**جزوه آموزش Pure Python**

**امیرحسین ارجمند**

**دانشگاه پارس**

**1401**

# مقدمه

این جزوه در خصوص آموزش برنامه نویسی پایتون با رویکرد عملگرا و همراه با آموزش سریع تدوین شده است .

بعد از مقدمه کوتاه برای هر قسمت ، دانشجویان ملزم هستند تمرینات مورد نظر را انجام داده و در اکانت گیت هاب خود منتشر کنند ، این تمرینات در جهت تکمیل موارد تدریس شده در جزوه و کلاس میباشد .

تمریناتی که در این جزوه آمده است گلچینی از تمریناتی است که در کتب مختلف آموزش پایتون در جهت تسریع یادگیری پایتون آمده است که انجام آنها الزامی است .

لطفا یادداشت هایی که در سر کلاس در خصوص برنامه نویسی گفته میشود به این جزوه اضافه کنید و تاکید میگردد این جزوه شامل همه مواردی که در سر کلاس عنوان میگردد نمیباشد .

با آرزوی موفقیت

امیرحسین ارجمند

# فصل 1 - نصب پایتون (Installation)

در این فصل ما نصب و اجرا مفسر پایتون (python interpreter) بر روی کامپیوتر را انجام میدهیم.

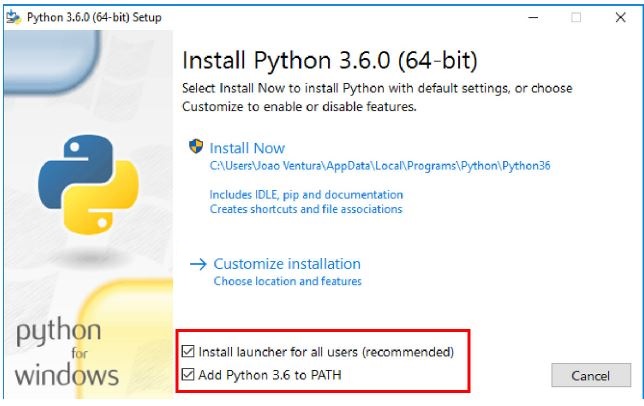
**نصب بر روی windows :**

1 . برای دانلود آخرین نسخه پایتون python3 برای ویندوز از طریق مرورگر خود وارد آدرس زیر شوید : <https://www.python.org/downloads/windos/> و بر روی دانلود کلیک کنید و بعد از دانلود بر روی فایل دانلود شده کلیک کنید تا فرآیند نصب پایتون آغاز شود . در زمان نگراش این جزوه آخرین نسخه پایتون 3.11.2 میباشد .

2 . مطمئن شوید که دو گزینه ی زیر در هنگام نصب تیک خورده باشد .

1 . Install launcher for all users

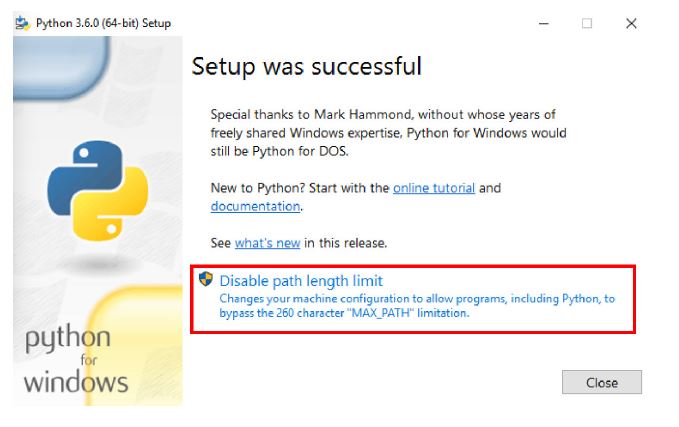
2 . Add python to PATH



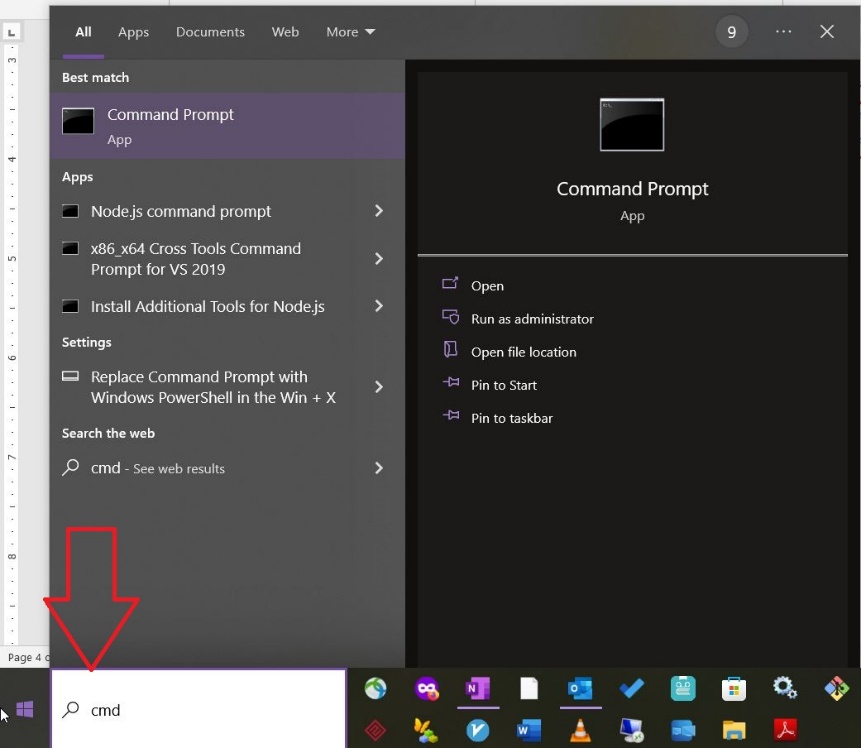
3 . بر روی Install Now کلیک کنید و اجازه دهید پایتون به صورت کامل بر روی سیستم شما نصب گردد .

4 . در هنگام نصب علاوه بر مفسر پایتون موارد دیگری هم مانند pip بر روی سیستم شما نصب میگردد ، pip ابزار مدیریت package و library های پایتون است که به شما اجازه میدهد library و package های مختلفی را بر روی سیستم خود نصب و از آنها استفاده کنید . ( در این خصوص بعدا بیشتر صحبت خواهیم کرد)

5 . پس از اتمام نصب صفحه زیر برای شما ظاهر میشود اگر پیام زیر برای شما نمایش داده شد بر روی آن مانند عکس زیر کلیک کنید و نصب پایتون برای شما به اتمام میرسد .



6 . برای اطمینان از نصب درست پایتون command line یا cmd خود را باز کنید ( در قسمت start ویندوز سرچ کنید cmd )



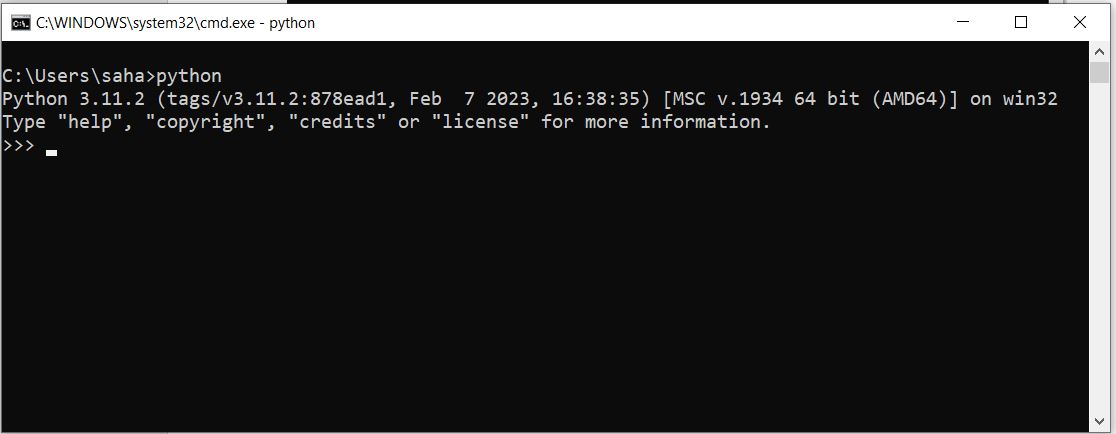
در محیط cmd ابتدا تایپ کنید : python –version

بعد از این دستور اگر پایتون به درستی بر روی سیستم شما نصب شده باشد ورژن پایتون را برای شما نمایش میدهد .

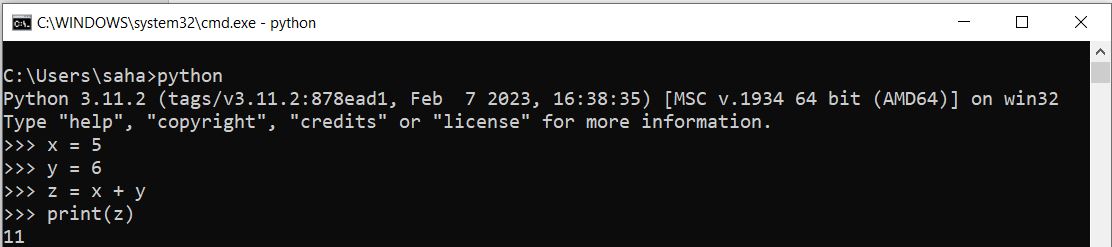


در مرحله بعد اگر در محیط cmd تایپ کنید python یا python3 اگر همه چیز به درستی نصب شده باشد شما باید Python REPL را ببنید .

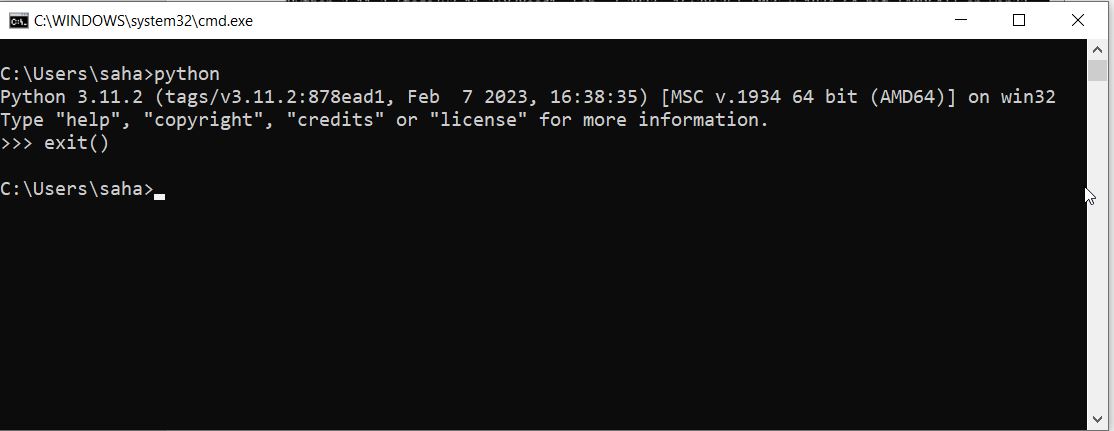
REPL یا Read Evalute, Print and Loop محیطی است که شما میتوانید کدهای کوچک پایتون خود را در آن برنامه نویسی کنید .



ما در این محیط برنامه کوچک پایتونی را که شامل جمع دو عدد است به صورت زیر پیاده سازی میکنیم :



برای خارج شدن از محیط REPL باید کد exit() را اجرا کنید .



