Parameter

Argument

به خاطر بیماری و ضعف خیلی نمیتونم تکمیل کنم این بحث رو ولی به اندازه ی کافی مطالب رو گفتم.

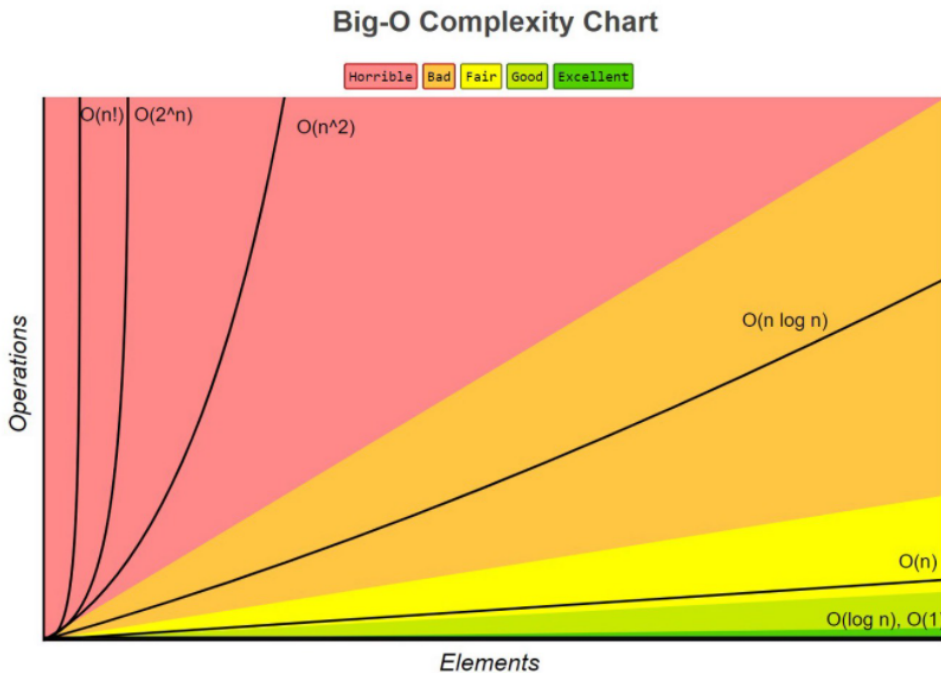
فردا میریم سراغ فایل ها، کار با فایل ها، فایل txt و بعد هم فایل های اکسل. دوباره تکرار میکنم که کل این مطالب صرفا دوره هست و قصد من آموزش نیست.

ممنون که دنبال میکنید 🌻🙌

کاری از : علی معینیان

In the name of god
21-day challenge with python
Part 12 out of 21

1 - Time complexity in python

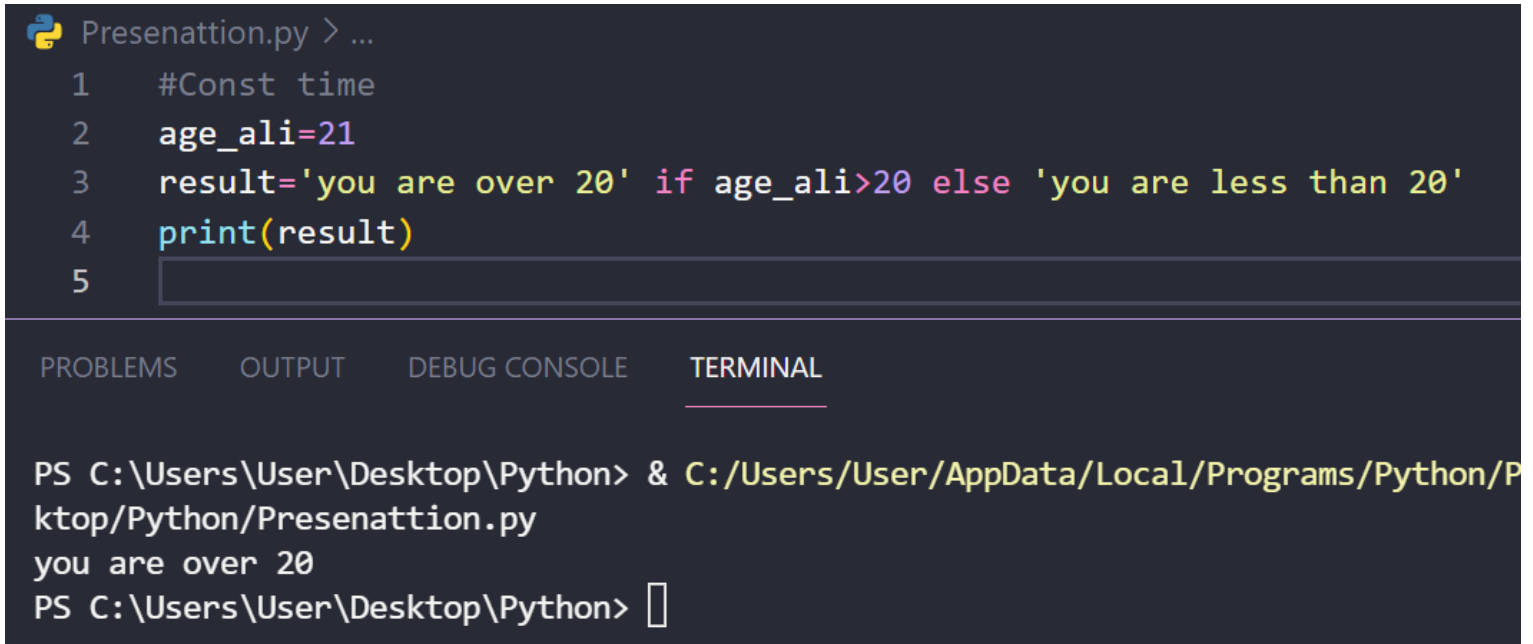


In computer science, time complexity is the computational complexity that describes the amount of time it takes to run an algorithm. Time complexity is commonly estimated by counting the number of elementary operations performed by the algorithm, supposing that each elementary operation takes a fixed amount of time to perform.

Now let's learn more about time complexities and let's see more examples about them in python :

1 - Constant Time — $O(1)$

It doesn't depend on the input data or no matter the size of input data, the running time always will be the same



```
Presenattion.py > ...
1 #Const time
2 age_ali=21
3 result='you are over 20' if age_ali>20 else 'you are less than 20'
4 print(result)
5
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python38-64/Python.exe C:/Users/User/Desktop/Python/Presenattion.py
you are over 20
PS C:\Users\User\Desktop\Python> █
```

2 - Logarithmic Time — $O(\log n)$

reduces the size of the input data in each step (it don't need to look at all values of the input data)

Algorithms with logarithmic time complexity are commonly found in operations on binary search.

Let's see what will be the binary search in python :

```

Presenattion.py > binary_search
1  # Returns index of x in arr if present, else -1
2  def binary_search(arr, low, high, x):
3      # Check base case
4      if high >= low:
5          mid = (high + low) // 2
6          # If element is present at the middle itself
7          if arr[mid] == x:
8              return mid
9          # If element is smaller than mid, then it can only
10         # be present in left subarray
11         elif arr[mid] > x:
12             return binary_search(arr, low, mid - 1, x)
13         # Else the element can only be present in right subarray
14         else:
15             return binary_search(arr, mid + 1, high, x)
16     else:
17         # Element is not present in the array
18         return -1
19 # Test array
20 arr = [ 2, 3, 4, 10, 40 ]
21 x = 10
22 # Function call
23 result = binary_search(arr, 0, len(arr)-1, x)
24 if result != -1:
25     print("Element is present at index", str(result))
26 else:
27     print("Element is not present in array")
28


```

You can find more explanations about binary search on the internet at [geeks for geeks](https://www.geeksforgeeks.org/) or where ever you like.

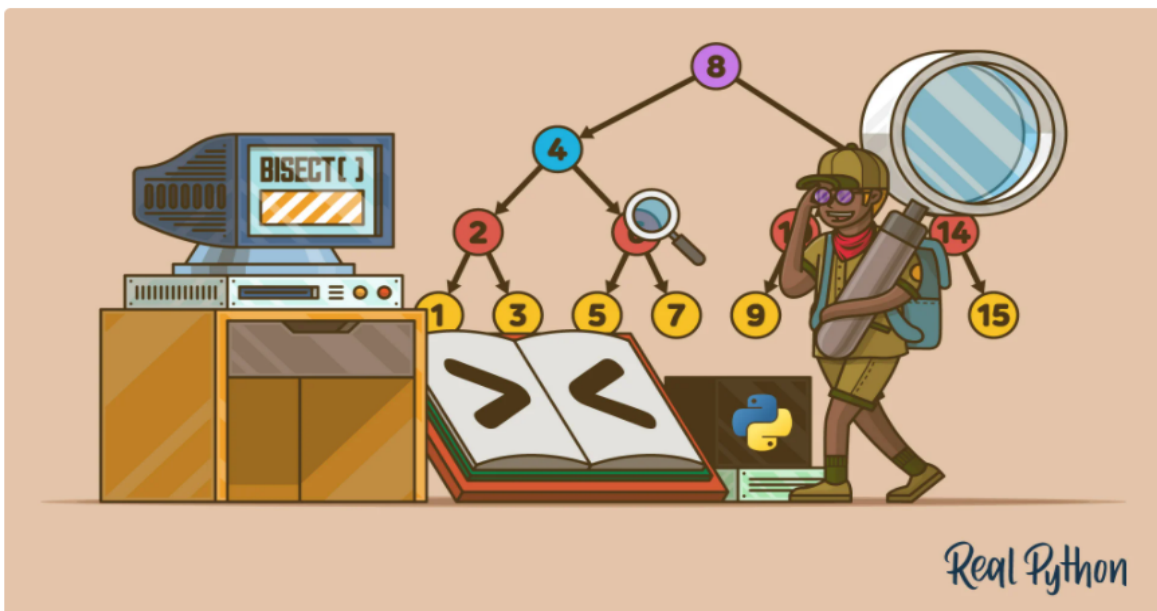
One more example of binary search :

Binary Search

	0	1	2	3	4	5	6	7	8	9
Search 23	2	5	8	12	16	23	38	56	72	91
	L=0	1	2	3	M=4	5	6	7	8	H=9
23 > 16 take 2 nd half	2	5	8	12	16	23	38	56	72	91
	0	1	2	3	4	L=5	6	M=7	8	H=9
23 > 56 take 1 st half	2	5	8	12	16	23	38	56	72	91
	0	1	2	3	4	L=5, M=5	H=6	7	8	9
Found 23, Return 5	2	5	8	12	16	23	38	56	72	91



If you really want to figure out how binary search works, you should read this article :





On the first attempt, the element in the middle happens to be a lemon. Since it's bigger than a strawberry, you can discard all elements to the right, including the lemon. You'll move the upper bound to a new position and update the middle index:



If you want to know more about these types of things you need to know more about hash and how it works but I won't explain hash here.

In python, we can perform a binary search by using a built-in function called: bisect

You can find elements with this built-in function through a list or whatever you like.

```

Presenation.py > ...
1 import bisect
2 sorted_fruits = ['apple', 'banana', 'orange', 'plum']
3 print(bisect.bisect_left(sorted_fruits, 'banana'))
4

```

def bisect_left(a: Sequence[_T], x: _T, lo: int=..., hi: int=...) -> int
Return the index where to insert item x in list a, assuming a is sorted.

The return value i is such that all e in a[:i] have e < x, and all e in a[i:] have e >= x. So if x already appears in the list, a.insert(i, x) will insert just before the leftmost x already there.

Optional args lo (default 0) and hi (default len(a)) bound the slice of a to be searched.

Full name: bisect.bisect_left

🔍 **bisect.bisect_left:** [function] Docs

We have another package for binary search in python :

Search projects

Help Sponsors Log in Register

binary-search 0.3.0

pip install binary-search

Latest version

Released: Aug 8, 2018

Binary search on python sorted sequences

Navigation

- Project description
- Release history
- Download files

Project links

Project description

Binary Search in Python

Binary search on already sorted sequences (list, tuple) to get the first element \geq or $>$ a value (similar to C++'s `std::lower_bound` and `std::upper_bound`).

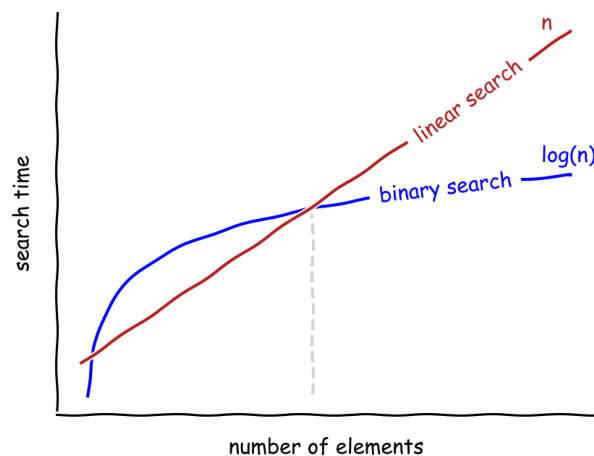
Requires python ≥ 3.5 for the `typing` module and ≥ 3.4 for the `enum` module, but you can easily extract the `search` function from `binary_search/__init__.py` if you need it for a lower version.

```
import binary_search as bs

sorted_sequence = (2, 5, 7, 9)
a = bs.search(sorted_sequence, 5)
print(a) # 1 - the index of the first element  $\geq 5$ 

b = bs.search(sorted_sequence, 6)
print(b) # 2 - the index of the first element  $\geq 6$  (element 2 with value 7)
```

Binary search is kind of great but don't forget this chart :



3 - Linear Time — $O(n)$

```
Presenattion.py > [e] names
1 list_names=['ali', 'mohammad', 'maryam', 'zahra', 'atefe', 'akbar']
2 for names in list_names:
3     if names == 'atefe':
4         print('True')
5     else:
6         print('False')
7
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe Presenattion.py
False
False
False
False
True
False
PS C:\Users\User\Desktop\Python> █
```

Let's see one more example :

```
# Linear Search in Python

def linearSearch(array, n, x):

    # Going through array sequentially
    for i in range(0, n):
        if (array[i] == x):
            return i
    return -1

array = [2, 4, 0, 1, 9]
x = 1
n = len(array)
result = linearSearch(array, n, x)
if(result == -1):
    print("Element not found")
else:
    print("Element found at index: ", result)
```

Let's figure out more about the concept of linear search in python :

1	3	5	4	7	9
---	---	---	---	---	---

List to be Searched for

Step - 2: If element is found, return the index position of the key.

1	3	5	4	7	9
---	---	---	---	---	---

↑
k≠7

1	3	5	4	7	9
---	---	---	---	---	---

↑
k≠7

1	3	5	4	7	9
---	---	---	---	---	---

↑
Key=7

1	3	5	4	7	9
---	---	---	---	---	---

↑
k≠7

Step - 3: If element is not found, return element is not present.

1	3	5	4	7	9
---	---	---	---	---	---

Key=7

4 - Quadratic Time — $O(n^2)$

An algorithm is said to have a quadratic time complexity when it needs to perform a linear time operation for each value in the input data

```
for x in data:  
    for y in data:  
        print(x, y)
```

Bubble sort is a great example of quadratic time.

Let's see the bubble sort in python :

```
def bubble_sort(our_list):
    # We go through the list as many times as there are elements
    for i in range(len(our_list)):
        # We want the last pair of adjacent elements to be (n-2, n-1)
        for j in range(len(our_list) - 1):
            if our_list[j] > our_list[j+1]:
                # Swap
                our_list[j], our_list[j+1] = our_list[j+1], our_list[j]
```

Now, let's populate a list and call the algorithm on it:

```
our_list = [19, 13, 6, 2, 18, 8]
bubble_sort(our_list)
print(our_list)
```

Output:

```
[2, 6, 8, 13, 18, 19]
```

5 - Exponential Time — $O(2^n)$

When we use to write recursive functions to calculate Fibonacci, that's it, we were using exponential time.

```
# Python program to display the Fibonacci sequence

def recur_fibo(n):
    if n <= 1:
        return n
    else:
        return(recur_fibo(n-1) + recur_fibo(n-2))

nterms = 10

# check if the number of terms is valid
if nterms <= 0:
    print("Plese enter a positive integer")
else:
    print("Fibonacci sequence:")
    for i in range(nterms):
        print(recur_fibo(i))
```


6 - Factorial — $O(n!)$

Big-o cheat sheet :

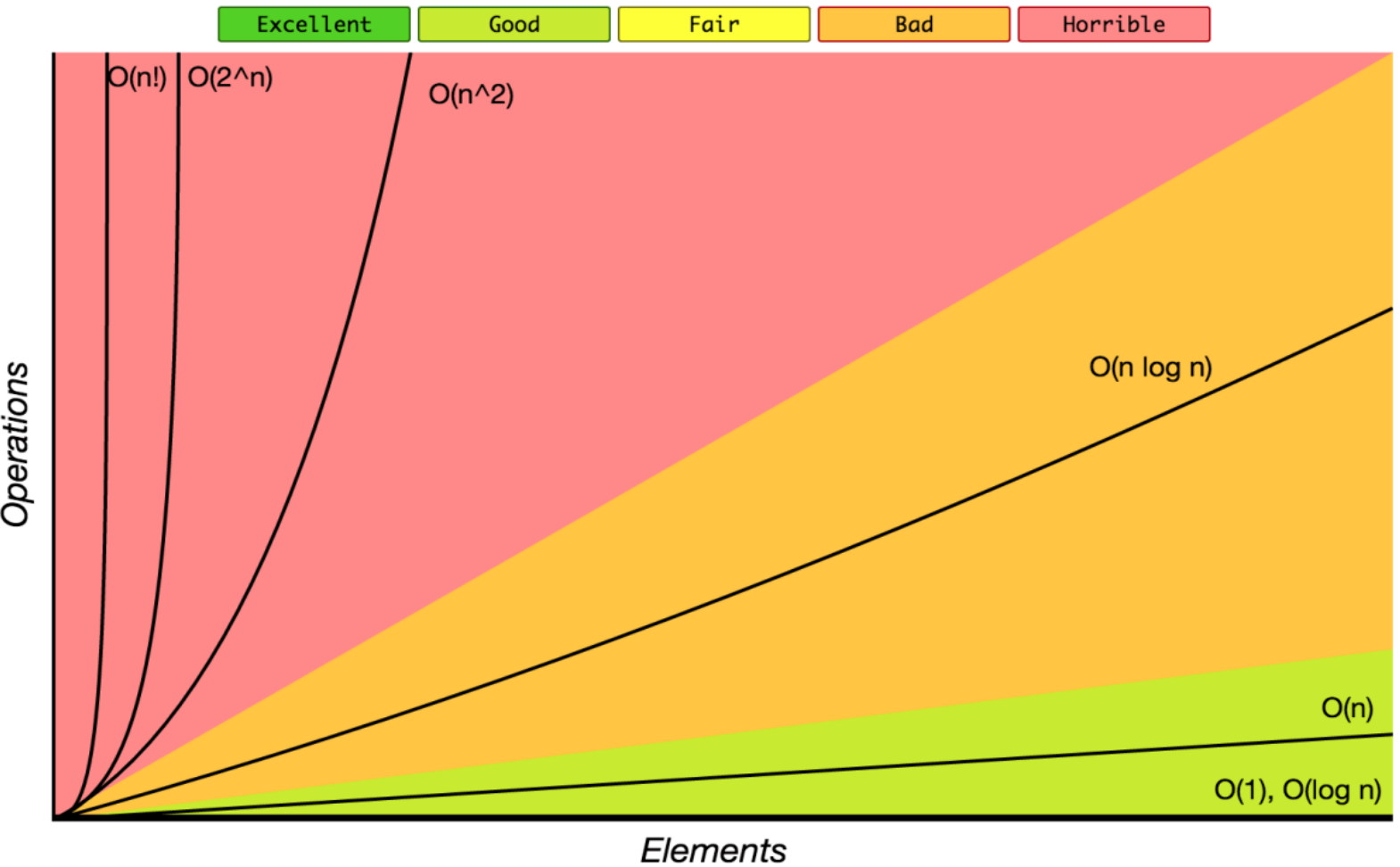
Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	$\theta(1)$	$\theta(n)$	$\theta(n)$	$\theta(n)$	$\theta(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Stack	$\theta(n)$	$\theta(n)$	$\theta(1)$	$\theta(1)$	$O(n)$	$O(n)$	$\theta(1)$	$\theta(1)$	$O(n)$
Queue	$\theta(n)$	$\theta(n)$	$\theta(1)$	$\theta(1)$	$O(n)$	$O(n)$	$\theta(1)$	$\theta(1)$	$O(n)$
Singly-Linked List	$\theta(n)$	$\theta(n)$	$\theta(1)$	$\theta(1)$	$O(n)$	$O(n)$	$\theta(1)$	$\theta(1)$	$O(n)$
Doubly-Linked List	$\theta(n)$	$\theta(n)$	$\theta(1)$	$\theta(1)$	$O(n)$	$O(n)$	$\theta(1)$	$\theta(1)$	$O(n)$
Skip List	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n \log(n))$
Hash Table	N/A	$\theta(1)$	$\theta(1)$	$\theta(1)$	N/A	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Binary Search Tree	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Cartesian Tree	N/A	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	N/A	$O(n)$	$O(n)$	$O(n)$	$O(n)$
B-Tree	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
Red-Black Tree	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
Splay Tree	N/A	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	N/A	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
AVL Tree	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
KD Tree	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$\theta(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
<u>Quicksort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
<u>Mergesort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n \log(n))$	$O(n)$
<u>Timsort</u>	$\Omega(n)$	$\theta(n \log(n))$	$O(n \log(n))$	$O(n)$
<u>Heapsort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n \log(n))$	$O(1)$
<u>Bubble Sort</u>	$\Omega(n)$	$\theta(n^2)$	$O(n^2)$	$O(1)$
<u>Insertion Sort</u>	$\Omega(n)$	$\theta(n^2)$	$O(n^2)$	$O(1)$
<u>Selection Sort</u>	$\Omega(n^2)$	$\theta(n^2)$	$O(n^2)$	$O(1)$
<u>Tree Sort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n^2)$	$O(n)$
<u>Shell Sort</u>	$\Omega(n \log(n))$	$\theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$
<u>Bucket Sort</u>	$\Omega(n+k)$	$\theta(n+k)$	$O(n^2)$	$O(n)$
<u>Radix Sort</u>	$\Omega(nk)$	$\theta(nk)$	$O(nk)$	$O(n+k)$
<u>Counting Sort</u>	$\Omega(n+k)$	$\theta(n+k)$	$O(n+k)$	$O(k)$
<u>Cubesort</u>	$\Omega(n)$	$\theta(n \log(n))$	$O(n \log(n))$	$O(n)$

Big-O Complexity Chart



The graph above shows the number of operations performed by the processor plotted against the number of input elements in the algorithm which in turn signifies which running time is the fastest. The one with the steepest gradient is going to take the most time to execute while the one with the lowest gradient is going to be the fastest because as the number of elements of the input of steeples gradient increases, the number of operations performed don't increase that much but in the case of steep gradients we can see that even with a small input, the number of operations performed are countless.

The next file will be about big-o and a great python package about it.

Thank you for your attention

By: Ali Moeinian

In the name of God
21-day challenge with python
Part 13 out of 21

Big-o packages in python

What is Big-o?

“Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity. It is a member of a family of notations invented by Paul Bachmann, Edmund Landau, and others collectively called Bachmann–Landau notation or asymptotic notation.”

