



PYTHON

ArcGIS Pro Notebook



ناهید نعمتی کوتنائی (تپسا)
دکتری جغرافیا و برنامه‌ریزی شهری
برنامه‌ریز شهری و تحلیلگر GIS

 [Dr.nemati.K](#)
 [@Nemati_k](#)
 ۰۹۱۱۲۲۳۰۷۹۸



محمدطاهر طاهرپور
دانشجوی ارشد مدیریت شهری
دانشگاه تهران

 [mtaherpoor](#)
 [@mtaherpoor](#)
 ۰۹۳۳۶۱۴۴۹۴۷



فهرست مطالب:

۳ Fundamentals of Python
۳ پایتون چیه؟
۳ معرفی چند منبع یادگیری پایتون
۳ Python Interfaces in ArcGIS Pro
۴ مرحله اول: ساخت پروژه برای کار با Notebook در ArcGIS Pro
۶ مرحله دوم: مرور اصول پایه زبان برنامه‌نویسی Python
۶ متغیرها یا Variables
۷ انواع داده یا data types
۸ توابع داخلی یا Built-in در پایتون
۹ تبدیل داده یا data Conversion
۱۰ عملگرها یا Operators در پایتون
۱۲ شرایط یا Conditionals در پایتون
۱۳ حلقه‌ها یا loops در پایتون
۱۴ توالی یا Sequences در پایتون
۱۵ عملگرها و متدهای انواع داده در پایتون
۱۶ Modules یا ماژولها در پایتون
Error! Bookmark not defined. کامنتها و داک استرینگها در پایتون
۱۷ توابع یا function
۱۸ String Functions توابع رشته‌ای
۱۹ حل چند تا تمرین
۲۰ جواب تمرین‌ها
۲۲ کتابخونه‌های پایتون

Fundamentals of Python

تو این جزوه اصول پایه پایتون رو با هم مرور می‌کنیم و از نوت بوک ArcGIS Pro واسه یادگیری بهتر کدها استفاده میکنیم. انتهای درسها چند تا تمرین به همراه جوابهاشون هم واسه یادگیری بهتر آوردیم. در انتهای جزوه هم چند تا از کتابخونه‌های مهم پایتون به همراه کاربردهاشون رو لیست کردیم.

مطالب این جزوه خلاصه مطالب Introduction to Python از سایت سولولرن و Data Visualization with Python از سایت Udemey هست.

پایتون چیه؟

پایتون یه زبان برنامه‌نویسی محبوب، کاربردی و خیلی قدرتمند هست و خیلی آسون میشه یادش گرفت، چون سینتکس ساده داره یعنی هم آسون نوشته میشه و هم آسون خونده میشه. ساختار داده یا Data structure پویا یا Dynamic و کتابخونه یا Libraryهای بزرگ داره. این زبان رو تو کل دنیا به بقیه زبانها ترجیح میدن. ما از این زبان استفاده میکنیم که بتونیم به ماشین دستورالعمل بدیم. هر دستورالعمل شامل یه سری عبارت یا Statement میشه که کامپیوتر باید پیگیریشون کنه. از پایتون میشه در زمینه نرم‌افزار و توسعه وب، علم داده یا Data science، یادگیری ماشین یا Machine Learning و زمینه‌های زیاد دیگه‌ای استفاده کرد.

معرفی چند منبع یادگیری پایتون

پیش فرضمون اینه که با زبان برنامه‌نویسی پایتون آشنایی داری. اگه اصلا با این زبان آشنایی نداری میتونی از سایت سولولرن، W³Schools یا Udemey استفاده کنی و آموزش حرفه‌ای ببینی. سایت سولولرن، هم آموزش‌هاش و هم گواهینامه پایان دوره‌هاش رایگان هست. سایت W³Schools آموزش‌های رایگان داره ولی واسه گرفتن گواهی پایان دوره باید هزینه کنی.

<https://www.sololearn.com/>

<https://www.w3schools.com/>

<https://www.udemy.com/>

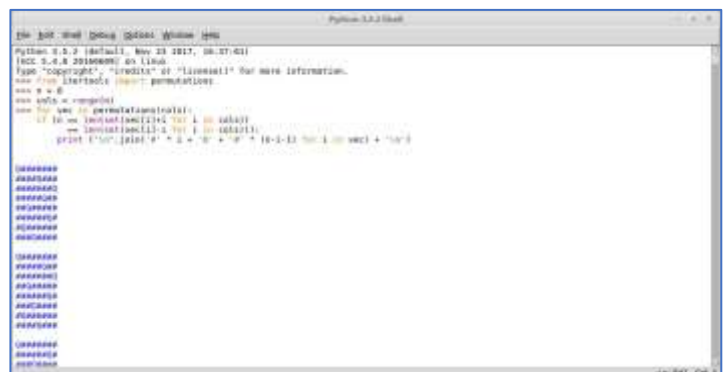
از این لینک هم میتونی آموزش‌های Udemey که فوق‌العاده هستن رو رایگان دانلود کنی:

<https://downloadly.ir>

Python Interfaces in ArcGIS Pro

به پنج شیوه زیر می‌تونی تو ArcGIS Pro کد پایتون بنویسی و ازش استفاده کنی:

۱- استفاده از **Python window**



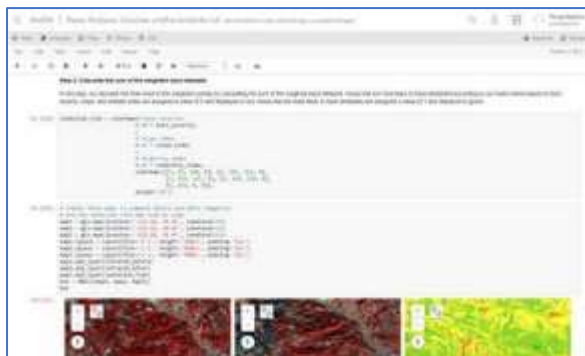
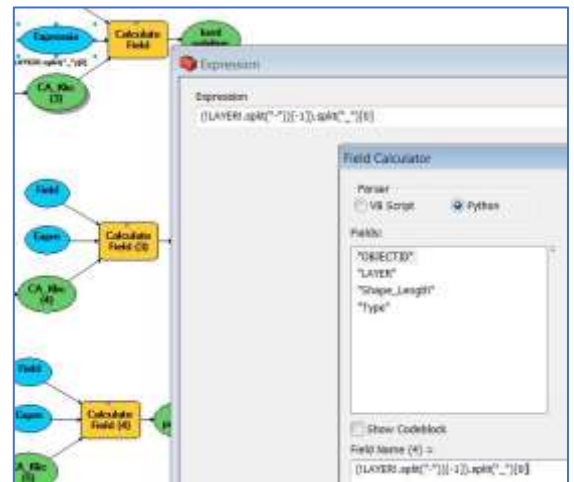
۲- استفاده از **Python IDE** (برای مثال IDLE) برای ساخت Script ها خارج از ArcGIS Pro