**نصب بر روی macOs :**

شما میتوانید آخرین نسخه پایتون را برای macOs از سایت زیر دانلود کنید :

<https://www.python.org/downloads/max-osx/>

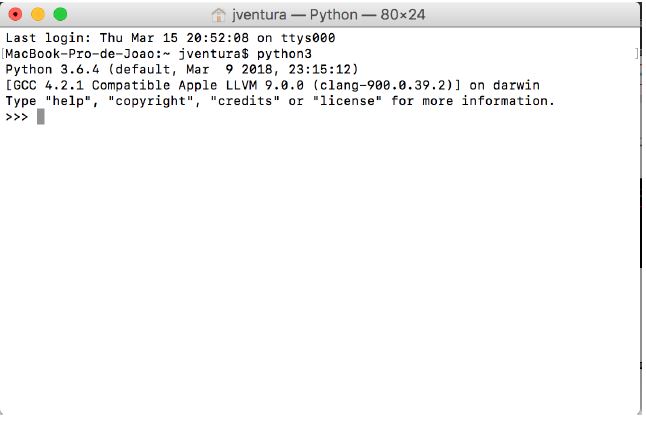
مطمئن شوید که شما آخرین نسخه پایتون 3 را دانلود میکنید (در این لحظه آخرین نسخه پایتون 3.11.2 میباشد)

شما همچنین میتوانید از Homebrew که یک پکیج منیجر در macOs است برای نصب پایتون استفاده کنید (<https://brew.sh/>)

برای نصب آخرین نسخه Python 3 از طریق Homebrew کافی است دستور زیر را در ترمینال خود وارد کنید :

brew install python3

راه دیگر استفاده از MacPorts Package manager است (<https://www.macports.org/>) و کافی است دستور port install python3 را در آن وارد کنید تا پایتون به صورت کامل برای شما نصب گردد .



در نهایت ترمینال خود را باز کنید و دستور python3 را وارد کنید اگر همه چیز به درستی انجام شده باشد شما باید محیط REPL که در بالاتر توضیح داده شد را ببینید .

برای بسته شدن این محیط دستور exit() را تایپ کنید .

**نصب بر روی Linux :**

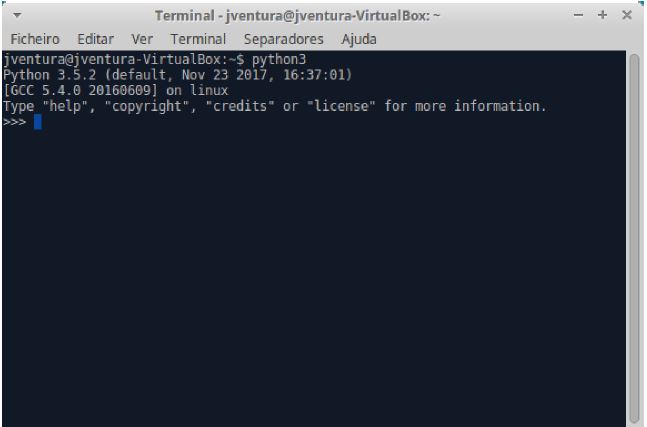
برای نصب پایتون بر روی لینوکس شما میتوانید آخرین نسخه Python 3 را از سایت زیر دانلود کنید :

<https://www.python.org/downloads/source/>

و یا از طریق پکیج منیجر خود و از طریق دستور apt-get پایتون را بر روی سیستم خود نصب کنید .

برای اطمینان از نصب Python 3 بر روی سیستم خود دستور python3 –version را در ترمینال تایپ و اجرا کنید .

در نهایت ترمینال خود را باز کنید و python3 را تایپ و اجرا کنید تا محیط REPL برای شما اجرا گردد برای بستن این محیط دستور exit() را تایپ و enter کنید .



# فصل 2 - نصب محیط برنامه نویسی VsCode

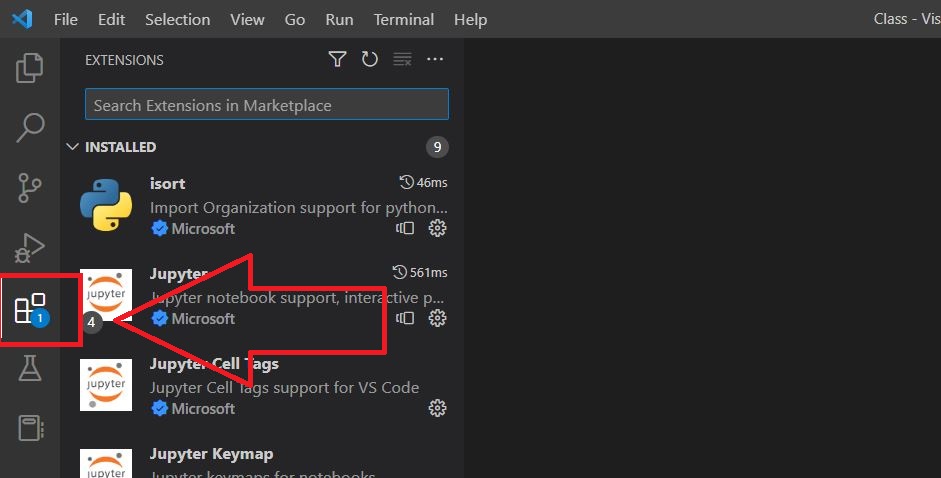
ما برای کد زدن بهتر و حرفه ای تر نیاز به محیط های حرفه ای برنامه نویسی داریم ، در این درس ما از نرم افزار VsCode استفاده میکنیم .

برای نصب VsCode کافی است آن را دانلود و در مسیر دلخواه نصب کنید .

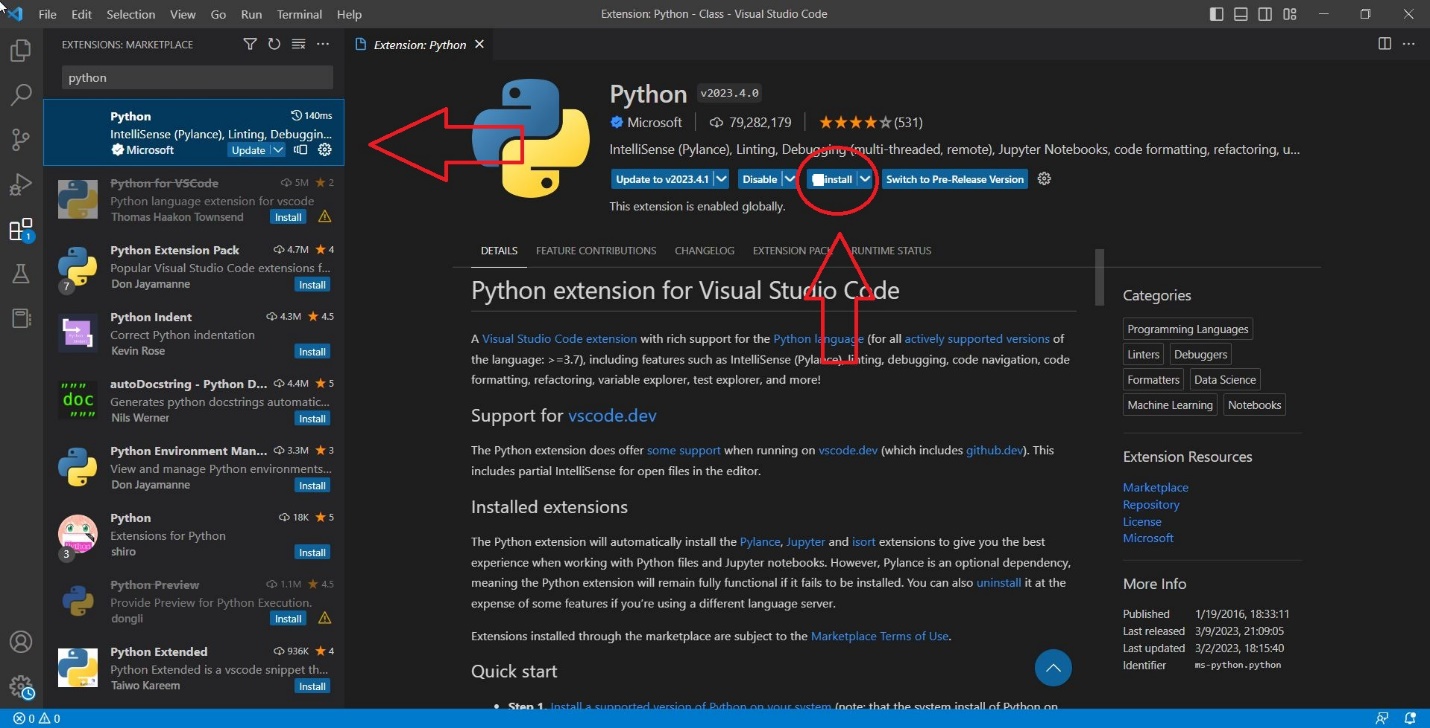
پس از نصب VsCode برنامه را اجرا کنید و مطابق مراحل زیر Extensions پایتون را بر روی VsCode نصب کنید .

این کار محیط برنامه نویسی VsCode را برای کار با پایتون آمده میکند .

1 . از نوار سمت چپ بر روی دکمه ای که در عکس نشان داده شده است کلیک کنید .



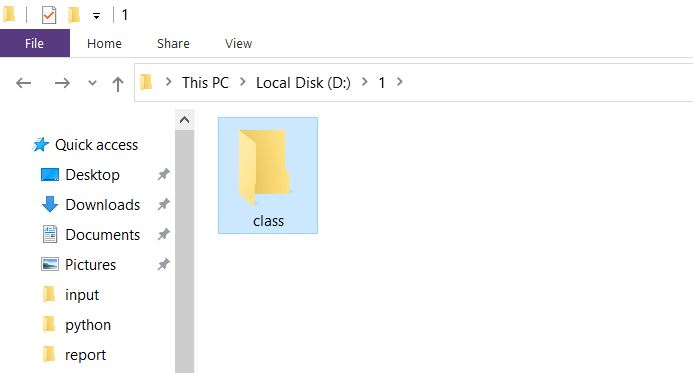
2 . در قسمت بالا (نوار جست و جو) search تایپ کنید python تا مانند عکس زیر اکستنشن پایتون برای شما به نمایش در بیاید :



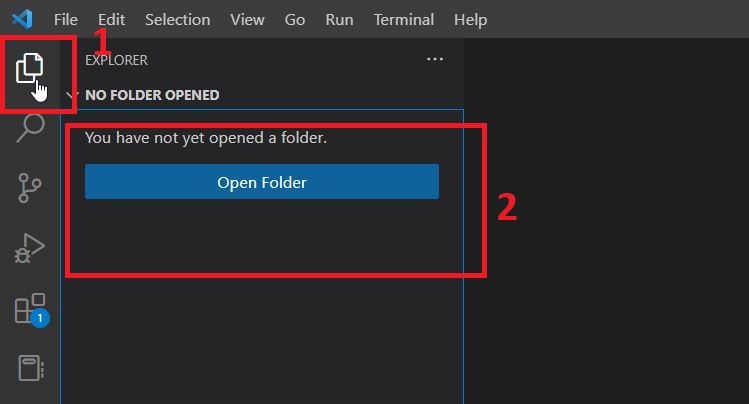
3 . بر روی دکمه install مطابق عکس بالا کلیک کنید تا اکستنشن پایتون برای شما نصب گردد .

# فصل 3 - اجرا یک کد ساده در محیط VsCode

1 . ما در این درس برای برنامه های خود یک Folder باز میکنیم و برنامه های خود را در آن پوشه انجام میدهیم ، نام این پوشه به دلخواه است برای مثال من نام آن را class میگذارم .



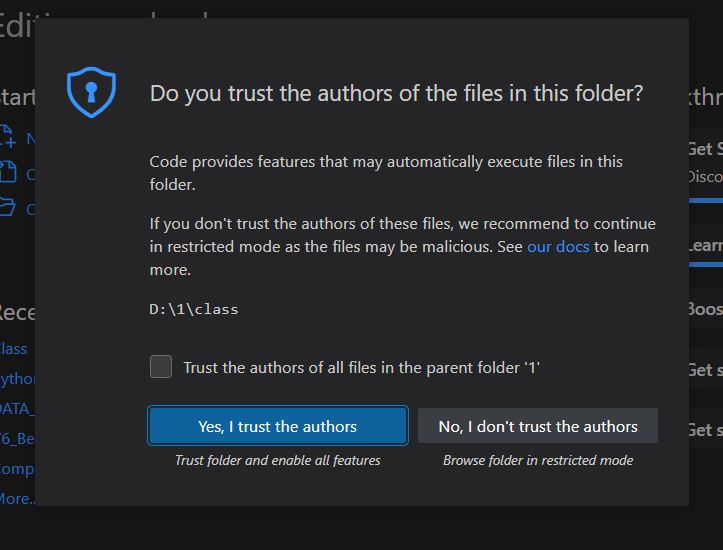
2 . سپس VsCode خود را باز میکنیم و این پوشه را به VsCode خود معرفی میکنیم :



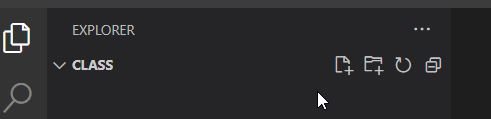
بر روی Open Folder کلیک میکنیم و مسیر پوشه ای که قرار است کدهای خود را در آن قرار دهیم به VsCode میدهیم .

