

# به نام خدا

آموزش پایتون با زبان ساده

Python Simple Training



تهیه کننده: راستین علیراده

تابستان ۱۴۰۲

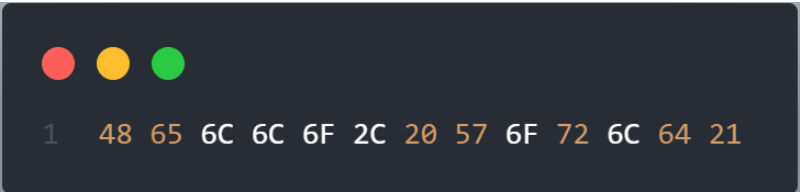
## مفاهیم پایه کامپیوتر و برنامه نویسی:

- **کامپیوتر:** یک ماشین الکترونیکی است که قادر به دریافت اطلاعات، پردازش و تبدیل آن به نتیجه است.
- کامپیوتر ها وابسته به قطعات الکترونیکی و سخت افزاری ای که از آنها تشکیل می شوند، کارایی، اندازه و گونه های متفاوتی دارند. کامپیوتر ها داده را از کاربر میگیرند و با سرعت بالایی عمل محاسبه و پردازش را انجام میدهند و پس از آن به شکلی دیگر، داده را به کاربر ارائه میکنند.
- مثال: ماشین حساب اعداد را از کاربر میگیرد (داده اولیه)، عمل محاسبه را انجام میدهد و خروجی (داده نهایی) مطلوبی به کاربر ارائه میدهد.
- **برنامه:** مجموعه ای از دستورات و تعلیمات است که برای کامپیوتر تهیه می شود تا وظیفه ای را انجام دهد.
- یک برنامه میتواند شامل توابع، حلقه ها، شرط ها، متغیر ها و... باشد که در ادامه به آنها خواهیم پرداخت.
- **برنامه نویسی:** عملی است که در آن شخص "برنامه نویس" به طور خلاقانه و منطقی برنامه ای را طراحی و خلق میکند.
- همانطور که اشاره شد **خلاقیت** کلمه کلیدی برنامه نویسی و همچنین راز یک برنامه نویس موفق است تا بتواند دستوراتی دقیق و منظم به کامپیوتر بدهد و خروجی مطلوبی دریافت کند.
- **زبان برنامه نویسی:** مجموعه ای از قواعد و سینتکس (syntax) ها است که برای نوشتن برنامه های کامپیوتری استفاده میشود.
- زبان های برنامه نویسی به برنامه نویسان امکان میدهند تا دستورات و تعلیمات مورد نیاز خود را به صورت قابل فهم برای کامپیوتر بیان کنند.

## انواع زبان برنامه نویسی:

- اگر بخواهیم به طور ساده زبان های برنامه نویسی در کامپیوتر را به ۳ دسته تقسیم بندی کنیم اینگونه میشود:
- **زبان برنامه نویسی سطح پایین:** این زبان ها به طور مستقیم با سخت افزار کامپیوتر ارتباط برقرار میکنند و دستورات و تعلیمات نزدیک به زبان ماشین را برای کامپیوتر فهم پذیر میکنند که برخی از آنها عبارتند از: زبان ماشین، زبان مجمع و زبان اسمبلی.

مثال: در زبان ماشین، دستوری مانند چاپ رشته "Hello, World!" به صورت بایت کد ماشین نوشته شده و برای هر پردازنده متفاوت است:



1 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 21

- زبان برنامه نویسی سطح میانی: این زبان ها بین زبان های سطح بالا و سطح پایین قرار دارند و قدرت و کنترل بیشتری را به برنامه نویسان ارائه میدهند و کمی بیشتر به زبان انسانی نزدیک هستند. برخی از آنها عبارتند از: C, C++, Pascal, JavaScript و....

مثال: در زبان C، میتوان دستور چاپ رشته "Hello, World!" را بدین گونه نوشت:

```
1 c
2 #include <stdio.h>
3
4 int main() {
5     printf("Hello, World!");
6     return 0;
7 }
```

- زبان برنامه نویسی سطح بالا: این زبان ها برای تسهیل فهم و استفاده برنامه نویسان طراحی شده اند و از سینتکس ها و قواعدی پیروی میکنند که به زبان انسانی نزدیک تر هستند. برخی از آنها عبارتند از: پایتون، جاوا، C، C++، جاوا اسکریپت و...

مثال: در زبان پایتون، یک دستور ساده مانند چاپ رشته "Hello, World!" را بدین گونه نوشت:

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

## معرفی و آشنایی با زبان برنامه نویسی پایتون (Python):

### تاریخچه پایتون:



پایتون یک زبان برنامه نویسی ساده، محبوب، رایگان، همه منظوره، شیءگرا و متن باز است که توسط "خیدو فان روسوم (Guido Van Rossum)" در سال ۱۹۹۱ در کشور هلند طراحی شد. روسوم علاقه زیادی به برنامه نویسی با زبان ABC داشت اما این زبان دارای یک سری مشکلات بود. از آنجاییکه روسوم ویژگی های ABC را خوب و کاربردی میدید تصمیم گرفت زبانی قدرتمندتر برای جایگزینی با ABC ارائه دهد.

روسوم که در تلاش بود تا نامی جدید برای زبان برنامه نویسی اش بیابد، ابتدا نام "B" را برای پروژه خود در نظر گرفت. پس از کمی پژوهش متوجه شد که زبانی با همین نام وجود دارد و همچنین او پیشنهادات مختلف اعضای گروه خود را پی در پی رد میکرد.



سرانجام روسوم تصمیم گرفت اولین نامی که به ذهنش میرسد را روی پروژه اش قرار دهد. احتمالاً با دیدن لوگوی اصلی زبان برنامه نویسی پایتون به یاد ماری با همین نام بیافتید اما درواقع سنگ بنای پایه گذاری این نام ارتباطی با مار پایتون ندارد و مربوط به سیرک تلویزیونی مورد علاقه روسوم یعنی Monty Python's Flying Circus است که در آن سال ها از شبکه تلویزیونی BBC پخش میشده است.

آموزش برنامه سازی با پایتون به زبان ساده

تهیه کننده: علیزاده

پس از آن نامگذاری، اولین چیزی که به ذهن هر شخص با شنیدن این نام می رسید، مار پایتون بود. روسوم تا مدت ها اجازه نمیداد که از تصویر مار پایتون به عنوان لوگوی این زبان استفاده شود؛ اما یکی از انتشاراتی که همیشه عکس جانوران را برای جلد کتاب های خود قرار میداد برای اولین بار تصویر مار پایتون را به عنوان جلد کتاب آموزش زبان برنامه نویسی پایتون قرار داد و از بعد آن نیز از تصویر دو مار زرد و آبی به عنوان نماد این زبان برنامه نویسی استفاده شد.