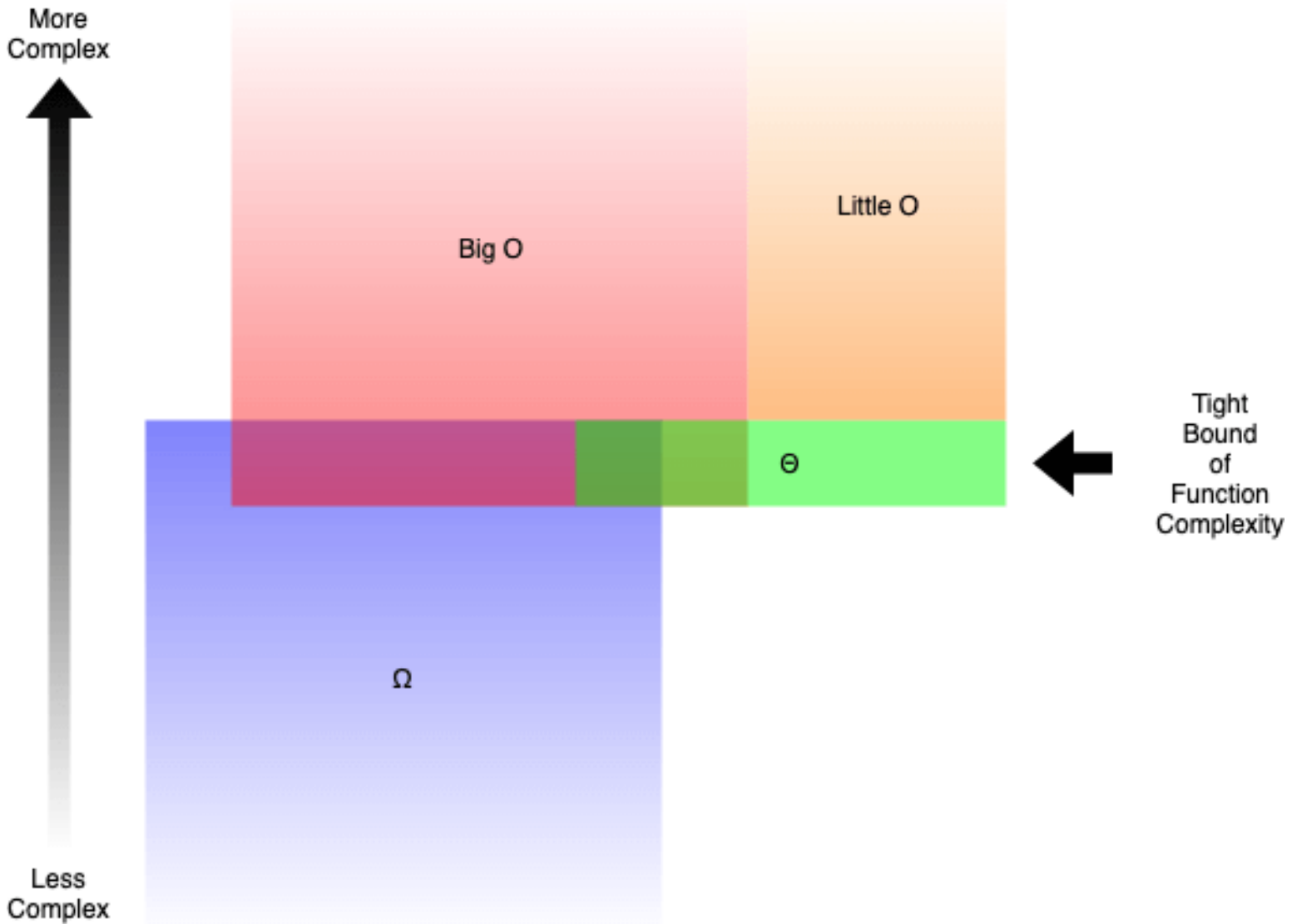
— Wikipedia’s definition of Big O notation

Big O: “ $f(n)$ is $O(g(n))$ ” iff for some constants c and N_0 , $f(N) \leq cg(N)$ for all $N > N_0$

Omega: “ $f(n)$ is $\Omega(g(n))$ ” iff for some constants c and N_0 , $f(N) \geq cg(N)$ for all $N > N_0$

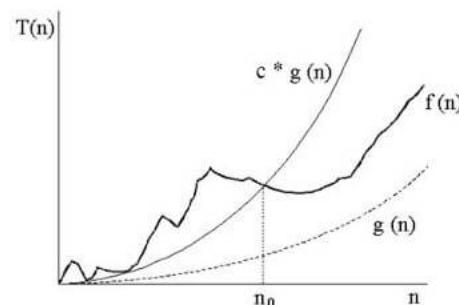
Theta: “ $f(n)$ is $\Theta(g(n))$ ” iff $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$

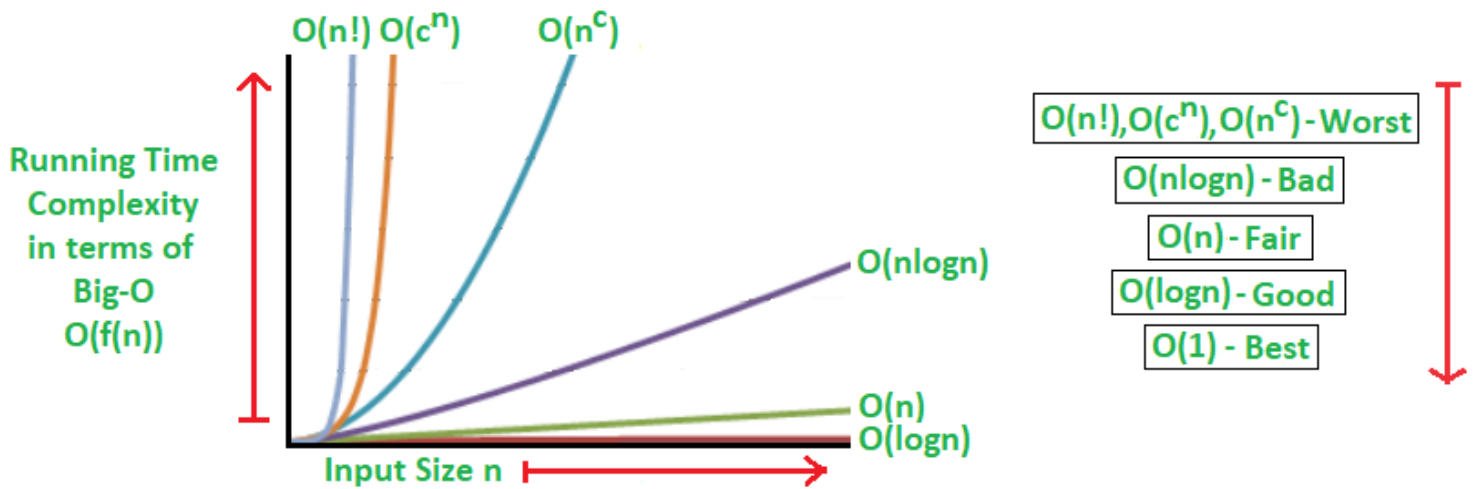
Little O: “ $f(n)$ is $o(g(n))$ ” iff $f(n)$ is $O(g(n))$ and $f(n)$ is not $\Theta(g(n))$



Big-Oh defined

- Big-Oh is about finding an *asymptotic upper bound*.
- Formal definition of Big-Oh:
 $f(N) = O(g(N))$, if there exists positive constants c, N_0 such that
 $f(N) \leq c \cdot g(N)$ for all $N \geq N_0$.
 - We are concerned with how f grows when N is large.
 - not concerned with small N or constant factors
 - Lingo: " $f(N)$ grows no faster than $g(N)$."





Let's review some points :

Algorithmic Examples of Runtime Analysis:

Some of the examples of all those types of algorithms (in worst-case scenarios) are mentioned below:

- *Logarithmic algorithm - $O(\log n)$ - Binary Search.*
- *Linear algorithm - $O(n)$ - Linear Search.*
- *Superlinear algorithm - $O(n \log n)$ - Heap Sort, Merge Sort.*
- *Polynomial algorithm - $O(n^c)$ - Strassen's Matrix Multiplication, Bubble Sort, Selection Sort, Insertion Sort, Bucket Sort.*
- *Exponential algorithm - $O(c^n)$ - Tower of Hanoi.*
- *Factorial algorithm - $O(n!)$ - Determinant Expansion by Minors, Brute force Search algorithm for Traveling Salesman Problem.*

Some examples :

- Ideal algorithm - $O(1)$ - Linear Search, Binary Search, Bubble Sort, Selection Sort, Insertion Sort, Heap Sort, Shell Sort.
- Logarithmic algorithm - $O(\log n)$ - Merge Sort.
- Linear algorithm - $O(n)$ - Quick Sort.
- Sub-linear algorithm - $O(n+k)$ - Radix Sort.

Common big-o functions :

Name	Big O
Constant	$O(c)$
Linear	$O(n)$
Quadratic	$O(n^2)$
Cubic	$O(n^3)$
Exponential	$O(2^n)$
Logarithmic	$O(\log(n))$
Log Linear	$O(n \log(n))$

Examples :

Constant Complexity ($O(C)$)

The complexity of an algorithm is said to be constant if the steps required to complete the execution of an algorithm remain constant, irrespective of the number of inputs. The constant complexity is denoted by $O(c)$ where c can be any constant number.

Let's write a simple algorithm in Python that finds the square of the first item in the list and then prints it on the screen.

```
def constant_algo(items):
    result = items[0] * items[0]
    print ()

constant_algo([4, 5, 6, 8])
```

In the above script, *irrespective of the input size*, or the number of items in the input list `items`, the algorithm performs only 2 steps: Finding the square of the first element and printing the result on the screen. Hence, the complexity remains constant.

Linear Complexity ($O(n)$)

The complexity of an algorithm is said to be linear if the steps required to complete the execution of an algorithm increase or decrease linearly with the number of inputs. Linear complexity is denoted by $O(n)$.

In this example, let's write a simple program that displays all items in the list to the console:

```
def linear_algo(items):
    for item in items:
        print(item)

linear_algo([4, 5, 6, 8])
```

Quadratic Complexity ($O(n^2)$)

The complexity of an algorithm is said to be quadratic when the steps required to execute an algorithm are a quadratic function of the number of items in the input. Quadratic complexity is denoted as $O(n^2)$. Take a look at the following example to see a function with quadratic complexity:

```
def quadratic_algo(items):  
    for item in items:  
        for item2 in items:  
            print(item, ' ', item)  
  
quadratic_algo([4, 5, 6, 8])
```

In the script above, you can see that we have an outer loop that iterates through all the items in the input list and then a nested inner loop, which again iterates through all the items in the input list. The total number of steps performed is $n * n$, where n is the number of items in the input array.

Do you remember that I introduced a very useful website that includes python packages and projects ? yes, PyPI.

In PyPI, you can find packages about big-o and omega but I won't use them because I want you to go and find them and learn how to work with the terminal of your IDE to understand how you should install those packages.

But I will give you a summary of them.



Search projects



Help

Sponsors

Log in

Register

big-O 0.10.1

pip install big-o



Latest version

Released: May 27, 2020

The package above is one of the greatest packages that give you a certain formula about your code and its execution time.

Let's see a simple example about Fibonacci :

```
python projects > recursive_fact.py > ...
```

```
1 import big_o
2 def fibo(n):
3     if n < 0:
4         return -1
5     if n < 2:
6         return n
7     else:
8         return fibo(n-1) + fibo(n-2)
9 print(big_o.big_o(fibo, big_o.datagen.n_, n_repeats=20, min_n=2, max_n=25)[0])
10
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe /recursive_fact.py
```

```
Exponential: time = -9.9 * 0.44^n (sec)
```

```
PS C:\Users\User\Desktop\Python>
```

All packages in PyPI has their own document and you can read them to learn how to work with those packages, for example :

Statistics

GitHub statistics:

★ Stars: 204

🔗 Forks: 33

📌 Open issues/PRs: 8

View statistics for this project via [Libraries.io](#) or by using [our public dataset on Google BigQuery](#)

Meta

License: LICENSE.txt

Author: [Pietro Berkes](#)

Maintainers



[Pietro Berkes](#)

For concreteness, let's say we would like to compute the asymptotic behavior of a simple function that finds the maximum element in a list of positive integers:

```
>>> def find_max(x):
...     """Find the maximum element in a list of positive integers."""
...     max_ = 0
...     for el in x:
...         if el > max_:
...             max_ = el
...     return max_
... 
```

To do this, we call `big_o.big_o` passing as argument the function and a data generator that provides lists of random integers of length N:

```
>>> import big_o
>>> positive_int_generator = lambda n: big_o.datagen.integers(n, 0, 10000)
>>> best, others = big_o.big_o(find_max, positive_int_generator, n_repeats=100)
>>> print(best)
Linear: time = -0.00035 + 2.7E-06*n (sec)
```

`big_o` inferred that the asymptotic behavior of the `find_max` function is linear, and returns an object containing the fitted coefficients for the complexity class. The second return argument, `others`, contains a dictionary of all fitted classes with the residuals from the fit as keys:

One of the greatest packages and in my idea the best package for big-o is :

big-O-calculator 0.0.9.8.4

pip install big-O-calculator

Released: Dec 29, 2020

It's really powerful and you must read it by yourself and try to find it out.

Here is the link: <https://pypi.org/project/big-O-calculator/>

Final cheat sheets :

Constant time $O(1)$



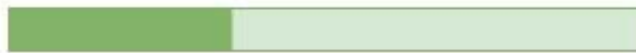
No matter how many elements we are working with, the algorithm/operation/whatever will always take the same amount of time

Linear time $O(\log n)$



You have this if doubling the number of elements you are iterating over doesn't double the amount of work. Always assume that searching operations are $\log(n)$

logarithmic time $O(n)$



Iterating through all elements in a collection of data. If you see a for-loop spanning from "0" to "array.length", you probably have "n" or linear time.

Quasilinear time $O(n \log n)$



You have this if doubling the number of elements you are iterating over doesn't double the amount of work. Always assume that any sorting operation is $n \cdot \log n$

Quadratic time $O(n^2)$



Every element in a collection has to be compared to every other element. Remember it if you went for shopping, after checkout and you need to go back to purchase another thing you have to checkout again.

Exponential time $O(2^n)$



Good

Bad

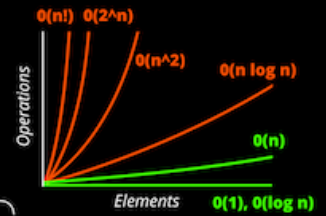
Very-Bad

LEGEND

TIME Complexity VS. SPACE Complexity

- Good Fair Bad
- Good Fair Bad

<BIG-O-CHEATSHEET>
 </>



DATA STRUCTURE
Operations

www.bigocheatsheet.com

ARRAY SORTING
Algorithms

DATA Structure	TIME Complexity								SPACE Complexity
	Average				Worst				
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Stack	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Queue	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Simply-Linked List	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Doubly-Linked List	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
Skip List	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n \log(n))$
Hash Table	N/A	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	N/A	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Binary Search Tree	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Cartesian Tree	N/A	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	N/A	$O(n)$	$O(n)$	$O(n)$	$O(n)$
B-Tree	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
Red-Black Tree	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
Splay Tree	N/A	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	N/A	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
AVL Tree	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	$O(n)$
KD Tree	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$\Theta(\log(n))$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$

ARRAY Algorithms	TIME Complexity				SPACE Complexity
	Best	Average	Worst	Worst	
	Quicksort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	
Mergesort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$	
Timsort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(1)$	
Heapsort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(1)$	
Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	
Selection Sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$	
Tree Sort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(n)$	
Shell Sort	$\Omega(n \log(n))$	$\Theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$	
Bucket Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n^2)$	$O(n)$	
Radix Sort	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$	
Counting Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$	
Cubesort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$	

Thank you for your attention

By: Ali Moeinian

21 day challenge with python



Part 14 out of 21

Work with txt files in python

Next file : csv files in python

سلامم ❤️

چون که کار کردن با فایل ها نکته های خیلی باحال خودش رو
داره تصمیم گرفتم کل کار کردن با فایل ها در پایتون رو به دو
قسمت تقسیم کنم :

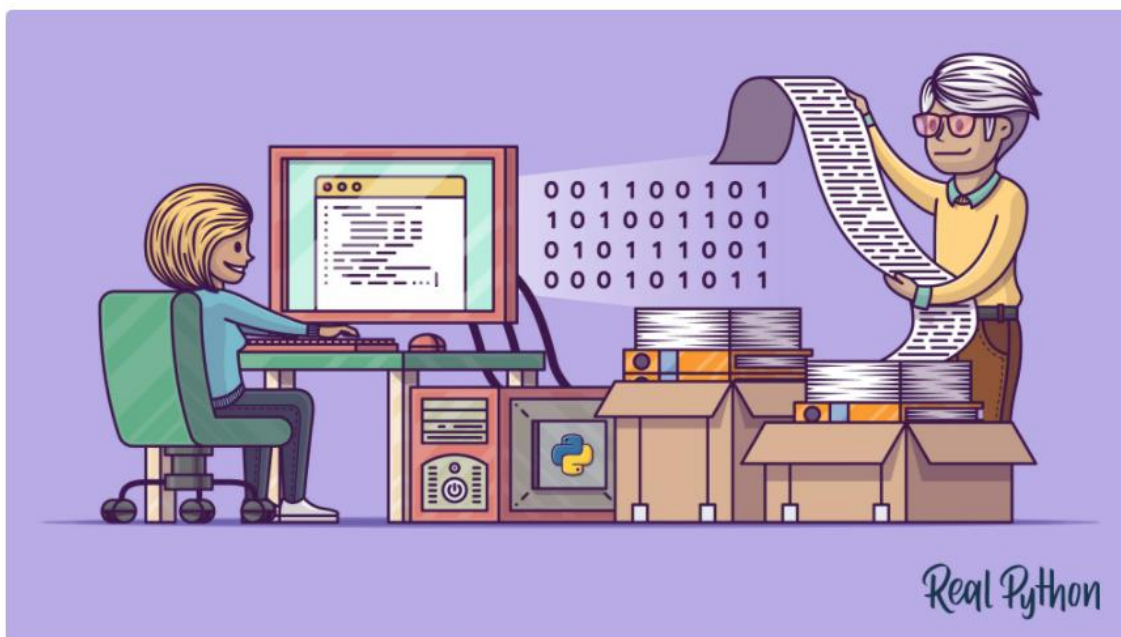
1 - کار با فایل های txt 2 - کار با فایل های csv

اما در اصل 3 نوع هستند، فایل های json هم هستند اما بعد از
یادگیری کار با فایل های csv دیگه بقیه اش کار سختی نیست.

راستی توی این قسمت و در روز بعدی یه مینی پروژه هم میزنیم
با هم با موضوعی که توی دنیای واقعی اتفاق افتاده .

البته روزش مشخص نیست، شاید یه پارت کامل از این دوره را
برای همون مینی پروژه اختصاص بدم.

خب قبل از شروع کار، دو تا از بهترین منابع برای یادگیری با
فایل ها رو میخواستم بهتون معرفی کنم :



Reading and Writing Files in Python (Guide)

Tutorials ▾ Student ▾ Jobs ▾ Courses

 GeeksforGeeks

Data Structures Algorithms Interview Preparation Topic-wise Practice C++ Java Python Competitive Programming Machine Learning Web Dev

Related Articles

[Destructors in Python](#)

[Inheritance in Python](#)

[Encapsulation in Python](#)

[Polymorphism in Python](#)

[Class method vs Static method in Python](#)

[File Handling in Python](#)

[Reading and Writing to text files in Python](#)

[Python Modules](#)

[Python Exception Handling](#)

[User-defined Exceptions in Python with examples](#)

<https://www.geeksforgeeks.org/courses/complete-interview-preparation-python-course/>



Reading and Writing to text files in Python

Difficulty Level : Easy • Last Updated : 19 Feb, 2021

Python provides inbuilt functions for creating, writing and reading files. There are two types of files that can be handled in python, normal text files and binary files (written in binary language, 0s and 1s).

- **Text files:** In this type of file, Each line of text is terminated with a special character called EOL (End of Line), which is the new line character ('\n') in python by default.
- **Binary files:** In this type of file, there is no terminator for a line and the data is stored after converting it into machine understandable binary language.

In this article, we will be focusing on opening, closing, reading and writing data in a text file.

Attention geek! Strengthen your foundations with the [Python Programming Foundation](#) Course and learn the basics.

To begin with, your interview preparations Enhance your Data Structures concepts

خب اول از همه نیاز داریم به یک فایل متنی :

```
python_challenge.txt - Python - Visual Studio Code
python_challenge.txt x
python projects > python_challenge.txt
1 Hello
2 this is a text file that i want to use in my program.
3 21 day challenge with python
4 by ali moeinian
5 im learning python every day and i really love it
6 im going to learn much more about back end development
7 and im going to start learn it by PHP and then Laravel.
8 i might start to read and learn more about Go and also django.
9 i havent decided about it yet but im sure im going to start my work with PHP and django, both.
```

دقت کنید که پسوند فایلتون باید `.txt` باشه 😊

خب اول از همه نیاز داریم این فایل رو وارد برنامه کنیم تا بتونیم باهاش کار کنیم :

```
python projects > Python_Challenge.py
1 with open(file, mode, encoding=str)
```

با دستور بالا میتونید فایل مد نظرتون رو در برنامه وارد کنید. کسانی که قبلا با پایتون کار میکردند و قدیمی این کار هستند شاید نحوه ی ورود فایلشون متفاوت باشه و اصلا از `with` استفاده نکنند اما همین `key word` باعث شده که کار ما برای

وارد کردن فایل ها خیلی راحت تر بشه و نیازی نیست دیگه از دستور `close` برای انتهای کارمون با فایل استفاده کنیم.

اما فایل بالا 3 تا آرگومان داره :

1 – مسیر فایل و خود فایل

2 – هدفی که برای استفاده از این فایل داریم (در آینده بیشتر میخونیم)

3 – فرمت کلی فایلمون

خب حالا بریم فایل رو وارد برنامه کنیم :

```
python projects > Python_Challenge.py
1 with open('python_projects\python_challenge.python_challenge.txt', mode, encoding=str):
2     pass
```

همونطور که میبینید با ادرس دهی کامل، تونستم فایل متنی خودم رو وارد برنامه کنم.

حالا بریم سراغ mode و ببینیم چه mode هایی رو میتونید استفاده کنیم ؟ جدول زیر خیلی بهمون کمک میکنه :

Mode	Description
'r'	Open a file for reading. (default)
'w'	Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
'x'	Open a file for exclusive creation. If the file already exists, the operation fails.
'a'	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
't'	Open in text mode. (default)
'b'	Open in binary mode.
'+'	Open a file for updating (reading and writing)

بجز جدول زیر، توضیحات پایین هم از سایت [geeks for geeks](https://www.geeksforgeeks.org/python-file-operation-0/) میتونه تکمیل بکنه این قسمت رو :

1. **Read Only ('r')** : Open text file for reading. The handle is positioned at the beginning of the file. If the file does not exist, raises I/O error. This is also the default mode in which file is opened.
2. **Read and Write ('r+')** : Open the file for reading and writing. The handle is positioned at the beginning of the file. Raises I/O error if the file does not exist.
3. **Write Only ('w')** : Open the file for writing. For existing file, the data is truncated and over-written. The handle is positioned at the beginning of the file. Creates the file if the file does not exist.
4. **Write and Read ('w+')** : Open the file for reading and writing. For existing file, data is truncated and over-written. The handle is positioned at the beginning of the file.
5. **Append Only ('a')** : Open the file for writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.
6. **Append and Read ('a+')** : Open the file for reading and writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end after the existing data.

ما توی این فایل هم از فایل‌مون متن رو میخونیم و ازش استفاده میکنیم و هم اینکه متن‌هایی که کاربرمون دوست داره رو وارد فایل متنی خواهیم کرد.

اما شاید بپرسید خب به چه دردی میخوره؟

من داشتم یک پروژه‌ی بانکی خیلی ساده مینوشتم که تصمیم گرفتم برای قسمت فیش بانکی که به کاربر داده میشه از همین قابلیت استفاده کنم و خیلیییی جالب شد.

خب اول از همه بریم فایل متنی رو چاپ کنیم در خروجی :

```
python_projects > Python_Challenge.py > txt_file
1 with open('python_projects\python_challenge.txt', 'r') as txt_file:
2     print (txt_file)
3
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe <_io.TextIOWrapper name='python_projects\python_challenge.txt' mode='r' encoding='cp1252'>
PS C:\Users\User\Desktop\Python> █
```

ای بابا ! چرا اینطوری شد پس ؟
بریم درستش کنیم :

```
python_projects > Python_Challenge.py > txt_file
1 with open('python_projects\python_challenge.txt', 'r') as txt_file:
2     print (txt_file.read())
3
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe <_io.TextIOWrapper name='python_projects\python_challenge.txt' mode='r' encoding='cp1252'>
Hello
this is a text file that i eant to use in my program.
21 day challenge with python
by ali moeinian
im learning python every day and i really love it
im going to learn much more about back end development
and im going to start learn it by PHP and then Laravel.
i might start to read and learn more about Go and also django.
i havent decided about it yet but im sure im going to start my work with PHP and django, both.
PS C:\Users\User\Desktop\Python> █
```

اهااان حالا درست شد 😊 باید از متد read برای خواندن کل متن استفاده میکردی.