Drawing Order	1	import arcpy
	2	
	3	# Input Data
Map	4	inputLine = arcpy.GetParameterAsText(0)
States	5	extentMask = "States"
	6	rasterLayer = "WorldRaster"
	7	
WorldRaster	8	length = arcpy.da.FeatureClassToNumpyArray(inputLine,
Value	9	arcpy.env.extent = extentMask
0.999977	10	
8.24803e-06	11	#Tools
Topographic	12	arcpy.Densify_edit(inputLine, "DISTANCE", length)
	13	
	14	arcpy.InterpolateShape_3d(rasterLayer, inputLine, "
	15	

۳- استفاده از arcpy.GetParameterAsText()

در Python Script Tool

۴- استفاده از Python Parser برای برچسب زدن یا labeling، محاسبات روی فیلدها تو جدول اطلاعاتی یا Field Calculator و محاسبات مقادیر در ساخت مدل یا ModelBuilder



۵- استفاده از ArcGIS Notebooks که بر اساس

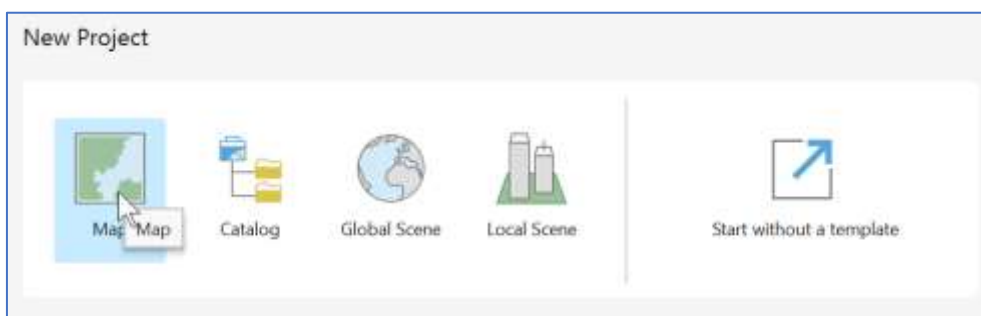
نوت بوک ژوپیتتر هست.

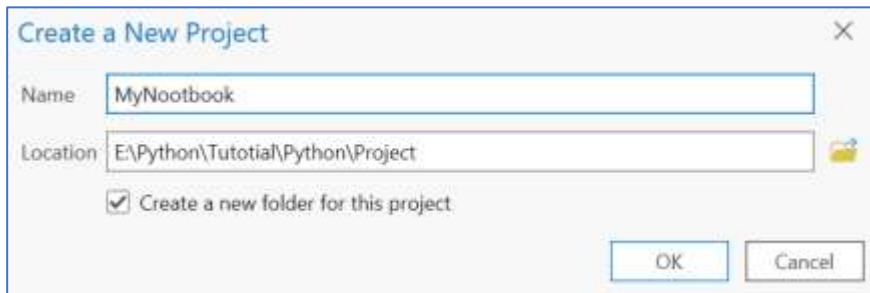
در مورد کار با همه این بخشها مطالب آموزشی آماده می‌کنیم، ولی تو این جزوه صرفاً می‌خوایم کدهای پایه پایتون رو تو نوت بوک ArcGIS Pro بنویسیم و تمرین کنیم. اگه به زبان پایتون آشنایی داری، بریم که نکات پایه‌اش رو با هم دوره کنیم و ببینیم که چطوری میشه از NootBook تو ArcGIS Pro واسه تمرین کدهایی که یاد می‌گیریم استفاده کنیم.

مرحله اول: ساخت پروژه برای کار با Notebook در ArcGIS Pro

نرم افزار ArcGIS Pro باز کن. تو صفحه خوش‌آمد گویی روی گزینه Map کلیک کن که یه پروژه جدید

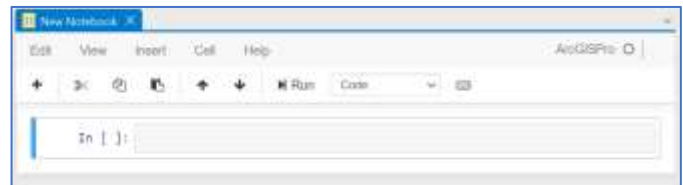
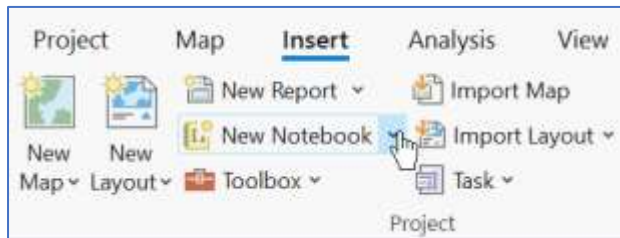
بساز.





بهش اسم و آدرس خروجی
 بده و OK کن که صفحه نرم افزار باز
 شه.

از سربرگ Insert وارد بخش project شو و روی گزینه New Notebook کلیک کن که پنجره اش برات باز شه.



این نتبوک، cell یا سلولهای مختلف داره. میتونی تو هر سلول یه چیزی تعریف کنی که میتونن به صورت جداگونه اجرا یا Run بشن. این سلولها امکان جابجایی دارن و کل کدهات رو به همراه اجراشون میتونی تو یه فایل ذخیره داشته باشی. در مورد این محیط جذاب تو ArcGIS Pro تو جزوه های بعدی بیشتر توضیح میدیم.

مرحله دوم: مرور اصول پایه زبان برنامه‌نویسی Python

تو جدولهای زیر توضیحات کلی در مورد اصول پایه پایتون آوردیم. چیزهایی که یاد می‌گیری رو حتما توی سلولهای نوت‌بوک بنویس که برات تمرین شه.

واسه رفتن به سلول بعدی، **شیفت و اینتر** رو بزن که کد سلولی که توش هستی رو اجرا کنه و بره تو یه سلول جدید.