## استفاده های پایتون:

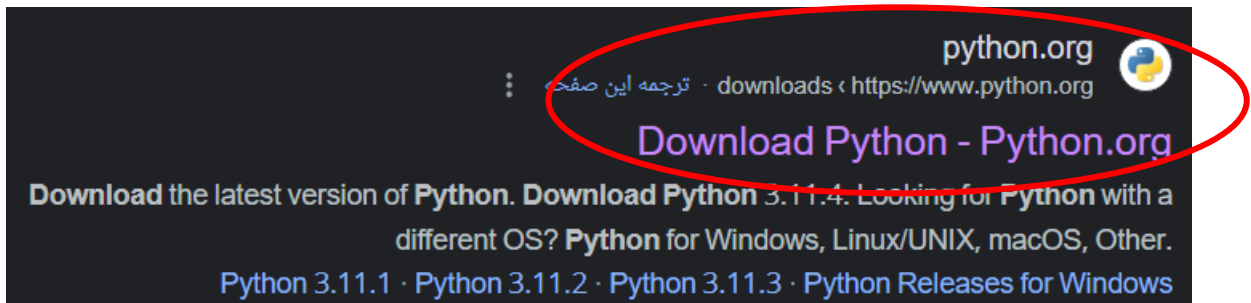
- توسعه وبسایت ها (سمت سرور): پایتون در توسعه صفحات وب بسیار محبوب است و با فریمورک هایی مانند Django, Flask برای ساخت صفحات وب با سرعت و دقت بالا بخصوص سمت سرور صفحات وب استفاده میشود.
- توسعه نرم افزار: از پایتون برای برنامه نویسی نرم افزار هایی با کاربرد های مختلف در اکثر پلتفرم های موجود مانند: ویندوز، لینوکس، اندروید، مک، وب اپ و... استفاده میشود که نشان از همه منظوره بودن این زبان است.
- ریاضیات و داده ها: با استفاده از کتابخانه های محبوبی مانند Numpy, Pymath, Scipy در پایتون میتوان محاسبات پیچیده ریاضی و سایر رشته ها را با آسودگی انجام داد و همین ویژگی پایتون را در سرعت بالای محاسبات محبوب کرده است.
- هوش مصنوعی: هوش مصنوعی نیز یکی از حوزه هایی است که پایتون و فریمورک های آن تمرکز ویژه ای بر آن دارند و کمپانی های بسیار بزرگی مانند گوگل و ناسا با استفاده از پایتون در تولیدات و اختراعات خود آن را به کار گرفته اند.
- اسکریپت نویسی سیستم: به علت قدرتمند بودن و سادگی پایتون میتوان با استفاده از آن فایل های سیستمی را مدیریت کرد و با پایگاه داده تعامل برقرار کرد یا وظایفی با استفاده از اسکریپت نویسی برای سیستم تعیین کرد.
- امنیت و شبکه: پایتون در حوزه هک و امنیت شبکه نیز کاربرد های فراوانی دارد. بسیاری از متخصصان این رشته برای شناسایی نقص های امنیتی و آزمایش امنیت یک پروژه از پایتون استفاده میکنند.

## چرا پایتون؟

- پایتون روی پلتفرم های مختلف مانند: مک، ویندوز، لینوکس و غیره قابل استفاده است.
- فرمت برنامه نویسی با پایتون شباهت بسیار زیادی با زبان انگلیسی دارد که برنامه نویسی با این زبان را ساده میکند.
- سادگی پایتون باعث شده که هنگام برنامه نوشتن با آن تعدادخطوط کمتری ایجاد شود و خروجی مناسبی ارائه دهد.
- پایتون بر روی یک سیستم مفسری ایجاد میشود. یعنی پایتون کد موردنظر را به صورت خط به خط و در زمان اجرا تفسیر میکند و کد را پس از نوشتن به راحتی میتوان اجرا کرد که سرعت پایتون را در مقایسه با زبان های کامپایلری که عملکرد پیچیده تر و سرعت پایین تری دارند و نیاز به ترجمه به یک زبان میانی دارند متمایز میکند.

نصب پایتون:

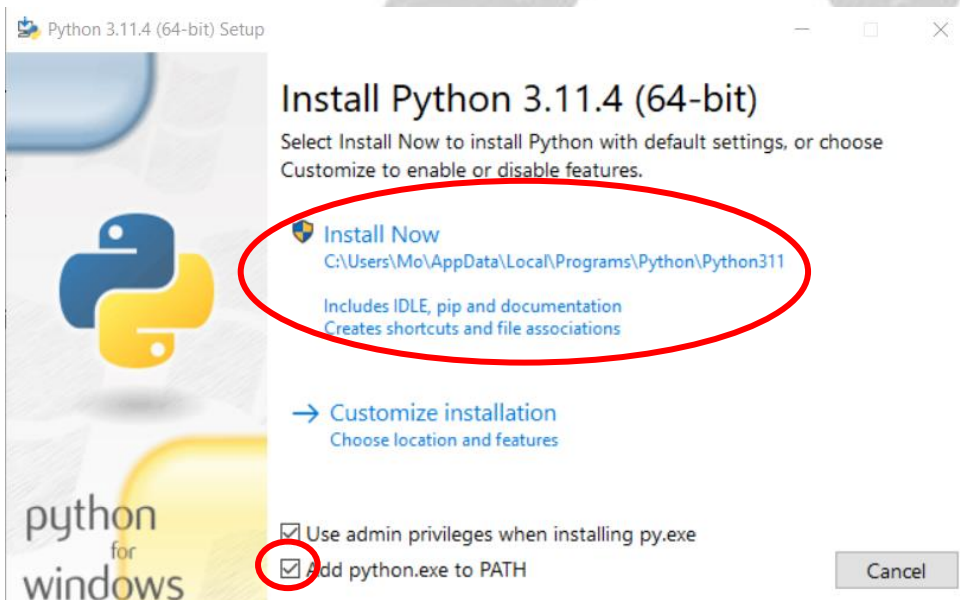
- برای نصب پایتون ابتدا به وبگاه رسمی پایتون یعنی [Python.org](https://www.python.org) رفته و در همان ابتدا نسخه پیشنهاد شده نرم افزار را با توجه به پلتفرم کاربر و نسخه آن دانلود میکنیم.