3 . اگر پس از انتخاب پوشه (select folder) با پیام زیر رو به رو شدید ابتدا گزینه

Trust the authors of all files in the parent folder ... را تیک بزنید و سپس بر روی دکمه Yes, I trust the authors کلیک کنید .



4. پس از انجام مراحل بالا پوشه مورد نظر به VsCode اضافه میگردد و ما میتوانیم برنامه های خود را از طریق VsCode بنویسیم در پوشه مورد نظر ذخیره کنیم .



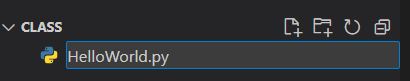
5 . برای ایجاد برنامه جدید در پوشه مورد نظر مانند عکس زیر عمل کنید و اولین پروژه برنامه نویسی خود را ایجاد کنید :



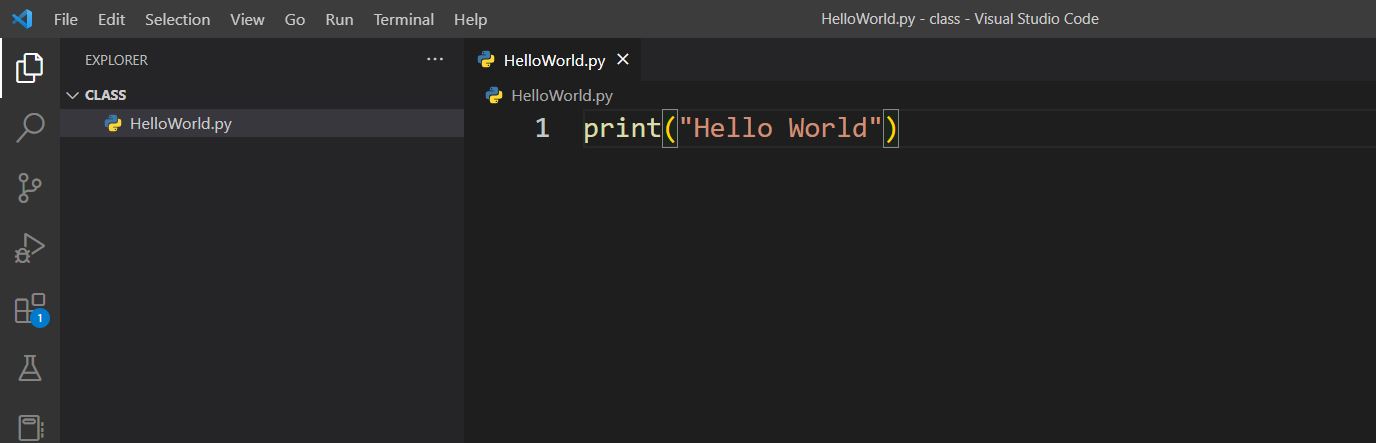
بعد از کلیک بر روی دکمه NewFile یک نام برای برنامه خود انتخاب کنید .

توجه کنید پسوند (format) فایل های برنامه پایتون .py است و حتما باید در انتهای همه فایل های پایتون پسوند py نوشته شود .

من نام برنامه را HelloWorld.py قرار دادم و دکمه enter را زدم تا فایل برای من ساخته شود .

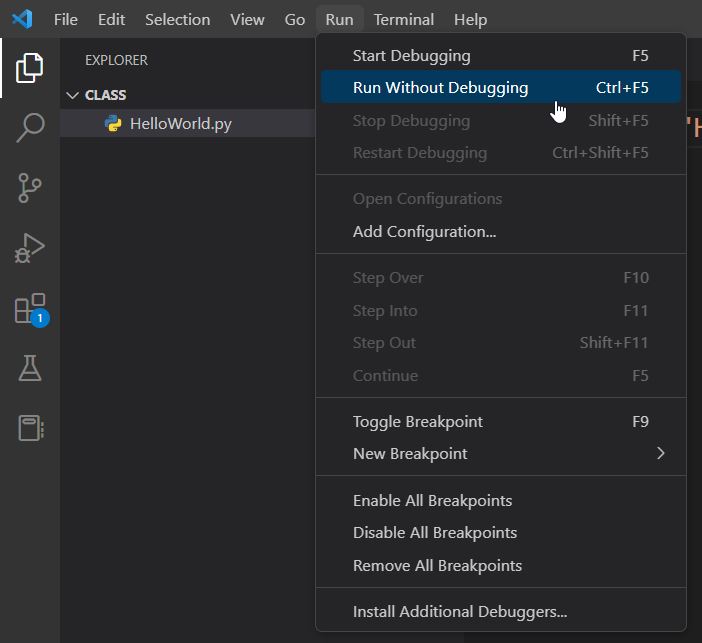


6 . بر روی فایل HelloWorld.py کلیک کنید و در سمت راست کد مورد نظر را مطابق عکس بنویسید .

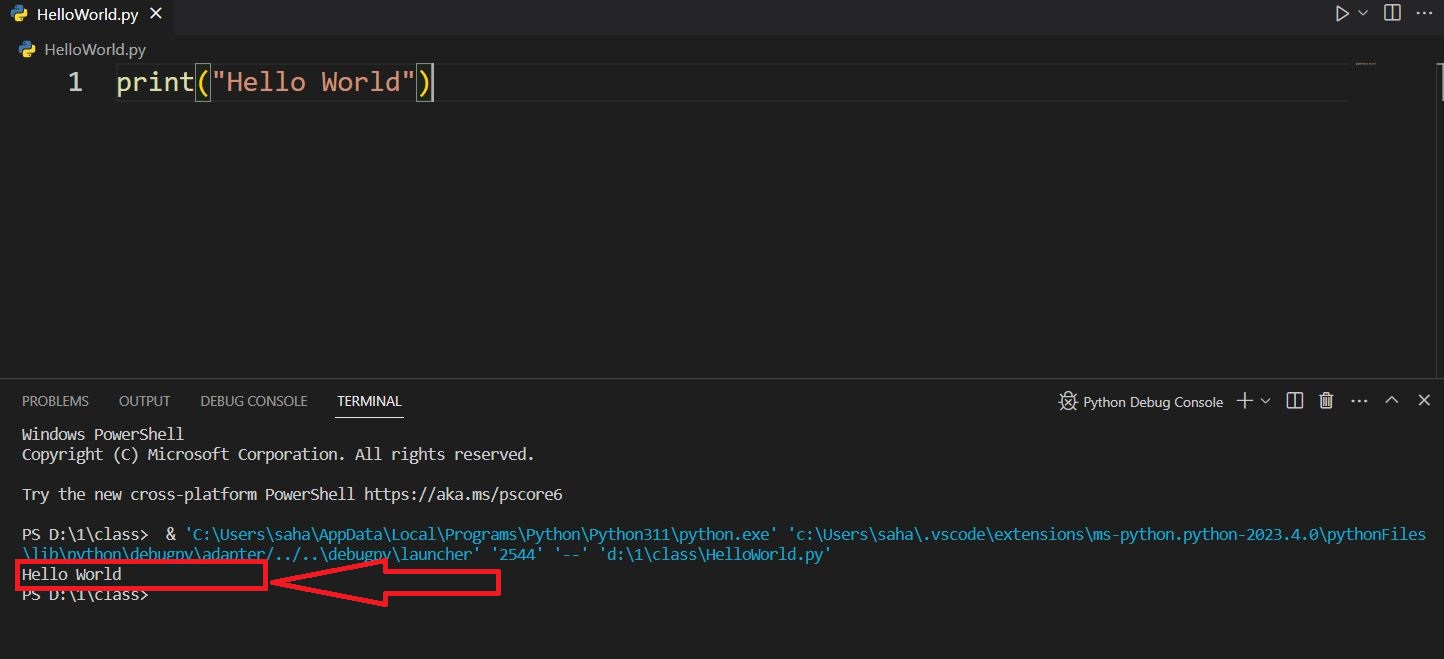


کد print در زبان پایتون چیزی را برای ما نمایش میدهد در این کد همانطور که مشخص است برنامه ای نوشته ایم که Hello World را برای ما نمایش دهد .

7 . برای اجرای کد از نوار ابزار بالا بر روی Run کلیک میکنیم و بر روی Run Without Debugging کلیک میکنیم تا برنامه ما اجرا شود .



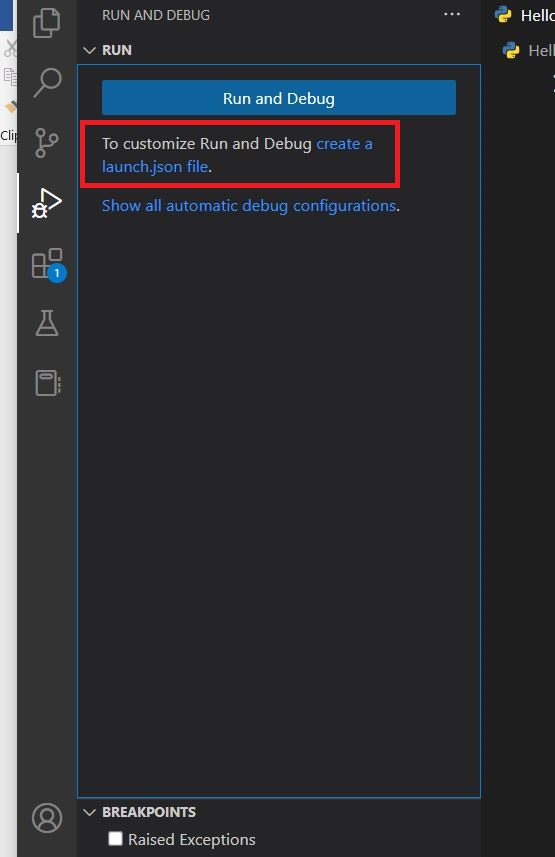
8 . کد اجرا شده باید به صورت زیر در ترمینال داخلی VsCode به نمایش در بیاید :



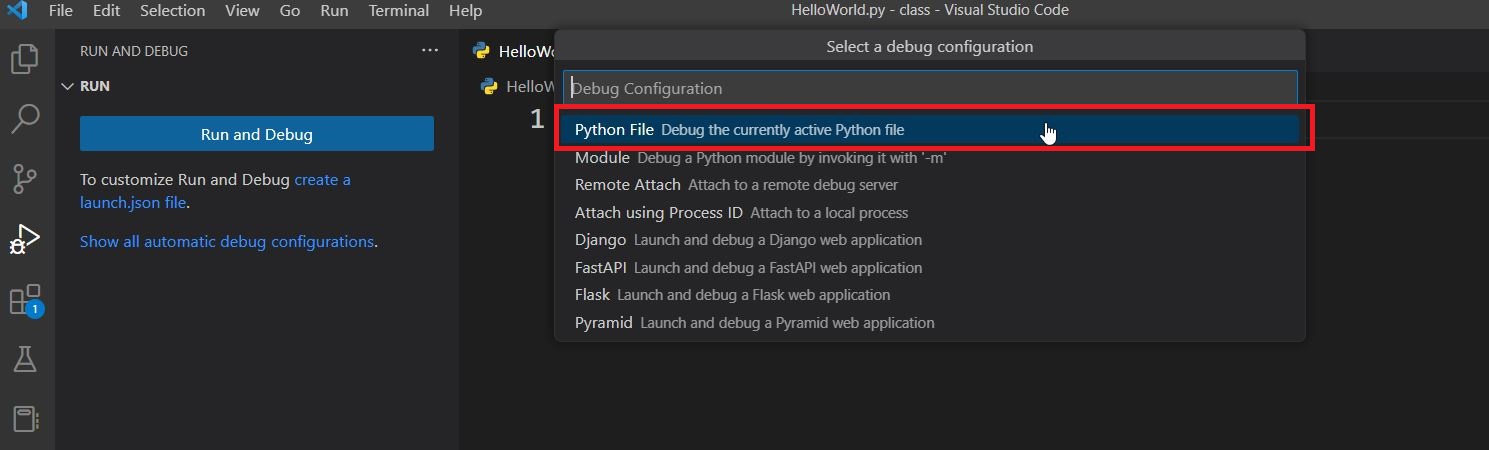
**تغییر محیط اجرا VsCode از حالت Internal به حالت External :**

ما میتوانیم محیط اجرایی VsCode را از حالت ترمینال داخلی ) Internal ) به حالت ترمینال خارجی (External) تغییر دهیم ، این کار به بهتر دیده شدن جواب اجرای برنامه های ما کمک میکند .