چطوری تعریف میشه؟	توضیحات	عنوان
<pre>A = ۱۰ Name = 'John' Mile = ۱۳۰٫۷ City_Name = "London"</pre> 	<p>✓ برای ذخیره کردن مقادیر (مثل اعداد، متنها و ...) استفاده میشه که بشه بعدا از این مقادیر در طول برنامه استفاده کرد. چون چیزی رو ذخیره میکنه نیاز به حافظه یا فضا داره.</p> <p>✓ نیازی به دستور اعلان نداره و فقط شامل اسم و مقدارش هست که با علامت = به هم وصل میشن.</p> <p>✓ واسه نامگذاری متغیرها هم محدودیت داری. با عدد شروع نکن. فقط با حروف یا آندرلاین شروع کن. از فاصله یا - بین کلمات استفاده نکن فقط _ بذار. پایتون به حروف کوچک و بزرگ حساس هست مثلا Age, age, AGE براش سه تا اسم متفاوت هستن. از اسامی که تو پایتون تعریف شده هم استفاده نکن مثل for, while, elif, if و ... واسه نامگذاری اسامی ترکیبی هم به یکی از دو شیوه زیر عمل کن:</p> <ul style="list-style-type: none"> ➤ Snake case یه روش محبوب واسه ایجاد نام متغیر در پایتون هست. تو این روش از زیرخط یا آندرلاین _ واسه جدا کردن اسامی چند کلمه‌ای در نام متغیر استفاده میشه. first_name ➤ Camel case هم روش محبوب دیگه واسه نامگذاری اسامی ترکیبی هست که اسم اول باحروف کوچک نوشته میشه و اسم بعدی به صورت چسبیده بهش می‌آد ولی اولش با حرف بزرگ شروع میشه مثل firstName 	<p>متغیرها یا Variables</p>

پایتون ۵ نوع داده داره:

- ۱- Numbers
- ۲- Strings
- ۳- Lists
- ۴- Tuples
- ۵- Dictionaries

یه تابع داریم به اسم `type()` که وقتی داخل اسم متغیر رو بنویسیم، نوع داده‌اش رو بهمون میده. یادت باشه که توابع با حروف کوچک نوشته میشن.

❖ سه نوع داده عددی یا **number** داریم:

۱- عدد صحیح یا **integers** که هم مثبت هست و هم منفی بدون اعشار

۲- عدد اعشاری یا **float** که مثبت و منفی هست و با اعشار.

۳- عدد پیچیده یا **complex** مثل $A+Bj$ که A بخش اصلی و واقعی هست و Bj بخش تخیلی یا تصوری. **Complex** خیلی تو پایتون کاربرد نداره.

❖ داده رشته یا متنی یا **Strings**

✓ توالی حروف، متن و ... رو مشخص میکنه.

✓ متن (میتونه عدد هم باشه) رو باید بذاری تو کوتیشن " " ، ' ' (تکی یا دوتایی فرق نداره فقط این ور و اون ور باید یکسان باشه) که اسمش بشه **string** یا رشته.

```
In [6]: type(A)
```

```
Out[6]: <class 'int'>
```

```
In [7]: type(Name)
```

```
Out[7]: <class 'str'>
```

```
In [8]: type(City_Name)
```

```
Out[8]: <class 'str'>
```

```
In [9]: type(Miles)
```

```
Out[9]: <class 'float'>
```

```
In [10]: a = 10
```

```
In [11]: b = 99.9
```

```
In [12]: c = 45.j
```

```
In [13]: type(a)
```

```
Out[13]: <class 'int'>
```

```
In [14]: type(b)
```

```
Out[14]: <class 'float'>
```

```
In [15]: type(c)
```

```
Out[15]: <class 'complex'>
```

```
In [17]: print("Welcome Python")
Welcome Python
```

```
In [*]: Name = input("What's your name?")
What's your name?
Tisa
```

```
In [18]: Name = input("What's your name?")
What's your name?Tisa
```

```
In [19]: Name
Out[19]: 'Tisa'
```

✓ از تابع `print()` واسه خروجی گرفتن یا ارسال پیام به اسکرین یا سایر وسایل نمایش استفاده میشه. اگه بخوای توی پرانتزش متن بنویسی باید بذاریش تو "" که تبدیل به استرینگ یا رشته بشه. اعداد به علامت نقل قول نیازی ندارن. میتونی اسم یه متغیر که از قبل تعریف شده رو هم بذاری تو پرانتز `print()` که مقدارش رو بهت خروجی بده.

✓ از تابع `input()` واسه گرفتن ورودی از کاربر استفاده میشه. باید براش متغیر تعریف کنی مثل شکل مقابل. وقتی کد رو با `Shift + enter` اجرا کنی یه کادر بهت میده که ورودی بهش بدی و بعد از اینکه چیزی نوشتی و اینتر زدی خارج از سلول بهت نشونش میده. اگه بخوای مجدد ببینی که چی وارد شده صرفاً اسم متغیر رو تو یه سلول جدا بنویس و اجراش کن.

✓ **نکته:** پیش فرض اطلاعاتی که کاربر وارد میکنه همیشه `String` هست. یعنی اگه عدد ۳ رو هم وارد کنه باز هم به صورت '۳' که رشته هست شناخته میشه.

چطوری تعریف میشه؟	توضیحات	عنوان
<pre> In [20]: variable_1 = 100 In [21]: type(variable_1) Out[21]: <class 'int'> In [22]: float(variable_1) Out[22]: 100.0 In [25]: type(float(variable_1)) Out[25]: <class 'float'> In [26]: str(variable_1) Out[26]: '100' In [27]: type(str(variable_1)) Out[27]: <class 'str'> </pre>	<p>گاهی نیاز داری ورودی که از کاربر میگیری رو به نوع دیگه‌ای تبدیل کنی چون با تابع <code>input()</code> کاربر هر چیزی وارد کنه به صورت رشته یا <code>string</code> برمیگیره.</p> <p>واسه تبدیل داده‌ها به هم:</p> <ul style="list-style-type: none"> ✓ دستور <code>int()</code>: تبدیل نوع داده به عدد صحیح ✓ دستور <code>float()</code>: تبدیل نوع داده به عدد اعشاری ✓ دستور <code>str()</code>: تبدیل نوع داده به رشته 	<p><i>تبدیل داده یا data Conversion</i></p>

عملگرها یا Operators عملوندها رو دستکاری میکنن. $5+8$ رو در نظر بگیر. 5 و 8 عملوند هستن و $+$ عملگر هست.
انواع عملگرها در پایتون:

۱- عملگرهای ریاضی یا Arithmetic operators:

✓ $+$ (به علاوه)، $-$ (منها)، $*$ (ضرب)، $/$ (تقسیم)، $\%$ (باقیمانده)، $**$ (توان)، $//$ (خارج قسمت یا تقسیم کف): یعنی وقتی دو تا عدد تقسیم شد عدد کف رو بده مثلا نتیجه تقسیم بشه ۸ ممیز ۳، بهت عدد ۸ رو برمیگردونه)

۲- عملگرهای مقایسه‌ای Comparison Operators:

✓ نتیجه این عملگر بولین (یه نوع داده تو پایتون) هست یعنی یا True برمیگرده یا False. **یادت باشه** که T و F رو با حرف بزرگ بنویسی.
✓ $==$ (مساوی)، $!=$ (نامساوی)، $<$ (کوچکتر از)، $>$ (بزرگتر از)، $<=$ (کوچکتر مساوی)، $>=$ (بزرگتر مساوی)