وبسایت رسمی دانلود پایتون Python.org

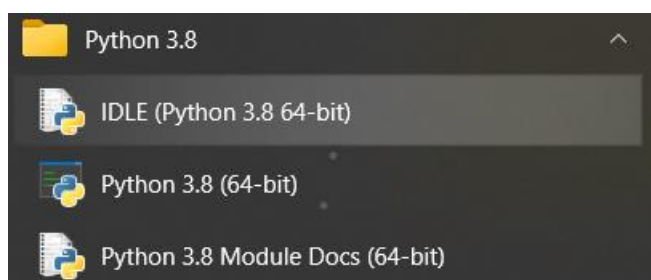


کلیک بر دکمه دانلود نسخه پیشنهادی بر اساس پلتفرم کاربر و نسخه آن

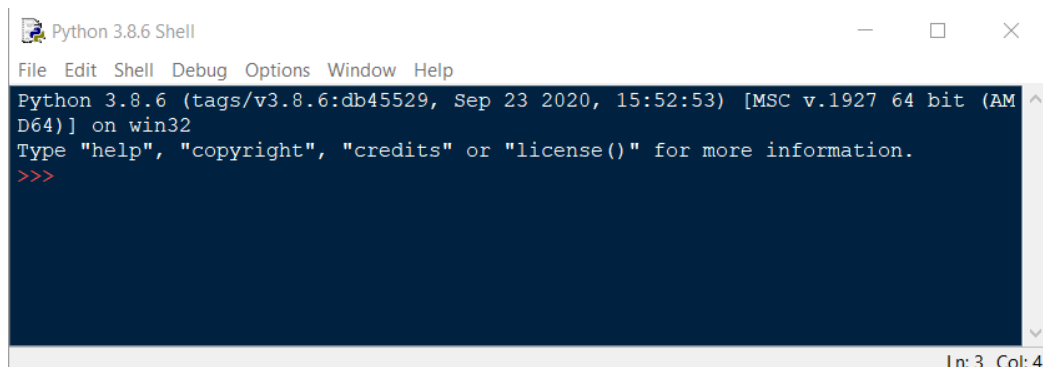


- پس از دانلود شدن فایل نصبی، آن را باز میکنیم و با رفتن به مسیر **Install Now** و همچنین قبول کردن دسترسی ادمین مسیر نصب را ادامه میدهیم. **هشدار ۱:** تیک روشن کردن **add to path** قبل از وارد شدن به مسیر نصب نهایی به هیچ وجه فراموش نشود چون این تیک درواقع اصلی ترین قسمت نصب است و درصورت فراموشی روشن کردن تیک، پیدا کردن آن در کنترل پنل دردسر خواهد بود.

**هشدار ۲:** در صورتی که تخصص ندارید و به انگلیسی وارد نیستید به هیچ وجه از مسیر **Customize Installation** استفاده نکنید و همه تیک ها را فعال نکنید.

استفاده از پایتون:

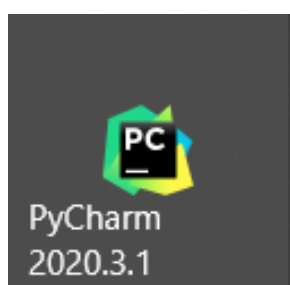
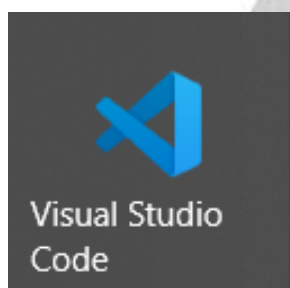
برای استفاده از پایتون میتوانید از IDLE یا همان محیط اجرای کد رسمی خود زبان برنامه نویسی پایتون استفاده کنید که با سرچ کردن IDLE در محیط دسکتاپ خود میتوانید به آن دسترسی پیدا کنید:



اما محیط برنامه نویسی اصلی پایتون یک سری مشکلاتی دارد. این بدین معنا نیست که پایتون به صورت کامل در آن اجرا نشود یا سرعت پایینی داشته باشد یا ناقص باشد. این محیط چون بر مبنای

تکنولوژی کمی قدیمی پایتون و سازگاری با تمام پلتفرم ها ایجاد شده ممکن است در نگاه اول دل برنامه نویس را بزند و زیاد جذاب به نظر نرسد و تصویری نه چندان جالب از تجربه استفاده از پایتون ایجاد کند. یا به عنوان مثال برنامه نویس هنگام کد زدن باید تمام پرانتز های باز و بسته و دستورات و کلمات کلیدی را به صورت دستی وارد کند که کمی خسته کننده است.

**راهکار:** برنامه نویس میتواند از محیط های ویرایش کد جایگزین استفاده کند که قابلیت پشتیبانی از پایتون را داشته باشند. به عنوان مثال بهترین محیط های جایگزین برای کدنویسی با پایتون میتوان به: Visual Studio Code, JetBrainsPyCharm, Atom Code Editor, Visual Studio و ده ها ویرایشگر دیگر اشاره کرد که سبک ترین و مناسب ترین ویرایشگر جامع در حال حاضر **Visual Studio Code** شناخته میشود که روی اکثر سیستم های ضعیف با حجمی در حدود ۹۰ الی ۱۰۰ مگابایت در دسترس است.

**چندی از مزیت های استفاده از ویرایشگر های کد (Code Editor):**

- سرعت بسیار بالا در تحلیل کد ها و اجرای آنها
- رنگ بندی مناسب اجزای کد و خسته نشدن برنامه نویس
- پوسته های مناسب با رنگ بندی ها ساده و زیبا برای دلنشین بودن کدزنی
- پشتیبانی از اکثر زبان های برنامه نویسی روز دنیا
- افزایش سرعت در کدزنی با استفاده از میانبر های کد
- استفاده از افزونه های کاربردی برای دقت بیشتر و نظم بیشتر کد
- اجرای درون برنامه ای کد و بدون نیاز به ران کردن مجزای cmd
- لینک کردن پروژه با تیم و کدنویسی اشتراکی با اعضای تیم
- دسترسی راحت تر به فایلای پوشه مربوط به پروژه