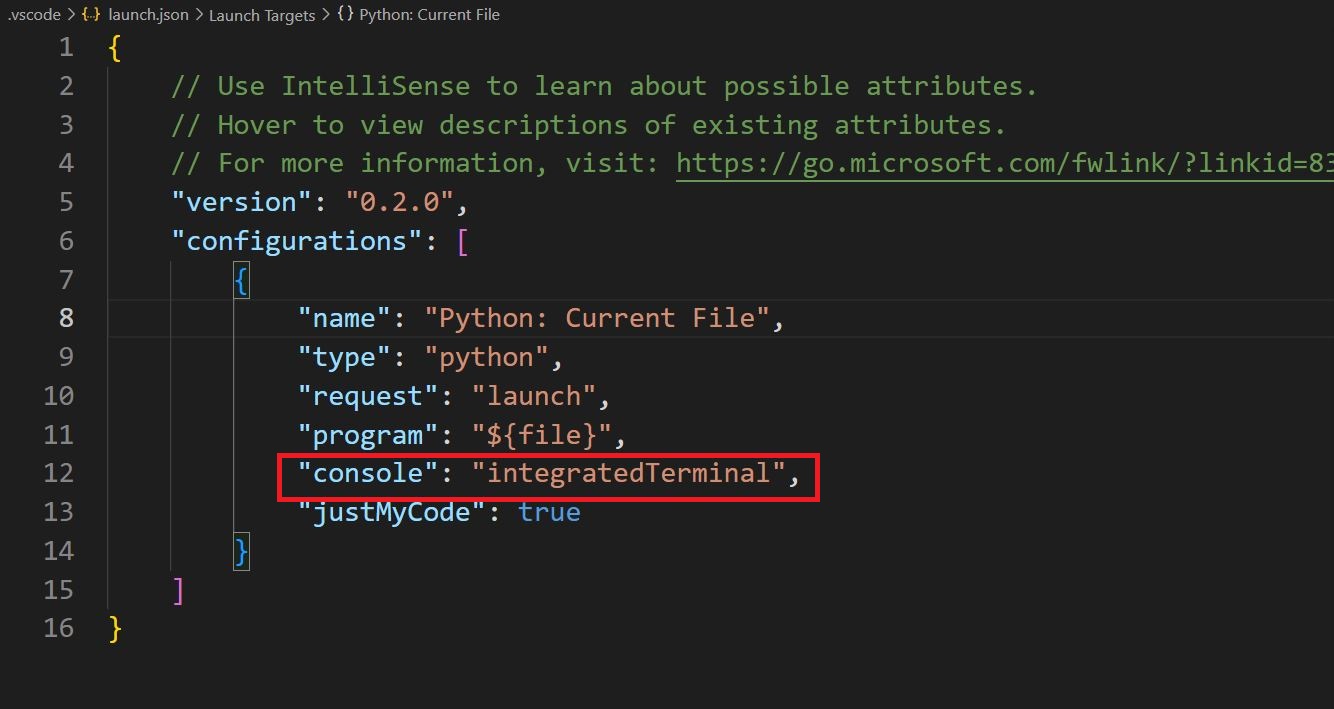
برای این کار در محیط VsCode دکمه های Ctrl + Shift + D را با یکدیگر فشار دهید ، در سمت چپ پنجره ای مانند عکس زیر ظاهر میشود .

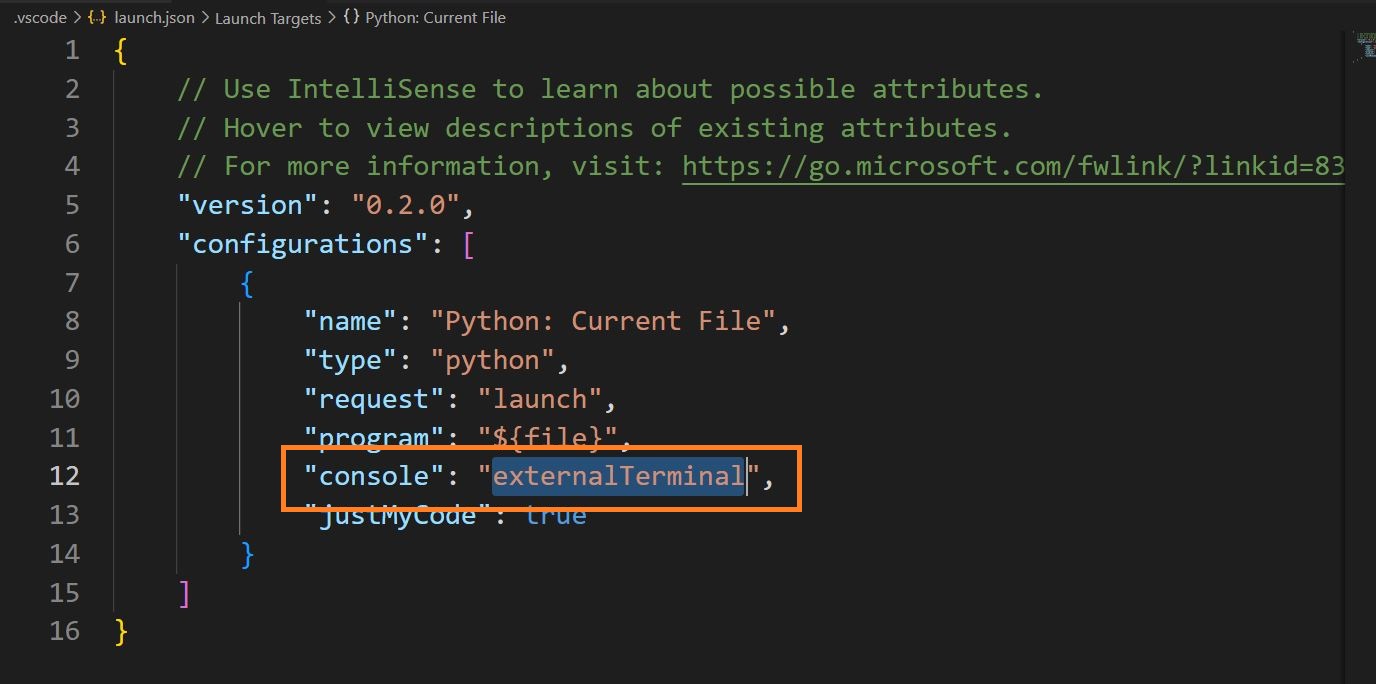


بر روی دکمه Create a launch.json file کلیک کنید ، سپس بر روی python file مطابق عکس زیر کلیک کنید .

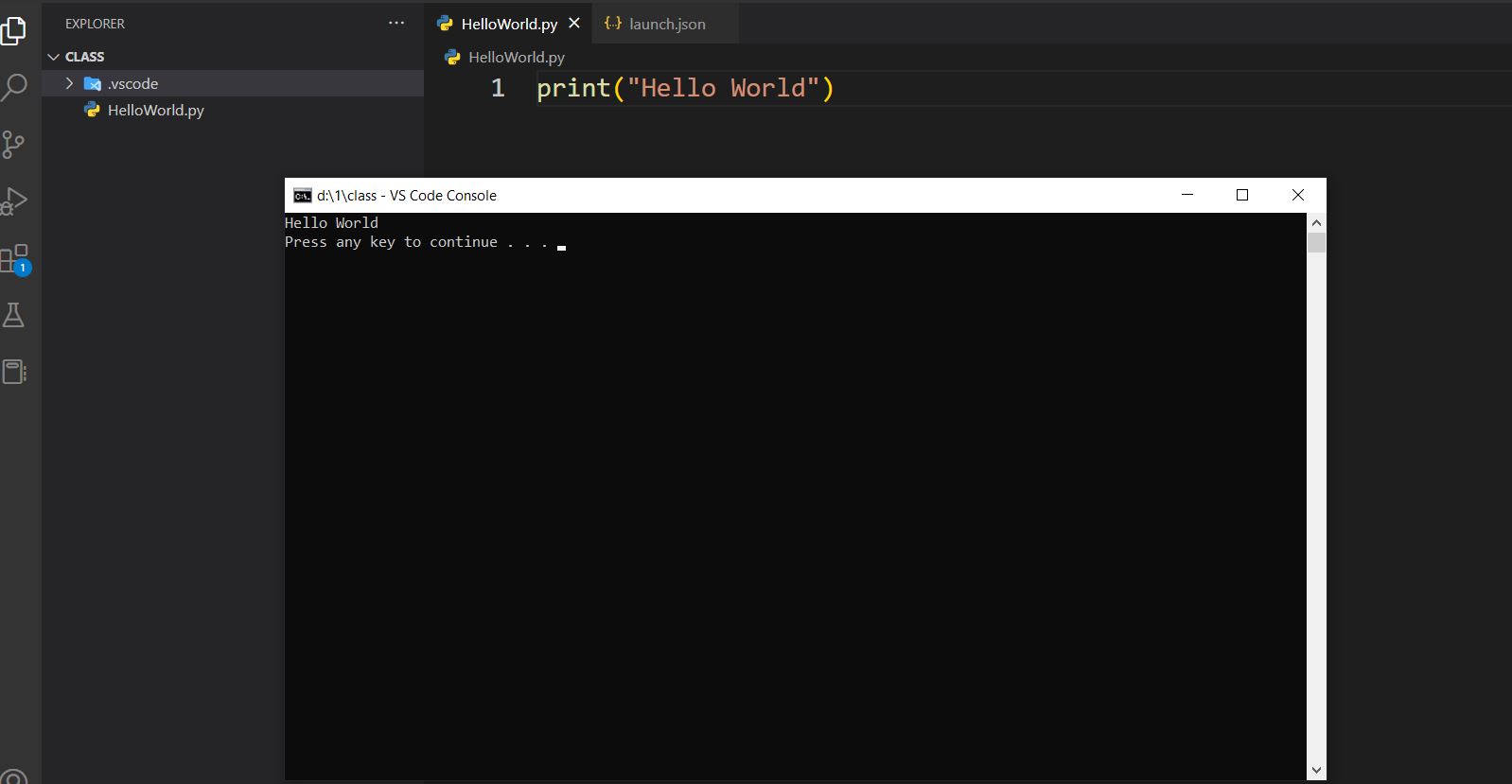


پس از این کار فایل تنظیمات برای شما ساخته میشود ، در قسمت console مقدار داخل " " را به externalTerminal تغییر دهید و سپس برای ذخیرع تغییرات دکمه های ctrl + s را بزنید تا فایل تنظیمات ذخیره گردد .





پس از این تغییرات یک بار برنامه Hello world را اجرا میکنیم مشاهده میکنیم که در ترمینال خارجی این برنامه برای ما اجرا میگردد .