۳- عملگرهای انتصاب یا Assignment operators:

✓ $=$ (مساوی)، $+=$ (یعنی عدد سمت راست رو با عدد سمت چپ جمع کن)، $-=$ (یعنی عددی که سمت راست می‌آد رو از عدد سمت چپ کم کن)

۴- عملگرهای منطقی یا logical operators:

✓ **and** (اگه همه شرایط برقرار باشه نتیجه True هست در غیر اینصورت False رو خروجی میده)
✓ **or** (اگه فقط یکی از شرایط برقرار باشه True برمیگرده و در غیر اینصورت False رو خروجی میده)

✓ **not** (مخالف مقایسه معنیش کن. اگه نتیجه True باشه False رو خروجی میده و برعکس)

❖ **یادت باشه که این سه تا رو با حروف کوچک بنویسی.** به جای علامت $=$ هم از علامت $==$ باید استفاده شه یعنی معادل.

```
In [28]: A = 9
         B = 15
```

```
In [29]: A + B
```

```
Out[29]: 24
```

```
In [30]: B - A
```

```
Out[30]: 6
```

```
In [31]: A / B
```

```
Out[31]: 0.6
```

```
In [32]: B * A
```

```
Out[32]: 6
```

```
In [33]: B ** 2
```

```
Out[33]: 225
```

```
In [1]: A = 9
         B = 15
```

```
In [3]: A == B
```

```
Out[3]: False
```

```
In [4]: A != B
```

```
Out[4]: True
```

```
In [5]: A < B
```

```
Out[5]: True
```

```
In [6]: A > B
```

```
Out[6]: False
```

```
In [7]: A <= B
```

```
Out[7]: True
```

```
In [8]: A >= B
```

```
Out[8]: False
```

```
In [11]: C = 1
```

```
In [13]: C += 1
```

```
In [17]: C -= 2
```

```
In [18]: C
```

```
Out[18]: 0
```

```
In [19]: A == 9 and B == 15
```

```
Out[19]: True
```

```
In [20]: A == 15 or B == 15
```

```
Out[20]: True
```

```
In [21]: not B == 15
```

```
Out[21]: False
```

چطوری تعریف میشه؟	توضیحات	عنوان
	<p>❖ کامنتها یه جور حاشیه‌نویسی واسه کد هستن که درکش رو آسونتر میکنن ولی روی نحوه اجرای کد تاثیری ندارن. در پایتون اگه اول عبارت یه # بذاری کامنت ساخته میشه و کامپیوتر تمام متن بعدش رو نادیده میگیره. به این علامت octothorpe یا علامت یه عدد یا هشتگ # هم میگن. میتونی تو کدت چندین کامنت داشته باشی.</p> <p>❖ داک استرینگها شبیه به کامنتها هستن و برای توضیح کد طراحی میشن ولی خیلی خاص‌تر هستن و سینتکس متفاوتی دارن. با گذاشتن چندتا رشته خط حاوی توضیح در مورد تابع، در زیر خط اول تابع نوشته میشن. Docstring "" ""ها یه جور سند هستن واسه اینکه بقیه دولوپرها از توابعی که مینویسی استفاده کنن. برخلاف کامنتهای مرسوم، داک استرینگها در طول زمان اجرای برنامه حفظ میشن. این کار به برنامه‌نویس اجازه میده که نظرات رو در زمان اجرا هم بررسی کنه.</p>	<p>کامنتها و داک استرینگها در پایتون</p>

چطوری تعریف میشه؟	توضیحات	عنوان
<pre>In [23]: age = int(input("Please enter your age: ")) if age < 21: print("You can't enter this place!") Please enter your age: 22 In [24]: number = int(input("Please enter number: ")) if number < 0: print("You number is negative") Please enter number: 1 In [27]: age = int(input("Please enter your age: ")) if age < 21: print("You can't enter this place!") else: print("Welcome") Please enter your age: 22 Welcome In [28]: number = int(input("Please enter number: ")) if number < 0: print("You number is negative") else: print("Positive") Please enter number: -1 You number is negative In [30]: Model = int(input("Please choose 1-3: ")) if Model == 1: print("You choose Rolls-Royce.") elif Model == 2: print("You choose Aston_Martin.") elif Model == 3: print("You choose Bentley.") else: print("Invalid Number") Please choose 1-3: 3 You choose Bentley.</pre>	<p>قبل از اینکه وارد Control statements یا عبارات کنترلی بشیم باید در مورد تصمیم‌سازی یا decision making بدونیم.</p> <ul style="list-style-type: none"> ➤ تصمیم‌سازی یعنی پیش‌بینی شرایطی که در طول اجرای برنامه انجام میشه و تشخیص اقدامات بر اساس این شرایط. در واقع شرایط درست و غلط رو بررسی میکنه که True هست یا False و بر اساس اون تصمیم میگیره. ➤ عبارت کنترلی واسه کنترل شرایط استفاده میشن و شامل: <ul style="list-style-type: none"> ✓ if یه عبارت کنترلی هست. if از یه عبارت منطقی استفاده میکنه، داده‌ها رو با هم مقایسه میکنه و نتیجه مقایسه رو خروجی میده. اگه شرایط برقرار نباشه یا False باشه، عبارات بعد از شرطی if اجرا میشن. ✓ else تو ترکیب با if می‌اد و به صورت جداگانه استفاده نمیشه. اگه شرایط if برقرار نباشه، شرط else اجرا میشه. میتونی else رو هم نیاری چون آوردنش اجباری نیست. ✓ elif چندین شرط رو بررسی میکنه و هر کدوم True بود همون رو برمیگردونه. در واقع اگه if برقرار نباشه سراغ elif‌ها. 	

مشق شب ۱:

یه کد بنویس که سن کاربر رو بگیره و به شکل زیر بهش خروجی بده:

- ۱- اگه زیر ۱۸ سال بود بهش بگه "تو کودک هستی".
- ۲- اگه بین ۱۸ تا ۳۵ بود بهش بگه "تو نوجوان هستی".
- ۳- اگه بین ۳۵ تا ۶۵ بود بهش بگه "تو جوان هستی".
- ۴- اگه بین ۶۵ تا ۹۰ بود بهش بگه "تو میانسال هستی".
- ۵- اگه ۹۰ به بالا بود (درغیر اینصورت یعنی اگه جز هیچ کدوم از گزینه‌های بالا نبود) بهش بگه "تو پیر هستی".