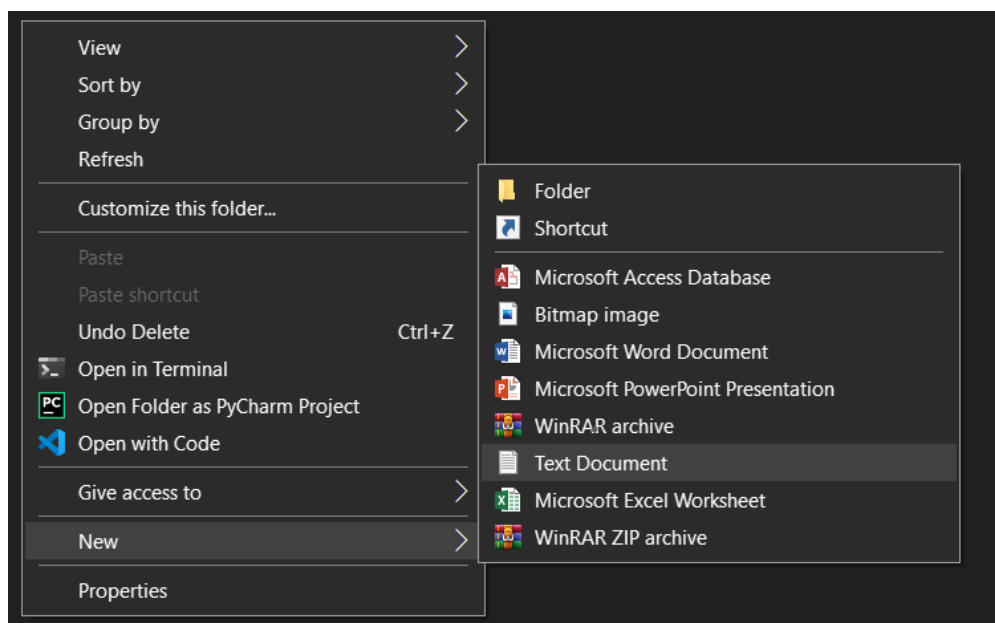
## شکل کلی زبان برنامه نویسی پایتون:

همانطور که اشاره شد پایتون یک زبان برنامه نویسی سطح بالا، تفسیری و شیء‌گرا است. اگر بخواهیم به نقشه کلی پایتون بنگریم، شکل کلی زبان پایتون شامل عناصر زیر میشود:

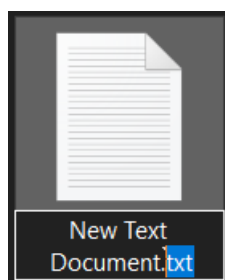
- **دستورات:** پایتون دارای دستورات مختلفی است که برای کنترل جریان اجرای برنامه، تعریف متغیرها، عملیات ریاضی و بسیاری از وظایف دیگر استفاده میشوند.
- **متغیرها و نوع داده ها:** برنامه نویس در پایتون میتواند متغیرها را تعریف کند و به آنها مقدار دهد. پایتون انواع داده های مختلفی را از جمله عدد صحیح (integer)، عدد اعشاری (float)، رشته (string) و بولین (Boolean) پشتیبانی میکند. همچنین پایتون اجازه میدهد تا متغیرهای پیچیده تری مانند لیست ها، دیکشنری ها و تاپل ها را نیز تعریف کنید که هر کدام فرمت و کاربرد خاص و جداگانه خود را دارند.
- **ساختارهای کنترلی:** پایتون دارای ساختارهای کنترلی مانند ساختارهای شرطی if-else، حلقه های تکرار for-while و سایر ساختارهای کنترلی مشابه است که به برنامه نویس امکان میدهد تا جریان اجرای برنامه را کنترل کند. دستورات شرطی بر اساس شرایط مشخصی تصمیم میگیرند که بخشی از کد اجرا شود یا نه و حلقه های تکرار برای تکرار یک بخش از کد بر اساس شرایط خاصی مشخص میشوند.
- **توابع و متدها:** در پایتون میتوان توابعی را تعریف کرد که یک تکه کد را اجرا کنند و نتیجه را برگردانند (return). توابع با استفاده از کلید واژه def تعریف میشوند و میتوانند با استفاده از پارامترها و آرگومان ها، ورودی مشخصی دریافت کنند و خروجی مطلوبی ارائه دهند. همچنین متدها نیز توابعی هستند که درون یک کلاس تعریف میشوند و با شیء مربوطه وابسته هستند.
- **کتابخانه ها:** پایتون دارای یک بسته فراوان از کتابخانه ها است که عملکرد و قابلیت های بیشتری را به برنامه های پایتونی اضافه میکنند. با استفاده از کتابخانه های پایتون، میتوان وظایف خاصی مانند کار با داده ها، تحلیل آماری، برنامه نویسی وب و بسیاری از وظایف دیگر را انجام داد. برای استفاده از کتابخانه های پایتونی میتوان از دستور import برای وارد کردن و به کار گرفتن کتابخانه مورد نظر به کد استفاده کرد.
- **شیء گرایی:** پایتون از مفهوم شیء‌گرایی پشتیبانی میکند که به برنامه نویس امکان می دهد کلاس ها و شیء ها را تعریف کند و از آن ها برای ساختار دهی به برنامه و مدیریت کد استفاده کند. کلاس ها مانند الگو های قالب برای ساختار دهی به برنامه و مدیریت کد عمل میکنند و شیء ها نمونه های خاصی از یک کلاس هستند که میتوانند دارای ویژگی ها (متغیر ها) و رفتار ها (متدها) باشند. با استفاده از شیء‌گرایی میتوان کد های خود را به صورت ماژولار و قابل توسعه طراحی کرد.

با استفاده از این عناصر، شما میتوانید برنامه های متنوعی را در پایتون بنویسید و از قابلیت های پویای این زبان برنامه نویسی بهره برداری کنید.

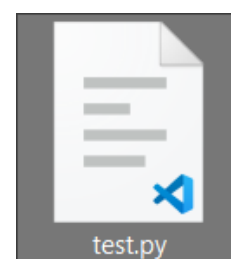
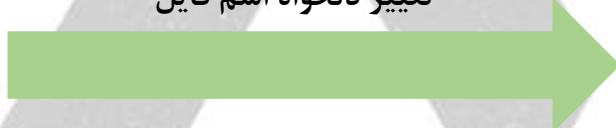


شروع پایتون نویسی:

برای شروع لازم است بدانید که فایل هایی که مربوط به زبان برنامه نویسی پایتون میشوند پسوندی با فرمت **(.py)** دارند و برای ایجاد یک فایل مربوط به پایتون ابتدا باید در پوشه ای که می‌خواهیم کد هایمان را قرار دهیم یک **text document** بسازیم و برای انجام اینکار ابتدا کلیک راست موس را فشرده و سپس از منوی **New** گزینه **text document** را انتخاب کنیم. پس از آن پسوند فایل را از **.txt** به **.py** تغییر می‌دهیم و تمام! فایل پایتونی ما ساخته میشود!

تغییر فرمت از **.txt** به **.py**

تغییر دلخواه اسم فایل



بعد از تغییر فرمت فایل و ایجاد فایل پایتونی، حال باید در هر محیطی که مورد نظر داریم فایل پایتون را باز کنیم و شروع کنیم به کد نویسی.