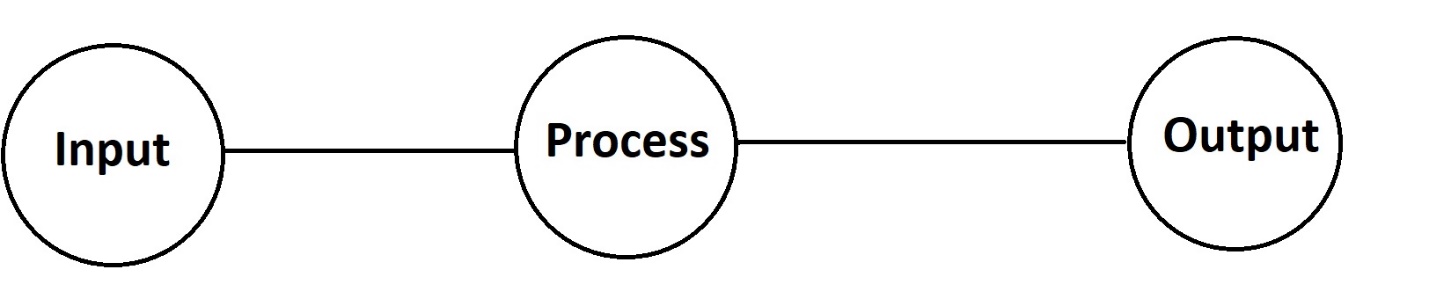
# فصل 4.1 - انواع دیتا در پایتون

هدف ما در برنامه نویسی پایتون به ترتیب در این 3 مرحله خلاصه میگردد :

1 . دریافت **دیتا** (Input)

2 . پردازش **دیتا** (Process)

3 . نمایش **دیتا** (Output)



چیزی که در 3 مرحله بالا مشترک است (دیتا) است ، ما دیتایی را به عنوان ورودی به برنامه خود میدهیم پردازشی بر روی آن دیتا انجام میدهیم و در نهایت نتیجه را نمایش میدهیم .

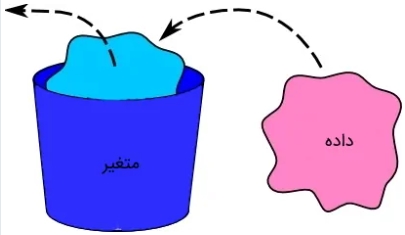
متغیر (variable) : ما در برنامه نویسی نیاز داریم دیتایی را به عنوان ورودی به برنامه خود بدهیم ، ما برای کار با دیتای نیاز به ذخیره سازی آن در حافظه کامپیوتر داریم برای این منظور دیتای خود را در مکان خاصی در حافظه کامپیوتر ذخیره میکنیم .

برای مثال زمانی که میگوییم :

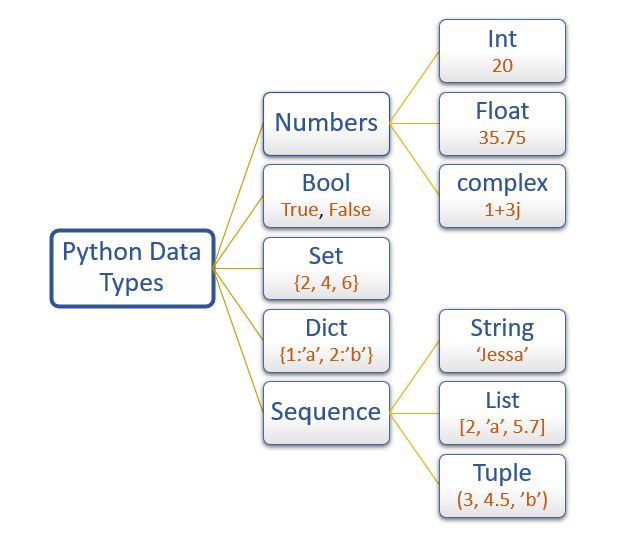
X = 2

ما متغییری به نام X را تعریف کرده ایم که مقدار 2 را در آن قرار داده ایم ، بعد از نوشتن کد بالا کامپیوتر فضایی در رم به X اختصصاص میدهد که مقدار 2 در آن ذخیره سازی شده است .

در واقع متغیر مانند ظرفی است که دیتایی درون آن ریخته میشود و از آن برای ذخیره دیتا در رم کامپیوتر استفاده میگردد .



انواع دیتا در پایتون :



# فصل 4.2 - Numbers and Strings

معرفی متغیرهای Int و String

در این فصل ما بر روی دیتا تایپ های پایه کار میکنیم که شامل اعداد (numbers) و استرینگ (strings) میباشد .

انواع data type : در زبان برنامه نویسی پایتون دیتا (اطلاعات) به چند نوع تقسیم میشوند .

ما در پایتون برای متوجه شدن تایپ دیتا از دستور type(x) استفاده میکنیم .

1 . Int (Integer) : اعداد صحیح

a = 2

print(a)

>> 2

print(type(a))

>> <class 'int'>

2 . float : اعداد اعشاری

b = 2.5

print(b)

>> 2.5

print(type(b))

>> <class 'float'>

در واقع در دو مثال بالا شما دو متغیر(Variables) به نام های a و b تعریف کرده اید که هر کدام عددی را در خود ذخیره کرده اند . متغیر a در خود یک عدد صحیح ذخیره کرده است در حالی که متغیر b در خود یک عدد اعشاری ذخیره کرده است ، پس از تعریف متغیر ما میتونیم از آنها در محاسبات خود استفاده کنیم .

print(a + b)

>> 4.5

print(a +b) \* 2

>> 9.0

print( 2 + 2 + 4 – 2/3)

>> 7.333333333333333

در واقع متغیر ظرفی است برای ذخیره اطلاعات در ram کامپیوتر به این صورت که زمانی که ما در برنامه خود یک متغییر برای مثال a = 2 تعریف میکنیم ظرفی به نام a برای ما ایجاد میشود که مقدار 2 در آن ریخته میشود ما میتوانیم در هر کجای برنامه خود از a که مقدار 2 را در خود ذخیره کرده است استفاده کنیم ، مقدار a تا زمانی که مقداری جدیدی نگرفته است همان 2 میماند ولی اگر در ادامه برنامه ما a = 3 را در برنامه خود بنویسیم مقدار 2 از متغیر a حذف میشود و مقدار جدید جایگزین آن میشود .

print(a + a)

>> 4

ما در پایتون برای تعریف اعداد ثابت برای مثال عدد PI از حروف بزرگ استفاده میکنیم :

PI = 3.14

print(PI \* 2 \* 2)

>> 12.56

عملگرهای ریاضی در پایتون (math operators):

a = 3

b = 5

جمع

print(a + b)

>> 8

تفریق

print(a – b)

>> -2

ضرب

print(a \* b)

>> 15

تقسیم

print(a / b)

>> 0.6

باقی مانده تقسیم

print(a%b)

>> 3

توان

print(a \*\* b)

>> 243

3. String :

پایتون همچنین از دیتا تایپ هایی از نوع نوشته (string) نیز ساپورت میکند . String در واقع دنباله ای از کارکترهایی است که به دنبال هم نوشته میشوند و همچنین میتوانند بین تک کوتیشن ‘ ’ یا دبل کوتیشن “ ” تعریف شود .

hi = “hello”

print(hi)

>> hello

bye = ‘goodbye’

print(bye)

>> goodbye

شما میتوانید دو string را با هم جمع ببندید به این صورت که در کنار هم آنها را نمایش دهد اما امکان جمع بستن متغیر هایی که جنس آنها با یکدگیر متفاوت میباشد وجود ندارد برای مثال جمع sting و integer امکان پذیر نیست .

print( hi + “ world” )

>> hi world

print( “hello” + 3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError : must be str, not int

همانطور که میبینیم در هنگام جمع بستن hello با عدد 3 پایتون به ما ارور must be str, not int را نشان میدهد .

همچنین ضرب در string ها کار تکرار یک string را انجام میدهد .

print( “Hello” \* 3 )

>> HelloHelloHello

تمرین با Integer :

1 . محاسبات ریاضی زیر را انجام دهید و تلاش کنید متوجه شوید هر کدام از اپراتور های ریاضی چه کاری انجام میدهند :

((3 / 2)), ((3 // 2)), ((3 % 2)), ((3\*\*2))

پیشنهاد میشود نگاهی به منابع سایت پایتون به این آدرس بیاندازید :

https://docs.python.org/3/library/ stdtypes.html#numeric-types-int-float-complex

2 . میانگین هر دنباله از اعداد زیر را محاسبه کنید :

(2, 4) , (4, 8, 9) , (12, 14/6, 15)

3 . حجم یک کره با فرمول 4/3 \* pi \* r^3 محاسبه میشود . حجم کره ای با شعاع 5 را محاسبه کنید ، پیشنهاد میشود یک متغیر به نام PI ایجاد کنید که مقدار آن برابر با 3.1415 باشد .

4 . با استفاده از اپراتور % چک کنید کدام یک از اعداد زیر زوج و کدام یک از اعداد فرد هستند :

(1 , 5, 20, 60/7)

راهنمایی : باقی مانده x / 2 همیشه 0 است اگر x زوج باشد .

تمرین با String :

پیشنهاد میشود نگاهی به داکیومنت های پایتون به این آدرس بیاندازید :

https://docs.python.org/3/library/stdtypes. html?#text-sequence-type-str

1 . حروف abc را در متغیری (variable) به نام s ذخیره کنید .

الف : از فانکشنی برای به دست آوردن طول s استفاده کنید . (تعداد حروف)

ب : عملیات های مورد نیازی انجام دهید که عبارت “abc” تبدیل به “aaabbbccc” شود . پیشنهاد میشود از string concatenation , string indexes استفاده شود ( ضرب و ایندکس های استرینگ)

2 . متغیر s را تعریف کنید به طوری که مقدار اولیه آن برابر با “aaabbbccc” باشد .

الف : از فانکشنی برای تغییر حرف a به X در استرینگ s استفاده کنید .

3 . چگونه میتوان همه حروف متغیر s که در بالا تعریف شد را به بزرگ و سپس به کوچک تبدیل کرد ؟

# فصل 5.1 – Lists

پایتون لیست یک نوع ساختار دیتا در پایتون است که قابلیت ذخیره گروهی از دیتاها را در خود دارد به این صورت که هر المنت از لیست میتواند تایپ متفاوتی از سایر المنت ها داشته باشد هرچند معمولا همه المنت های یک لیست از یک جنس تعریف میشوند .

لیست ها با استفاده از براکت ها تعریف میشوند ، ما به هر المنت در لیست از طریق اندیس (index) میتوانیم دسترسی پیدا کنیم به این صورت که اندیس 0 به ما اولین آیتم از لیست را میدهد .

l = [1, 2, 3, 4, 5]

print( l[0] )

>> 1

print ( l[1])

>> 2

تمرین : چگونه میتوان به عدد 4 از لیست دسترسی پیدا کرد ؟

گاهی اوقات شما تنها نیاز به قسمتی از لیست دارید یا به اصطلاح sublist . ما در پایتون برای ایجاد این sublist که قسمت کوچکی از یک لیست بزرگتر است از slicing استفاده میکنیم ،به این صورت که از اندیس اول و آخری که میخواهیم به آنها دسترسی داشته باشیم استفاده میکنیم .

l = [ ‘a’ , ‘b’, ‘c’, ‘d’, ‘e’]

print ( l[1:3] )

>> [‘b’, ‘c’]

در نهایت اوپراتور های ریاضی هم برای لیست ها تعریف شده است به این صورت که اگر ما از جمع استفاده کنیم دو لیست را در کنار هم نمایش میدهد :

[1, 2] + [3,4]

>> [1, 2, 3, 4]

[1,2] \* 2

>> [1, 2, 1, 2]

تمرین با list :

1 . یک لیست به نام l ایجاد کنید که مقادیر روبه رو در آن تعریف شده باشد ( [1, 4, 9, 10, 23] )

پیشنهاد میشود به داکیومنت های سایت پایتون در خصوص لیست نگاهی بیاندازید :

https://docs.python.org/3.5/tutorial/introduction.html#lists

الف . از طریق list slicing یک sublist به صورت [4, 9] و [10, 23] ایجاد کنید .

ب . مقدار 90 را به انتهای لیست اضافه کنید (append) .

ج. میانگین مقادیری که در لیست است را محاسبه کنید . ( شما میتوانید از فانکشن های sum , len استفاده کنید)

د . ساب لیست [4, 9] را حذف کنید .