اما بعضی وقتا دوست داری متنت رو خط به خط بخونی،
اینجاست که متد استفاده شده باید یکم متفاوت باشه :

```
python_projects > Python_Challenge.py > txt_file
1 with open('python_projects\python_challenge.txt', 'r') as txt_file:
2     print (txt_file.readlines())
3
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop/Python/python_projects/Python_Challenge.py
['Hello\n', 'this is a text file that i eant to use in my program.\n', '21 day challenge with python \n', 'by ali moeinian\n', 'im learning python every day and i really love it \n',
'im going to learn much more about back end development\n', 'and im going to start learn it by PHP and then Laravel.\n', 'i might start to read and learn more about Go and also django
.\n', 'i havent decided about it yet but im sure im going to start my work with PHP and django, both.']
PS C:\Users\User\Desktop\Python>
```

خب همونطور که میبینی خیلییی خروجیمون زشت شد
میتونی تعیین کنی تا چه کاراکتری رو دوست داری توی خروجی
خودت ببینی :

```
python_projects > Python_Challenge.py > txt_file
1 with open('python_projects\python_challenge.txt', 'r') as txt_file:
2     print (txt_file.read(36))
3
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Pyt
Hello
this is a text file that i ean
PS C:\Users\User\Desktop\Python>
```

همونطور که میبینید، مثلا من خواستم صرفا تا کاراکتر 36 رو
در خروجی ببینم.

بریم ببینیم اصلا فایلمون چند خط بوده و توی هر خط چه چیزی
رو نوشتیم :

```
python_projects > Python_Challenge.py > txt_file
1 with open('python_projects\python_challenge.txt', 'r') as txt_file:
2     reader_object=txt_file.readlines()
3     counter=0
4     for lines in reader_object:
5         counter+=1
6         print('line ( %i ) : %s ' %(counter,lines))
7
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/De
line ( 1 ) : Hello
line ( 2 ) : this is a text file that i eant to use in my program.
line ( 3 ) : 21 day challenge with python
line ( 4 ) : by ali moeinian
line ( 5 ) : im learning python every day and i really love it
line ( 6 ) : im going to learn much more about back end development
line ( 7 ) : and im going to start learn it by PHP and then Laravel.
line ( 8 ) : i might start to read and learn more about Go and also django.
line ( 9 ) : i havent decided about it yet but im sure im going to start my work with PHP and django, both.
PS C:\Users\User\Desktop\Python> █
```

😊 اینجاست که خواندن خط به خط برنامه به دردمون میخوره

میتونی تعداد حروف دل خواهت رو هم بشماری، مثلا من تعداد حروف a رو داخل متنم میخوام بشمارم :

```
python_projects > Python_Challenge.py > txt_file
```

```
1 with open('python_projects\python_challenge.txt', 'r') as txt_file:  
2     reader=txt_file.read()  
3     if 'a' in reader:  
4         print('you have (%i) a in your text . ' %(reader.count('a')))  
5  
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Pyth  
you have (34) a in your text .  
PS C:\Users\User\Desktop\Python> █
```

حتی میتونید برای این قسمت برنامهتون input تعریف کنید تا کاربرتون به دلخواه این کار را انجام بده.

حتی میتونید کلمات رو پیدا کنید و هر بلایی دوست داشتید سرشون بیارید.

اما بریم سراغ یکسری برنامه ی خیلی جالب که خودم تونستم بنویسمشون واقعا خیلی براشون وقت گذاشتم.

اما قبلش، نیاز دارم یکسری عدد الکی وارد فایل متنی بکنم چون قراره با اعداد توی متن کار کنیم :

```
python_projects > python_challenge.txt
1 Hello
2 this is a text file that i want to use in my program.
3 21 day challenge with python
4 by ali moeinian
5 im learning python every day and i really love it
6 im going to learn much more about back end development
7 and im going to start learn it by PHP and then Laravel.
8 i might start to read and learn more about Go and also django.
9 i havent decided about it yet but im sure im going to start my work with PHP and django, both.
10 my phone number is : 21615464511
11 my postal code : 5161511511561
12 my another phone number : 0913 2563 5665
13 my birth : 644131231321
```

همه ی اعداد بالا فرضی و الکی هستند

```
python_projects > Python_Challenge.py > txt_file
1 with open('python_projects\python_challenge.txt', 'r') as txt_file:
2     # find numbers in a text file
3     numbers=[]
4     for line in txt_file:
5         words = line.split()
6         for i in words:
7             for letter in i:
8                 if(letter.isdigit()):
9                     numbers.append(letter)
10
11     print(''.join(numbers))
12
13
14
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Pyt
21091288965355161511561091325635665644131231321
PS C:\Users\User\Desktop\Python> █
```

ولی خب اخه این اعداد بالا که خیلی به دردمون نمیخوره !

```
python_projects > Python_Challenge.py > txt_file
1 with open('python_projects\python_challenge.txt', 'r') as txt_file:
2     # find numbers in a text file
3     numbers=[]
4     for line in txt_file:
5         words = line.split()
6         for i in words:
7             for letter in i:
8                 if(letter.isdigit()):
9                     numbers.append(letter)
10
11     print(' '.join(numbers))
12     print(numbers)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop/Python/python_projects/Python_Challenge.py
21091288965355161511511561091325635665644131231321
['2', '1', '0', '9', '1', '2', '8', '8', '9', '6', '5', '3', '5', '5', '1', '6', '1', '5', '1', '1', '5', '1', '1', '5', '6', '1', '0', '9', '1', '3', '2', '5', '6', '3', '5', '6', '6', '5', '6', '4', '4', '1', '3', '1', '2', '3', '1', '3', '2', '1']
PS C:\Users\User\Desktop\Python> █
```

ای وای! باز هم که بد تر شد 😞

```
python_projects > Python_Challenge.py > txt_file
1 with open('python_projects\python_challenge.txt', 'r') as txt_file:
2     # find numbers in a text file
3     numbers=[]
4     for line in txt_file:
5         words = line.split()
6         for i in words:
7             for letter in i:
8                 if(letter.isdigit()):
9                     numbers.append(letter)
10                    if len(numbers) == 11:
11                        print(' '.join(numbers))
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop/Python/python_projects/Python_Challenge.py
0 9 1 2 8 8 9 6 5 3 5
PS C:\Users\User\Desktop\Python> █
```

خب خب خب 😊 انگار بهتر شد !

تا اینجا فقط یکم بازی بازی کردیم با این کد ها و فایل متنیومون.

اما به نظرتون ایراد کد بالایی چیه ؟

واقعا چرا نمیشد اعداد توی یک متن رو مجزی بیرون آورد و به عنوان یک خروجی نشون داد ؟؟؟

قطعا میشه 😊 اما دوست دارم خودتون رو به چالش بکشید و یک فایل متنی رو پیدا کنید که حاوی اعداد باشه و اعداد اون رو بدون اینکه تغییری بکنه توی خروجی نمایش بدید، خیلی جالب میشه.

اما بریم و در نهایت توی فایلمون متن بنویسیم :
توی برنامه ازت میخوام اسمتو وارد کنی و اسمت وارد فایل
متنمون میشه.

```
python_projects > Python_Challenge.py > txt_file
1 with open('python_projects\python_challenge.txt', 'w') as txt_file:
2     InputText=str(input('Please enter your name : '))
3     txt_file.write(InputText)
4     print('Write in empty file done...')
5
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python38-64/Scripts/python.exe python_projects\Python_Challenge.py
Please enter your name : ali moeinian
Write in empty file done...
PS C:\Users\User\Desktop\Python> 
```

خب حالا بریم فایل رو چک کنیم ببینیم چی شد :

```
Python_Challenge.py python_challenge.txt x
```

```
python_projects > python_challenge.txt
1 ali moeinian
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python38-64/Scripts/python.exe python_projects\Python_Challenge.py
Please enter your name : ali moeinian
Write in empty file done...
PS C:\Users\User\Desktop\Python> 
```

بله! اسممون وارد فایل شده 😊

ولی متن های قبلیمون چی شد؟؟

شاید بعضی وقت ها نخواهیم متن هامون رو از دست بدیم، خب پس باید چیکار کنیم؟
باید mode رو تغییر بدیم.

```
python_projects > Python_Challenge.py > txt_file
```

```
1 with open('python_projects\python_challenge.txt', 'a+') as txt_file:
2     InputText=str(input('Please enter your name : '))
3     txt_file.write(InputText)
4     print('Write in empty file done...')
5
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Pyth
Please enter your name : ali moeinian *****
Write in empty file done...
PS C:\Users\User\Desktop\Python> █
```

Mode فایل رو به a+ تغییر دادم و بریم نتیجه رو ببینیم :

```
python_projects > python_challenge.txt
```

```
1 Hello
2 this is a text file that i eant to use in my program.
3 by ali moeinian
4 im learning python every day and i really love it
5 im going to learn much more about back end development
6 and im going to start learn it by PHP and then Laravel.
7 i might start to read and learn more about Go and also django.
8 i havent decided about it yet but im sure im going to start my work with PHP and django, both.
9 my phone number is : 09128896535
10 ali moeinian ***** ←
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Us
Please enter your name : ali moeinian *****
Write in empty file done...
PS C:\Users\User\Desktop\Python> █
```

خب این هم از فایل مربوط به فایل های متنی 🙌🌸
سعی کردم کامل بگم و چالش های باحال رو هم جواب بدم.
قطعا موارد و نکات خیلی بیشتری هست اما هدف ما صرفا دوره
کردن اصل مطالب هست.

خیلی ممنون که دنبال میکنید 🌲❤️

کاری از : علی معینیان

21 day challenge with python



Part 15 out of 21

Work with CSV files in python

بریم سراغ فایل های CSV و ببینیم چرا باید کار با اونها رو یاد بگیریم؟

قبل از شروع توضیحات و مثال ها، بهترین منابع برای یادگیری کار با فایل های CSV رو بهتون معرفی میکنم :

Table of Contents

- csv — CSV File Reading and Writing
 - Module Contents
 - Dialects and Formatting Parameters
 - Reader Objects
 - Writer Objects
 - Examples

Previous topic

File Formats

Next topic

configparser — Configuration file parser

This Page

Report a Bug
Show Source

csv — CSV File Reading and Writing

Source code: [Lib/csv.py](#)

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases. CSV format was used for many years prior to attempts to describe the format in a standardized way in [RFC 4180](#). The lack of a well-defined standard means that subtle differences often exist in the data produced and consumed by different applications. These differences can make it annoying to process CSV files from multiple sources. Still, while the delimiters and quoting characters vary, the overall format is similar enough that it is possible to write a single module which can efficiently manipulate such data, hiding the details of reading and writing the data from the programmer.

The `csv` module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats.

The `csv` module’s `reader` and `writer` objects read and write sequences. Programmers can also read and write data in dictionary form using the `DictReader` and `DictWriter` classes.

See also:

[PEP 305 - CSV File API](#)

The Python Enhancement Proposal which proposed this addition to Python.

سایت بالایی که داکيومنت های خود پایتون هست و بسیار عالی توضیح داده، توصیه میکنم حتما مطالعه کنید و از مثال هاش استفاده کنید.

Related Articles

- Introduction ▼
- Input/Output ▼
- Operators ▼
- Data Types ▼
- Control Flow ▼
- Functions ▼
- Python OOP ▼
- Exception Handling ▼
- File handling ▼
- Python on Regex ▼

Working with csv files in Python



Difficulty Level : Medium • Last Updated : 22 Jul, 2021

This article explains how to load and parse a CSV file in Python.

First of all, what is a CSV ?

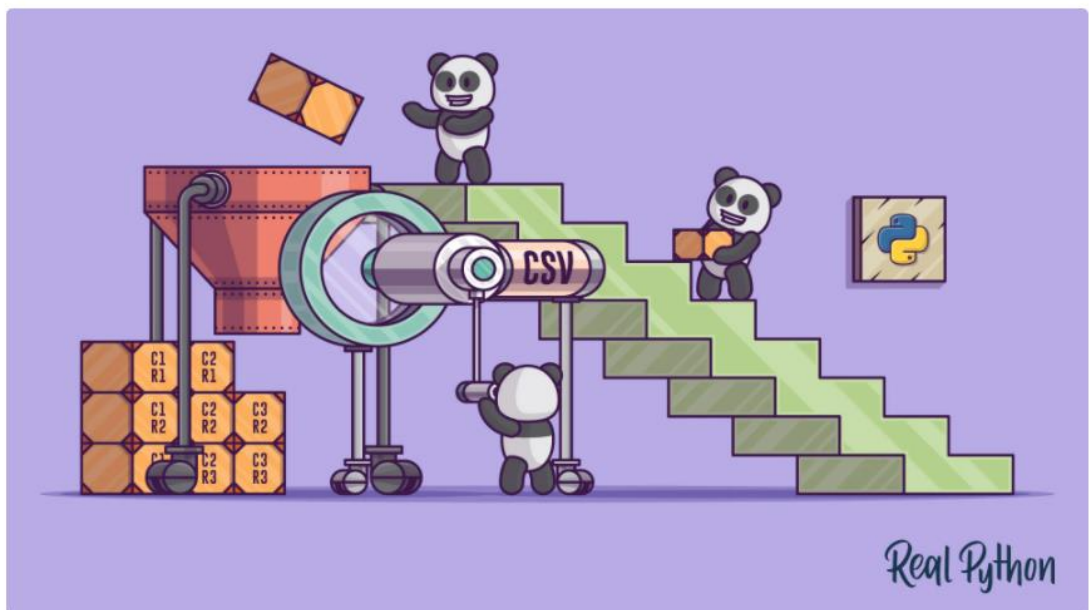
CSV (Comma Separated Values) is a simple **file format** used to store tabular data, such as a spreadsheet or database. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

For working CSV files in python, there is an inbuilt module called [csv](#).

8/10 Student success ratio
9/10 course completion ratio

A live learning experience to put you among the top skilled professionals!

Make it happen



Reading and Writing CSV Files in Python

by Jon Fincher • Jul 16, 2018 • 78 Comments • data-science intermediate python

به نظرم بهترین توضیحات برای این قسمت، سایت بالایی هست.
کلا این سایت real python خیلیییی از لحاظ اینطور مقالات کامل
هست و واقعا اگر دوست دارید حتی ریز ترین نکات کار با هر
موضوعی رو یادگیرید، مقاله های این سایت رو مطالعه کنید.

اول از همه، ببینیم اصلا csv چیه ؟

Csv = comma seprated values

ولی اصلا به چه درد میخوره ؟ بهترین جواب رو از سایت
Real python میتونید بخونید :

Let's face it: you need to get information into and out of your programs through more than just the keyboard and console. Exchanging information through text files is a common way to share info between programs. One of the most popular formats for exchanging data is the CSV format. But how do you use it?

من یک فایل csv فرضی و الکی آماده کردم، بریم ببینیم چطوریه ؟
راستی یادتون نره پسوند فایل های مربوطه حتما باید CSV. باشه.

```
example.csv X ← csv.py
python_projects > example.csv
1 name,age,city
2 ali,21,esfahan
3 mohammad,20,tehran
4 hossein,25,sanandaj
5 maryam,65,eilam
6 akbar,70,gilan
7 asghar,66,kordestan
8 jaber,23,khoozestan
9 kazem,90,ghazvin
```

فایل بالایی 3 تا ستون با عنوان های نام، سن و شهر داره و قراره توی این فایل با این 3 تا ستون و آدم ها و اطلاعات توش یکم بازی کنیم.

اما دقیقا عین کار با فایل های متنی، باید قبل از شروع کار، فایل مربوطه رو بیاریم توی برنامه :

```
python_projects > Python_Challenge.py > csv_file
1 with open('python_projects\example.csv', 'r') as csv_file:
2     pass
```

دقیقا عین وارد کردن فایل های متنی، هیچ فرقی نداره توی این قسمت حتی قسمت mode هم قوانینش همونه 😊

برای خواندن فایل CSV نیاز داریم عین قبل، یک خواننده تعریف کنیم و از متد های مربوطه استفاده کنیم :

```
python_projects > Python_Challenge.py > csv_file
```

```
1 with open('python_projects\example.csv', 'r') as csv_file:  
2     reader=csv_file.read()  
3     print(reader)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/  
name,age,city  
ali,21,esfahan  
mohammad,20,tehran  
hossein,25,sanandaj  
maryam,65,eilan  
akbar,70,gilan  
asghar,66,kordestan  
jaber,23,khoozestan  
kazem,90,ghazvin  
PS C:\Users\User\Desktop\Python> █
```

خب تا اینجا که همه چیز عین قبله و تفاوتی احساس نمیشه.

اما نیاز داریم تا کا ماژول CSV رو وارد برنامه کنیم تا کارمون راحت تر و درست تر بشه :

نزدیک یک ساعت میشه درگیر این ارور لعنتی شدم و اصلا نمیزاره
کارم رو پیش ببرم :

```
python_projects > challenge.py > csvfile
```

```
1 import csv
2 with open('python_projects\example.csv') as csvfile:
3     handle = csv.reader(csvfile)
4     for items in handle:
5         print(items)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop
Traceback (most recent call last):
  File "c:\Users\User\Desktop\Python\python_projects\challenge.py", line 1, in <module>
    import csv
  File "c:\Users\User\Desktop\Python\python_projects\csv.py", line 5, in <module>
    reader=csv.reader(file)
AttributeError: partially initialized module 'csv' has no attribute 'reader' (most likely due to a circular import)
PS C:\Users\User\Desktop\Python> █
```

هر کاری کردم درست نشد واقعا !

اعصاب داغون !

بای

21 day challenge with python



Part 16 out of 21

Work with CSV files in python

After bug

سلااااااام ✨

دیروز با یک باگی برخورد کردم که برای اولین بار بود میدیدمش و هر چقدر هم سرچ کردم به جواب درستی نرسیده بودم.

اما داستان از این قرار بود که ما اومدیم و ماژول CSV رو وارد برنامه کردیم.

من حواسم نبود که یک فایل پایتون به اسم `csv.py` داشتم توی همون فولدر و در اصل برنامه داشت این فایل پایتون رو به جای ماژول وارد برنامه میکرد.

مورد بعدی اینکه باگ اصلا ترس نداره و خجالت نکشید ازش، واقعا من اگر به خطا برخورد نمیکردم و باهاشون دست و پنجه نرم نمیکردم، پیشرفتی در کارم نبود.

و اینکه دیروز یاد گرفتم علاوه بر نامگذاری متغیر ها، خیلی مهمه که توی نام گذاری فایل هامون هم دقت کنیم که این مشکلات ریز پیش نیاد.

بریم سراغ کارمون با یک خلاصه ی ریز از فایل گذشته .

خب ما فایل های CSV رو تعریف کردیم و این هم از فایل خودمون :

```
python_projects > example.csv
1  'name','age','city'
2  'ali',20,'esfahan'
3  'mohammad',20,'eilam'
4  'maryanm',21,'sistan'
5  'zahra',30,'tehran'
6  'kazem',45,'semnan'
7  'siroos',18,'esfahan'
8  'elham',14,'tehran'
9  'fateme',65,'tehran'
```

و بعد هم اومدیم و فایل رو وارد برنامه کردیم :

```
python_projects > challengen.py > csvfile
1  import csv
2  with open('python_projects\example.csv') as csvfile:
3      pass
```

خب بریم و فایلمون رو بخونیم :

```
python_projects > 🐍 challengen.py > 📄 csvfile
1 import csv
2 with open('python_projects\example.csv') as csvfile:
3     reader = csv.reader(csvfile, delimiter=' ', quotechar='|')
4     print(reader)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop/Python/python_projects/challengen.py
<_csv.reader object at 0x00000238761D13C0>
PS C:\Users\User\Desktop\Python> █
```

ای باباااااااا ! چی شد؟؟؟ بریم درستش کنیم :

```
python_projects > 🐍 challengen.py > 📄 csvfile
1 import csv
2 with open('python_projects\example.csv') as csvfile:
3     reader = csv.reader(csvfile,)
4     print(list(reader))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop/Python/python_projects/challengen.py
[["name", "age", "city"], ["ali", '20', "esfahan"], ["mohammad", '20', "eilam"], ["maryam", '21', "sistan"], ["zahra", '30', "tehran"], ["kazem", '45', "semnan"], ["siroos", '18', "esfahan"], ["elham", '14', "tehran"], ["fateme", '65', "tehran"]]
PS C:\Users\User\Desktop\Python> █
```



بهتر شد ولی باز هم ایراد داره :

اولا که خیلی ظاهر بدی داره

دوما اینکه خیلی شلوغه

سوما اینکه نباید اون سر تیترو های ستون ها رو نشون بده

بریم درستش کنیم 😊

```
python_projects >  challeng.py >  csvfile
```

```
1 import csv
2 with open('python_projects\example.csv') as csvfile:
3     reader = csv.reader(csvfile)
4     next(reader)
5     for items in reader:
6         print(items)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/
ktop/Python/python_projects/challeng.py
```

```
['ali', '20', 'esfahan']
['mohammad', '20', 'eilam']
['maryanm', '21', 'sistan']
['zahra', '30', 'tehran']
['kazem', '45', 'semnan']
['siroos', '18', 'esfahan']
['elham', '14', 'tehran']
['fateme', '65', 'tehran']
```



```
PS C:\Users\User\Desktop\Python> 
```

 اهان

الان خوشگل شد و دیگه شلوغ هم نیست.

خب توی مرحله ی بعدی ما نیاز داریم تا بتونیم به محتویات داخل هر ستون دسترسی داشته باشیم تا بتونیم کار هایی که میخوایم رو انجام بدیم :

مثلا به اسم ها دسترسی میخوایم داشته باشیم :

```
python_projects >  challengen.py >  csvfile  
1 import csv  
2 with open('python_projects\example.csv') as csvfile:  
3     reader = csv.reader(csvfile)  
4     next(reader)  
5     for items in reader:  
6         print(items[0])
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Python/Python/python_projects/challengen.py  
ali  
mohammad  
maryanm  
zahra  
kazem  
siroos  
elham  
fateme  
PS C:\Users\User\Desktop\Python> |
```

فهمیدی چیکار کردم ???

قوانین لیست ها - ایندکس ها و 😊 خودشه !

مثلا میخواهیم میانگین همه ی سن ها رو بدست بیاریم :

```
python_projects > 📄 challengen.py > [🔍] csvfile
```

```
1 import csv
2 import statistics
3 with open('python_projects\example.csv') as csvfile:
4     reader = csv.reader(csvfile)
5     next(reader)
6     📌 list_ages=[]
7     for items in reader:
8         change_float=float(items[1])
9         list_ages.append(change_float)
10    print('list of ages are : ' + str(list_ages))
11    print('average is : ' + str(statistics.mean(list_ages)))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python39-64/Python.exe python_projects/challengen.py
list of ages are : [20.0, 20.0, 21.0, 30.0, 45.0, 18.0, 14.0, 65.0]
average is : 29.125
PS C:\Users\User\Desktop\Python> █
```

به همین سادگی و زیبایی 🥰

اما راستی statistics چیه؟؟

یادته گفتم حتما سعی کن داکيومنت های پایتون رو بخونی؟؟

یه جایی که فکرشو نمیکنی دستتو میگیره !

این هم از کاربرد های باحال این کتابخونه که خیلی راحت کمکم کرد

بتونم میانگین رو بگیرم. 😊

یکم چیز هایی که از قبل یاد گرفتیم رو با مواردی که الان یاد گرفتیم
قاطی کنیم ببینیم چی میشه :

```
python_projects > 📄 challeng.py > [🔍] csvfile
```

```
1 import csv
2 import statistics
3 with open('python_projects\example.csv') as csvfile:
4     reader=csv.reader(csvfile)
5     next(reader)
6     print(list(map(lambda item: item[0], reader)))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python38-64/Python/python_projects/challeng.py
['ali', 'mohammad', 'maryam', 'zahra', 'kazem', 'siroos', 'elham', 'fateme']
PS C:\Users\User\Desktop\Python> █
```

چقدر راحت شد 😊 😊

کل کار با فایل های CSV مربوط به دسترسی به ازلاعات داخل ستون ها و سطر هاست و اگر این کار رو به خوبی یاد بگیریم، دیگه بقیه اش بازی کردن با کد هاست.