برای شروع می‌خواهیم یک برنامه خیلی ساده پایتونی بنویسیم؛ در پایتون از تابع **print** برای چاپ کردن مقداری که می‌خواهیم استفاده میکنیم. برای چاپ کردن رشته **"Hello, World!"** اینگونه کد را مینویسیم:

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

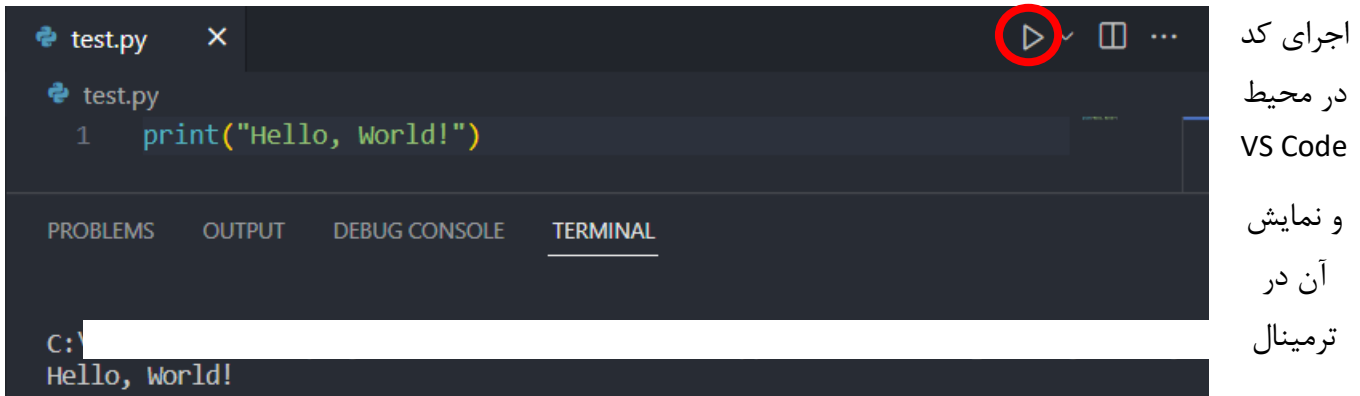
کد وارد شده:

```
C:\>python test.py
Hello, World!
```

خروجی پس از اجرا:



- برای اجرای کد های پایتون، در صورتی که برنامه نویس از ویرایشگر های کد استفاده کند، با نصب افزونه مربوط به پایتون میتواند کد را در ترمینال درون خود ویرایشگر اجرا کند و نسبت به ترمینال اصلی دسکتاپ بی نیاز است.



- اما اگر برنامه نویس از ویرایشگر کد استفاده نکند، میتواند کد را در محیط کنسول یا ترمینال خود دسکتاپ یا cmd در ویندوز استفاده کند. برای این کار باید ترمینال را در مسیر فولدر فایل باز کند و دستور زیر را وارد کند:

## python <نام فایل>.py




وارد کردن cmd در مسیر فولدر فایل کد



دستور اجرا کردن برنامه

**نکته:** اهمیت add to path اینجا ظاهر میشود و اگر تیک مربوط به آن هنگام نصب روشن نمیبود، اجرای فایل در دسدر بود.

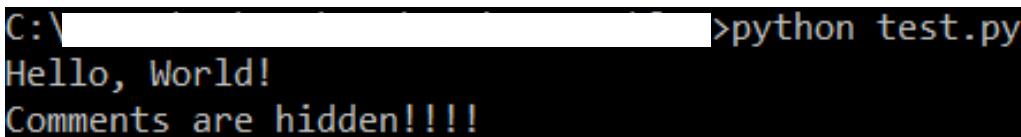
- مهم ترین بخش کد های پایتون، مربوط به پرانتز و مقداری است که درون پرانتز باز و بسته قرار میگیرد. در برنامه ما "Hello, World!" مقدار تابع print است که درون پرانتز ( ) باز و بسته قرار میگیرد.
- برای چاپ کردن مقدار هایی که شامل چندین کاراکتر است مانند Hello, World! ، باید مقدارمان را درون دابل کوتیشن (") یا کوتیشن (') باز و بسته قرار گیرد وگرنه کد ما اجرا نمیشود مگر اینکه متغیری با آن داشته باشیم که در درس های بعدی توضیح داده می شود.
- پایتون به حروف بزرگ و کوچک حساس است و ما برای نوشتن توابع مختلف نمیتوانیم از ترکیبی از حروف بزرگ کوچک استفاده کنیم چون به طور پیشفرض پایتون فقط توابع با حروف کوچک را اجرا میکند مثلاً برای تابع print نمیتوانیم از Print, PRINT, pRiNt استفاده کنیم.



```

1 print("Hello, World!") #this is a comment
2 # another comment
3 # And Anotherrrrrr Comment # Commenttttssss
4 # this is comment again: print("Hello, World!")
5 print("Comments are hidden!!!!") #Yes comments are hidden

```



```

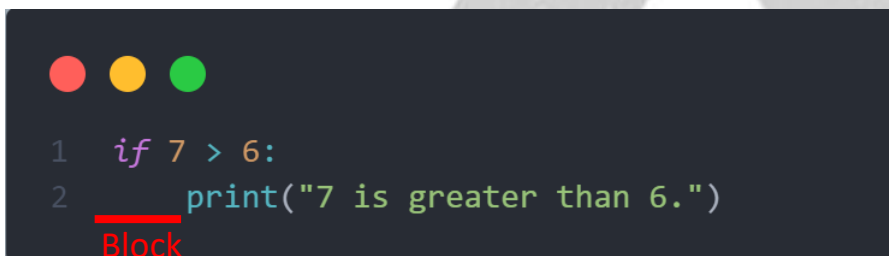
C:\>python test.py
Hello, World!
Comments are hidden!!!!

```

- کامنت ها در زبان های برنامه نویسی نقش مهمی بازی میکنند. کامنت به برنامه نویس کمک میکند تا به کد خود جلوه کامل تر، خوانا تر و منظم تری دهد که اگر برنامه نویس دیگری قصد

داشت آن کد ها را تحلیل کند به مشکل بر نخورد و کاربرد هر قسمت در کد را متوجه شود یا حتی اگر پس از مدتی خود برنامه نویس اصلی به کد رجوع کرد علت استفاده از هر قسمت را متوجه شود. به گونه ای میتوان گفت که کامنت ها در کد نقش یادداشت را دارند. برای کامنت کردن مطلبی در پایتون باید از **مربع** یا **هشتگ** (#) استفاده کنیم. به گونه ای که هر مطلبی در هر حالتی بعد از کامنت بیاید، مفسر آن را نادیده میگیرد یعنی از نماد مربع تا خط بعد از آن کامنت میشود و هیچ تاثیری در کد ندارد.