# فصل 5.2 – List comprehensions

List comprehensions یک روش ساده برای ساخت لیست است . این لیست از دو براکت و یک عبارت که شامل کلمه "for" میباشد تشکیل شده است . نتیجه نهایی آن عبارت یک لیست است که با عبارت نوشته شده برابر است . در اینجا مثالی آورده شده است که مربع (توان 2) اعداد یک لیست را در لیست دیگری ایجاد کرده ایم .

[x \* x for x in [0, 1, 2, 3]]

>> [0, 1, 4, 9]

باتوجه به انعطاف پذیری List comprehensions به صورت عمومی معمولا از فانکشن range استفاده میکنند که یک بازه از اعداد را برای ما باز میگرداند .

[ x\*x for x in range(4)]

>> [0, 1, 4, 9]

گاهی اوقات ممکن است شما بخواهید فیلتری بر روی نتیجه قرار دهید . کلمه “if” میتواند در این کیس ها استفاده شود :

[ x for x in range(10) if x % 2 == 0]

>> [0, 2, 4, 6, 8]

مثال بالا همه اعداد زوج بین بازه (رنج) 0 تا 10 را برای ما بازمیگرداند .

موارد بیشتر در خصوص list comprehensions را میتوانید در آدرس زیر پیدا کنید :

https://docs.python.org/3/tutorial/datastructures.html#listcomprehensions

تمرین با list comprehensions :

1 . با استفاده از list comprehensions لیستی ایجاد کنید که مربع 10 عدد 1 تا 10 را به ما بدهد .

2 . با استفاده از list comprehensions لیستی ایجاد کنید که مکعب 20 عدد 1 تا 20 را به ما بدهد .

3 . یک list comprehensions بسازید که شامل همه اعداد زوج در بازه 0 تا 20 باشد ، و همچنین یک لیست دیگر بسازید که شامل اعداد فرد در همین بازه باشد .

4 . یک لیست بسازید که شامل مربع (توان 2) اعداد زوج در بازه 0 تا 20 باشد .

الف : جمع (sum) اعداد درون لیست را از طریق فانکشن sum به دست آورید .

5 . یک لیست بسازید که شامل مربع (توان 2) اعداد زوج در بازه 0 تا 20 باشد ، اما اعدادی که بخش پذیر بر 3 هستند نباید در لیست وجود داشته باشند ، به عبارت دیگر هر عددی که بر 2 بخش پذیر است در لیست باشد و هر عددی که بر 3 بخش پذیر است در لیست نباشد ، نتیجه باید به این صورت باشد : [4, 16, 64, 100, 196, 256]

پیشنهاد : کلمه “and” را در داکیومنت های پایتون سرچ کنید .

# فصل 6.1 – Modules and functions

در این فصل ما در خصوص ماژول و تابع (فانکشن – function) صحبت خواهیم کرد . فانکشن یک بلاک کد است که کار خاصی را انجام میدهد . یک ماژول یک فایل پایتون است که شامل متغیرها ) variables) و فانکشن ها و موارد دیگری میباشد .

برای شروع vscode خود را باز کنید و کار با ماژول math را که شامل فانکشن های ریاضی است شروع میکنیم :

import math

math.cos(0.0)

>> 1.0

math.radians(275)

>> 4.799655442984406

توابع (function) مجموعه ای از دستورالعمل ها هستند که زمانی که فراخوانی میشود دستورات داخل تابع اجرا میشوند .

در مثال پایین یک فانکشن با نام “do\_hello” تعریف کرده ایم که زمانی که فراخوانی میشود دو عبارت را print میکند .

def do\_hello():

print(“Hello”)

print(“World”)

do\_hello()

>> Hello

>> World

مطمئن شوید قبل از print در فانکشن do\_hello از tab (فاصله) استفاده کرده اید . tab و space در پایتون مشخص میکنند که کد مربوطه به ساختار قبل از خود وابسته است و زیرمجموعه ساختار قبل از خود است . برای مثال ، عبارت print داخل فانکشن do\_hello حتما باید دارای tab باشد که مشخص کند این print مربوط به فانکشن do\_hello میباشد .

فانکشن ها همچنین میتوانند پارامتر هایی دریافت کنند که مقادیری را برمیگرداند (return) :

def add\_one(val):

print(“Function got value”, val)

return val + 1

value = add\_one(1)

>> function got value 1

value

>> 2

تمرین با math module :

پیشنهاد : نگاهی به داکیومنت های پایتون در خصوص ماژول math که در آدرس زیر قرار داده شده است بیاندزید :

https://docs.python.org/3/ library/math.html

1 . بزرگترین مقسوم علیه مشترک از جفت اعداد زیر را به دست آورید : (15، 21) ، (152، 200) ، (1988، 9765)

2 . لگاریتم بر مبنای 2 اعداد زیر را محاسبه کنید : 0 ، 1 ، 2 ، 6 ، 9 ، 15

3 . از طریق فانکشن input از کاربر درخواست کنید یک عدد را وارد کند و پس از آن عدد را برنامه نمایش دهد .

الف ) sin , cos عدد داده شده را محاسبه و نتیجه را نمایش دهید .

مطمئن شوید عدد داده شده توسط کاربر به int کست (cast) شده باشد .

تمرین با function :

1 . فانکشنی با نام add2 ایجاد کنید که دو عدد را به عنوان آرگیومنت دریافت کند و جمع آن اعداد را در خروجی return کند همچنین فانکشنی با نام add3 ایجاد کنید که سه عدد را به عنوان آرگیومنت دریافت کند و جع آنها را نمایش دهد .

2 . فانکشنی پیاده سازی کنید ) Implement) که دو عدد را به عنوان آرگیومنت دریافت کند و عدد بزرگتر را return کند . از عبارت if برای مقایسه دو عدد استفاده کنید .

پیشنهاد : بهتر است داکیومنت پایتون در آدرس زیر را مطالعه کنید :

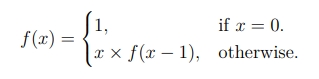
https://docs.python.org/3/ tutorial/controlflow.html#if-statements

3 . فانکشنی با نام is\_divisible پیاده سازی کنید که دو پارامتر با نام های a , b دریافت کند و اگر a بر b بخش پذیر بود true برگداند در غیر این صورت false برگرداند . (از اوپراتور ریاضی % استفاده کنید . )

4 . یک فانکشن با نام average ایجاد کنید که یک لیست را به عنوان آرگیومنت بتوانیم به آن بدهیم و میانگین اعداد درون لیست را به ما بدهد . ( از sum , len میتوانید استفاده کنید )

# فصل 6.2 – توابع بازگشتی (Recursive functins)

. در برنامه نویسی کامپیوتر تابع بازگشتی (Recursive function) به صورت ساده تابعی است که خودش را صدا میزند . برای مثال تابع فکتوریل را میبینیم :



برای مثال ما فکتوریل 5 را محاسبه میکنیم :

5! = 5 \* 4!

= 5 \* 4 \* 3!

= 5 \* 4 \* 3 \* 2!

= 5 \* 4 \* 3 \* 2 \* 1

= 120

فکوریل 5 عبارت است از 5 بار فکتوریل 4 و الی آخر . در نهایت فکتوریل 1 (یا 0) برابر است با 1 که حالت بازگشتی به اتمام میرسد ، در پایتون ما برای این حالت ها میتوانیم تابع بازگشتی (recursive function) بنویسیم .

def factorial(x) :

if x == 0:

return 1

else:

return x \* factorial(x -1)

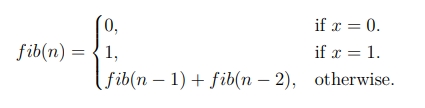
اتفاقی که در تابع بازگشتی میافتد این است که باید یک نقطه پایه (base) وجود داشته باشد که زمانی که فانکشن تمام میشود مجدد بتواند به نقطه پایه برگردد و بازگشت داشته باشد . در این مثال فکتوریل ما میدانیم که فکتوریل 0 میشود 1 ، و فکتوریل عددهای بزرگرتر از 0 بستگی به فکتوریل اعداد قبل دارد تا زمانی که به فکتوریل 0 برسد .

تمرین با recursive function :

1 . فانکشن فکتوریلی پیاده سازی کنید و آن را تست کنید که درست کار کند با استفاده از چندین مقدار متفاوت آن را تست کنید و همچنین اعداد به دست آمده را با ماشین حساب مقایسه کنید .

2 . یک تابع بازگشتی پیاده سازی کنید که جمع n عدد صحیح را به ما بدهد (از 0 شروع شود) ، n در این مثال پارامتر فانکشن ما میباشد . با فکر کردن در خصوص نقطه پایه (base case) کار خود را شروع کنید .

3 . دنباله فیبوناچی عبارت است از اعدادی که هر عدد از حاصل جمع دو عدد قبل به دست می آید . با توجه به عبارت بازگشتی زیر فانکشن fib(n) را پیاده سازی کنید .



. نتیجه را چک کنید برای اعداد اولیه این دنباله :

0 , 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …

# فصل 7.1 – Iteration and loops - For

در این فصل ما در خصوص حلقه (loops) و تکرار صحبت میکنیم ، ما از loop ها در برنامه نویسی کامپیوتر برای اتوماتیک سازی فرآیندهای تکرار پذیر استفاده میکنیم .

در پایتون ما دو نوع حلقه داریم که عبارت است از for و while که در اینجا ما در خصوص حلقه for صحبت میکنیم .

حلقه for به شما اجازه میدهد تمام آیتم های یک لیست را پیمایش کنید و هرکاری که میخواهید میتوانید با آن آیتم ها انجام دهید . برای مثال ، اجازه بدهید یک لیست بسازیم و print کنیم مربع (توان 2) هر آیتمی که در لیست وجود دارد .

for value in [0, 1, 2, 3, 4, 5]

print (value \* value)

>>

0

1

4

9

16

25

این ساختار بسیار ساده اما قدرتمند میباشد . حلقه for پایه بسیار از چیزها در برنامه نویسی میباشد . برای مثال ، شما قبلا از فانکشن sum(list) استفاده کرده اید که همه آیتم های درون یک لیست را جمع میبندد ، اما در اینجا ما همین کار را با حلقه for انجام میدهیم :

mylist = [1, 5, 7]

sum = 0

for value in mylist:

sum = sum + value

print(sum)

>> 13

به صورت کلی ، شما یک متغیر به نام sum ایجاد کردید و هر value را که یک آیتم از لیست بود به آن اضافه کردید .

گاهی اوقات ، به جای استفاده از value در یک لیست ، شما ممکن است بخواهید با ایندکس های یک لیست کار کنید ، اما نکته ای که وجود دارد این است که این آیتم ها درون لیست هستند . در اینجا مثالی میزنیم که پیمایش کنیم بر روی لیست و هر آیتم از لیست را به همراه ایندکس(اندیس) آن نمایش میدهیم .

mylist = [1, 5, 7]

for i in range(len(mylist)):

print( “Index: “, i, “Value: “, mylist[i])

>> Index : 0 Value: 1

>> Index : 1 Value: 5

>> Index : 2 Value: 7

شما میتوانید بببنید که ما بر روی خود لیست پیمایش نکردیم اما بر روی “range” ی که از طول (length) به دست آمده است پیمایش انجام داده ایم . فانکشن range به ما یک لیست به خصوص را میدهد .

list(range(3))

>> [0, 1, 2]

بنابراین ، زمانی که شما از range استفاده میکنید شما بر روی mylist پیمایش نمیکنید بلکه بر روی اعداد لیست با استفاده از ایندکس ها شما به آیتم های لیست دسترسی پیدا میکنید .

پیشنهاد : برای اطلاع پیدا کردن بیشتر در خصوص range میتوانید به داکیومنت زیر در سایت پایتون مراجعه کنید :

https://docs.python.org/3/tutorial/controlflow. html#the-range-function

. گاهی اوقات ممکن است شما دو چیز را لازم داشته باشید ( ایندکس ها و مقادیر ) indexes and values ، در این زمان ها شما میتوانید از فانکشن enumerate function استفاده کنید .