حلقه‌ها عباراتی هستند که همیشه باهاشون کد رو به صورت تکراری اجرا کرد. دو نوع حلقه داریم:

➤ **حلقه while:** تا وقتی که شرط برقرار باشه اجرا میشه. کد مقابل می‌گه که تا زمانی‌که $i < 10$ یا $i < 20$ هست یه فعالیتی رو انجام بده. حلقه while نیاز به شمارنده داره یعنی باید $i = 0$ رو که تعریف کردیم بعد انتهای حلقه $i += 1$ رو بنویسیم. اگه به i یکی یکی اضافه نکنیم، حلقه تا ابد ادامه پیدا میکنه.

➤ **حلقه for:** بهت اجازه میده که یک یا چندین فعالیت رو برای هر شرایط اجرا کنی. کد مقابل روی کلمه Python تکرار انجام میده و هر حرفش رو تو هر تکرار خروجی میده و وقتی به آخرین حرف رسید حلقه متوقف میشه.

کلا از حلقه‌های for واسه وقتی که از قبل میدونی تعداد تکرار چند تا هست استفاده کن و از حلقه while واسه وقتی که یه شرط هست که باید بهش بررسی. حواست به تورفتگیهای کد بلوک حلقه و : باشه.

از ساختارهای کنترلی هم همیشه واسه حلقه‌ها استفاده کرد که بشه جریان نرمال حلقه‌ها و پیامدشون رو تغییر داد. این ساختارهای کنترلی شامل موارد زیر هستن و با شرط if داخل حلقه می‌آن:

➤ **break:** بیشتر برنامه‌نویسها از break تو حلقه‌ها استفاده میکنن که تو فرایند در حال انجام کد، وقفه ایجاد کنن. هر وقت کد به break رسید اجرای برنامه متوقف میشه.

➤ **continue:** ساختار کنترلی continue واسه شرایطی هست که میخوایم از روی یه چیزی بپریم. تو کد مقابل عدد ۲ خروجی گرفته نمیشه یا از روش میپره.

➤ **pass:** این ساختار نمیخواد هیچ دستوری یا کدی رو اجرا کنه. در واقع هیچ اتفاقی نمی‌افته. ازش به عنوان placeholder یا نگهدارنده مکان استفاده میشه.

تابع **range()** به تابع از پیش تعریف شده تو پایتون هست و بر اساس مقداری که بهش میدیم اجرا میشه. رنج دو تا آرگومان یا مقدار قبول میکنه که ابتدا و انتهای رنج رو بهمون میده. مثلا $range(0, 10)$ از عدد ۰ تا یکی مونده به ۱۰ یعنی ۱۴ رو خروجی میده. آرگومان سوم گام هست یعنی $range(3, 30, 4)$ به رنج از ۳ تا ۲۹ میده ولی هر عدد با ۴ جمع میشه. اگه ۴ تا آرگومان بهش بدیم خطا میده. علامت * قبل از range هم کمک میکنه که خروجی به صورت افقی باشه نه عمودی. با حلقه for هم میتونی بنویسیش: `for i in range()`. اینکار به شکل مرسوم واسه تکرار یه کد به تعداد دفعات مشخص استفاده میشه. با لیست هم میشه نوشتش: `x = list(range())`

```
In [4]: i = 0
while i < 10:
    print(i)
    i += 1

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
In [1]: i = 0
while i < 20:
    print("hello world")
    i += 1

hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
```

```
In [1]: for characters in "Python":
    print(characters)

P
y
t
h
o
n
```

```
In [7]: Number = 0
while Number < 10:
    print(Number)
    if Number == 5:
        break
    Number += 1

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

```
In [1]: Number = 0
while Number < 20:
    if Number == 2:
        Number += 1
        continue
    print(Number)
    Number += 1

0
1
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

```
In [2]: for character in "python":
    pass

Traceback (most recent call last)
File ~\Desktop\python\python\python.py, line 1, in <module>
    for character in "python":
IndentationError: expected an indented block (<string>, line 1)
```

```
In [4]: range(0,15)
Out[4]: range(0, 15)

In [5]: print(*range(0,15))
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

In [6]: print(*range(3,30,4))
3 7 11 15 19 23 27

In [7]: range(3,10,5,5)

TypeError                                 Traceback
(most recent call last)
In [7]:
Line 1:     range(3,10,5,5)

TypeError: range expected at most 3 arguments, got 4
```

پایتون ۶ تا توالی داخلی داره. در واقع انواع داده در پایتون یه توالی هستن. هر مقدار در این نوع داده‌ها ایندکس داره که از ۰ شروع میشه. یعنی ایندکس عدد سوم میشه ۲، ایندکس عدد پنجم میشه ۴ و ...

در مورد دو نوع داده یا توالی یعنی Strings و numbers با هم یاد گرفتیم ۴ تا توالی دیگه شامل موارد زیر هستن:

➤ **List**: همه کاره هست. داده‌هاش با کاما از هم جدا مشن و دورشون براکت مربعی میاد [].

➤ **Tuples**: تاپلها شبیه لیست هستن ولی immutable یا تغییرناپذیر هستن. ما به این داده‌های تغییر ناپذیر واسه فایل‌های پیکربندی یا config نیاز داریم. با کاما جدا میشن و میرن تو پرانتز (). اگه بین اعدادش کاما نذاریم نوع داده رو integer یا عدد صحیح برمیگردونه.

➤ **Dictionary**: شامل key-value هست. بین key و value دونقطه یا : کولن میاد و دورشون آکولاد {Key: Value}. مقادیر کلیدها میتونن مشابه باشن. کلیدهای مشابه هم میشه داشت. ولی آخرین کلید و مقدار رو در نظر میگیره و خروجی میده.

➤ **Sets**: واسه این خوبه که اعداد تکراری رو از مجموعه برداره و فقط مقادیر اصلی رو خروجی بده. ستها هم آکولاد و ویگول قبول میکنن ولی key-value ندارن. {,}

واسه دسترسی به هر کدوم از داده‌های انواع داده باید ایندکسش رو صدا بزنینم. یعنی موقعیتش رو تو نوع داده مشخص کنیم. با [] و گذاشتن شماره ایندکس داخلش اینکار انجام میشه.

✓ اگه از آخر به اول بخوایم بشمریم باید عدد منفی بدیم. -۱ همون عدد آخر هست.

✓ اگه تو [] دو تا عدد بیاریم که با دونقطه : از هم جدا شدن، در واقع میخوایم که مقدار اولین ایندکس تعیین شده رو تا یکی مونده به آخرین ایندکس بهمون بده.