اما یک عالمه چیز دیگه مونده از فایل های CSV که واقعا بعضی هاش خیلی حرفه ای میشه و ما هدفمون صرفا دوره کردنه.

حتما اگر با CSV ها کار میکنید، بهتون توصیه میکنم داکيومنت خود پایتون در این باره رو یک بار هم که شده مطالعه بکنید، مخصوصا قسمتی که با دیکشنری ها شروع به کار میکنه و اونها رو وارد فایل های CSV میکنه :

```
>>> import csv
>>> with open('names.csv', newline='') as csvfile:
...     reader = csv.DictReader(csvfile)
...     for row in reader:
...         print(row['first_name'], row['last_name'])
...
Eric Idle
John Cleese

>>> print(row)
{'first_name': 'John', 'last_name': 'Cleese'}
```



```
class csv.DictWriter(f, fieldnames, restval='', extrasaction='raise', dialect='excel', *args,
**kws)
```

Create an object which operates like a regular writer but maps dictionaries onto output rows. The *fieldnames* parameter is a [sequence](#) of keys that identify the order in which values in the dictionary passed to the `writerow()` method are written to file *f*. The optional *restval* parameter specifies the value to be written if the dictionary is missing a key in *fieldnames*. If the dictionary passed to the `writerow()` method contains a key not found in *fieldnames*, the optional *extrasaction* parameter indicates what action to take. If it is set to `'raise'`, the default value, a `ValueError` is raised. If it is set to `'ignore'`, extra values in the dictionary are ignored. Any other optional or keyword arguments are passed to the underlying `writer` instance.

Note that unlike the `DictReader` class, the *fieldnames* parameter of the `DictWriter` class is not optional.

A short usage example:

بریم مثالش رو با هم ببینیم :

```
python_projects >  challeng.py >  csvfile
```

```
1 import csv
2 import statistics
3 with open('python_projects\example.csv') as csvfile:
4     reader=csv.DictReader(csvfile)
5     next(reader)
6     for rows in reader:
7         print(rows)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/python_projects/challeng.py
```

```
{'name': 'mohammad', 'age': '20', 'city': 'eilam'}
{'name': 'maryanm', 'age': '21', 'city': 'sistan'}
{'name': 'zahra', 'age': '30', 'city': 'tehran'}
{'name': 'kazem', 'age': '45', 'city': 'semnan'}
{'name': 'siroos', 'age': '18', 'city': 'esfahan'}
{'name': 'elham', 'age': '14', 'city': 'tehran'}
{'name': 'fateme', 'age': '65', 'city': 'tehran'}
PS C:\Users\User\Desktop\Python> 
```

کلا خیلی باحاله با ستون ها و سطر ها بازی کردن .

فردا یک مینی پروژه ای که یکی از افراد لینکدین باهش برخورد کرد در دنیای واقعی رو با هم حل میکنیم ، عنوانش رو میگم بهتون و خودتون روش فکر کنید . چالش خوبی میشه براتون :

در یک مرکز بهداشتی، لیست 450 csv نفره داریم که شامل نام و نام خانوادگی، سن، کد ملی و وضعیت واکسن افراد است.

افرادی که واکسن زده اند، وضعیت آنها با Vaccinated مشخص شده و افرادی که واکسن نزده اند با Not vaccinated معین شده اند.

حال ما میخواهیم افراد واکسینه نشده را پیدا کنیم و اطلاعات کامل آنها را در فایل txt ذخیره کنیم.

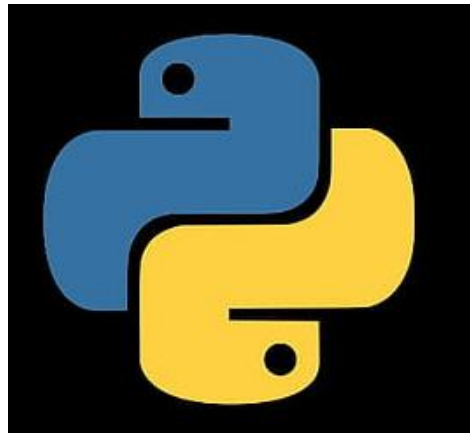
یک تمرین خیلی باحال از هر چیزی که یاد گرفتیم.

فردا با هم حلش میکنیم!

ممنون که دنبال میکنید  

کاری از : علی معینیان

21 day challenge with python



Part 17 out of 21

**A Python challenge that
includes almost everything we
read**

سلااااااااام ✨

خب قرار شد امروز با هم یک تمرین خیلی باحال انجام بدیم.

این تمرین رو من 2 یا 3 روز پیش توی لینکدین دیدم که یکی از دوستان باهاش درگیر بود و خیلی جالب بود برام و پیش خودم گفتم حتما یکی از قسمت های این چالش رو به این تمرین اختصاص بدم.

بزارید عین پست ایشون رو بزارم و ببینیم که مشکل چی بوده ؟



Mohammad Mohammadi • 2nd

Web Developer

1h • 🌐

یکی از بهترین لذت های برنامه نویسی حل مشکلات دیگرانه 😊

این کد شاید ۲۰ خط نشه ولی کار چندین روز از کادر درمان کم میکنه

چون مامانم جزو کادر درمان هست معمولا باید لیست افراد رو از سامانه بگیره بفرسته به مرکز

(در صورتی که خودشون هم دسترسی دارن)

(تو اون سامانه یه دکمه نداشتن واسه اکسترکت لیست 😞👤)

و امروز لیست افراد واکسن نزده رو مادر من باید میومد ۱۲۰۰ نفر رو اسم و فامیل و کد

ملی و ... رو تک تک کپی میکرد تو اکسل ولی با ۲۰ خط کد اون کار که احتمالا چند روز

زمان میبرد و به شدت حوصله بر بود تو ۱۰ دقیقه انجام میشه

بعضی وقتا برنامه نویسی واقعا لذت بخشه 😎

#موقت

اما برای اینکه یکم تفاوت داشته باشه چند تا قسمتش رو من دست کاری میکنم که یکم بیشتر کار بیره :

1 – قطعا دیتا های ما 1200 نفر نمیتونه باشه و با 20 نفر این کار رو انجام میدیم

2 – نتایج نهایی رو در فایل txt نشان خواهیم داد

3 – مشخصات موجود در فایل CSV :

✓ نام و نام خانوادگی

✓ سن

✓ شماره ملی

✓ وضعیت واکسن : Vaccinated – Not vaccinated

مورد آخر اینکه من تا به حال این برنامه رو ننوشتم و اولین بار هست دارم روش فکر میکنم، پس مرحله به مرحله فکر هام رو مینویسم براتون .

برو بریم! 😎

اول از همه یک فایل پایتون میخواهیم :

challeng.py ×

python_projects > challeng.py > ...

```
1
2
3
4
5
6
```

در مرحله ی بعدی یک سری فایل CSV میخواهیم با اطلاعات فرضی شامل 20 نفر :

example.csv ×

python_projects > example.csv

```
1  FullName,age,ID,Vaccinated situation
2  Ali Moeinian,20,123,Vaccinated
3  Ali kazemi,22,111,Not Vaccinated
4  maryam esmaeili,45,236,Vaccinated
5  abbas hosseini,36,693,Not Vaccinated
6  mojtaba vahdati,30,333,Not Vaccinated
7  fateme ghasemi,63,313,Vaccinated
8  shima tabatabaei,51,548,Not Vaccinated
9  paria rostami,20,789,Not Vaccinated
10 reyhane kazemi,29,952,Vaccinated
11 Ali fooladi,31,348,Not Vaccinated
12 kazem fooladian,78,693,Vaccinated
13 laleh hadadi,90,117,Not Vaccinated
14 hossein Moeinian,41,256,Not Vaccinated
15 mehrdad moradi,44,300,Not Vaccinated
16 atefe vaezi,25,110,Vaccinated
17 bahar shirooei,29,220,Not Vaccinated
18 shakila dadkhah,12,852,Vaccinated
19 majid karami,33,100,Not Vaccinated
20
```

خب اول از همه اینکه شماره ملی هر کس یک کد 3 رقمی هست که مشخصه الکیه و دوم اینکه چون اطلاعات دقیقی از فایل اصلی نداشتیم، این فایل رو به طور فرضی طراحی کردم.

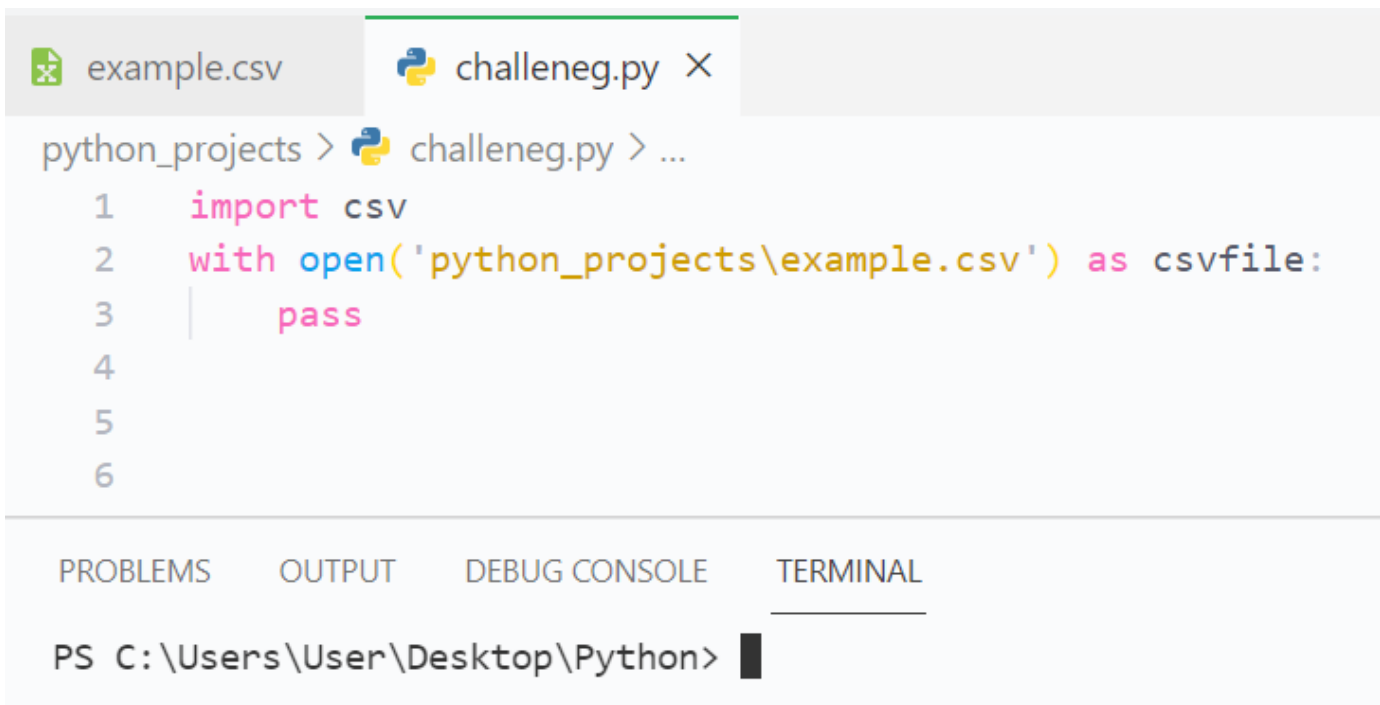
مهم برنامه ای خواهد بود که مینویسیم نه این اطلاعات 😊

و در آخر هم نیاز دارم یک فایل `txt` بسازم تا در آخر جواب ها رو داخلش به چاپ برسونیم.

راستی میتونی این کار رو هم نکنی و از `mode : a+` استفاده کنی چون خودش میتونه فایل دلخواه رو برات بسازه.

بریم سراغ برنامه :

1 – اول از همه من پیش خودم میگم خب باید فایل مشخصات رو وارد برنامه کنم و همچنین ماژول `CSV` رو هم فراموش نمیکنم.



```
example.csv | challeng.py X
python_projects > challeng.py > ...
1 import csv
2 with open('python_projects\example.csv') as csvfile:
3     pass
4
5
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> █
```

2 – الان باید پیام کل فایل رو بخونم و ببینم آیا ایرادی هست یا نه؟
ترجیح میدم فایل رو مثل دیکشنری بخونم و از متد دیکشنری استفاده
کنم.

```
example.csv  challeng.py X
python_projects > challeng.py > csvfile
1  import csv
2  with open('python_projects\example.csv') as csvfile:
3      reader = csv.reader(csvfile)
4      for rows in reader:
5          print(rows)
6

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

challeng.py
['FullName', 'age', 'ID', 'Vaccinated situation']
['Ali Moeinian', '20', '123', 'Vaccinated']
['Ali kazemi', '22', '111', 'Not Vaccinated']
['maryam esmaeili', '45', '236', 'Vaccinated']
['abbas hosseini', '36', '693', 'Not Vaccinated']
['mojtaba vahdati', '30', '333', 'Not Vaccinated']
['fateme ghasemi', '63', '313', 'Vaccinated']
['shima tabatabaei', '51', '548', 'Not Vaccinated']
['paria rostami', '20', '789', 'Not Vaccinated']
['reyhane kazemi', '29', '952', 'Vaccinated']
['Ali fooladi', '31', '348', 'Not Vaccinated']
['kazem fooladian', '78', '693', 'Vaccinated']
['laleh hadadi', '90', '117', 'Not Vaccinated']
['hossein Moeinian', '41', '256', 'Not Vaccinated']
['mehrdad moradi', '44', '300', 'Not Vaccinated']
['atefe vaezi', '25', '110', 'Vaccinated']
['bahar shirooei', '29', '220', 'Not Vaccinated']
['shakila dadkhah', '12', '852', 'Vaccinated']
['majid karami', '33', '100', 'Not Vaccinated']
PS C:\Users\User\Desktop\Python>
```

دو تا ایراد داریم اینجا :

- اون تیتر های اول هر ستون هست که نباید باشه
- به صورت دیکشنری نیست

بریم درستش کنیم.

```
example.csv  challeng.py X
python_projects > challeng.py > csvfile
1 import csv
2 with open('python_projects\example.csv') as csvfile:
3     reader = csv.DictReader(csvfile)
4     for rows in reader:
5         print(rows)
6
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/challeng.py
{'FullName': 'Ali Moeinian', 'age': '20', 'ID': '123', 'Vaccinated situation': 'Vaccinated'}
{'FullName': 'Ali kazemi', 'age': '22', 'ID': '111', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'maryam esmaeili', 'age': '45', 'ID': '236', 'Vaccinated situation': 'Vaccinated'}
{'FullName': 'abbas hosseini', 'age': '36', 'ID': '693', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'mojtaba vahdati', 'age': '30', 'ID': '333', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'fateme ghasemi', 'age': '63', 'ID': '313', 'Vaccinated situation': 'Vaccinated'}
{'FullName': 'shima tabatabaei', 'age': '51', 'ID': '548', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'paria rostami', 'age': '20', 'ID': '789', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'reyhane kazemi', 'age': '29', 'ID': '952', 'Vaccinated situation': 'Vaccinated'}
{'FullName': 'Ali fooladi', 'age': '31', 'ID': '348', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'kazem fooladian', 'age': '78', 'ID': '693', 'Vaccinated situation': 'Vaccinated'}
{'FullName': 'laleh hadadi', 'age': '90', 'ID': '117', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'hossein Moeinian', 'age': '41', 'ID': '256', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'mehrdad moradi', 'age': '44', 'ID': '300', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'atefe vaezi', 'age': '25', 'ID': '110', 'Vaccinated situation': 'Vaccinated'}
{'FullName': 'bahar shirooei', 'age': '29', 'ID': '220', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'shakila dadkhah', 'age': '12', 'ID': '852', 'Vaccinated situation': 'Vaccinated'}
{'FullName': 'majid karami', 'age': '33', 'ID': '100', 'Vaccinated situation': 'Not Vaccinated'}
PS C:\Users\User\Desktop\Python>
```

خیلی جالب شد 😊 اگر از متد DictReader استفاده کنی، خودش به طور اتوماتیک اون سر تیتر ها رو دیگه توی خروجی نشون نمیده و علتش هم اینه که سر تیتر ها رو به طور اختصاصی به هر کدوم از دیتا ها اختصاص داده.

3 – خب حالا باید به قسمت شرایط واکسن دسترسی داشته باشم.

```
python_projects > challengeg.py > [🔍] csvfile
1  import csv
2  with open('python_projects\example.csv') as csvfile:
3      reader = csv.DictReader(csvfile)
4      for rows in reader:
5          print(rows['Vaccinated situation'])
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
challengeg.py
Vaccinated
Not Vaccinated
Vaccinated
Not Vaccinated
Not Vaccinated
Vaccinated
Not Vaccinated
Not Vaccinated
Vaccinated
Not Vaccinated
Vaccinated
Not Vaccinated
Not Vaccinated
Not Vaccinated
Vaccinated
Not Vaccinated
Vaccinated
Not Vaccinated
PS C:\Users\User\Desktop\Python> █
```

به همین سادگی 😊 😁

4 – الان نیاز داریم به برنامه بفهمونم اگر وضعیت واکسن هر فرد "واکسن نزده" بود، بیا و از بقیه جداش کن.

```
example.csv  challengen.py ×
python_projects > challengen.py > csvfile
1  import csv
2  with open('python_projects\example.csv') as csvfile:
3      reader = csv.DictReader(csvfile)
4      for rows in reader:
5          if rows['Vaccinated situation'] == 'Not Vaccinated':
6              print(rows)
7
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:
challengen.py
{'FullName': 'Ali kazemi', 'age': '22', 'ID': '111', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'abbas hosseini', 'age': '36', 'ID': '693', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'mojtaba vahdati', 'age': '30', 'ID': '333', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'shima tabatabaei', 'age': '51', 'ID': '548', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'paria rostami', 'age': '20', 'ID': '789', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'Ali fooladi', 'age': '31', 'ID': '348', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'laleh hadadi', 'age': '90', 'ID': '117', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'hossein Moeinian', 'age': '41', 'ID': '256', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'mehrdad moradi', 'age': '44', 'ID': '300', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'bahar shirooei', 'age': '29', 'ID': '220', 'Vaccinated situation': 'Not Vaccinated'}
{'FullName': 'majid karami', 'age': '33', 'ID': '100', 'Vaccinated situation': 'Not Vaccinated'}
PS C:\Users\User\Desktop\Python> █
```

خیلی جالب شد واقعا !

به همین سادگی از توی اون لیست تونستم افرادی که واکسن نشده اند رو بیرون بکشیم.

5 – نیاز داریم که این اطلاعات رو بریزیم توی یک فایل txt که اسمش رو هم میزاد : Not_Vaccinated.txt

```
example.csv  challengen.py ×  not vaccinated.txt

python_projects > challengen.py > ...
1  import csv
2  with open('not vaccinated.txt','a+') as Not_Vaccinated_people:
3      with open('python_projects\example.csv') as csvfile:
4          reader = csv.DictReader(csvfile)
5          Not_Vaccinated=[]
6          for rows in reader:
7              if rows['Vaccinated situation'] == 'Not Vaccinated':
8                  print(rows)
9
```

باید موارد خارج شده را وارد فایل متنی کنیم :

```
example.csv  challengen.py ×  not vaccinated.txt

python_projects > challengen.py > Not_Vaccinated_people
1  import csv
2  with open('not vaccinated.txt','a+') as Not_Vaccinated_people:
3      with open('python_projects\example.csv') as csvfile:
4          reader = csv.DictReader(csvfile)
5          Not_Vaccinated=[]
6          for rows in reader:
7              if rows['Vaccinated situation'] == 'Not Vaccinated':
8                  Not_Vaccinated_people.write(rows)
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.
Traceback (most recent call last):
  File "c:\Users\User\Desktop\Python\python_projects\challengen.py", line 8, in <module>
    Not_Vaccinated_people.write(rows)
TypeError: write() argument must be str, not dict
PS C:\Users\User\Desktop\Python> 
```

باز هم ارور ! ارور رو بخونیم ببینیم چی میگه :

میگه که ورودی write() نمیتونه دیکشنری باشه 😞

خب باشه . میریم و نوع ورودی رو تغییر میدیم :

```
example.csv  challeng.py ×  not vaccinated.txt

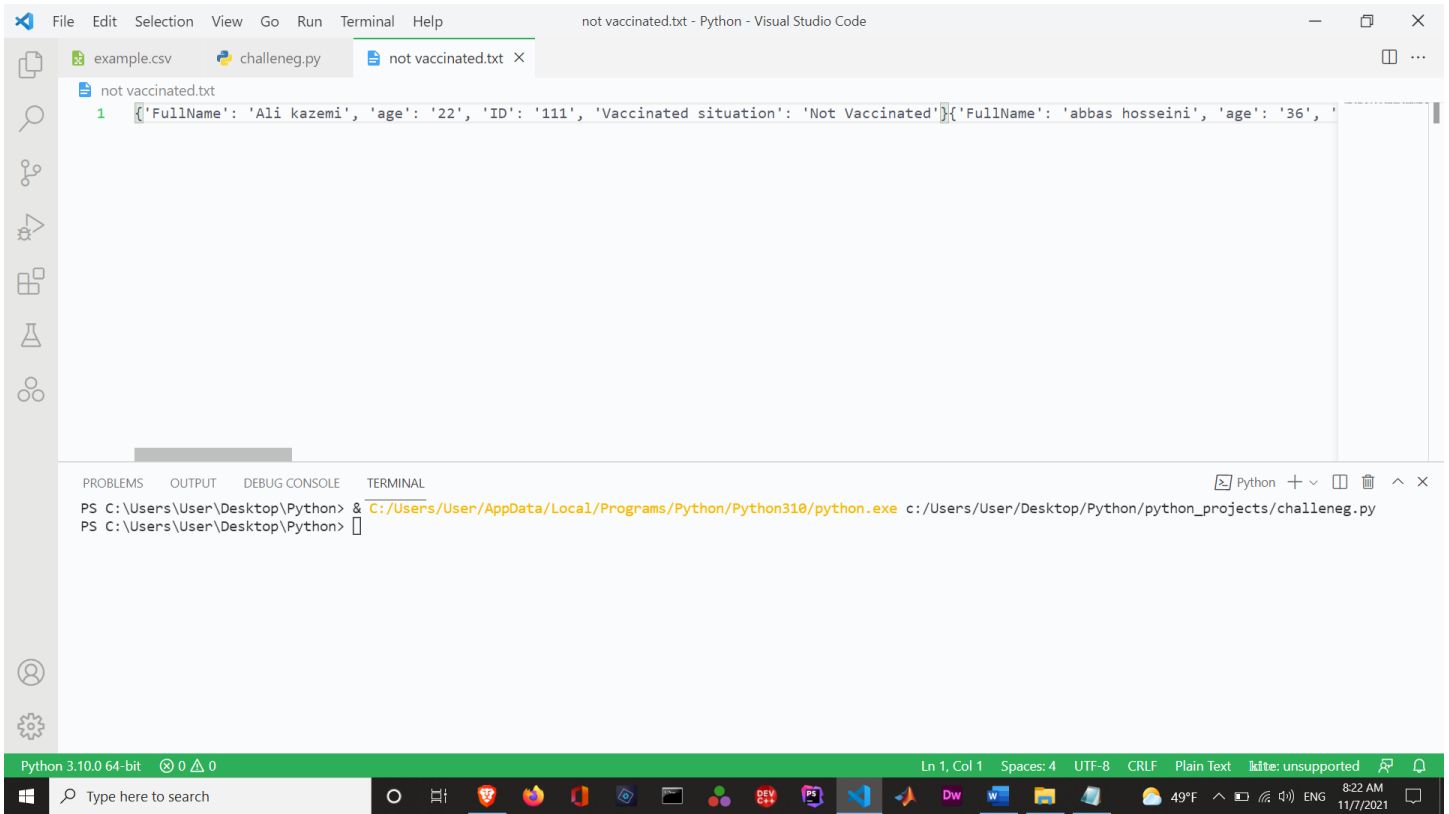
python_projects > challeng.py > Not_Vaccinated_people
1  import csv
2  with open('not vaccinated.txt','a+') as Not_Vaccinated_people:
3      with open('python_projects\example.csv') as csvfile:
4          reader = csv.DictReader(csvfile)
5          Not_Vaccinated=[]
6          for rows in reader:
7              if rows['Vaccinated situation'] == 'Not Vaccinated':
8                  Not_Vaccinated_people.write(str(rows))
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python
PS C:\Users\User\Desktop\Python> █
```

خب خب خب ! درست شد و هیچ اروری نداد 🙌

بریم توی فایل متنی و ببینیم ایا مواردی که میخوایم منتقل شدند یا نه؟



The screenshot shows the Visual Studio Code interface. The editor window displays a file named 'not vaccinated.txt' with the following content:

```
1 [{"FullName": 'Ali kazemi', 'age': '22', 'ID': '111', 'Vaccinated situation': 'Not Vaccinated'}]
```

The terminal window shows the command prompt with the following output:

```
PS C:\Users\User\Desktop\Python> C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop/Python/python_projects/challengen.py  
PS C:\Users\User\Desktop\Python>
```

The status bar at the bottom indicates the file is a Python 3.10.0 64-bit text file with 1 line and 1 column. The system tray shows the date and time as 8:22 AM on 11/7/2021.