این فاکشن کار شمارش را برای ما انجام میدهد .

mylist = [1, 5, 7]

for i, value in enumerate(mylist):

print(“Index:” , i, “Value: “, value)

>> Index : 0 Value : 1

>> Index : 1 Value : 5

>> Index : 2 Value : 7

این را به یاد داشته باشید که اولین مقدار در پایتون لیست همیشه ایندکسی برابر با 0 دارد .

# فصل 7.2 – Iteration and loops - While

همچنین ما حلقه های while را داریم که به ما این اجازه را میدهند که یک عبارت را تکرار کنیم در شرایطی که یک شرط مشخص true (درست) است .

برای مثال ، برنامه پایین از عدد 10 شروع میکند و تا زمانی که n بزرگتر از 0 است اعداد را نمایش میدهد و هر بار 1 را از n کم میکند زمانی که n بزرگتر از 0 است شرط n > 0 برقرار است و زمانی که n = 0 میشود حلقه متوقف میشود .

n = 10

while n > 0:

print(n)

n = n -1

>>

10  
9  
8  
7  
6  
5  
4  
3  
2  
1

هیچگاه باتوجه به شرطی که در عبارت while ست شده است 0 برای ما به نمایش در نخواهد آمد .

تمرین با حلقه for :

پیشنهاد : برای این فصل شاید مشتاق باشید که داکیومنت های پایتون را در آدرس زیر مطالعه کنید :

https://docs.python.org/3/tutorial/controlflow.html#for-statements

1 . یک فانکشن با نام add ایجاد کنید که یک لیست را به عنوان پارامتر دریافت کند و جمع همه المان های درون لیست را به شما نمایش دهد . از حلقه for برای پیمایش بر روی المنت ها استفاده کنید .

2 . یک فانکشن با نام دلخواه ایجاد کنید که یک لیست را به عنوان پارامتر دریافت کند و max مقداری که درون لیست است را برای ما برگرداند . هر بار که بر روی لیست پیمایش میکنید ممکن است بخواهید ماکزیموم مقدار پیدا شده را ذخیره کنید تا با مقدار بعدی در پیمایش مقایسه کنید .

3 . فانکشنی مانند تمرین قبل ایجاد کنید که در خروجی برای ما یک لیست برگرداند که اولین مقدار آن max value باشد که در تمرین قبلی پیدا کرده ایم و دومین مقدار ایندکسی باشید که این value در لیست دارد ، ما در قبل max value را پیدا کرده ایم در اینجا صرفا نیاز داریم که موقعیت آن را در لیست پیدا کنیم .

4 . فانکشنی پیاده سازی کنید که یک لیست را به عنوان ورودی دریافت کند و معکوس آن را برگرداند . برای این کار شما میتوانید یک لیست خالی ایجاد کنید و المنت ها را بر اساس نظم به لیست خالی اضافه کنید .

پیشنهاد : برای چک کردن آنچه که میتوانیم با لیست ها انجام دهیم میتوانید لینک زیر را مطالعه نمایید :

https: //docs.python.org/3/tutorial/datastructures.html#more-on-lists

5 . فانکشنی با نام is\_sorted ایجاد کنید که یک لیست را به عنوان پارامتر دریافت کند و اگر اعداد در این لیست به ترتیب باشند True برگرداند . برای مثال [1,2,2,3] به ترتیب است در صورتی که [1,2,3,4] به ترتیب نمیباشد . پیشنهاد : شما باید اعداد درون لیست را با عدد بعدی مقایسه کنید .

6 . یک فانکشن به نام is\_sorted\_dec ایجاد کنید که مانند فانکشن بالا یک لیست را به عنوان ورودی دریافت کند و آن را به ترتیب بزرگتر به کوچکتر (معوکس) مرتب سازی کند و برگرداند .

7 . یک فانکشن به نام has\_duplicates ایجاد کنید که یک لیست را به عنوان پارامتر دریافت کند و چک کند آیا لیست مقادیر تکراری دارد یا خیر . شما ممکن است نیاز داشته باشید که دو حلقه for تعریف کنید که به ازای هر مقدار value شما باید چک کنید مقدار تکراری درون لیست دارید یا خیر .

تمرین با حلقه while :

1 . یک فانکشن پیاده سازی کنید که یک عدد دریافت کند به عنوان پارامتر و آن را نمایش دهد .

هر بار یکی از عدد اصلی کم کند و به ترتیب نمایش دهد که عدد مورد نظر فرد است یا زوج تا زمانی که عدد به 0 میرسد .

>> even\_odd(10)

Even number : 10

Odd number : 9

Even number : 8

Odd number : 7

Even number: 6

Odd number: 5

Even number: 4

Odd number: 3

Even number: 2

Odd number: 1

# فصل 8 – Dictionaries

در این فصل ما با دیکشنری در پایتوت کار میکنیم . Dictionary یک ساختار دیتا data strucature در پایتون است که در آن ایندکس ها به جای عدد از کلید (key) استفاده میکنند . مثال پایین یک دیکشنری را نشان میدهد که ایندکس های آن نام دانش آموزان و مقادیر value های آن سن دانش آموزان را نشان میدهد .

ages = {

"Peter": 10,

"Isabel": 11,

"Anna": 9,

"Thomas": 10,

"Bob": 10,

"Joseph": 11,

"Maria": 12,

"Gabriel": 10,

}

print(ages["Peter"])

>>10

امکان این وجود دارد که بر روی محتوای dictionary پیمایش کنیم از طریق "items" مانند زیر :

for name, age in ages.items():

print(name, age)

>>

Peter 10

Isabel 11

Anna 9

Thomas 10

Bob 10

Joseph 11

Maria 12

Gabriel 10

کلید ها key در دیکشنری همیشه لازم نیست از نوع string باشند و از هر تایپی میتوانند باشند برای مثال در نمونه پایین کلید یک int و value یک لیست میباشد .

d = {

0: [0, 0, 0],

1: [1, 1, 1],

2: [2, 2, 2],

}

d[2]

>> [2, 2, 2]

همچنین شما میتوانید در قسمت value در دیکشنری از یک دیکشنری دیگر استفاده کنید .

students = {

"Peter": {"age": 10, "address": "Lisbon"},

"Isabel": {"age": 11, "address": "Sesimbra"},

"Anna": {"age": 9, "address": "Lisbon"},

}

students['Peter']

>> {'age': 10, 'address': 'Lisbon'}

students['Peter']['address']

>> 'Lisbon

این ساختار در خصوص پیاده سازی اطلاعات سلسله مراتبی کاملا مفید است .

تمرین با dictionary :

پیشنهاد : بهتر است داکیومنت های موجود در خصوص dictionary را در مسیر زیر مطالعه کنید :

https://docs.python.org/3/library/stdtypes.html# mapping-types-dict

. دیکشنری زیر را در نظر بگیرید :

ages = {

"Peter": 10,

"Isabel": 11,

"Anna": 9,

"Thomas": 10,

"Bob": 10,

"Joseph": 11,

"Maria": 12,

"Gabriel": 10,

}

1 . چه تعداد دانش آموز در این دیکشنری وجود دارند ؟ تلاش کنید از طریق فانکشن len به این سوال جواب دهید .

2 . فانکشنی پیاده سازی کنید که دیکشنری ages را به عنوان پارامتر دریافت کند و میانگین سن دانش آموزان را برگرداند . با استفاده از متد items همانطور که در بالا توضیح داده شد امکان دسترسی به سن دانش آموزان را داریم .

3 . فانکشنی پیاده سازی کنید که دیکشنری ages را به عنوان پارامتر دریافت کند و نام پیرترین (بالاترین سن) دانش آموز را برگرداند .

4 . فانکشنی پیاده سازی کنید که دیکشنری ages و یک عدد n را به عنوان پارامتر دریافت کند و یک دیکشنری جدید برگرداند که هر دانش آموز n سال پیرتر شده است . برای مثال

new\_ages(ages, 10) در واقع جواب نهایی دیکشنریی مانند ages خواهد بود با این تفاوت که سن هر دانش آموز 10 سال بزرگتر از مقادیر ages خواهد بود .

تمرین با sub-dictionaries :

دیکشنری زیر را در نظر بگیرید :

students = {

"Peter": {"age": 10, "address": "Lisbon"},

"Isabel": {"age": 11, "address": "Sesimbra"},

"Anna": {"age": 9, "address": "Lisbon"},

}

1 . چه تعداد دانش آموز در دیکشنری وجود دارند ؟ تلاش کنید با استفاده از فانکشن مناسب این کار را انجام دهید .

2 . فانکشنی پیاده سازی کنید که دیکشنری students را دریافت کند و سن میانگین دانش آموزان را برگرداند .

3 . فانکشنی پیاده سازی کنید که دیکشنری studneds و یک آدرس را به عنوان پارامتر دریافت کند و یک دیکشنری برگرداند که مقدار key آن آدرس داده شده و مقدار value آن لیستی از همه دانش آموزانی باشد که در دیکشنری studends وجود دارد .

# فصل 9.1 – Classes

در برنامه نویسی Object oriented یا شی ء گرا (OOP) ، یک class ساخاری است که به ما اجازه میدهد دسته بندی انجام دهیم که این دسته بندی بر روی properties ها که در کلاس ما به آنها attribute گفته و فانکشن ها که در کلاس ما به آن method میگویم انجام میگردد .

کلاس پایین را در نظر بگیرید که ما یک شخص person را تعریف کردیم همراه با properties های name و age و همراه با متدی که نام آن greet است .

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def greet(self):

print("Hello, my name is %s!" % self.name)

اکثریت کلاس ها نیاز به متد سازنده (constructor method ) که با (\_\_init\_\_) مشخص میشود دارند تا به attribute های کلاس مقدار اولیه داده شود ، در مثال قبل cunstructor ی که در کلاس تعریف شده است نام شخص (person name) و سن شخص (age) را دریافت کرده است و آن را در نمونه ای (instance) از کلاس ذخیره سازی کرده است . (مراجعه شود به کلمه self در کد نوشته شده) ، در نهایت متد greet نام شخص ذخیره شده در instance مورد نظر را نمایش میدهد .

instance های کلاس از طریق instance سازی object سازی ایجاد میشوند . در این جا ما میتوانیم دو obj از کلاس را بسازیم :

a = Person("Peter", 20)

b = Person("Anna", 19)

a.greet()

>> Hello, my name is Peter!

b.greet()

>> Hello, my name is Anna!

print(a.age) # We can also access the attributes of an object

20

تمرین با class :

پیشنهاد : استفاده کنید از داکیومنت های پایتون در خصوص class ها که در آدرس زیر آمده است :

https://docs.python.org/3/tutorial/classes

1 . کلاسی به نام Rectangle پیاده سازی کنید که مختصات یک مستطیل را که به صورت (x1, y1) , (x2, y2) است در خود ذخیره کند .

2 . متد سازنده کلاس class constructor را با استفاده از پارامتر های (x1, x2, y1, y2 ) ایجاد کنید و آنها را در هر instance از کلاس ذخیره کنید با استفاده از کلمه “self”

3 . متد های width() , height() را پیاده سازی کنید که به ترتیب طول و ارتفاع مستطیل را برگدانند . دو object از instance Rectangle بسازید و این دو متد را تست کنید .

4 . متد مساحت را با نام area پیاده سازی کنید به صورت که مساحت مستطیل را برگداند .

5 . متد محیط را با نامه circumference ایجاد کنید به صورتی که محیط مستطیل را برگداند .

6 . یکی از Obj های ایجاد شده را در print قرار دهید تا ببنید چه چیزی برای شما نمایش میدهد . متد \_\_str\_\_ متدی است که زمانی که شما از obj های خود print میگیرید به شما نمایش داده میشود ، متد \_\_str\_\_ را به گونه ای پیاده سازی کنید که زمانی که از obj پرینت میگیریم مختصات را به صورت (x1, y1) و (x2, y2) به شما نمایش دهد .