❖ اگه عدد اول حذف شده باشه منظور از ایندکس ۰ هست یا اولین عدد.

```
In [8]: MyList = [1,2,3,4,5,6,7,8]
In [9]: type(MyList)
Out[9]: <class 'list'>
In [10]: MyList_2 = [1, 2, 3, 'apple', 12, 20, 'a', 'b', 10, 8]
In [11]: type(MyList_2)
Out[11]: <class 'list'>
In [12]: MyList_3 = [1,2,3,[1,2,3], 'apple', 'banana', ['apple', 'banana']]
In [13]: type(MyList_3)
Out[13]: <class 'list'>
```

```
In [17]: MyTuples = (1,2,3,4,5,6,7,8)
In [18]: type(MyTuples)
Out[18]: <class 'tuple'>
In [19]: MyTuples_2 = (1, 2, 3, 'apple', 12, 20, 'a', 'b', 10, 8)
In [20]: type(MyTuples_2)
Out[20]: <class 'tuple'>
In [21]: MyTuples_3 = (1,2,3,[1,2,3], 'apple', 'banana', ['apple', 'banana'])
In [22]: type(MyTuples_3)
Out[22]: <class 'tuple'>
```

```
In [28]: MyDict = {'Iran':4, 'USA':1, 'UK':2, 'France':3, 'Austria':4, 'Greece':5}
In [29]: type(MyDict)
Out[29]: <class 'dict'>
In [30]: MyDict_2 = {}
In [31]: type(MyDict_2)
Out[31]: <class 'dict'>
In [32]: MyDict_3 = {'Germany':10, 'Austria':10, 'France':10}
In [33]: type(MyDict_3)
Out[33]: <class 'dict'>
```

```
In [45]: MySet_2 = {'a', 'b', 'a', 'd', 'a'}
In [46]: type(MySet_2)
Out[46]: <class 'set'>
In [47]: MySet_3 = {1,2,2,3,3,3,4,4,5}
In [48]: type(MySet_3)
Out[48]: <class 'set'>
```

```
In [49]: MySet_3
Out[49]: {1, 2, 3, 4, 5}
```

```
In [50]: String_1 = "1234567"
In [51]: List_1 = [1,2,3,4,5,6,7]
In [52]: Tuple_1 = (1,2,3,4,5,6,7)
In [53]: Set_1 = {"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o"}
In [54]: Set_2 = {1,2,3,4,5,6,7}
```

```
In [57]: String_1[2]
Out[57]: '3'
In [58]: List_1[5]
Out[58]: 6
In [59]: Tuple_1[1]
Out[59]: 2
```

```
In [60]: String_1[-2]
Out[60]: '6'
In [61]: List_1[-5]
Out[61]: 3
In [62]: Tuple_1[-1]
Out[62]: 7
```

```
In [65]: String_1[2:5]
Out[65]: '345'
In [66]: List_1[:3]
Out[66]: [1, 2, 3]
In [67]: Tuple_1[2:4]
Out[67]: (3, 4)
```

از تابع `len()` واسه بدست آوردن تعداد اعضای انواع داده استفاده میشه.

❖ واسه اضافه کردن به مجموعه داده‌ها همیشه به شیوه‌های مختلفی اقدام کرد مثل += یا استفاده از متدهای `append()`، `add()` و ... **یادت باشه که به Tuples نمیتونی مقدار اضافه کنی چون غیرقابل تغییر هست.**

❖ با متد `pop()` میشه به عنصر رو از به نوع داده حذف کرد. **یادت باشه که از Tuples نمیتونی چیزی رو حذف کنی.**

❖ با تابع `count()` میشه تعداد تکرار به عضو تو مجموعه رو شمرد.

❖ برای اضافه کردن دو تا مجموعه `Set` و یا دو تا مجموعه `dictionary` به هم از `update()` استفاده کن.

❖ تابع `insert()` به آیتم جدید رو تو موقعیت داده شده به لیست اضافه میکنه. اولین آرگومانش موقعیت ایندکس رو مشخص میکنه. در حالیکه دومین پارامتر آیتمی هست که باید در اون موقعیت جا داده بشه.

❖ تابع `index()` اولین آیتمی که با اون گزینه تطبیق داره رو پیدا میکنه و ایندکسش رو برمیگردونه. اگه گزینه مورد نظر تو لیست نباشه خطا میده.

❖ `max(list)` بزرگترین عدد لیست رو برمیگردونه.

❖ `min(list)` کوچکترین عدد لیست رو برمیگردونه.

❖ `list.count(item)` تعداد اون آیتمی که تو پرانتزش می‌آد رو میشمره که چندبار تو لیست اومده.

❖ `list.remove(item)` آیتمی که تو پرانتزش اومده رو از لیست پاک میکنه.

❖ `list.reverse()` آیتمهای به لیست رو معکوس میکنه.

```
In [66]: string_1 = "1234567"
In [67]: list_1 = [1,2,3,4,5,6,7]
In [68]: tuples_1 = (1,2,3,4,5,6,7)
In [69]: dict_1 = {'One':1, 'Two':2, 'Three':3, 'Four':4, 'Five':5, 'Six':6, 'Seven':7}
In [70]: set_1 = {1,2,3,4,5,6,7}
```

```
In [68]: len(string_1)
Out[68]: 7

In [69]: len(list_1)
Out[69]: 7

In [70]: len(dict_1)
Out[70]: 7

In [71]: len(tuples_1)
Out[71]: 7
```

```
In [72]: string_1 = "P"
In [73]: list_1.append(0)
In [74]: set_1.add(8)
In [75]: dict_1["Eight"] = 8
In [76]: string_1
Out[76]: '12345678'
In [77]: list_1
Out[77]: [1, 2, 3, 4, 5, 6, 7, 0]
In [78]: set_1
Out[78]: {1, 2, 3, 4, 5, 6, 7, 8}
In [79]: dict_1
Out[79]: {'One': 1, 'Two': 2, 'Three': 3, 'Four': 4, 'Five': 5, 'Six': 6, 'Seven': 7, 'Eight': 8}
```

```
In [80]: Tuples_1.append(8)
-----
AttributeError                                Traceback (most recent call last)
In [80]:
Line 1: Tuples_1.append(8)
-----
AttributeError: 'tuple' object has no attribute 'append'
```

```
In [82]: list_1.pop()
Dict_1.pop("Eight")
Out[82]: 8

In [83]: list_1
Out[83]: [1, 2, 3, 4, 5, 6, 7]

In [84]: Dict_1
Out[84]: {'One': 1, 'Two': 2, 'Three': 3, 'Four': 4, 'Five': 5, 'Six': 6, 'Seven': 7}
```

```
In [85]: Tuples_1.pop()
-----
AttributeError                                Traceback (most recent call last)
In [85]:
Line 1: Tuples_1.pop()
-----
AttributeError: 'tuple' object has no attribute 'pop'
```

```
In [86]: string_1 = "London"
List_2 = ["L", "o", "n", "d", "o", "n"]
Tuples_2 = ("L", "o", "n", "d", "o", "n")

In [87]: string_1.count("n")
Out[87]: 2

In [88]: List_2.count("n")
Out[88]: 2
```

```
In [89]: dict_1 = {"One":1, "Two":2, "Three":3, "Four":4, "Five":5, "Six":6, "Seven":7}
In [90]: dict_1.update(dict_2)
In [91]: dict_1.update(dict_2)

In [92]: dict_1
Out[92]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

In [93]: dict_1
Out[93]: {'One': 1, 'Two': 2, 'Three': 3, 'Four': 4, 'Five': 5, 'Six': 6, 'Seven': 7, 'Eight': 8, 'Nine': 9, 'Ten': 10}
```

```
In [1]: import math
In [2]: dir(math)
Out[2]: ['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfin', 'ite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
```

```
In [3]: help(math)
Help on built-in module math:
NAME
math
DESCRIPTION
This module provides access to the mathematical functions
defined by the C standard.
FUNCTIONS
acos(x, /)
Return the arc cosine (measured in radians) of x.
The result is between 0 and pi.
acosh(x, /)
Return the inverse hyperbolic cosine of x.
asin(x, /)
Return the arc sine (measured in radians) of x.
The result is between -pi/2 and pi/2.
```

```
In [4]: help(math.factorial)
Help on built-in function factorial in module math:
factorial(x, /)
Find x!.
Raise a ValueError if x is negative or non-integral.
```

```
In [1]: #floor(8.7)
Out[1]: 8
In [2]: help(math.floor)
Help on built-in function floor in module math:
floor(x, /)
Return the floor of x as an Integral.
This is the largest integer <= x.
In [4]: help(math.floor)
Help on built-in function floor in module math:
floor(x, /)
Return the floor of x as an Integral.
This is the largest integer <= x.
```