- فاصله ها یا همان space ها نقش مهمی در پایتون دارند و نباید بی جا و بی دلیل از آنها استفاده کرد وگرنه کد به مشکل برمیخورد.



```

1 if 7 > 6:
2     print("7 is greater than 6.")

```

Block

- در کد روبرو با مفهوم بلوک (blocks) آشنا میشویم. بلوک ها نیز نقش مهمی در کد دارند و در صورت رعایت نکردن حالت بلوکی کد، برنامه ما به مشکل بر میخورد

و اجرا نمیشود. بلوک ها در گونه های مختلفی در اکثر زبان های برنامه نویسی وجود دارند اما در پایتون به صورت ۴ عدد فاصله یا یک عدد تب (tab) ظاهر میشوند.

**توجه:** نیازی به توجه به مفهوم کد نیست. فقط لازم است بدانید که در اکثر مواقعی که دونقطه (:) گذاشته میشود و سراغ خط بعدی میرویم باید حداقل یک بلوک از ابتدای خط فاصله داشته باشیم. اکثر ویرایشگر های کد به صورت خودکار این عمل را انجام میدهند و نیازی به وارد کردن دستی بلوک توسط برنامه نویس نیست.

## کتابخانه ها در پایتون (libraries):

کتابخانه ها در پایتون به مجموعه ای از کدها، توابع و ابزارهای کاربردی گفته میشود که به صورت مجزا نوشته شده و به برنامه های پایتونی کمک میکنند. کتابخانه ها در واقع پکیج هایی هستند که توابع و ابزار های مختلفی را برای انجام وظایف خاص فراهم میکنند. به شیوه ای دیگر میتوان گفت که کتابخانه ها، افزونه های کاربردی زبان برنامه نویسی پایتون حساب میشوند.

آموزش برنامه سازی با پایتون به زبان ساده

تهیه کننده: علیزاده

با استفاده از کتابخانه ها، برنامه نویسان قادر به استفاده از توابع و قابلیت های مهم و پرکاربرد دیگری هستند که توسط افراد دیگر نوشته شده و به اشتراک گذاشته شده اند. این کتابخانه ها به برنامه نویسان امکان کاهش زمان و تلاش لازم برای پیاده سازی وظایف مختلف و خاص را میدهند و قابلیت های قدرتمندی را به پروژه های پایتون اضافه میکنند.

```
1 import translate # این کتابخانه برای ترجمه استفاده میشود
2 import pygame # این کتابخانه برای بازی سازی استفاده میشود
3 import tkinter # این کتابخانه برای گرافیکی استفاده میشود
4 from tkinter import Tk # در اینجا قسمتی از یک کتابخانه را استفاده میکنیم
5 import math # این کتابخانه برای ریاضیات استفاده میشود
6 import persian # این کتابخانه برای زبان فارسی استفاده میشود
7 import wikipedia # این کتابخانه برای استفاده از ویکی پدیا وارد میشود
```

به عنوان مثال در کد بالا میبینیم که چندین کتابخانه در کنار هم برای کدنویسی بهتر وارد شده اند که هر کدام وظیفه خاص خود را دارند. برای استفاده کردن از یک کتابخانه از دستور `import` و برای استفاده کردن از قسمتی از یک کتابخانه از دستور `from` استفاده میکنیم.

**نکته ۱:** بیش از هزاران کتابخانه داریم که هر کدام کاربرد خاص خودشان را دارند. بنابر میزان خاص بودن نیز شیوه متفاوتی در نوشتار کد خود دارند که با رجوع به اسناد مربوط به هر کتابخانه، آنها را میآموزید.

**نکته ۲:** هر کتابخانه ای که بخواهیم استفاده کنیم باید در اول کد پیش از ساختن هر متغیر و تابعی آن کتابخانه هارا وارد کنیم و سپس به کدنویسی بپردازیم. اگر وسط کد، کتابخانه `import` به پاسخ مطلوبی نمیرسیم.

**نکته ۳:** هر کتابخانه شناسه اختصاری خود را دارد که با وارد کردن دستور زیر در `cmd` و همچنین شناسه اختصاری کتابخانه، میتوانیم کتابخانه مربوطه را نصب کنیم و از آن استفاده کنیم.

## <شناسه اختصاری کتابخانه> pip install

```
C:\> pip install persian
Requirement already satisfied: persian in c:\python38\lib\site-packages (0.5.0)
WARNING: You are using pip version 22.0.4; however, version 23.1.2 is available.
You should consider upgrading via the 'c:\python38\python.exe -m pip install --upgrade pip' command.
```

**نکته ۴:** هیچ کتابخانه ای بدون نصب داشتن از پیش عمل نمیکند و کد شما به مشکل بر میخورد پس قبل از شروع کدنویسی حتما کتابخانه هایی که نیاز دارید را نصب کنید.

**نکته ۵:** برای آشنایی با شناسه اختصاری هر کتابخانه و آموزش های مربوط به آن و اسناد آن میتوانید به پایگاه وب [pypi.org](https://pypi.org) رجوع کنید.

شیء ، کلاس و متد:

- **کلاس:** کلاس یک الگوی تعریف شده است که شامل متغیرها و توابع است. این الگو مشخص میکند که یک شیء از چه ویژگی ها و رفتار هایی برخوردار است. برای تعریف یک کلاس در پایتون از کلیدواژه `class` استفاده می شود.

```

1 class person:
2     def __init__(self, name, age):
3         self.name = name
4         self.age = age
5
6     def greet(self):
7         print("Hello, my name is", self.name)

```

- **شیء:** شیء یا نمونه یک نمونه مشخص از یک کلاس است. با استفاده از یک کلاس، میتوانید شیئی از آن ایجاد کنید و به ویژگی ها و روش های آن دسترسی داشته باشید. برای ایجاد شیء در پایتون از نام کلاس به عنوان یک تابع استفاده میشود.

```

1 person1 = person("John", 25)

```

- **متد:** متد ها یا توابع درون کلاس ها تعریف میشوند و به شیء مربوطه مربوط میشوند. آنها رفتار ها و عملکرد های کلاس را تعریف میکنند. متدها در پایتون معمولاً با استفاده از کلیدواژه `def` تعریف میشوند و نیاز به پارامتر `self` دارند که به شیء فعلی ارجاع میدهد.

```

1 person1.greet()

```

اگر درنهایت در ترکیبی از کدهای بالا، کد چپ را داشته باشیم، خروجی راست را خواهیم داشت:

```

1 class person:
2     def __init__(self, name, age):
3         self.name = name
4         self.age = age
5
6     def greet(self):
7         print("Hello, my name is", self.name)
8
9 person1 = person("John", 25)
10 person1.greet()