# فصل 9.2 – Inheritance - وراثت

در برنامه نویسی شیء گرا OOP ، وراثت inheritance حالتی است که هر subclass میتواند از کلاسی دیگر attribute ها و method های آن را به ارث ببرد و به کلاس اجازه میدهد عملکرد کلاس بالاتر خود را داشته باشد . برای مثال از کلاس person که در بالا تعریف شد ما میتوانیم یک کلاس زیرکلاس subclass تعریف کنیم که افراد با 10 سال سن را در خود جای داده باشد :

class TenYearOldPerson(Person):

def \_\_init\_\_(self, name):

super().\_\_init\_\_(name, 10)

def greet(self):

print("I don't talk to strangers!!")

نشانه اینکه کلاس TenYearOldPerson یک subclass از کلاس Person است در خط اول مشخص است ، در داخل پرانتز آمده است که این کلاس از کلاس Person ارث بری کرده است .

سپس ما متد constructor کلاس TenYearOldPerson را نوشته ایم که فقط یک name که نام شخص است دریافت میکند ، اما در نهایت ما از constructor کلاس پدر یا superclass استفاده کرده ایم که در آن مشخص کرده ایم که name هما نامی است که در constuctor گرفته ایم و 10 سن شخص است ، در نهایت متد greet مجدد پیاده سازی شده است .

تمرین با Inheritance :

از کلاس Rectangle استفاده کنید برای تمرینات زیر :

1 . یک کلاس به نام Square (مربع) ایجاد کنید که از کلاس Rectangle ارث بری کند .

2 . متد سازنده کلاس constructor را برای کلاس Square پیاده سازی کنید . متد constructor فقط باید دارای x1, y1 و همچنین سایز مربع .

3 . دو Obj بسازید از کلاس Sauare و متد area را فرابخوانید و آن را نمایش دهید .

# فصل 10.1 – Iterators

همانطور که در قبل دیدید ،در پایتون ما از حلقه for برای پیمایش iterate بر روی محتوای یک object استفاده میکردیم .

for value in [0, 1, 2, 3, 4, 5]:

print(value)

>>

0

1

4

9

16

25

Object هایی که میتوانند با حلقه for پیمایش شوند iterators نامیده میشوند یعنی این Object ها قابلیت پیمایش دارند برای مثال ما میتوانیم بر روی همه آیتم های یک لیست پیمایش کنیم . بنابراین یک iterator عبارت است از یک object که از پروتکل تکرار iteration پیروی میکند .

فانکشن iter که یک فانکشن built-in (فانکشن هایی که به صورت پایه ای در زبان پایتون وجود دارند) است میتواند برای ساخت Object های با قابلیت پیمایش iterator object استفاده گردد ، در صورتی که فانکشن next میتواند برای پیمایش تدریجی بر روی محتوا مورد استفاده قرار گیرد .

my\_iter = iter([1, 2, 3])

print(my\_iter)

>> <list\_iterator object at 0x10ed41cc0>

next(my\_iter)

>> 1

next(my\_iter)

>> 2

next(my\_iter)

>> 3

next(my\_iter)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

StopIteration

اگر آیتم دیگری وجود نداشته باشد که با متد next آن را نمایش دهد ، iterator ارور StopIteration را نمایش میدهد .

# فصل 10.2 – Iterators classes

Iterator قابلیت این را دارد که به صورت یک کلاس پیاده سازی شود . شما فقط نیاز به پیاده سازی 2 متد در کلاس خود دارید :

متد \_\_next\_\_ و متد \_\_iter\_\_

در اینجا یک مثال از کلاسی است که فانکشن range را شبیه سازی میکند که همه مقادیر بین a تا b را برای ما برمیگرداند .

class MyRange:

def \_\_init\_\_(self, a, b):

self.a = a

self.b = b

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self):

if self.a < self.b:

value = self.a

self.a += 1

return value

else:

raise StopIteration

به صورت کلی با هر بار صدا زدن next ، به variable بعدی به جلو حرکت میکند و a را برای ما برمیگرداند . زمانی که مقدار a < b برابر با false شد ارور StopIteration را برای ما نمایش میدهد .

myrange = MyRange(1, 4)

next(myrange)

>> 1

next(myrange)

>> 2

next(myrange)

>> 3

next(myrange)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

StopIteration

اما یک موضوع مهمتر این است که شما میتوانید iterator class را در یک حلقه for استفاده کنید :

for value in MyRange(1, 4):

print(value)

>>

1

2

3

تمرین با iterators :

1 . یک iterator class پیاده سازی کنید که مربع (توان 2) همه اعداد بین a تا b را برای ما برگرداند .

2 . یک iterator class پیاده سازی کنید که همه اعداد زوج بین 1 تا n را برای ما برگرداند .

3 . یک iterator class پیاده سازی کنید که همه اعداد فرد بین 1 تا n را برای ما برگرداند .

4 . یک iterator class پیاده سازی کنید که همه اعداد بین n تا 0 را برای ما برگرداند (به این صورت باشد که از عدد بزرگتر به عدد کوچکتر برای ما نمایش دهد)

5 . یک iterator class پیاده سازی کنید که دنباله فیبوناچی از آیتم اول تا آیتم n ام را برای ما برگرداند . شما میتواند چک کنید تعریف دنباله فیبوناچی را در قسمت فانکشن ها . این ها اعداد اولیه دنباله فیبنوناچی هستند :

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …

6 . یک iterator class پیاده سازی کنید که تمام جفت اعداد متوالی را از 0 تا n به صورت زیر برای ما برگرداند :

(0, 1) , (1, 2), (2,3) ….

# فصل 11 – Generators

اگر شما فصل قبل را مطالعه کرده باشید ،میدانید که iterators آبجکت هایی هستند که به طور منظم با حلقه های for مورد استفاده قرار میگیرند. در سوی دیگر ،iterator ها آبجکت هایی هستند که پروتوکل های iteration را پیاده سازی میکنند .حال generator ها در پایتون یک راه راحت برای پیاده سازی یک iterator است . به جای classs ،generator یک فانکشن است که یک مقدار را هر لحظه بر میگرداند . کلمه yield در generator مورد استفاده قرار میگیرد . در اینجا مثالی از generator وجود دارد که اعداد موجود بین دو عدد را نشان میدهد :

def myrange(a, b):

while a < b:

yield a

a += 1

مانند iterator ها generator ها میتوانند با حلقه های for مورد استفاده قرار بگیرند .

for value in myrange(1, 4):

print(value)

>>

1

2

3

در واقع میتوان گفت که generator ها شبیه iterator ها رفتار میکنند .

seq = myrange(1,3)

next(seq)

>> 1

next(seq)

>> 2

next(seq)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

StopIteration

موضوع جالب در خصوص generator ها کلمه ی yield است . کلمه yield خیلی شبیه return عمل میکند ،اما شبیه return نیست، این کلمه به فانکشن اجازه میدهد بعد از اتمام مجدد کار خود را از سر بگیرد و متوقف نشود . به عبارت دیگر ، هر زمان که مقدار بعدی از یک generator مورد نیاز است ،پایتون فانکشن را بیدار میکند و فانکشن اجرای خود را از خط yield از سر میگیرد گویی که هرگز از فانکشن خارج نشده است .

فانکشن های generator میتوانند فانکشن های دیگری را درون خود داشته باشند . برای مثال ،بسیار رایج است از فانکشن range برای پیمایش iterate بر روی دنباله ای از اعداد استفاده کنیم :

def squares(n):

for value in range(n):

yield value \* value

تمرین با generator :

1 . یک generator پیاده سازی کنید که نام آن squares باشد که مربع (توان 2) همه اعداد از a تا b را برای ما yield کند .این generator را با یک حلقه for تست کنید و هر مقدار yield شده را print کنید .

2 . یک generator بسازید که همه اعداد زوج بین 1 تا n را yield کند .

3 . یک generator دیگر بسازید که همه اعداد فرد بین 1 تا n را yield کند .

4 . یک generator پیاده سازی کنید که همه اعداد بین n تا 0 را برگرداند .

5 . یک generator بسازید که دنباله فیبوناچی را برای ما برگرداند که از المان اول شروع شده باشد و تا n المان پیش برود .

اعداد ابتدایی این دنباله به این صورت میباشد :

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,89, …

6 . یک generator پیاده سازی کنید که جفت اعداد زیر را از 0 تا n به صورت زیر برای ما برگرداند :

(0, 1), (1, 2), (2, 3). . .

# فصل 12.1 – Coroutines

coroutine در پایتون شبیه به generator ها هستند اما به جای تولید دیتا ، بیشتر به عنوان مصرف کنندگان دیتا استفاده میشوند . به عبارت دیگر ،coroutine فانکشن هایی هستند که شروع به کار میکنند هربار که یک مقدار با استفاده از متد send به آن ارسال شود .

موضوعی که در خصوص coroutine وجود دارد این است که از کلمه yield در سمت راست مساوی قرار داده شده است . در اینجا مثالی از coroutine وجود دارد که print میکند مقادیر که به آن ارسال شده است :

def coroutine():

print('My coroutine')

while True:

val = yield

print('Got', val)

co = coroutine()

next(co)

>> My coroutine

co.send(1)

>> Got 1

co.send(2)

>> Got 2

co.send(3)

>> Got 3

صدا زدن ابتدایی next ضروری است تا coroutine به جلو حرکت کند . همانطور که میبینید coroutine اجرا شد و عبارت های مورد نظر را print کرد . در نهایت کد به عبارت yield میرسد جایی که منتظر میماند تا مجدد اجرا شود . سپس هر زمانی که یک value به coroutine ارسال شود (از طریق send) ، فانکشن coroutine کار خود را از yield از سر میگیرد ، value ارسالی را در val کپی میکند و آن را print میگیرد .

coroutine ها میتوانند بسته شوند با استفاده از متد close() .

co.close()

co.send(4)

>> Traceback (most recent call last):

File "<stdin>", line 1, in <module>

StopIteration

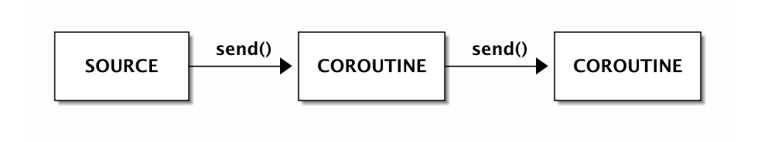
تمرین با coroutine :

1 . یک coroutine به نام square ایجاد کنید که print کند مربع (توان 2) هر value یی که به آن ارسال میشود .

2 . پیاده سازی کنید coroutine ی به نام minimize که مقادیر را درون خود نگهداری و ذخیره کند و در نهایت print کند minimum مقداری که به فانکشن ارسال شده است .

# فصل 12.2 – Pipelines – خط لوله

از Coroutine ها میتوان برای پیاده سازی خط لوله انتقال دیتا استفاده کرد زمانی که یک coroutine دیتا را به coroutine بعدی ارسال میکند در یک خط لوله انتقال دیتا . Coroutine ها با استفاده از متد send دیتا را در لوله های انتقال دیتا هدایت میکنند .



. مثالی میزنیم از یک خط لوله کوچک جایی که یک مقدار ارسال میشود به courotine که مربع (توان 2) دیتا را تولید میکند و پس از آن دیتا را send میکند به coroutine ی که مصرف کنند دیتا است که آن را در نهایت print میکند .

def producer(consumer):

print("Producer ready")

while True:

val = yield

consumer.send(val \* val)

def consumer():

print("Consumer ready")

while True:

val = yield

print('Consumer got', val)

Coroutine های بالا باید حتما " آغاز بشوند" با استفاده از next قبل از آنکه هر value ای با استفاده از send به couroutine ها ارسال شوند .

cons = consumer()

prod = producer(cons)

next(prod)

>> Producer ready

next(cons)

>> Consumer ready

prod.send(1)

>> Consumer got 1

prod.send(2)

>> Consumer got 4

prod.send(3)

>> Consumer got 9

همچنین ، با استفاده از coroutine دیتا میتواند به مقاصد متفاوتی ارسال شود . در مثال زیر دو مصرف کننده دیتا (consumer) تعریف شده است ،اولین مصرف کننده فقط print میکند اعداد در رنج 0 تا 10 را و دومین مصرف کننده فقط اعداد بین 10 تا 20 را print میکند .