ماژولها فایل‌هایی هستند که شامل کدهای پایتون میشن و میتونن توابع، کلاسها و متغیرها رو تعریف کنن. علاوه بر اون میتونن شامل کدهای قابل اجرا باشن.

واسه اینکه از کدهای آماده استفاده کنیم و مجدد نویسیمشون از ماژولها استفاده میکنیم.

با عبارت `import` میشه یه ماژول رو صدا زد. ماژولها فقط یکبار لود میشن.

داخل کتابخونه پایتون ready-made modules وجود داره که توسط دولوپرها نوشته شده و میشه از اینترنت و گیت هاب پیدا بشون کرد.

یکی از این ماژولها، ماژول ریاضی یا `math` هست که محاسبات ریاضی رو برامون انجام میده. وقتی واردش کنیم به همه محاسبات ریاضی میتونیم دسترسی داشته باشیم.

با تابع `dir()` میشه فهمید که چه چیزهایی در ماژول `math` وجود داره.

اگه بخوایم ببینیم که این ماژول چیکار میکنه از تابع `help()` استفاده میکنیم. تمام عملگرهای ریاضی و وظایفشون رو برات می‌اره.

اگه فقط یکی از عملگرها رو بخوایم ببینیم باید با نقطه به کلمه `math` بچسبونیمش.

اگه بخوایم یکی از توابع ریاضی رو بیاریم از `from import ... math` به همراه نام تابع استفاده میکنیم.

```
In [6]: def Sum(A, B):
        """This function sum the variables"""
        print(A+B)

In [7]: Sum(4, 5)
8
```

```
In [9]: def Sum(A,B,C,D):
        print(A+B+C+D)

In [10]: def Multiply(A):
        print(A*2)

In [11]: Sum(3,4,5,6)
18

In [12]: Multiply(5)
10
```

```
In [13]: Multiply(Sum(3,4,5,6))
18
-----
TypeError                                 Traceback (most recent call last)
In [13]: Multiply(Sum(3,4,5,6))
          line 1: Multiply(Sum(3,4,5,6))

In [10]: Multiply(A)
          line 2: print(A*2)

TypeError: unsupported operand type(s) for *: 'NoneType' and 'int'
```

```
In [14]: def Sum(A,B,C,D):
        return A+B+C+D

In [15]: def Multiply(A):
        return A*2

In [16]: Multiply(Sum(10,10,10,10))
Out[16]: 88
```

```
In [17]: def Sum(a,b):
        return a+b

In [18]: Sum(10,14)
Out[18]: 24

In [19]: Sum_1 = lambda a,b : a+b

In [20]: Sum_1(10,14)
Out[20]: 24
```

توابع به ساختار هستن که میتونی به صورت تکراری ازشون استفاده کنیم مثل `print()`. همیشه باهاش کارها رو تو یه مرحله انجام داد. توابع فقط وقتی اجرا میشن که صداشون بزیم.

پایتون تعداد خیلی خیلی زیادی توابع داخلی و توابعی که توسط کاربر تعریف میشه داره. یعنی میتونیم خودمون هم تابع تعریف کنیم. واسه تعریفش از `def` استفاده میکنیم.

عبارت `return`:

وقتی بخوایم یه مقدار رو که تو تابع دریافت کردیم، تو یه متغیر نگه داریم یا ذخیره کنیم از `return` استفاده میکنیم و بعد از این متغیر میتونیم تو برنامه استفاده کنیم.

تو کد مقابل اگه بخوایم با تابع پرینت یه تابع تو در تو بنویسیم خطا دریافت میکنیم. به جای `print()` باید از `return` استفاده کنیم.

توابع `lambda expression`:

توابع ناشناس یا Anonymous رو همیشه با `lambda` تعریف کرد. باهاش میشه توابع رو در یه خط تعریف کرد. با یه تابع کوتاه تعریف میشه و چندین آرگومان و یه عبارت یا `expression` داره. معمولاً به صورت توابع ناشناس داخل یه تابع دیگه تعریف میشن.

```
In [12]: nums = [4, 5, 6]
msg = "Numbers: {0} {1} {2}".format(nums[0], nums[1], nums[2])
print(msg)

Numbers: 4 5 6

In [13]: a = "{x}, {y}".format(x=5, y=12)
print(a)

5, 12
```

```
In [14]: x = ", ".join(["spam", "eggs", "ham"])
print(x)

spam, eggs, ham
```

```
In [15]: str = "some text goes here"
x = str.split(' ')
print(x)

['some', 'text', 'goes', 'here']
```

```
In [16]: x = "Hello ME"
print(x.replace("ME", "world"))

Hello world
```

```
In [17]: print("This is a sentence.".upper())
print("AN ALL CAPS SENTENCE".lower())

THIS IS A SENTENCE.
an all caps sentence
```

انواع توابع رشته‌ای:

- رشته‌ها به تابع `format()` دارن که این امکان رو فراهم میکنه که با کمک نگهدارنده مکان یا `placeholder` مقادیری در داخلش قرار بگیرن. هر آرگومان تابع `format` در رشته در موقعیت مربوط به خودش قرار میگیره که این کار با کمک آکولادها `{}` تعیین میشه. میتونی نگهدارنده‌های مکان رو به جای اعداد ایندکس نامگذاری کنی.
- `join()` با کمک یه جداکننده، یه لیست از رشته رو به یه رشته دیگه وصل میکنه.
- `split()` متضاد `join()` هست. یه رشته رو با یه جداکننده مشخص داخل یه لیست برمیگردونه.
- `replace()` یه زیررشته رو در یه رشته با یکی دیگه جایگزین میکنه.
- `lower()` و `upper()` رشته‌های با حروف کوچک و بزرگ رو به شرایط درخواستی تغییر میده.