خب این فایل ها منتقل شد، اما اخه اینکه خیلی زشت شد، نه ؟؟؟
من دوست داشتم خروجی دقیقا اینطوری بشه :

```
{'FullName': 'Ali kazemi', 'age': '22', 'ID': '111', 'Vaccinated situation': 'Not Vaccinated'}  
{'FullName': 'abbas hosseini', 'age': '36', 'ID': '693', 'Vaccinated situation': 'Not Vaccinated'}  
{'FullName': 'mojtaba vahdati', 'age': '30', 'ID': '333', 'Vaccinated situation': 'Not Vaccinated'}  
{'FullName': 'shima tabatabaei', 'age': '51', 'ID': '548', 'Vaccinated situation': 'Not Vaccinated'}  
{'FullName': 'paria rostami', 'age': '20', 'ID': '789', 'Vaccinated situation': 'Not Vaccinated'}  
{'FullName': 'Ali fooladi', 'age': '31', 'ID': '348', 'Vaccinated situation': 'Not Vaccinated'}  
{'FullName': 'laleh hadadi', 'age': '90', 'ID': '117', 'Vaccinated situation': 'Not Vaccinated'}  
{'FullName': 'hossein Moeinian', 'age': '41', 'ID': '256', 'Vaccinated situation': 'Not Vaccinated'}  
{'FullName': 'mehrddad moradi', 'age': '44', 'ID': '300', 'Vaccinated situation': 'Not Vaccinated'}  
{'FullName': 'bahar shirooei', 'age': '29', 'ID': '220', 'Vaccinated situation': 'Not Vaccinated'}  
{'FullName': 'majid karami', 'age': '33', 'ID': '100', 'Vaccinated situation': 'Not Vaccinated'}  
PS C:\Users\User\Desktop\Python>
```

بریم درستش کنیم :

```
example.csv  challeng.py ×  not vaccinated.txt

python_projects > challeng.py > Not_Vaccinated_people
1  import csv
2  with open('not vaccinated.txt', 'w') as Not_Vaccinated_people:
3      with open('python_projects\example.csv') as csvfile:
4          reader = csv.DictReader(csvfile)
5          Not_Vaccinated=[]
6          for rows in reader:
7              if rows['Vaccinated situation'] == 'Not Vaccinated':
8                  Not_Vaccinated_people.write(str(rows) + '\n')
9
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop/Python/python_projects/challeng.py
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop/Python/python_projects/challeng.py
PS C:\Users\User\Desktop\Python> █
```

بریم نتیجه رو ببینیم :

```
not vaccinated.txt
1  {'FullName': 'Ali kazemi', 'age': '22', 'ID': '111', 'Vaccinated situation': 'Not Vaccinated'}
2  {'FullName': 'abbas hosseini', 'age': '36', 'ID': '693', 'Vaccinated situation': 'Not Vaccinated'}
3  {'FullName': 'mojtaba vahdati', 'age': '30', 'ID': '333', 'Vaccinated situation': 'Not Vaccinated'}
4  {'FullName': 'shima tabatabaei', 'age': '51', 'ID': '548', 'Vaccinated situation': 'Not Vaccinated'}
5  {'FullName': 'paria rostami', 'age': '20', 'ID': '789', 'Vaccinated situation': 'Not Vaccinated'}
6  {'FullName': 'Ali fooladi', 'age': '31', 'ID': '348', 'Vaccinated situation': 'Not Vaccinated'}
7  {'FullName': 'laleh hadadi', 'age': '90', 'ID': '117', 'Vaccinated situation': 'Not Vaccinated'}
8  {'FullName': 'hossein Moeinian', 'age': '41', 'ID': '256', 'Vaccinated situation': 'Not Vaccinated'}
9  {'FullName': 'mehrdad moradi', 'age': '44', 'ID': '300', 'Vaccinated situation': 'Not Vaccinated'}
10 {'FullName': 'bahar shirooei', 'age': '29', 'ID': '220', 'Vaccinated situation': 'Not Vaccinated'}
11 {'FullName': 'majid karami', 'age': '33', 'ID': '100', 'Vaccinated situation': 'Not Vaccinated'}
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop/Python/python_projects/challeng.py
PS C:\Users\User\Desktop\Python> & C:/Users/User/AppData/Local/Programs/Python/Python310/python.exe c:/Users/User/Desktop/Python/python_projects/challeng.py
PS C:\Users\User\Desktop\Python> █
```

Python 3.10.0 64-bit 0 0 0 Ln 12, Col 1 Spaces: 4 UTF-8 CRLF Plain Text kate: unsupported 8:25 AM 11/7/2021

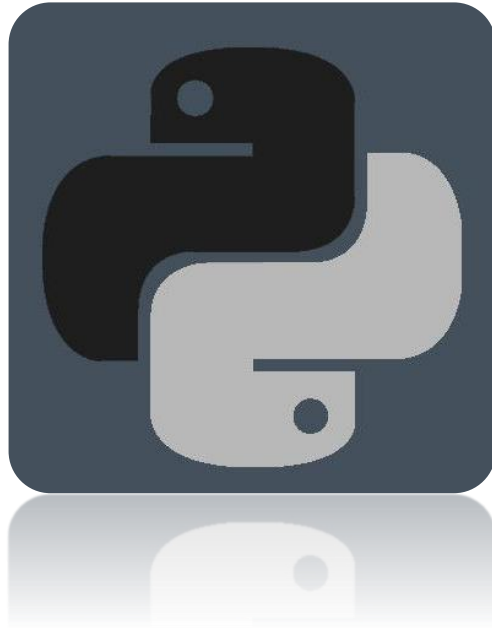
این هم از یک مینی تسک خیلی باحال که من خودم به شخصه خیلییی حال کردم باش و نکته ی جالب تر برای من این بود که واقعا از قبل روش فکر نکرده بودم و دقیقا هم زمان با همین چالش نوشتنش و بهترین قسمتش هم مرحله ای نوشتنش بود که دقیقا افکارم و چیز هایی که توی ذهنم میومد رو مرحله به مرحله نوشتم.

امیدوارم که همه چی خوب بوده باشه و خوشحال میشم اگر ایرادی در کدم هست رو بهم حتما گوش زد کنید.

ممنونم که من رو دنبال میکنید 🎁🌸

کاری از : علی معینیان

21 day challenge with python



Part 18 out of 21

OOP, Good or Bad?

OOP: Object Oriented Programming

سلام!!!!!! 😊

رسیدیم به آخرین بحث دوره ی خودمون !

شی گرای و برنامه نویسی شی گرا 😞

اما توی این پارت سراغ شی گرای نمیرم .

همیشه یه دعوا ی قدیمی بوده بین OOP و Functional Programming

که قراره حسابی توی این پارت بهش فکر کنیم و ببینیم چرا این

اختلاف نظر وجود داشته ؟

اصلا شی گرای از کجا اومد ؟

چرا اومد ؟

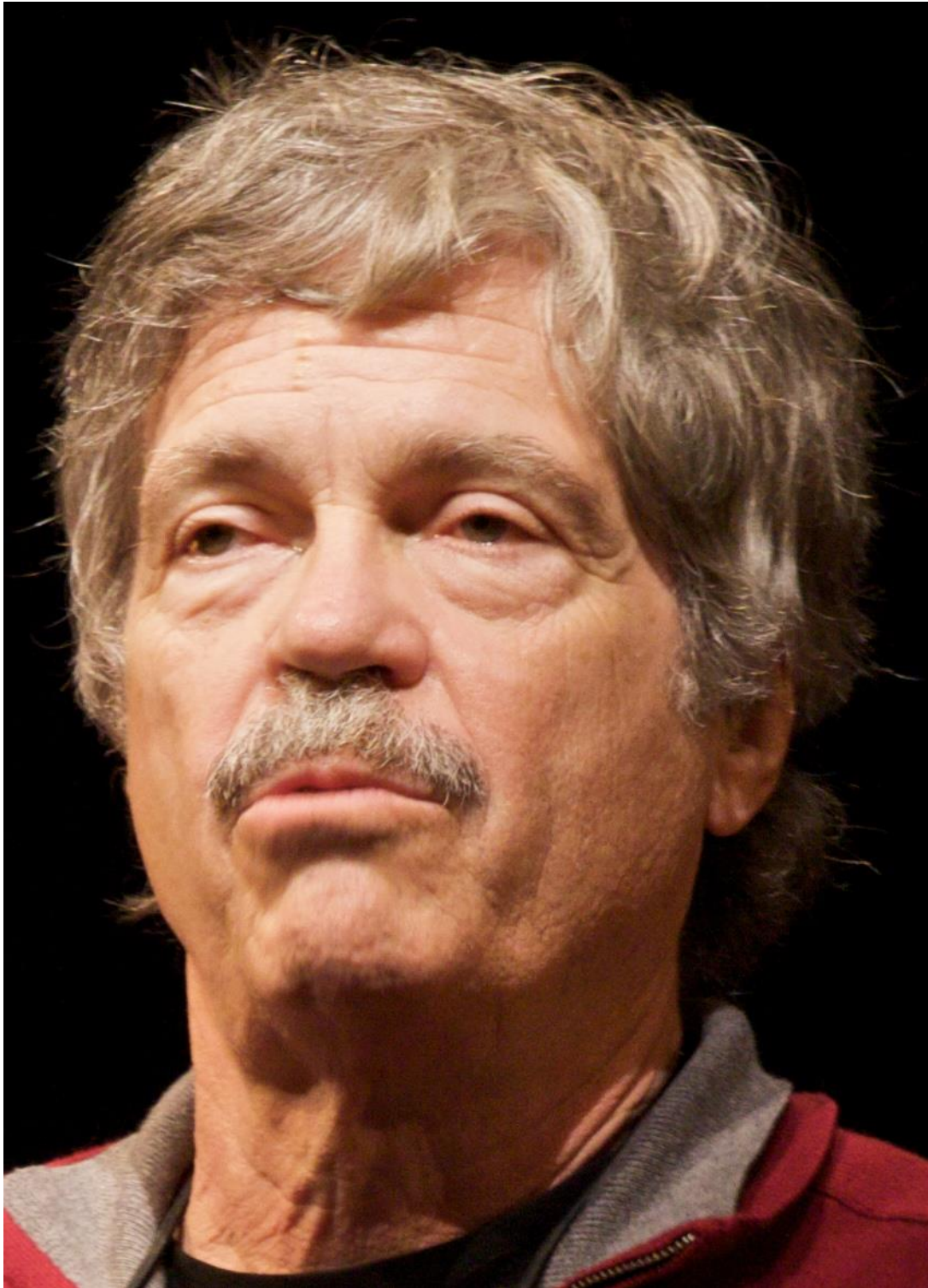
به چه علتی فراگیر شد ؟



**Functional vs.
Procedural Programming
vs. OOP**

اول از همه بریم ببینیم کدوم بنده خدایی برای اولین بار مفهوم شی
گرایی رو وارد برنامه نویسی کرد . طبق تحقیقات من و با یک سرچ
ساده میشه فهمید کی برای اولین بار اومده این کار رو کرده :

Alan Kay



بله 😞 ایشون برای اولین بار ظاهرا اومدن و مفهوم شی گرای را وارد برنامه نویسی کردند.

طبق نوشته ها و صحبت های آلن کی، ایشون در دبیرستان که بوده، در سال های 1966 برای اولین بار ایده ی شی گرای زده به سرش ! و خیلی ها اصلا نمیفهمیدن که منظور آلن از شی گرای چیه ؟ به همین علت خیلی ها از خودشون شروع کردن به تفسیر کردن که منظور آلن چی بوده ؟

که خود این شخص واقعا کلافه میشه و میاد و این پیام رو منتشر میکنه :

"I'm sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea. The big idea is messaging."

~ Alan Kay

توی این پیام میاد و میگه من متاسفم که از اصطلاح object استفاده کردم، چون باعث میشه مردم بر روی ایده های کوچک تر فکر کنند. منظور من در اصل پیام رسانی بوده است !

اما حالا یک مساله ی دیگر پیش اومد ! پیام رسانی؟؟ 😞

دولوپر ها هم مثل من و شما گفتند برادر، حالت خوبه ؟

پیام رسانی چیه؟؟؟ حالا مشکل ما فهم این شی گرایه بود، الان باید
بفهمیم منظورت از پیام رسانی چیه!

که آلن در سال 2003 میاد و هدفش و تفکرش رو در ارتباط با برنامه
نویسی شی گرا، به طور واضح بیان میکنه:

*“OOP to me means only messaging, local retention and protection and hiding
of state-process, and extreme late-binding of all things.”*

~ Alan Kay



بریم ببینیم اصلا چی میگه این آقای آلن کی:

In other words, according to Alan Kay, the essential ingredients of OOP are:

- Message passing
- Encapsulation
- Dynamic binding

خیلی جالب شد!

کسی که برای اولین بار مفهوم شی گزایی را وارد دنیای برنامه نویسی کرده، اصلاً حرفی از زیر کلاس، پولیمورفیسم و وراثت نزده! و در اصل اون ها رو ضروری نمیدونسته برای بحث OOP! باز هم بریم به طور دقیق تر ببینیم آقای آلن چی میگه:

The Essence of OOP

The combination of message passing and encapsulation serve some important purposes:

- **Avoiding shared mutable state** by encapsulating state and isolating other objects from local state changes. The only way to affect another object's state is to ask (not command) that object to change it by sending a message. State changes are controlled at a local, cellular level rather than exposed to shared access.
- **Decoupling** objects from each other — the message sender is only loosely coupled to the message receiver, through the messaging API.
- **Adaptability and resilience to changes** at runtime via late binding. Runtime adaptability provides many great benefits that Alan Kay considered essential to OOP.

و باز هم مشکلات شروع شد !

“[...] the whole point of OOP is not to have to worry about what is inside an object. Objects made on different machines and with different languages should be able to talk to each other [...]” ~ Alan Kay

بعد از مدتی، آقای آلن کی، یکم داستان رو جالب تر کرد و گفت :

“My math background made me realize that each object could have several algebras associated with it, and there could be families of these, and that these would be very very useful.”

~ Alan Kay

خب یکم الان همه گیج ایم 😞 😞
چی شد اصلا ؟؟؟؟؟

بزارید یکم بیشتر بحث کنیم :

با گذشت زمان، هر چیزی پیشرفت میکنه و تغییر میکنه.

از تعاریف اولیه ی oop گرفته تا هر ان چیزی که داخل مباحث شی گزایی پیش میاد !

پس خیلی گیر نمیدیم که آلن اول چی گفته و الان برداشت ما از شی گزایی چیه .

Object-oriented programming

From Wikipedia, the free encyclopedia

"Object-oriented" redirects here. For other meanings of object-oriented, see [Object-orientation](#).

"Object-oriented programming language" redirects here. For a list of object-oriented programming languages, see [List of object-oriented programming languages](#).

Object-oriented programming (OOP) is a [programming paradigm](#) based on the concept of "objects", which can contain [data](#) and code: data in the form of [fields](#) (often known as [attributes](#) or [properties](#)), and code, in the form of procedures (often known as [methods](#)).



A feature of objects is that an object's own procedures can access and often modify the data fields of itself (objects have a notion of [this](#) or [self](#)). In OOP, computer programs are designed by making them out of objects that interact with one another.^{[1][2]} OOP languages are diverse, but the most popular ones are [class-based](#), meaning that objects are [instances](#) of [classes](#), which also determine their [types](#).

Many of the most widely used programming languages (such as C++, Java, Python, etc.) are [multi-paradigm](#) and they support object-oriented programming to a greater or lesser degree, typically in combination with [imperative](#), [procedural programming](#). Significant object-oriented languages include: [Java](#), [C++](#), [C#](#), [Python](#), [R](#), [PHP](#), [Visual Basic.NET](#), [JavaScript](#), [Ruby](#), [Perl](#), [SIMSCRIPT](#), [Object Pascal](#), [Objective-C](#), [Dart](#), [Swift](#), [Scala](#), [Kotlin](#), [Common Lisp](#), [MATLAB](#), and [Smalltalk](#).

جدای از بحثمون بگم که اگر واقعا به فهم عمیق شی گزایی علاقه دارید حتما ویکی پدیا رو مطالعه کنید . خیل کامل و جامع توضیح داده شی گزایی و همه ی مفاهیم داخلش رو.

بریم فلسفه ی شی گزایی رو یکم بخونیم، ببینیم اصلا چرا هست ؟
مگه برنامه نویسی فانکشنال مشکلش چی بود ؟

اومدن و گفتن که برنامه نویسی باید بیاد و به دنیای واقعی ما نزدیک تر بشه! یعنی چی ؟

یعنی ما توی دنیای واقعی خودمون اشیا رو داریم و هر شی یکسری ویژگی داره و اشیا مختلف ممکنه ویژگی های یکسانی هم داشته باشند که میشه نقطه اشتراک اونها  

خب زبان C رو همه باهوش اشنایی دارند !
و ما همه میدونیم فرق اصلی C با C++ چیه !
اما جالبه بدونید قبل از اینکه شی گزایی به سی پلاس پلاس اضافه
بشه، خود آقای آلن کی اومد و small talk رو ساخت 😊
اما small talk چیه ؟

Smalltalk

From Wikipedia, the free encyclopedia

This article is about the programming language. For other uses, see [Small talk \(disambiguation\)](#).

Smalltalk is an [object-oriented](#), [dynamically typed](#) [reflective programming language](#). Smalltalk was created as the language underpinning the "new world" of computing exemplified by "human-computer symbiosis".^[4] It was designed and created in part for [educational](#) use, specifically for [constructionist learning](#), at the Learning Research Group (LRG) of [Xerox PARC](#) by [Alan Kay](#), [Dan Ingalls](#), [Adele Goldberg](#), [Ted Kaehler](#), [Diana Merry](#), [Scott Wallace](#), and others during the 1970s.

The language was first generally released as Smalltalk-80. Smalltalk-like languages are in active development and have gathered loyal communities of users around them. ANSI Smalltalk was ratified in 1998 and represents the standard version of Smalltalk.^[5]

Smalltalk took second place for "most loved programming language" in the [Stack Overflow](#) Developer Survey in 2017,^[6] but it was not among the 26 most loved programming languages of the 2018 survey.^[7]

این زبان برنامه نویسی، توسط خود آلن کی و یکسری محق دیگه ساخته شد و جایی خوندم که آلن کی معتقد به جز small talk هیچ زبانی نتونسته شی گزایی رو اونطور که توی ذهن آلن کی بوده پیاده سازی کنه (معتبر نیست)

یک عالمه فیلم از آلن کی روی یوتوب هست که خوبه برید و ببینید و به حرفاش گوش بدید.

اما بزارید یکم بحث رو جذاب تر کنیم 😊 😞

اولین زبانی که از OOP به طور رسمی استفاده کرده چی بود؟

زبانی بود به نام Simula در سال 1967 !

خب بریم ببینیم آلن چی گفته و نظرش چیه :

- I didn't like the way Simula I or Simula 67 did inheritance (though I thought Nygaard and Dahl were just tremendous thinkers and designers). So I decided to leave out inheritance as a built-in feature until I understood it better.

پس بریم سراغ خود small talk و ببینیم آلن کی خودش توی این زبان برنامه نویسی چه کار خاصی برای برنامه نویسی شی گرا کرده؟؟

کلا این آقا با هر شی گرایی ای در زبان های دیگه مخالف بوده و گفته این مفاهیم، اون چیزی نیست که من میخوام (حرکت عجیبه)

Object Oriented Programming with Smalltalk

by Bryce Hendrix

داکیومنت اصلی اسمال تاک در ارتباط با شی گرای و مفاهیم اون تقریبا 200 صفحه است، قسمت های جالبش رو براتون میزارم :

Lecture 2: Classes and Instances

- **Class**

- A template for objects that share common characteristics.
- Includes an object's state variable and methods

- Ex: Vehicle class

```
Vehicle
Velocity
Location
Color
Weight
Start( )
Stop( )
Accelerate( )
```

- **Instance**

- A particular occurrence of an object is defined by a class
 - Classes are sometimes thought of as factories. If we had an automobile factory, the class would be the factory and the automobiles would be the instances of that factory.
- Each instance has its own values for instance variables
 - Each automobile has its own engine, hood, doors, etc.
- All instances of a class share the same method
 - Methods are the functions that are applicable to all instances of a class.
 - The method `accelerate` is applicable to all automobiles
 - Ex: A road contains many instances of vehicles, all different colors, going different speeds, starting, stopping, accelerating, etc
 - Ex: cars on a road. It is important to note that `car1` and `car3` are not the same object, but are the both instances of the class `Car`. `car1` and `car3` are equivalent, but not equal. Equality implies they are the same object.

```
aRoad = Road new.
car1 = Car new withColor: red withSpeed 30.
car2 = Car new withColor: blue withSpeed 45.
car3 = Car new withColor: red withSpeed 30.
```

- **Class Hierarchy**

- Allows sharing of state and behavior
 - Subclasses are able to use the methods and variables of the parent classes.
- Each class refines / specializes its ancestors
- Child can add new state information
 - A Land Vehicle adds the state information regarding the number of axles
- Child can add, extend or override parent behavior
 - All Vehicles can be driven, but all types of vehicles require different sets of methods to drive
- Superclass is the parent and subclass is a child
- Abstract class holds common behavior & characteristics, concrete classes contain complete characterization of actual objects
 - In the Vehicle example, Vehicle is the Superclass, and Sailboat, Speed boar, Jet, Helicopter, Car and Truck are the concrete classes.