```

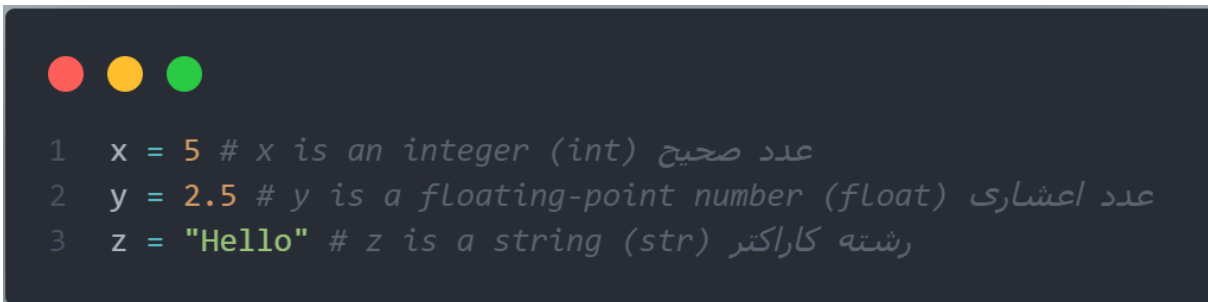
Hello, my name is John

در اینجا ما کلاسی با نام `person` ساختیم که اشیائی با محتویات `john` , 25 , داشت و آنها را به ترتیب `name` , `age` تعریف کردیم و با متدی که داشتیم از `greet` استفاده کردیم تا سن و نام را بدون نوشتن در دابل کوتیشن ، دریافت کنیم.

متغیر ها (variables) و داده ها در پایتون:

متغیر: در پایتون، متغیرها به منظور ذخیره و استفاده از داده‌ها تعریف می‌شوند. برای تعریف یک متغیر در پایتون، نام متغیر را مشخص کرده و سپس مقدار آن را به آن اختصاص می‌دهیم. در اینجا متغیر X تعریف شده و مقدار ۵ به آن اختصاص داده شده است.

در پایتون، نیازی به تعریف نوع داده‌ای متغیر نیست، به این معنی که تعیین نوع داده‌ای متغیر برای پایتون به صورت خودکار انجام می‌شود و با توجه به نوع داده‌ای مقدار متغیر، نوع آن به صورت خودکار تعیین می‌شود. برای مثال:

**نکات:**

- نام متغیر باید با یکی از حروف الفبا (a-z or A-Z) یا علامت \_ شروع شود.
- نمی‌تواند شامل کاراکترهای غیرمجاز مانند #, ?, ^, \$, . باشد.

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	

- نمی‌توان از کلمات رزرو شده در پایتون برای نام متغیر استفاده کرد:

- نام متغیر نباید دارای فضای خالی (spaces) باشد.

- اسامی متغیرها نسبت به بزرگی و کوچکی حروف حساس هستند. در پایتون دو حرف مانند a و A دو کاراکتر مختلف به حساب می‌آیند.

با توجه به قابلیت‌های متغیرهای پایتون، شما می‌توانید از آنها در کدنویسی خود استفاده کنید و به راحتی با داده‌های مختلف کار کنید.

**داده:** داده در پایتون به هر نوع اطلاعاتی گفته می‌شود که در برنامه‌نویسی استفاده می‌شود. در پایتون، داده‌ها می‌توانند به صورت مختلفی تعریف شوند که به انواع داده‌ها در صفحه بعدی می‌پردازیم.

## آموزش برنامه سازی با پایتون به زبان ساده انواع داده:

تهیه کننده: علیزاده



```
1 x = 10 # int
2 y = 3.14 # float
```

- اعداد: به صورت عددی صحیح (integer) و اعشاری (float) می‌توانند تعریف شوند. مثال:



```
1 name = "John"
```

- رشته‌ها: مجموعه‌ای از حروف و کاراکترها که بین دو علامت نقل قول تعریف می‌شوند. مثال:



```
1 my_list = [1, 2, "three", True]
```

- لیست‌ها: مجموعه‌ای از داده‌های متفاوت با ترتیب مشخص، که در بین دو علامت [] و با استفاده از کاما جدا می‌شوند. مثال

- دیکشنری‌ها: مجموعه‌ای از داده‌های متفاوت که هر کدام با یک کلید (key) منحصر به فرد شناخته می‌شوند. مثال



```
1 my_dict = {"name": "John", "age": 30, "is_student": True}
2 # dictionaryVar = {Key1:Value1, Key2:Value2, Key3:Value3}
```

- تاپل‌ها: مجموعه‌ای از داده‌های متفاوت با ترتیب مشخص، که در بین دو علامت () و با استفاده از کاما جدا می‌شوند و بعد از تعریف قابل تغییر نیستند. مثال:



```
1 my_tuple = ("apple", "banana", "cherry")
```

- مجموعه‌ها: مجموعه‌ای از داده‌های تکراری نشده و بدون ترتیب مشخص، که در بین دو علامت {} و با استفاده از کاما جدا می‌شوند. مثال:



```
1 my_set = {"apple", "banana", "cherry"}
```

- Boolean: شامل دو مقدار true یا false می‌باشد. مثال:



```
1 my_bool1 = True
2 my_bool2 = False
```

به مثال زیر توجه کنید:



```
1 num1 = num2 = num3 = num4 = num5 = 10
2 message1 = message2 = message3 = "Hello World!"
3
4 print(num1)
5 print(num4)
6 print(message1)
7 print(message3)
```

```
10
10
Hello World!
Hello World!
```

دقت کنید که برای متغیرهای تعریف شده در حالت بالا یک خانه حافظه تخصیص داده می شود، یعنی مقدار ۱۰ در حافظه ذخیره شده و متغیرهای num1 و num2 و num3 و num4 و num5 به آن خانه از حافظه اشاره می کنند. همچنین می توان چند متغیر را تعریف کرد و برای هر یک از آن ها مقدار جداگانه ای مشخص نمود:



```
1 num1, num2, message1 = 10, 12.5, "Hello World!"
2
3 print(num1)
4 print(num2)
5 print(message1)
```

```
10
12.5
Hello World!
```

در کد بالا مقدار num1 برابر ۱۰، num2 برابر ۱۲٫۵ و message1 برابر Hello World! می باشد. در پایتون، متغیرها هم باید تعریف و هم مقداردهی شوند. یعنی اگر متغیری را تعریف کرده و به آن مقداری را اختصاص ندهید و برنامه را اجرا کنید با خطا مواجه می شوید:



```
1 number
2
3 print(number)
```

```
Traceback (most recent call last):
  File "test.py", line 1, in <module>
    number
NameError: name 'number' is not defined
```