حل چند تا تمرین

تمرین ۱: دو تا متغیر به اسم a و b بساز و به هر کدام یک مقدار عددی یا رشته‌ای اختصاص بده و از شون کنار هم خروجی بگیر.

تمرین ۲: یه لیست بساز که عضوهایش شامل red, blue, green, yellow باشه. بعد یه عضو دیگه به اسم black بهش اضافه کن و از سومین عضو لیست و تعداد اعضای لیست خروجی بگیر.

تمرین ۳: از تک تک اعضای لیست خروجی بگیر.

تمرین ۴: یه کد بنویس که موقع خروجی گرفتن از اعضای لیست وقتی به رنگ سبز رسید اجرا کد متوقف شه.

تمرین ۵: برای لیست یه شرط بنویس که هر وقت کد به رنگ زرد رسید پرینت شه 'yellow' و در غیر اینصورت 'not yellow' خروجی گرفته شه.

تمرین ۶: به تابع تعریف کن که تمام اعضای لیست بالا رو چک کنه و اگه قرمز رو پیدا کرد 'red' رو خروجی بده و در غیر اینصورت بنویسه 'not red'.

تمرین ۷: یه کد بنویس که از کاربر رنگ مورد علاقه‌اش رو بگیره و بعد چیزی که کاربر وارد کرده رو خروجی بده.

تمرین ۸: یه کد بنویس که تو اعضای لیست چک کنه و ببینه آیا رنگی که کاربر وارد کرده قرمز هست یا نه اگه نبود 'not red' رو خروجی بده.

جواب تمرین‌ها

جواب تمرین ۱:

```
In [19]: a = "Hello"
         b = "world"
         print (a, b)

Hello world
```

جواب تمرین ۲: لیست رو با [,] درست میکنیم. برای اضافه کردن عضو بهش از متد append() استفاده میکنیم و برای تعداد اعضا از تابع len().

```
In [24]: list = ['red', 'blue', 'green', 'yellow']
         list.append('black')
         print(list[2])
         print(len(list))

green
5
```

جواب تمرین ۳: باید یه حلقه for بنویسیم.

```
In [31]: for colors in list:
         print(colors)

red
blue
green
yellow
black
```

جواب تمرین ۴: باید یه حلقه for به همراه یه if و همینطور break بنویسیم. اگه بخوایم از روی چیزی بپریم باید به جای break از continue استفاده کنیم. تو کد زیر اگه print(colors) رو ببریم زیر break کلمه 'green' تو خروجی نمی‌آد.

```
In [32]: for colors in list:
         print(colors)
         if colors == "green":
             break

red
blue
green
```

جواب تمرین ۵: باید یه if و else داخل حلقه for بنویسیم.

```
In [37]: for colors in list:
         if colors == "yellow":
             print(colors)
         else:
             print("not yellow")

not yellow
not yellow
not yellow
yellow
not yellow
```

جواب تمرین ۶: باید به تابع با def با به آرگومان مثلا c تعریف کنیم و تو بلوکش به شرط با if و else بنویسیم. بعد باید تابع رو صدا بزنینم. واسه اینکه این بررسی روی همه اعضای لیست انجام بشه باید به حلقه for قبل از صدا زدن تابع بنویسیم. یادت باشه که وقتی میخوای تابع رو صدا بزنی تو دل حلقه for باشه یعنی تورفتگی داشته باشه. یادت باشه که از عبارت return واسه نگهداری یا ذخیره به مقدار تو به متغیر که از تابع دریافت کردیم استفاده میکنیم که بعدا بتونیم ازش تو برنامه استفاده کنیم. یعنی به جای print() از return استفاده کن و وقتی تابع رو صدا زدی بذارش تو تابع

.print()

```
In [51]: def checkRed(c):
         if c == 'red':
             return 'red'
         else:
             return 'not red'
         for colors in list:
             print(checkRed(colors))

red
not red
not red
not red
not red
```

جواب تمرین ۷: واسه گرفتن ورودی از کاربر باید از تابع input() استفاده کنیم و واسه خروجی گرفتن از تابع print(). یادت باشه که تابع input() رو با متغیر تعریف کنی.

```
In [64]: x = input('What is your favourite color: ')
         print(x)

What is your favourite color: orange
orange

In [65]: x

Out[65]: 'orange'
```

جواب تمرین ۸: کافیه تو کد بالا به جای print(x) عبارت print(checkRed(x)) رو وارد کنی.

```
In [66]: x = input('What is your favourite color: ')
         print(checkRed(x))

What is your favourite color: orange
not red
```

کتابخانه‌های پایتون

پایتون کتابخانه‌های بی‌نظیری برای data Science علم داده، Machine learning یادگیری ماشین، Artificial Intelligence هوش مصنوعی، Deep learning یادگیری عمیق و ... دارد. چند تا از مهمترین کتابخانه‌های پایتون که در سال ۲۰۲۳ معرفی شده که یادگیریشون ضروری هست رو اینجا آوردیم.

- ✚ Numpy
- ✚ Pandas
- ✚ Matplotlib
- ✚ Scikit_Learn
- ✚ PyTorch
- ✚ Natural Language Toolkit (NLTK)
- ✚ Seaborn
- ✚ SciPy
- ✚ Plotly
- ✚ OpenCv
- ✚ LightGBM
- ✚ Eli5
- ✚ PyBrain
- ✚ Squarify
- ✚ Permetrics
- ✚ ...

دسته‌بندی پایین هم به تجربه به دست اومده:

➤ برای کار با داده‌ها یا data:

numpy as np ✓

pandas as pd ✓

matplotlib.pyplot as plt ✓

➤ برای زیبایی بیشتر نمودارها

seaborn as sns ✓

squarify ✓

➤ برای چک کردن داده‌های موجود در یک جدول

missingno as msno ✓

➤ برای یادگیری ماشین

scikit_learn ✓

permetrics ✓

➤ برای کار با داده‌های جغرافیایی

geopandas as gpd ✓

geoplotlib ✓

arcpy ✓

➤ برای کار با داده‌های رستری

rasterio ✓

تو جزوه‌های بعدی چند تا از این کتابخانه‌های مهم رو به همراه کاربردهاشون آموزش میدیم. البته پیش‌نیازش، تسلط به این جزوه هست، پس حسابی این جزوه رو تمرین کن که درک جزوه‌های بعدی برات راحت‌تر شه.