Lecture 3: Messages, Methods, and Programming in Smalltalk

- **Messages**
 - A message specifies what behavior an object is to perform
 - Only way to communicate with an object
 - Implementation is left up to the receiver object
 - Ex: Ask the baker to bake a cake. We don't care how he does it.
 - Ex: `baker bakeCake.`
 - State Information can only be accessed via messages
 - Ex: I want to know how old you are (one of your state variables), so I ask you. I don't care how you compute your age, all I care about is the answer.
 - Ex: `baker age.`
 - The receiver object always returns a result (object).
 - A lot of the time a receiver is modified and it doesn't make sense to return something, so the argument is returned
 - Ex: `#(a b c) at: 3 put: #d` returns `#d`
- **Methods**
 - A method specifies how a receiver object performs a behavior.
 - Executed in response to a message
 - Must have access to data (must be passed, or contained in object)
 - If there is no access passed or contained in the object, what can be done?
 - Needs detailed knowledge of data
 - Can manipulate data directly
 - Can modify instance variables of the object receiving the message
 - Ex: `#(a b c) at: 3 put: #d.` modifies the collection which is the instance variable
 - Returns an object as a result of its execution
 - Since a method is executed in response to a message, and we have already said all messages return an object, it should only make sense that the method returns an object as the result of its execution
 - Has same name as the message name
 - Ex: `#(a b c) size.` `size` is the message called by the receiver, and the `size` method is the method in class `Array` to be executed
 - Visual Works does no type checking on arguments, although the types should be type-compatible.
 - Method returns the receiver object by default, unless explicitly returned
 - Ex: Bob is asked to bake a cake. Bob's 'bake' method explicitly says to return a cake, rather than returning himself to the requester.
 - Ex: the `at:` method of class `Interval`
 - Explicitly returns a temp variable

```
at: anInteger
    "Answer the number at index position
    anInteger in the receiver interval."
    | answer |
    anInteger > 0
        ifTrue: [
            answer := beginning + (increment *(anInteger
                - 1)).
            (increment < 0
                and: [answer between: end and:
                    beginning])
                ifTrue: [^answer]
```

اینجا مفهوم message گفته شده که میتونه جالب باشه و بفهمیم آلن

چی گفته 😞

Lecture 7: The Object Class

- The Object class is the main class from which all other classes are derived.
- Any and every kind of object in Smalltalk can respond to the messages defined by the Object class
- All methods of the Object class are inherited to overridden
- **Functionality of an object**
 - Determined by its class
 - Two ways to test functionality
 - Comparing object to a class or superclass to test membership or composition
 - `receiver isKindOf: aClass`
 - tests if the receiver is a member of the hierarchy of aClass
 - `anInteger isKindOf: Integer` returns true
 - `receiver isMemberOf: aClass`
 - tests if the receiver is of the same class
 - `anInteger isMemberOf: Magnitude` returns false
 - `receiver respondsTo: aSymbol`
 - tests if the receiver knows how to answer aSymbol
 - `anInteger respondsTo: #sin` returns true
 - `anInteger respondsTo: #at:` returns false
 - Querying the object for its class
 - `receiver class`
 - `#(1 2 3) class` returns Array
- **Comparison of objects**
 - Comparison and equivalence are very similar, but should not be confused
 - `==` is used to test if the receiver and argument are the same object
 - `#(a b c) class == Array` returns true
 - `#(a b c) == #(a b c) copy` returns false
 - `=` is used to test if the receiver and argument represent the same component
 - `#(a b c) class = Array` returns true
 - `#(a b c) = #(a b c) copy` returns true
 - Other comparison operations
 - `receiver ~= anObject`
 - Not equal
 - `receiver ~~ anObject`
 - Not Equivalent
 - `receiver hash`
 - `hash` provides a nice way of telling objects apart, too much trust should not be placed in comparing objects of the same class, as `hash` is often trivialized (as in the example below, `Array` uses `size` as the hash function).
 - Ex:

```
a := 3.147 hash. ← 132
b := 3.14 hash. ← 287
c := #(1 2 3) ← 3
d := #(3 4 5) ← 3
```
- **Copying objects**
 - `deepCopy` has been removed since VisualWorks 1.0
 - Two methods for copying:
 - `copy` returns another instance just like the receiver. Usually `copy` is simply a shallow copy, but some classes override it.
 - Does not copy the objects that the instance variables contain, but copies the "pointer" to the objects.

Lecture 8: Messages & Methods

- Messages are what is passed between objects
- Methods are what is defined in a class to act on an instance of the class
- **Message Expressions**
 - Receiver-object message-selector arguments
 - Unary
 - Receiver message-selector
 - Parsed left to right
 - Ex: `Time now.`
 - Ex: `8 squared.`
 - Binary
 - Receiver message argument
 - Parsed left to right
 - Ex: `1 + 2 * 3.` (Note: returns 9)
 - Parenthesis do the expected
 - Ex: `1 + (2 * 3).` (returns 7)
 - Keyword
 - Receiver message arguments
 - Ex:

```
aString = 'ABC'.
aString at: 3 put: $D. (Note: returns 'D', aString equals #(ABD))
```

 - Important to note that `'ABC' at: 3 put: $D` returns `$D`
 - `aString` is the object
 - `at` is the keyword message-selector
 - `3` is the argument
 - `'C'` is the object
 - `put` is the keyword message-selector
 - `$D` is the argument (`$D` is a literal)
 - Parentheses change order
 - Precedence *always* left to right
 - Separated by periods, unless temp variable declaration or comment
 - **Method Lookup**
 - A method and a message-selector must be exactly the same, or no method will be found by the method lookup
 - The methods defined for the receiver's class first
 - If no match, the superclass is searched
 - Path continues through Object unless a method is found.
 - `self` refers to receiver, lookup starts within the class of the receiver
 - `super` refers to receiver, lookup starts in superclass of receiver

قسمت های باحال کتابش رو به نگاهی حتما بندازید مفاهیم خیلی جالبی توشه!

یکم بیشتر بریم ببینیم این آقای آلن کی رو کجا میتونیم پیدا کنیم؟

Quora



Search Quora

Quora uses cookies to improve your experience. [Read M](#)



Alan Kay



19,063 followers · 1 following

Follow

Notify me

Ask



Still trying to learn how to think better

[Profile](#) [537 Answers](#) [0 Questions](#) [0 Posts](#) [19.1K Followers](#) [Following](#) [Edits](#) [Activity](#)

Profile

Most recent ▾



Alan Kay · [Follow](#)

Still trying to learn how to think better · Sat

Why do some media amplify cognition more than others?

A comprehensive answer is beyond the size of a Quora response. But we are "thinking" all the time, and part of our thinking is highly influenced by the information given to us by our senses: especially kinesthetics/touch, visual/configurative, [syn](#) (more)



28



Alan Kay · [Follow](#)

I'm the "Alan Kay" in question (try Google for the usual misinformation) · Thu

How can I incorporate a historic/literary approach to a topic about computer science?


I like to bring in important events in the history of computing when writing about contemporary issues. The main reason is that a very high percentage of computerists

خب بریم ببینیم آن کی در پاسخ به برخی سوالات در ارتباط با ایده ی خودش چه جواب هایی داده ؟

Object-Oriented Programming

C++ (programming language)

Programming Languages

Computer Programming 

What did Alan Kay mean by, "I made up the term object-oriented, and I can tell you I did not have C++ in mind."?

 Answer

 Follow · 32

 Request



4 Answers

قسمت های مهم پاسخ آن رو هایلایت کردم :



Alan Kay, Have designed a few programming languages



Updated Oct 25, 2017 · Upvoted by Alan Mellor, [Started programming 8 bit computers in 1981](#) and Bulat Ziganshin, [27 years of C++ and counting](#)

It's hard to praise too highly the programming languages that are the bridge from one way of looking at programming to much better ways of looking at programming. The two greatest such in the 60s were Lisp and Simula. Perhaps the greatest single conception of a software system of the 60s was Sketchpad.

As explained elsewhere on Quora, and in "The Early History of Smalltalk", I had chance encounters with Sketchpad and Simula in my first week of grad school in late 66, that shocked me into a realization about "computers as basic and universal units" via the connections and parallels with other like things, such as biological structures, computers on networks, processes in time-sharing systems, general systems of parts intercommunicating, and so forth. **I started to think about dynamic languages to make such processes, and how the processes could be made efficient and parsimonious enough to be universal.**

This attracted great attention, and 10 years later in the 80s “object-oriented languages” started to appear. The only ones that had some of the same flavor as Smalltalk were several versions of Lisp. There were several “object Pascals”, even an “Object COBOL”!, and of course, C++.

It's important to realize that C++ was part of a chain of ideas starting around 1979 by Bjarne Stroustrup who was also shocked into action from encountering Simula. He was not trying to “steal from Smalltalk” in any way. Here's what he says in his history paper:

C++ was designed to provide Simula's facilities for program organization together with C's efficiency and flexibility for systems programming. ... The goal was modest in that it did not involve innovation, and preposterous in both its time scale and its Draconian demands on efficiency and flexibility.

Elsewhere, he takes pains to say that he's “not trying to do what Smalltalk at Xerox Parc did”. He was essentially trying to do with C what Simula did with Algol.

His approach was via “Abstract Data Types” (which the co-inventor of Simula — Dahl — also liked), and this is the way “classes” in C++ are generally used. And, similar to Simula being a preprocessor to Algol, C++ was a preprocessor to C: the “classes” were program code structuring conventions but didn't show up as objects during runtime.

And for a variety of reasons, some of them not good, some reasonable, C++ got very popular.

و حرف خیلی مهمی که خود آرن نوشتہ:

I don't think that “real OOP” as we thought of it then, is the way to go in the future (and didn't then). Consider Sketchpad, and that it is programmed in terms of constraints that the system solves to guarantee a very close fit between what was desired and what is manifested. This is an early glimpse into “requirements-based programming”. It has something like objects — hard to get away from the main idea — but is “relational” rather than message-based (the messages are implicit, etc.) Sketchpad was a tour de force in many ways, including its solvers. But they didn't go far enough to handle general requirements. Today I think this is doable via a half dozen new techniques plus enormously larger machine capacities.

بریم سوال های بعدی رو ببینیم (داستان داره جالب میشه) :

What is Alan Kay's definition of Object Oriented?



Alan Kay, Had something to do with "Object-Oriented Programming"

Answered Sep 14, 2016

Originally Answered: What is Alan Kay's definition of Object Oriented?

The other answers are well worth reading. An interesting question arises about terminology: to what extent is it reasonable to try to retain original definitions versus having a term get "softer" in meaning as time passes and new and additional conceptions are formed? (The latter happens in part because of the ways we spontaneously evolve our languages, so trying to legislate against it doesn't work.)

However, I think "colonizing" a term to get reflected status is not a good practice, because it really weakens the central ideas (for example, C++ is called an "object-oriented language" — and most people argue "it is!" — but it is much too far from the ways I was thinking to be included in any definition I would come up with).

Part of the problem here is that I made a mistake with how the term was coined — I should have picked something else — in hindsight: "server-oriented programming"?

In any case, the "server" metaphor — mentioned by Eric des Courtis below — is good enough here (since I've written about and answered questions about "objects" in the past, and especially in the "The Early History of Smalltalk" written for the ACM ca 1993.).

There is a bit of a red herring here because the power of a comprehensive universal building block can also be its downfall. For example, a “server” could choose to allow its encapsulation to be violated — e.g. by making its services to closely resemble data structures acted on by procedures. Here, in my opinion, we would be simulating quite the wrong kinds of things, and devolving back into weak and fragile programming styles. (That is my view of what has mostly happened with “objects” — “real objects” never showed up because most people wanted to retain their data oriented style, etc.)

We could argue that the definition was incomplete — even: poor. It allowed too much discretion on the parts of programmers (this was partly because we used it for our own purposes at Parc and thought — not terribly accurately — that -we- had sufficient discretion to use it wisely (certainly not 100% of the time!))

And ... this November will be the 50th anniversary of my “recognition” of the powers of the simple idea of making everything from “encapsulated servers exchanging non-command messages”.

Even though — in my opinion again — the simple idea of making computation systems “be like computers on the Internet” still isn’t generally recognized, much more is needed in programming and systems building than “a great recognition” from 50 years ago which had enormous relative power for about two decades.

For example — today and tomorrow — we should be programming in terms of “requirements and goals” that can manifest a workable system (possibly needing a super-computer).

Even though — in my opinion again — the simple idea of making computation systems “be like computers on the Internet” still isn’t generally recognized, much more is needed in programming and systems building than “a great recognition” from 50 years ago which had enormous relative power for about two decades.

For example — today and tomorrow — we should be programming in terms of “requirements and goals” that can manifest a workable system (possibly needing a super-computer).

We should be able to optimize a system like this without touching the requirements and goals part, etc. The feeling of such programming should be like the CAD-SIM-FAB cycles in more developed parts of engineering. In other words, we want to devote most of our attention into “the whats” rather than “the hows”, use most of our energy for design, and we’d like to “ship the design!” (that would be a good slogan for the next few years).

Just as the great language Lisp was first for programming, but then became “a very high level machine code” for higher level ideas, we should see that what was powerful about direct programming with the kinds of object systems we made at Parc and subsequently so many years ago, should now be retained for structural integrity and other pragmatic reasons, but that the code should now be automatically written from much higher level sources. This doesn’t mean that “objects are now hidden”, but that they should be part of the “modeling and designing of ideas and processes” that is the center of what programming needs to be.

12.1K views · View 101 upvotes · View 1 share · Answer requested by Mark Miller



101



1



8



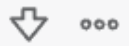
و باز هم سوال جالب بعدی :



Sohail Morady · May 11, 2019

Alan Kay I can't understand one thing. You say "non-command messages", but what if our object models a "Person" and we really need to choose a name for him? shouldn't this be a "command" to the object? I think there's something here that I don't understand.

Upvote · 1 Reply



Alan Kay · May 10, 2020

This is a good question, and one that comes up in both Lisp and OOP. Hint: why did McCarthy invent "LABEL"?

I'm not so much into trying to copy the physical world as some, but this is an interesting analogy. A parent may try to force a name on a kid, but quite a few kids come up with a different name for themselves internally, and some even carry this through to an external name via the courts

In Lisp and OOP, objects can be bound to various variables (these are external names), they can have their own notion of "name", etc.

But they also have an "ID" which is bestowed by the universe at birth. (Part of the difference is the difference between "EQ" and "EQUAL" in Lisp.)

Upvote · 1 Reply



Sohail Morady I've seen that you talk about "thinking in systems" a lot. So I've found a...

خیلی جالب شد واقعا و میفهمیم همیشه منتقدانی بودند به این شی گرای و آلن هم تقریبا جوابشون رو داده و از خودش دفاع کرده اما از طرفی میگه که شی گرای هایی که در زبان های الان هست، اون چیزی نیست که در ذهن من بود !

Alan Kay

C++ Vs. Java

Object-Oriented Programming

C# (programming language)

C++ (programming language)

+5



What are the differences between Alan Kay's OOP and today's OOP in C++, Java, and C#?



Answer



Follow · 9



Request



Alan Mellor, Self taught programmer starting in 1981



Updated Feb 10, 2020

The biggest difference is that objects have direct connections to each in these languages.

To call a method on some other object, you need a reference to it. A way of identifying precisely which object to call the method on.

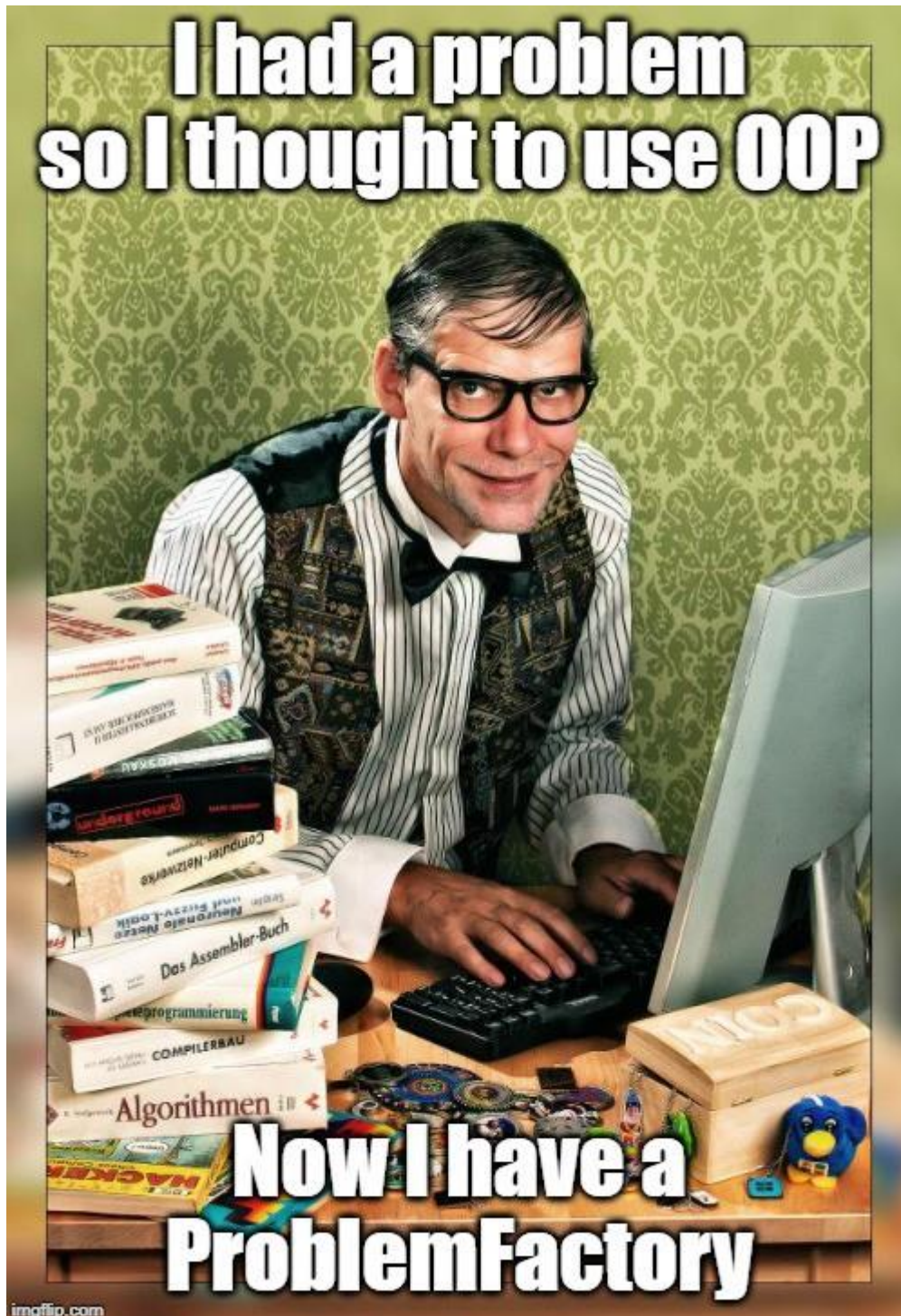
In Kay's analogy of messages as enzymes in the bloodstream, the two objects are unaware of each others' existence. An object releases a message which is received by all objects. The ones that wish to respond do so.

I feel that the type systems are also different, but I lack the expertise to analyse that.

For certain another difference is that a running program was a live system in the Kay / Smalltalk language world. You could change objects whilst the code keeps running.

That is simply not present in any mainstream language today. It affects how we think about building software.

خب ما تا اینجا فلسفه ی وجودی برنامه نویسی ش گرا رو خوندم و
درباره اش یاد گرفتیم.



برنامه نویسی شی گرا یا برنامه نویسی تابع گرا ؟

در قسمت 19 میریم سراغ مخالفان برنامه نویسی شی گرا.
در قسمت 20 دلایل موافقان برنامه نویسی شی گرا رو بررسی میکنیم
و در قسمت 21 شی گرایی در پایتون رو بررسی خواهیم کرد.

ممنون که دنبال میکنید  

کاری از : علی معینیان

21 day challenge with python

~~Object Oriented Programming~~

Part 19 out of 21

Why OOP is Bad ?

Is it a mistake?

سلامم ✓

توی قسمت قبلی، اومدیم و با فلسفه ی وجودی شی گراییی آشنا شدیم.
توی این قسمت میخوایم بریم سراغ مخالفان برنامه نویسی شی گرا.

از نظرات متخصصان میخونیم، تا معتبر ترین مقاله در این حوزه تا
نظرات برنامه نویسان عادی.

خب مهم ترین و جالب ترین مقاله ای که تا به حال اومده و به ضد
برنامه نویسی شی گرا حرف زده رو شاید تا به حال باهش برخورد
کرده باشید :



 BetterProgramming

ARCHIVE

WRITE FOR US

SUPPORT US

This is your **last** free member-only story this month. [Upgrade for unlimited access.](#)

Object-Oriented Programming — The Trillion Dollar Disaster

Why it's time to move on from OOP



Ilya Suzdalnitski

Follow



Jul 10, 2019 · 27 min read ★



بریم و با هم بررسی کنیم ببینیم چی میگه :

نویسنده ی این مقاله، ایلیا (فامیلیشو نمیتونم بخونم) هست که به اختصار ایلیا صداش میکنیم.


کل بحثی که ایلیا داره میکنه، داره میگه که من با کد های پیچیده مخالفم.

ایلیا اعتقاد داره هر کدی که نوشته میشه باید قابل اعتماد باشه، باید سادگی خودش رو حفظ کنه و بقیه اش مهم نیست !
اون توی مقاله اش میگه که : مسئولیت ما به عنوان یک توسعه دهنده نرم افزار باید کاستن پیچیدگی باشد.

ایلیا به نکته ای که در فایل قبلی اوردم هم اشاره میکنه و میگه حتی خود مخترع شی گراییی هم، منتقد شی گراییی ای هست که در حال حاضر داره استفاده میشه.

ایلیا در جایی از مقاله اش میگه من خودم موافق شی گراییی هستم که آلن کی وارد دنیای برنامه نویسی کرد و در اصل مخالف شی گراییی هستم که در زبان های برنامه نویسی در حال انجام هست.

در قسمت اول مقاله اش با یک جمله ی جالب برخورد کردم :

This post sums up my first-hand decade-long journey from Object-Oriented to Functional programming.  No matter how hard I try, I can no longer find use cases for OOP. I have personally seen OOP projects fail because they become too complex to maintain.

. . . .

ایلیا در اصل داره با پیچیدگی شی گرایی مخالفت میکنه و معتقده، شی گرایی کد ها رو پیچیده میکنه .

ایلیا میگه که ما قبل از نوشتن برنامه، اون رو روی کاغذ برنامه ریزی میکنیم و شی گرایی شاید اونجا خیلی کمکون بکنه ولی در عمل صرفا پیچیدگی پروژه رو زیاد میکنه و بعد هم باعث شکست اون پروژه خواهد شد.

ایشون هم دیدش همانند آلن کی هست و معتقده این شی گرایی که امروزه ما داریم استفاده میکنیم، از هدف اصلی خودش خیلی دور شده!

توی این قسمت از مقاله اش، به طور خیلی جالبی کلا بحث شی
گرایی رو رد میکنه :

OOP is not natural for the human brain, our thought process is centered around “doing” things — go for a walk, talk to a friend, eat pizza. Our brains have evolved to do things, not to organize the world into complex hierarchies of abstract objects.

چرا این قسمت مهمه ؟

خب ما گفتیم برنامه نویسی ها سعی کردند همیشه واقع بین باشند و
برنامه نویسی رو به زندگی واقعی انسان ها نزدیک کنند !
اما اینجا، ایلیا میگه اصلا بحث شی گرایی با طبیعت انسان ها
ناسازگاره !

در قسمت بعدی مقاله، ایلیا میاد و از چارچوب های برنامه نویسی نام
میبره و میگه ما در برنامه نویسی 2 تا چارچوب کلی داریم که برنامه
نویس ها چه خوب باشن و چه بد، از این دو چارچوب استفاده میکنند :

1 – برنامه نویسی تابع گرا

2 – برنامه نویسی شی گرا

و برای چارچوب مناسب، سه تا عنصر رو معرفی میکنه :

A good programming framework helps us to write reliable code. First and foremost, it should help reduce complexity by providing the following things:

1. Modularity and reusability
2. Proper state isolation
3. High signal-to-noise ratio

و حالا میاد و اولین ایراد فنی خودش رو به برنامه نویسی شی گرا میگیره با توجه به چارچوبی که تعریف کرد :

Unfortunately, OOP provides developers too many tools and choices, without imposing the right kinds of limitations. Even though OOP promises to address modularity and improve reusability, it fails to deliver on its promises (more on this later). OOP code encourages the use of shared mutable state, which has been proven to be unsafe time and time again. OOP typically requires a lot of boilerplate code (low signal-to-noise ratio).

در این قسمت خیلی سفت و سخت میاد و از برنامه نویسی تابع گرا یا همون Functional programming دفاع میکنه و دلایل خودش را هم بیان میکنه :

The one thing that Functional Programming does really well is it helps us write *reliable* software. The need for a debugger almost disappears completely. Yep, no need to step through your code and watch variables. I personally haven't touched a debugger in a very long time.

The best part? If you already know how to use functions, then you're already a functional programmer. You just need to learn how to make the best use of those functions!

I'm not preaching Functional Programming, I don't really care what programming paradigm you use writing your code. I'm simply trying to convey the mechanisms that Functional Programming provides to address the problems inherent with OOP/imperative programming.

در این قسمت ایلیا میاد و اشاره ای به Erlang و smal talk میکنه!
اسمال تاک رو درباهره اش در فایل قبلی گفتم.
ایلیا معتقدتهنها جاهایی که شی گرایی به مفهوم واقعی استفاده شده،
داخل همین دو تا زبان برنامه نویسی هست.

و خیلی جالبه که حرف های خودم در قسمت قبل رو دوباره دارم اینجا هم میبینم .

Alan Kay's big idea was to have independent programs (cells) communicate by sending *messages* to each other. The state of the independent programs would *never be shared* with the outside world (encapsulation).

That's it. OOP was *never intended* to have things like inheritance, polymorphism, the "new" keyword, and the myriad of design patterns.

ایلیا در این قسمت طرفداری خیلی جالبی از Erlang میکنه و میگه معتبر ترین زبان برنامه نویسی دنیاست !

(جالب شد باید برم درباره ی ارلنگ بیشتر بخونم)

With OOP-inflected programming languages, computer software becomes more verbose, less readable, less descriptive, and harder to modify and maintain.

— Richard Mansfield

و باز در این قسمت، ایراد بزرگ پیچیدگی رو به برنامه نویسی شی گرا وارد میکنه :

The *most important* aspect of software development is keeping the code complexity down. Period. None of the fancy features matter if the codebase becomes impossible to maintain. Even 100% test coverage is worth nothing if the codebase becomes too *complex* and *unmaintainable*.

What makes the codebase complex? There are many things to consider, but in my opinion, the top offenders are: shared mutable state, erroneous abstractions, and low signal-to-noise ratio (often caused by boilerplate code). All of them are prevalent in OOP.

و باز هم طرفداری از برنامه نویسی تابع گرا :

In Functional Programming, state typically is being *isolated*. You always know where some state is coming from. State is never scattered across your different functions. In OOP, every object has its own state, and when building a program , you have to keep in mind the state of *all* of the objects that you currently are working with.

به مهم ترین قسمت این مقاله میرسیم 😞

توی این قسمت، ایلیا میاد و برنامه نویسی شی گرا رو با دنیای واقعی مقایسه میکنه و کاملا مخالفت خودش رو با شباهت برنامه نویسی شی گرا و دنیای واقعی اعلام میکنه :

Methods/Properties

The methods or properties that provide access to particular fields are *no better* than changing the value of a field directly. It doesn't matter whether you mutate an object's state by using a fancy property or method — the *result is the same: mutated state.*

The real world is not hierarchical

OOP attempts to model everything as a hierarchy of objects. Unfortunately, that is not how things work in the real world. Objects in the real world interact with each other using messages, but they mostly are independent of each other.

Inheritance in the real world

OOP inheritance is not modeled after the real world. The parent object in the real world is unable to change the behavior of child objects at run-time. Even though you inherit your DNA from your parents, they're unable to make changes to your DNA as they please. You do not inherit "behaviors" from your parents, you develop your own behaviors. And you're unable to "override" your parents' behaviors.

The real world has no methods

Does the piece of paper you're writing on have a "write" *method*? No! You take an empty piece of paper, pick up a pen, and write some text. You, as a person, don't have a "write" method either — you make the decision to write some text based on outside events or your internal thoughts.

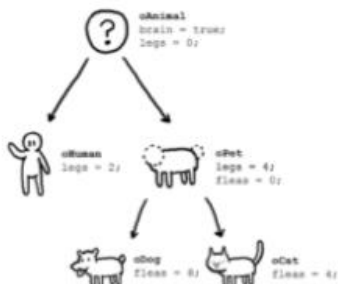
و در نهایت هم می‌گه :

The problem factory

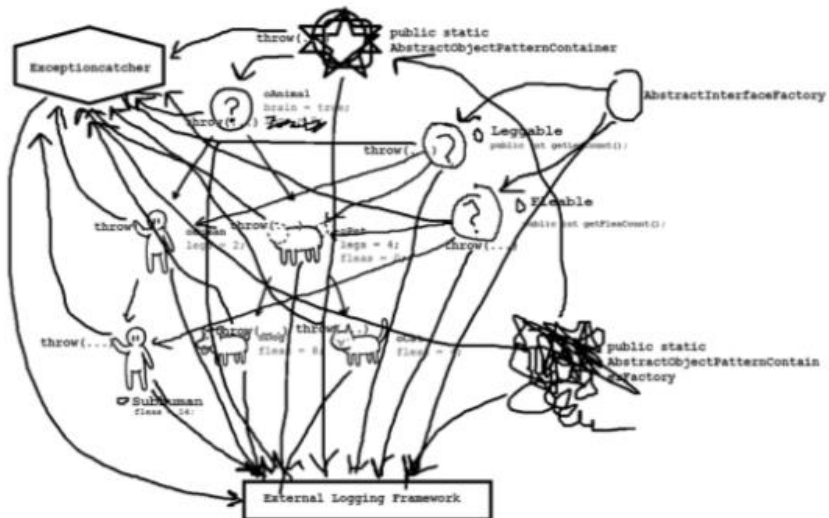
In fact, it is impossible to write good and maintainable Object-Oriented code.

این هم تیر آخر :

What OOP users claim



What actually happens



و دیگه تا آخر این مقاله یکسری مثال ها رو میاره و به طرفدارای برنامه نویسی شیگرا می‌گه بیاید و از اعتقاداتتون دفاع کنید و یکسری مثال ها وسر گذشت جاوا و سی شارپ رو میاد می‌گه !

این از کلیات مقاله ی ایلیا (فامیلشو بلد نیستم) که معروف ترین مقاله بر ضد برنامه نویسی شی گراست !

اما بریم سراغ یک سری آدم دیگه و حرفای بقیه ی مخالفان رو هم گوش کنیم :

ترجمه ی این مقاله رو باهم بررسی خواهیم کرد.

Goodbye, Object Oriented Programming



Charles Scalfani Jul 23, 2016 · 10 min read

من چند دهه است که با زبان های شی گرا برنامه نویسی می کنم. اولین زبان شی گرایی که استفاده کردم ++ C و سپس Smalltalk و در آخر .NET و Java بود.

خیلی مشتاق بودم که از مزایای وراثت ، محصورسازی و چند ریختی استفاده می کردم. سه ستون شی گرایی.

مشتاق بودم وعده "قابلیت استفاده مجدد" را بدست آورم و از دانش کسانی که پیش از من در این منظره جدید و جالب بوده اند استفاده کنم.

نمی توانم هیجانم را هنگام فکرکردن به شبیه سازی دنیای واقعی در کلاس ها توصیف کنم و انتظار داشتم همه چیز درست پیش برود.

بیشتر از این نمیتوانستم اشتباه کنم!

خیلی از منتقدین مخالفان برنامه نویس شی گرا اعتقاد دارند که این مخالفان، الا شی گرایی رو درک نکردند، اما این مقاله از شخصیه که با شی گراییسال های طولانی کار کرده و ببینیم چی میگه :

یک نقل قول عالی توسط جو آرمسترانگ ، خالق ارلانگ وجود دارد:

مشکل زبانهای شی گرا این است که آنها این همه محیط ضمنی را با خود حمل میکنند. شما یک موز می خواستید اما آنچه شما گرفتید یک میمون، موز و کل جنگل بود

مقاله ی این شخص، برخلاف مقاله ی ایلیا، بیشتر با مثال ها میاد و برنامه نویسی شی گرا رو به چالش میکشه و نقاط ضعف اون رو بیان میکنه.

قول های شکسته شده

خوب ، شی گرایی مطمئناً در روزهای اولیه وعده های زیادی داده است. و این وعده ها هنوز به برنامه نویسان ساده لوح که در کلاسهای درس نشسته اند ، وبلاگ ها را می خوانند و دوره های آنلاین را می گذرانند ، داده می شود.

برای درک اینکه چطور شی گرایی به من دروغ گفت ، سالها طول کشید. من هم بسیار چشم بسته و بی تجربه اعتماد داشتم.

خداحافظ ، برنامه نویسی شی گرا.

حالا چیکار کنیم؟

سلام ، برنامه نویسی functional. کار با شما در طول چند سال گذشته بسیار خوب بوده است.

فقط می دانید ، من هیچ یک از وعده های شما را ارزش نمیدانم. من باید آن را ببینم تا آن را باور کنم.

جمع بندی نهایی :

من علاوه بر مقالاتی که در این قسمت آوردم، خیلی بلاگ ها و نظرات رو از برنامه نویس ها در سطح اینترنت خوندم، چه ابرانی چه خارجی !

90 درصد از افراد مخالف با برنامه نویسی شی گرا حرفشون اینه :

تا وقتی که برنامه نویسی فانکشنال هست و خیلی ساده تر و مطمئن تر هست، چرا از شی گرایی استفاده کنیم و کدمون رو پیچیده تر کنیم ؟

کل حرف ها و بحث ها سر فانکشنال نوشتن یا شی گرایی نوشتن، همین 2 خط بالاییه !

اما واقعا درست میگن ???

در قسمت بعدی، نظرات طرفداران برنامه نویسی شی گرا رو بررسی میکنم تا ببینیم اونها چی میگن ؟

ممنونم که من رو دنبال میکنید 😊 ✨

کاری از : علی معینیان

21 day challenge with python

Object Oriented Programming



Part 20 out of 21

Why OOP is Good ?

سلام!!!!!! 😊

رسیدیم به قسمت 20 🌸 😍

توی این قسمت، نظرات، مقالات و صحبت های موافقان شی گرای
رو بررسی خواهم کرد. ببینیم چرا شی گرای خوبه؟

من بعد از یک عالمه جستجو کردن، هیچ مقاله ی معتبری نتونستم پیدا
کنم که جوابی باشه بر مقالاتی که دیروز در ارتباط با بدی ها شی
گرای شرح دادم براتون (این اولین نکته ی جالب).

اول بریم سراغ مفاهیم شی گرای :

شی (Object) : شی یک موجودیت فیزیکی یا مفهوم کلی است بگونه ای که دارای هویت بوده و قادر به بروز رفتار و
ثبت حالات خود می باشد.

صفت (property): هر شی یکسری خصوصیات دارد که به آنها صفت گفته می شود. که در واقع یک مقدار و ارزش
مشخصی برای آن به ازای هر شی می تواند وجود داشته باشد. مانند طول، ارتفاع، رنگ و ...

روش (Method) : هر شی در واقع یک سری رفتار دارد که به آن ها روش یا متد، گفته می شود. متد در واقع پاسخ
هایی است که آن شی در مقابل تحریکات محیط از خود نشان می دهد.

کلاس (Class) : به مجموعه ای از اشیا که دارای ویژگی و رفتار مشترک باشند، کلاس می گویند. یک class نمونه ی
اولیه ای است که هر object از روی آن ساخته می شود. کلاس دانشجو، کلاس انسان، کلاس ماشین و ...

کلاس ها مانند نقشه های ساختمان هستند. یک کلاس، نقشه ایجاد یک شی از کلاس است. همانطور که می توانیم
خانه های زیادی از روی یک نقشه بسازیم، می توانیم تعدادی شی از روی یک کلاس، نمونه سازی کنیم.

مزایای برنامه نویسی شی گرا :

مزایای برنامه نویسی شی گرا چیست؟

- افزایش امنیت برنامه
- کاهش هزینه نگهداری
- قابلیت استفاده مجدد
- تحلیل ساده تر برنامه
- قابلیت سازمان دهی بهینه تر کدها
- عدم نیاز به نوشتن کدهای تکراری و قابلیت هایی که قبلا پیاده سازی شده اند و صرف جویی در استفاده از منابع
- قابلیت تقسیم برنامه به برنامه های کوچک تر اما مستقل

در مطالعاتی که داشتم، اکثر کسانی که از برنامه نویسی شی گرا دفاع میکنند، به دو موضوع کلی اشاره میکنند:

1 – قابلیت استفاده مجدد و تغییر پذیری در برنامه های شی گرا بسیار بیشتر از برنامه های تابع گراست.

2 – برنامه نویس شی گرا باعث ساده تر شدن پیاده سازی خواهد شد.

خب، اینجا دو تا موضوع پیش میاد :

1 – برنامه نویسی شی گرا در چه مواقع میتونه باعث ساده سازی پیاده سازی کد ها بشه ؟

2 – برنامه نویسی تابع گرا در چه زمان میتونه بهتر از برنامه نویسی شی گرا ظاهر بشه ؟

پس لازمه ما این دو چارچوب برنامه نویسی رو با هم مقایسه کنیم.

#1. Definition

Functional Programming



Functional programming emphasizes on evaluation of functions.

OOP



Object oriented programming based on concept of objects.

#2. Data

Functional Programming



Functional programming uses immutable data.

OOP



Object oriented uses the mutable data.

#3. Model

Functional Programming



Functional programming does follow declarative programming model.

OOP



Object oriented programming does follow imperative programming model.

#4. Support

Functional Programming



Parallel programming supported by Functional Programming.

OOP



Object oriented programming does not support parallel programming.

#5. Execution

Functional Programming



In Functional programming, the statements can be executed in any order.

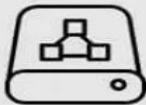
OOP



In OOPs, the statements should be executed in particular order.

#6. Iteration

Functional Programming



In Functional programming, recursion is used for iterative data.

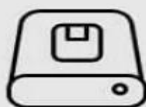
OOP



In OOPs, loops are used for iterative data.

#7. Element

Functional Programming



The basic elements of functional programming are Variables and Functions.

OOP



The basic elements of object oriented programming are objects and methods.

#8. Use

Functional Programming



The functional programming is used only when there are few things with more operations.

OOP



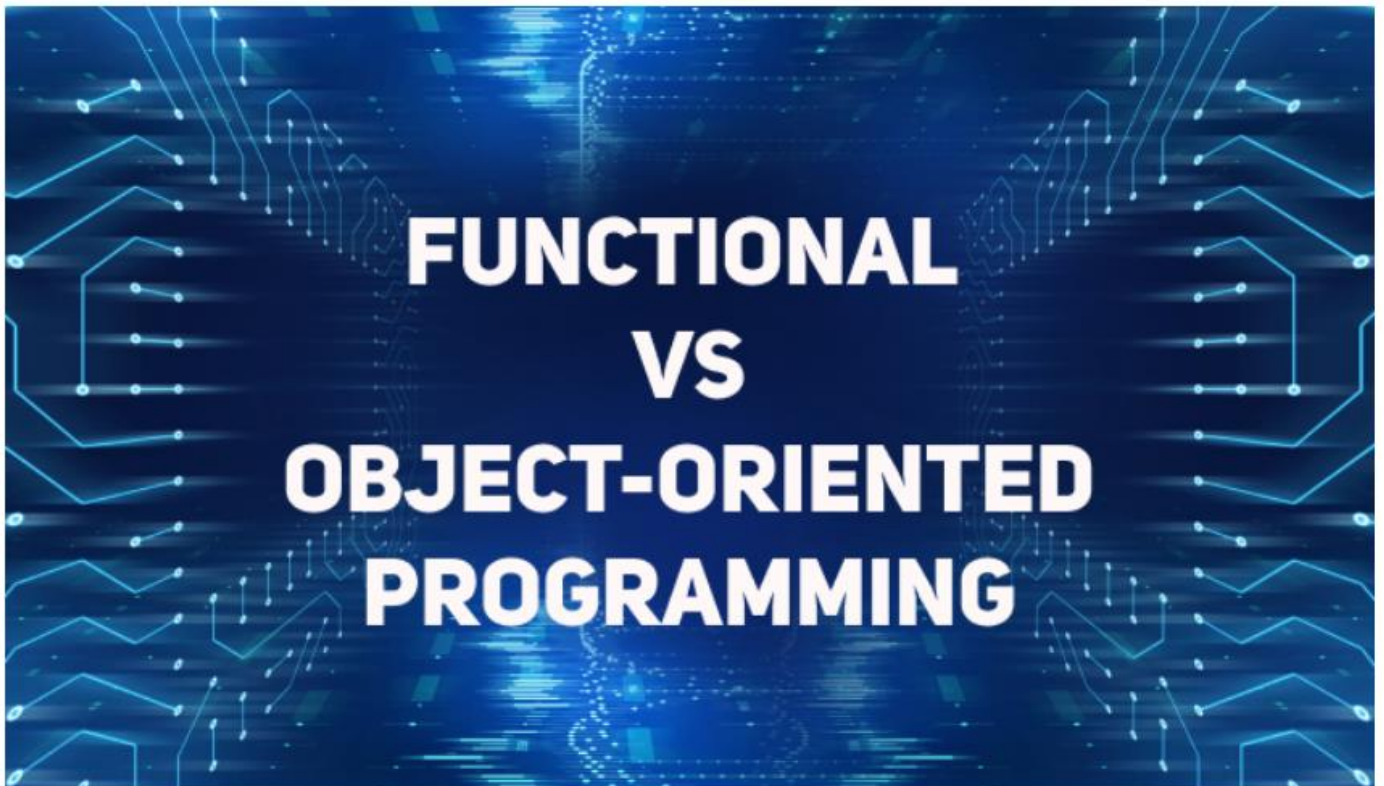
Object oriented programming is used when there are many things with few operations.

یک نمونه از مقایسه ی برنامه نویسی های شی گرا و تابع گرا رو دیدیم.

قسمت های مهم مقاله ی زیر:

Functional Programming VS Object Oriented Programming (OOP) Which is better....?

 shaistha fathima Jul 9, 2019 · 5 min read



برنامه نویسی تابع گرا :

Functional programming provides the advantages like efficiency, lazy evaluation, nested functions, bug-free code, parallel programming. In simple language, functional programming is to write the function having statements to execute a particular task for the application. Each small function does its part and only its part. The function can be easily invoked and reused at any point. It also helps the code to be managed and the same thing or statements does not need to be written again and again. It allows for very modular and clean code that all works together in harmony.

برنامه نویسی شی گرا :

The main deal with OOP is the ability to encapsulate data from outsiders. Encapsulation is the ability to hide variables within the class from outside access — which makes it great for security reasons, along with leaky, unwanted or accidental usage. Most programmers using object oriented design say that it is a style of programming that allows you to model real world scenarios much simpler. This allows for a good transition from requirements to code that works like the customer or user wants it to.

نتیجه گیری جالب این مقاله :

Object-oriented languages are good when you have a *fixed set of operations on things*, and as your code evolves, you primarily *add new things*. This can be accomplished by adding new classes which implement existing methods, and the existing classes are left alone.

Functional languages are good when you have a *fixed set of things*, and as your code evolves, you primarily *add new operations on existing things*. This can be accomplished by adding new functions which compute with existing data types, and the existing functions are left alone.

To put it simply, When you're working across different boundaries, OOP is an excellent method to keep everything packaged up and secure from unwanted external usage. Where as, Functional programming works well when complexity is contained.

و در نهایت :

Finally, to conclude, it is always up to the programmers or developers to choose the programming language concept that makes their development productive and easy.

نهایتاً می‌گه بسته به خودتونه که از کدام چارچوب استفاده کنید 😞

اما 😞 هنوز نمیتونیم فراموش کنیم که شی گرای بی باعث پیچیده تر شدن کد های میشه که میتونی به سادگی اونها رو فانکشنال بنویسیم! البته همیشه گفت همیشه این حرف درسته، ولی مطمئن طرفداران برنامه نویس شی گرا هم این حرف رو قبول دارند.

یکی از مقالات جالبی که از خوبی های شیگرایی به طرز جالبی دفاع میکنه رو بهتون معرفی میکنم، اگر دوست داشتید میتونید مطالعه کنید.

[Home](#) » [Software Development](#) » [Software Development Tutorials](#) » [Programming Languages Tutorial](#) » Advantages of OOP

Programming Languages Tutorial

- Programming Languages Basics
- Algorithm In Programming
- Web Development Apps In Go Programming
- Back End Programming Languages
- Best Programming Languages
- Code Generator Tools
- Duck Number
- Evil Number



و اما بریم سراغ سایت Geeks for Geeks که خیلی خلاصه اومده
و توضیح داده :

Benefits of OOP

- We can build the programs from standard working modules that communicate with one another, rather than having to start writing the code from scratch which leads to saving of development time and higher productivity,
- OOP language allows to break the program into the bit-sized problems that can be solved easily (one object at a time).
- The new technology promises greater programmer productivity, better quality of software and lesser maintenance cost.
- OOP systems can be easily upgraded from small to large systems.
- It is possible that multiple instances of objects co-exist without any interference,
- It is very easy to partition the work in a project based on objects.
- It is possible to map the objects in problem domain to those in the program.
- The principle of data hiding helps the programmer to build secure programs which cannot be invaded by the code in other parts of the program.
- By using inheritance, we can eliminate redundant code and extend the use of existing classes.
- Message passing techniques is used for communication between objects which makes the interface descriptions with external systems much simpler.
- The data-centered design approach enables us to capture more details of model in an implementable form.

Disadvantages of OOP

- The length of the programmes developed using OOP language is much larger than the procedural approach. Since the programme becomes larger in size, it requires more time to be executed that leads to slower execution of the programme.
- We can not apply OOP everywhere as it is not a universal language. It is applied only when it is required. It is not suitable for all types of problems.
- Programmers need to have brilliant designing skill and programming skill along with proper planning because using OOP is little bit tricky.
- OOPs take time to get used to it. The thought process involved in object-oriented programming may not be natural for some people.
- Everything is treated as object in OOP so before applying it we need to have excellent thinking in terms of objects.

مقاله ای ایلیا رو یادتونه ؟ (قسمت 19)

این مقاله بر خلاف ایلیا اومده و مطالبی رو نوشته، خواندنش خالی از لطف نیست، گرچه به نظر من نتونسته خیلی خوب بیاد و جواب بده.

A reaction to "Object-Oriented Programming — The Trillion Dollar Disaster" article.

Published on September 14, 2019



Philippe Roy

Co-Founder of EveryListing.com.

3 articles

+ Follow

بعد از جستجوی های زیاد، من نتونستم هیچ مقاله یا سایتی پیدا کنم که بر ضد برنامه نویسی تابع گرا صحبت کنه!

اما در قسمت قبل دیدیم چه ایراداتی بر برنامه نویسی شی گرا وارد بود!

این نشانه ی چیه ؟

بهترین نتیجه ای که میشه تا الان گفت :

به خاطر استفاده ی گسترده از برنامه نویسی شی گرا، امکان حذف این چارچوب نیست، حال چه درست باشد و چه از پایه غلط.

پس بهترین کار آن است که در جای درست، تصمیم گیری کنیم که بهتر است از برنامه نویسی شی گرا استفاده کنیم یا برنامه نویسی تابع گرا.

من و امثال من که تازه وارد دنیای برنامه نویسی شده ایم شاید نتونیم با شی گرایی ارتباط خوبی بگیریم و برامون عجیب باشه، اما به نظرم بهترین منبع یادگیری شی گرایی دوره ی زیر داخل سایت مکتب خونه که رایگان هم هست (دوره از : حمید دانشجو)



در قسمت بعدی، میریم سراغ شی گزایی در پایتون و چالشمون به
پایان میرسه 😊

ممنون که من رو دنبال میکنید 😊

کاری از : علی معینیان

21 day challenge with python



Last Part

Object Oriented Programming in Python

☆ سلام ☆

و رسیدیم به آخرین قسمت چالش 21 روز دوره ی پایتون !

قسمت آخر : شی گرایی در پایتون

قبل از شروع این قسمت باید بگم من خودم خیلی در ارتباط با شی گرایی پایتون خیلی حرفه ای نیستم و قطعاً توش ایراداتی دارم اما سعی میکنم مطالب اصلی رو بیان کنم و داکيومنت های خوب رو معرفی کنم.

قبل از اینکه خودم با مثال ها در ارتباط با شی گرایی پایتون توضیح بدم، باید بهترین منابعی که به خودم هم در فهم شی گرایی کمک کرده اند رو بهتون معرفی کنم :

1 – دوره ی سایت مکتب خونه



آموزش رایگان شیء گرایی در پایتون

مکتب خونه

حمید دانشجو

آخرین به روزرسانی: ۲۲ خرداد ۱۴۰۰
زمان مطالعه: ۷ دقیقه

برنامه‌نویسی شیء‌گرا در پایتون – یک راهنمای مقدماتی برای مبتدیان

برنامه نویسی ۷۳۲۷ بازدید

پایتون برای هر چیزی از رزبری پای تا یادگیری ماشین استفاده می‌شود. با این وجود اگر می‌خواهید با هر نوع پروژه بزرگی کار کنید، باید شیوه کار پایتون با برنامه‌نویسی شیء‌گرا (OOP) از جمله مفهوم کلاس در برنامه نویسی به زبان پایتون را بدانید. این مقاله مفاهیم کاملاً ابتدایی برنامه‌نویسی شیء‌گرا در پایتون را بررسی می‌کند.

فهرست مطالب این نوشته [پنهان کردن]

۱. پایتون اساساً چیست؟
۲. پیش‌نیازهای راه‌اندازی پایتون
۳. مفاهیم اساسی پایتون: کلاس‌ها
۴. متغیرهای خصوصی در پایتون کدام هستند؟
۵. درک وراثت در پایتون
۶. دانش پایتون خود را بیش از این بسط دهید
۷. فیلم آموزش برنامه نویسی شیء‌گرا در پایتون

آموزش اکسل Excel

آموزش برنامه‌نویسی

آموزش پایتون Python

آموزش پریمیر Premiere

آموزش فتوشاپ Photoshop

آموزش وردپرس Wordpress

آموزش ریاضیات دانشگاهی

3 – مقاله ی بسیار کامل و جالب سایت real python



Object-Oriented Programming (OOP) in Python 3

Table of Contents

Python OOPs Concepts

Object Oriented Programming in Python | Set 2 (Data Hiding and Object Printing)

OOP in Python | Set 3 (Inheritance, examples of object, subclass and super)

Class method vs Static method in Python

Class or Static Variables in Python

Changing Class Members in Python

Constructors in Python

Destructors in Python

Inheritance in Python

Types of inheritance Python

Encapsulation in Python



Python OOPs Concepts

Difficulty Level : Easy • Last Updated : 24 Aug, 2021

In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

Main Concepts of Object-Oriented Programming (OOPs)

- Class
- Objects
- Polymorphism
- Encapsulation
- Inheritance



5 – و در نهایت کامل ترین توضیحات :

9. Classes

Classes provide a means of bundling data and functionality together. Creating a new class creates a new *type* of object, allowing new *instances* of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state.

Compared with other programming languages, Python's class mechanism adds classes with a minimum of new syntax and semantics. It is a mixture of the class mechanisms found in C++ and Modula-3. Python classes provide all the standard features of Object Oriented Programming: the class inheritance mechanism allows multiple base classes, a derived class can override any methods of its base class or classes, and a method can call the method of a base class with the same name. Objects can contain arbitrary amounts and kinds of data. As is true for modules, classes partake of the dynamic nature of Python: they are created at runtime, and can be modified further after creation.

In C++ terminology, normally class members (including the data members) are *public* (except see below *Private Variables*), and all member functions are *virtual*. As in Modula-3, there are no shorthands for referencing the object's members from its methods: the method function is declared with an explicit first argument representing the object, which is provided implicitly by the call. As in Smalltalk, classes themselves are objects. This provides semantics for importing and renaming. Unlike C++ and Modula-3, built-in types can be used as base classes for extension by the user. Also, like in C++, most built-in operators with special syntax (arithmetic operators, subscripting etc.) can be redefined for class instances.

(Lacking universally accepted terminology to talk about classes, I will make occasional use of Smalltalk and C++ terms. I would use Modula-3 terms, since its object-oriented semantics are closer to those of Python than C++, but I expect that few readers have heard of it.)

تعاریف کلی مفاهیم موجود در شی گرایى :

- **Class** – A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- **Class variable** – A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- **Data member** – A class variable or instance variable that holds data associated with a class and its objects.
- **Function overloading** – The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved.
- **Instance variable** – A variable that is defined inside a method and belongs only to the current instance of a class.
- **Inheritance** – The transfer of the characteristics of a class to other classes that are derived from it.
- **Instance** – An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.
- **Instantiation** – The creation of an instance of a class.
- **Method** – A special kind of function that is defined in a class definition.
- **Object** – A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.
- **Operator overloading** – The assignment of more than one function to a particular operator.

کمی تعاریف را بیشتر بهش توجه کنیم :

Class

A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created. It is a logical entity that contains some attributes and methods.

To understand the need for creating a class let's consider an example, let's say you wanted to track the number of dogs that may have different attributes like breed, age. If a list is used, the first element could be the dog's breed while the second element could represent its age. Let's suppose there are 100 different dogs, then how would you know which element is supposed to be which? What if you wanted to add other properties to these dogs? This lacks organization and it's the exact need for classes.

و مثال ساخت یک کلاس :

```
python_projects >  1-class.py > ...
```

```
1 # create an empty class for employees
2 class employees:
3     pass
4
```

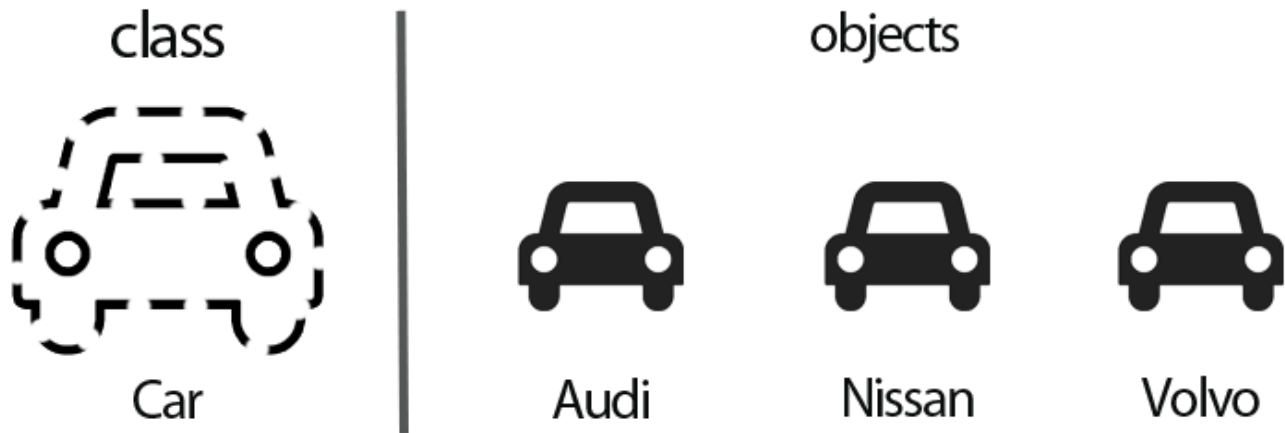
Objects

The object is an entity that has a state and behavior associated with it. It may be any real-world object like a mouse, keyboard, chair, table, pen, etc. Integers, strings, floating-point numbers, even arrays, and dictionaries, are all objects. More specifically, any single integer or any single string is an object. The number 12 is an object, the string "Hello, world" is an object, a list is an object that can hold other objects, and so on. You've been using objects all along and may not even realize it.

مثال ساخت object :

```
python_projects >  1-class.py > ...  
1 # create an empty class for employees  
2 class employees:  
3     pass  
4  
5 #make instance (object) of your class  
6 ali=employees()  
7 hossein=employees()  
8
```

فهم بهتر مفهوم کلاس و آبجکت :



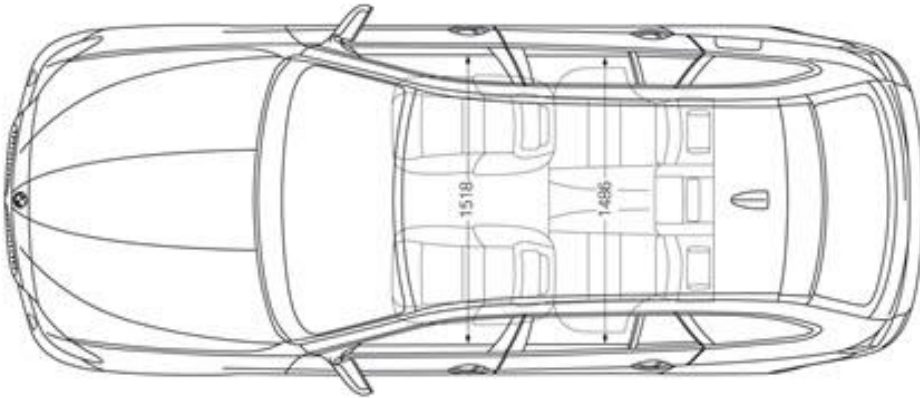
و در قسمت بعدی کمی روی آبجکت ها بیشتر تمرکز میکنیم :

An object consists of :

- **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
- **Behavior:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
- **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

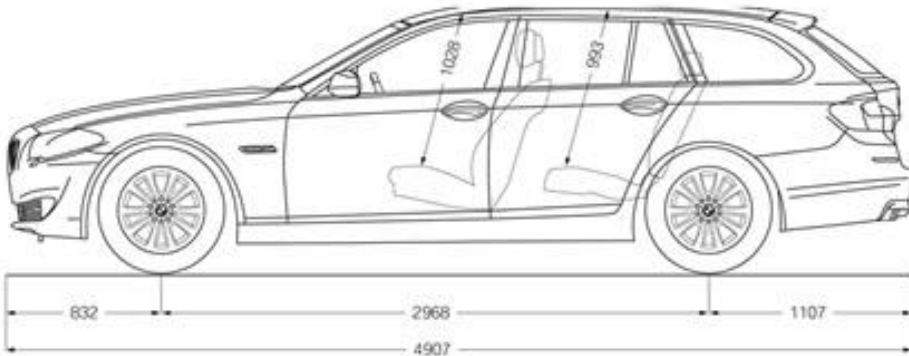
یک عکس خیلی جالب برای فهم بهتر بحث شی گرایی :

class



Methods:

```
drive()
stop()
.
.
.
```



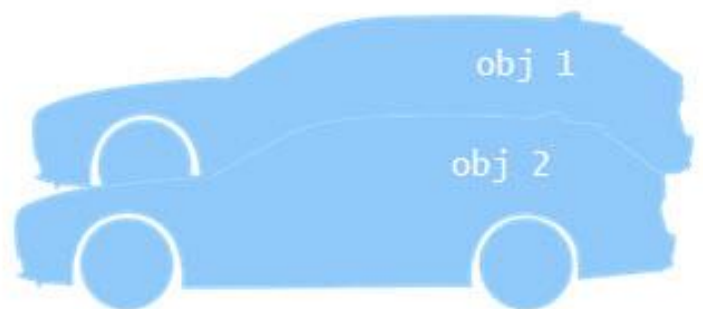
Attributes:

```
color = ?
fuel_capacity = ?
max_speed = ?
.
.
.
```

objects



```
color = #F06292
fuel_capacity = 40
max_speed = 160
```



```
color = #64B5F6
fuel_capacity = 20
max_speed = 80
```

همیشه وقتی در پایتون یک کلاس را میسازیم حتما باید از متد های مختلفی استفاده کنیم، یکی از این متد ها، `__init__` هست :

You can give `.__init__()` any number of parameters, but the first parameter will always be a **variable** called `self`. When a new class instance is created, the instance is automatically passed to the `self` parameter in `.__init__()` so that new **attributes** can be defined on the object.

Let's update the `Dog` class with an `.__init__()` method that creates `.name` and `.age` attributes:

Python

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

Notice that the `.__init__()` method's signature is indented four spaces. The body of the method is indented by eight spaces. This indentation is vitally important. It tells Python that the `.__init__()` method belongs to the `Dog` class.

Notice that the `.__init__()` method's signature is indented four spaces. The body of the method is indented by eight spaces. This indentation is vitally important. It tells Python that the `.__init__()` method belongs to the `Dog` class.

In the body of `.__init__()`, there are two statements using the `self` variable:

1. `self.name = name` creates an attribute called `name` and assigns to it the value of the `name` parameter.
2. `self.age = age` creates an attribute called `age` and assigns to it the value of the `age` parameter.

Attributes created in `.__init__()` are called **instance attributes**. An instance attribute's value is specific to a particular instance of the class. All `Dog` objects have a `name` and an `age`, but the values for the `name` and `age` attributes will vary depending on the `Dog` instance.

On the other hand, **class attributes** are attributes that have the same value for all class instances. You can define a class attribute by assigning a value to a **variable** name outside of `.__init__()`.

بریم توی مثال خودمون ببینیم(حتما به کامنت ها دقت کنید):

```
class employee:
    #Dunder( double underline between them ) = magic methodes(functions)
    #find mmagic methodes : dir(variable)
    #method for initialization
    #we use __init__ to share all common things between employees
    def __init__(self,FirstName,LastName,payment):
        #self : refere to the object of the class
        ...

        self is object holder , for example:
        we made the object name : ali
        and we have :
        self.FirstName=FirstName
        so it make this at final:
        ali.FirstName=FirstName
        exactly like the file : 1-class.py
```

و تکمیل مثال خودمون :

```
class employee:
    #Dunder( double underline between them ) = magic methodes(functions)
    #find mmagic methodes : dir(variable)
    #method for initialization
    #we use __init__ to share all common things between employees
    def __init__(self,FirstName,LastName,payment):
        #self : refere to the object of the class
        ...

        self is object holder , for example:
        we made the object name : ali
        and we have :
        self.FirstName=FirstName
        so it make this at final:
        ali.FirstName=FirstName
        exactly like the file : 1-class.py
        ...

        self.FirstName=FirstName
        self.LastName=LastName
        self.payment=payment
        #create email
        self.mail=str(self.FirstName) + "_" + str(self.LastName) + "@gmail.com"
    # define action in a class = make functioj in a class body
```

Example

Following is the example of a simple Python class –

```
class Employee:
    'Common base class for all employees'
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print "Total Employee %d" % Employee.empCount

    def displayEmployee(self):
        print "Name : ", self.name, ", Salary: ", self.salary
```

- The variable *empCount* is a class variable whose value is shared among all instances of a this class. This can be accessed as *Employee.empCount* from inside the class or outside the class.
- The first method *__init__()* is a special method, which is called class constructor or initialization method that Python calls when you create a new instance of this class.
- You declare other class methods like normal functions with the exception that the first argument to each method is *self*. Python adds the *self* argument to the list for you; you do not need to include it when you call the methods.

حال نیاز داریم برای این کلاسی که تعریف کرده ایم، شی بسازیم :

```
#make object of our class
ali=employee('ali', 'moeinian', '2000')
hossein=employee('hossein', 'nejadnik', '1500')
```

یا در مثال دوم :

```
class Employee:
    'Common base class for all employees'
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print "Total Employee %d" % Employee.empCount

    def displayEmployee(self):
        print "Name : ", self.name, ", Salary: ", self.salary

"This would create first object of Employee class"
emp1 = Employee("Zara", 2000)
"This would create second object of Employee class"
emp2 = Employee("Manni", 5000)
emp1.displayEmployee()
emp2.displayEmployee()
print "Total Employee %d" % Employee.empCount
```

یا بریم مثال داخل خود داکيومنت اصلی پایتون رو ببینیم :

```
>>> class Complex:
...     def __init__(self, realpart, imagpart):
...         self.r = realpart
...         self.i = imagpart
...
>>> x = Complex(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

```
class Dog:

    kind = 'canine'          # class variable shared by all instances

    def __init__(self, name):
        self.name = name    # instance variable unique to each instance

>>> d = Dog('Fido')
>>> e = Dog('Buddy')
>>> d.kind          # shared by all dogs
'canine'
>>> e.kind          # shared by all dogs
'canine'
>>> d.name         # unique to d
'Fido'
>>> e.name         # unique to e
'Buddy'
```

شما میتونید توابع مختلفی رو بر حسب نیازتون در کلاس بنویسید و از اونها استفاده کنید.

بزارید بریم سراغ مثال خودمون و تکمیلش کنیم (حتما به کانت ها توجه کنید) :

```
1  '''
2  class variable : property of the class
3
4  instance variable : unique property of the object of the class
5  '''
6  # Question : we want to add 10% to payment of each employee
7  class employee:
8      pay_rising=0.1
9
10     def __init__(self,FirstName,LastName,payment):
11         self.FirstName=FirstName
12         self.LastName=LastName
13         self.payment=payment
14         self.mail=str(self.FirstName) + "_" + str(self.LastName) + "@gmail.com"
15
16     def fullname(self):
17         return 'Full Name : %s %s ' %(self.FirstName,self.LastName)
18
19     def payment(self):
20         return 'Last Payment : %s' %(self.payment)
21
22     def payment_increase(self):
23         self.payment += float(self.payment * self.pay_rising)
24         return 'New Payment : %s' %(self.payment)
25
26 ali=employee('ali', 'moeinian', 2000)
27 hossein=employee('hossein', 'nejadnik', 1500)
28
29 print('=== Before Increase ===')
30 print(employee.payment(ali))
31 print(employee.payment(hossein))
32
33 print('\n=== After 10% ===')
34 print(employee.payment_increase(ali))
35 print(employee.payment_increase(hossein))
36
37 print('\n=== After 15% just for ali ===')
38 ali.pay_rising=0.15
39 print(employee.payment_increase(ali))
40 print(employee.payment_increase(hossein))
```

اینجا همون قسمتی که دعوا بود درباره اش!
قابلیت تغییر پذیری در شی گرای بیشر از برنامه نویسی تابع گراست
که مثالش رو هم دیدید.

مثال های دیگر :

Python

```
class Dog:
    species = "Canis familiaris"

    def __init__(self, name, age):
        self.name = name
        self.age = age

    # Instance method
    def description(self):
        return f"{self.name} is {self.age} years old"

    # Another instance method
    def speak(self, sound):
        return f"{self.name} says {sound}"
```

Python

```
>>> miles = Dog("Miles", 4)

>>> miles.description()
'Miles is 4 years old'

>>> miles.speak("Woof Woof")
'Miles says Woof Woof'

>>> miles.speak("Bow Wow")
'Miles says Bow Wow'
```

یکسری اتریبیوت ها هم هستند که به ما کمک میکنند :

```
ali=employee('ali', 'moeinian', 2000)
hossein=employee('hossein', 'nejadnik', 1500)

#Find attributes that we can use for our objects :print(dir(ali))

print(ali.__dict__)
print('-----')
print(hossein.__dict__)
print('-----')
print(employee.__dict__)
```

برای فهم مفهوم ارث بری خوبه که توضیحات مثال خودمون رو خوب

بخونید (میدونم کامنت نوشتنم در این قسمت به شدت افترضه ولی میخوام صرفاً برای فهم بیشتر باشه) :

```
#inheritance : ارث بری
'''
ارث بری به ان معناست که ویژگی های یک کلاس به ابجکت ما نسبت داده میشود
حال میتواند همه ی ویژگی ها به ارث برسد و میتواند کمی از آنها به ارث برسد
خود ابجکت هم میتواند یکسری ویژگی های خاص داشته باشد
'''

print('----- single inheritance -----'.title())
class father:
    def feature_dad(self):
        print('this is father part.'.title())
#when we write : (father) it means we are working with inheritance
class child(father):
    def feature_child(self):
        print('this is children part.'.title())
# make object with father
person=father()
person.feature_dad()
# person.feature_child is not acceptable
#-----
وقتی ابجکت را با کلاس به ارث برنده میسازیم هم ویژگی های کلاس اصلی را داریم و هم ویژگی های خودش
person1=child()
person1.feature_child()
person1.feature_dad()
```

```
class Person(object):
    # __init__ is known as the constructor
    def __init__(self, name, idnumber):
        self.name = name
        self.idnumber = idnumber

    def display(self):
        print(self.name)
        print(self.idnumber)

    def details(self):
        print("My name is {}".format(self.name))
        print("IdNumber: {}".format(self.idnumber))

# child class
class Employee(Person):
    def __init__(self, name, idnumber, salary, post):
        self.salary = salary
        self.post = post

        # invoking the __init__ of the parent class
        Person.__init__(self, name, idnumber)

    def details(self):
        print("My name is {}".format(self.name))
        print("IdNumber: {}".format(self.idnumber))
        print("Post: {}".format(self.post))
```

یا یک مثال دیگه :

بحث شی گرای و اواقعا بحثیه که هر چقدر بیشتر واردش میشیم،
مفاهیم خیلییی زیادی رو میتونیم درک کنیم و واقعا من نمیتونم تمام
مباحثش رو در این پی دی اف مطرح کنم.

اما در حد معرفی و چند مثال میتونیم موارد بیشتری از شی گرای رو
با هم درک کنیم :

Polymorphism

Polymorphism simply means having many forms. For example, we need to determine if the given species of birds fly or not, using polymorphism we can do this using a single function.

```
) class Bird:
)     def intro(self):
)         print("There are many types of birds.")
)
)     def flight(self):
)         print("Most of the birds can fly but some cannot.")

class sparrow(Bird):

    def flight(self):
        print("Sparrows can fly.")

class ostrich(Bird):

    def flight(self):
        print("Ostriches cannot fly.")

obj_bird = Bird()
obj_spr = sparrow()
obj_ost = ostrich()

obj_bird.intro()
obj_bird.flight()

obj_spr.intro()
obj_spr.flight()

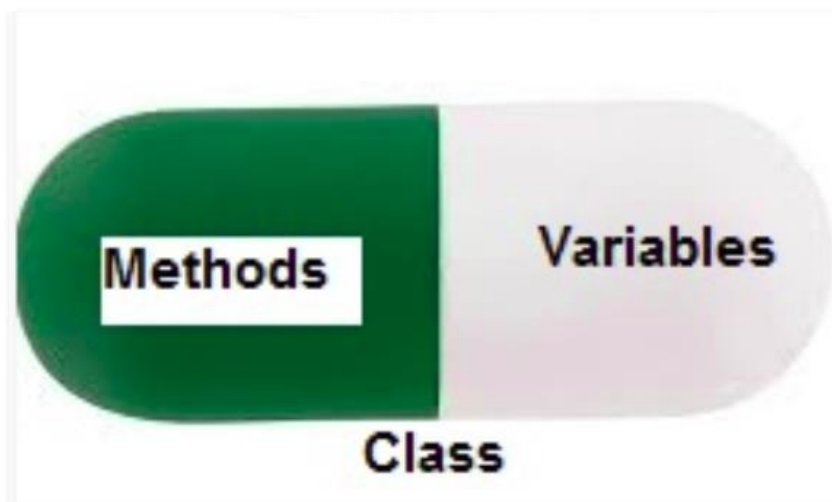
obj_ost.intro()
obj_ost.flight()
```

و اما رسیدیم به یکی از پایه ای ترین مفاهیم شی گزایی که از 3 رکن اصلی شی گزایی بود که آن کی تعریف کرده بود :

Encapsulation

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data. To prevent accidental change, an object's variable can only be changed by an object's method. Those types of variables are known as private variables.

A class is an example of encapsulation as it encapsulates all the data that is member functions, variables, etc.




```
# Python program to
# demonstrate private members

# Creating a Base class
class Base:
    def __init__(self):
        self.a = "GeeksforGeeks"
        self.__c = "GeeksforGeeks"

# Creating a derived class
class Derived(Base):
    def __init__(self):

        # Calling constructor of
        # Base class
        Base.__init__(self)
        print("Calling private member of base class: ")
        print(self.__c)

# Driver code
obj1 = Base()
print(obj1.a)

# Uncommenting print(obj1.c) will
# raise an AttributeError

# Uncommenting obj2 = Derived() will
# also raise an AttributeError as
# private member of base class
# is called inside derived class
```

قطعا موارد بالا که ذکر کردم همه ی بحث شی گزایی نیست.
فهم بحث شی گزایی و انواع مواردی که داخلش دخیله، واقعا نیاز به
یکسری مطالعات وسیع داره.

سعی کردم کلیات این بحث رو معرفی کنم و عین قبل که پازل رو
کامل میکردیم، تکمیل پازل بحث شی گزایی رو به خودتون بسپارم.

در نهایت، باید بگم این 21 فایلی که من درست کردم و به اشتراک
گذاشتم، صرفا با هدف دوره ی مباحث مهم پایتون بود.
من خودم هیچ نوع ادعایی در حرفه ای بودن در پایتون ندارم و بسیار
تازه کارم.

میشه گفت دنیای تخصصی پایتون تازه از این نقطه شروع میشه 😊

خیلی ممنون که در این 21 روز من رو دنبال کردید 🌸

و تمام 🙌❤️

کاری از : علی معینیان