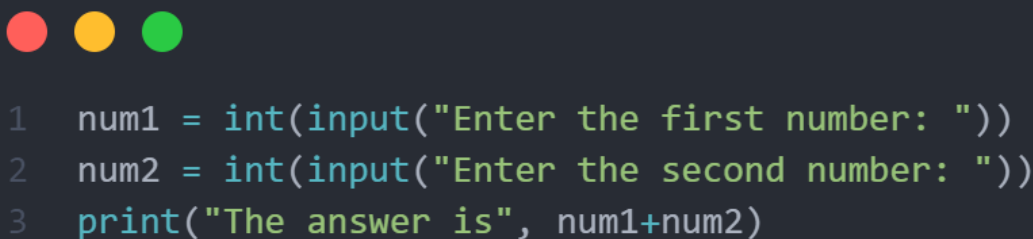


دستورات ورودی و دستورات خروجی در پایتون:

**دستورات ورودی:** دستورات ورودی (Input) در پایتون به کار می‌روند تا اطلاعات را از کاربر یا منبع دیگری که به برنامه شما وصل است دریافت کنند. برای خواندن داده ها از تابع `input()` استفاده می‌شود. این تابع با فراخوانی، متنی را به کاربر نمایش می‌دهد تا او بتواند داده مورد نظر خود را وارد کند.

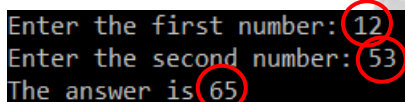
**دستورات خروجی:** دستورات خروجی (Output) در پایتون از طریق تابع `print()` صورت می‌گیرد. این تابع می‌تواند متغیرها و محاسبات را چاپ کند.

مثال ۱: برنامه ای بنویسید که ۲ عدد صحیح از کاربر بگیرد و آن‌ها را جمع کند:



```
1 num1 = int(input("Enter the first number: "))
2 num2 = int(input("Enter the second number: "))
3 print("The answer is", num1+num2)
```

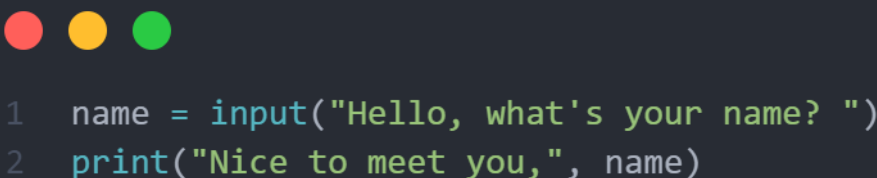
- در این برنامه ما دو ورودی با نام های `num1` و `num2` گرفتیم.
- پس از آن، از حاصل جمع `num1` و `num2` چاپ و خروجی گرفتیم.
- همچنین پیش از تابع `input` مشخص کردیم که برنامه ما با اعداد صحیح کار میکند یعنی `int` ها.
- در قسمت چاپ نیز، پیش از حاصل جمع، متن "The answer is" را قرار دادیم تا برنامه جلوه بهتری پیدا کند.



```
Enter the first number: 12
Enter the second number: 53
The answer is 65
```

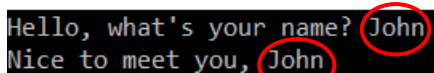
خروجی ای که از کد میگیریم بدین صورت است:

مثال ۲: برنامه ای بنویسید که نام کاربر را ورودی بگیرد و به او خروجی خوشامدگویی بدهد:



```
1 name = input("Hello, what's your name? ")
2 print("Nice to meet you,", name)
```

- در این برنامه ما یک ورودی با نام `name` گرفتیم.
- پس از آن، پیام خوشامدگویی ای که در نظر داریم را همراه با نام کاربر چاپ و خروجی میگیریم.



```
Hello, what's your name? John
Nice to meet you, John
```

خروجی ای که از کد میگیریم بدین صورت است:

پایتون و ریاضیات:

**عملگر ها:** نمادهایی هستند که اعمال خاصی را انجام می دهند و بر عملوند ها تاثیر میگذارند.

**عملوند ها:** مقادیری که عملگرها بر روی آنها عملی انجام می دهند و از عملگرها تاثیر میبینند.

مثال: در عبارت  $6+12-4$ ، اعداد 6, 12, 4 عملوند محسوب میشوند و +, - عملگر محسوب میشوند.

**عملگر های ریاضی:**

پایتون از عملگر های ریاضی برای انجام محاسبات استفاده میکند:

+ : برابر است با حاصل جمع عملوند ۱ با عملوند ۲  $var1 = var2 + var3$

- : برابر است با حاصل تفریق عملوند ۱ از عملوند ۲  $var1 = var2 - var3$

\* : برابر است با حاصل ضرب عملوند ۱ در عملوند ۲  $var1 = var2 * var3$

/ : برابر است با حاصل تقسیم عملوند ۱ بر عملوند ۲  $var1 = var2 / var3$

% : برابر است با باقیمانده تقسیم عملوند ۱ بر عملوند ۲  $var1 = var2 \% var3$

\*\* : برابر است با عملوند ۱ به توان عملوند ۲  $var1 = var2 ** var3$

// : برابر است با تقسیم عملوند ۱ بر عملوند ۲ ( نمایش حاصل بصورت عدد صحیح )  $var1 = var2 // var3$

مثال: محاسبات تمام حالت های عملگر های ریاضی با دو عملوند ۱۰ و ۲۰:

```
1 num1 = 10
2 num2 = 20
3
4 print(f"{num1} + {num2} =", num1+num2) # حاصل جمع
5 print(f"{num1} - {num2} =", num1-num2) # حاصل تفریق
6 print(f"{num1} * {num2} =", num1*num2) # حاصل ضرب
7 print(f"{num1} / {num2} =", num1/num2) # حاصل تقسیم
8 print(f"{num1} \% {num2} =", num1%num2) # باقیمانده تقسیم
9 print(f"{num1} ** {num2} =", num1**num2) # توان
10 print(f"{num1} // {num2} =", num1//num2) # حاصل تقسیم به صورت عدد صحیح
```

نکته: حرف f را قبل از باز کردن نقل قول گذاشتیم، علت آن استفاده از دستور فرمت است. فرمت در این کد به ما کمک کرده تا نام متغیر را درون کد قرار دهیم بدون اینکه نیاز باشد مقدار متغیر را به صورت دستی وارد کنیم. در این کد ما نام متغیر را درون {} قرار داده ایم.

```
10 + 20 = 30
10 - 20 = -10
10 * 20 = 200
10 / 20 = 0.5
10 \% 20 = 10
10 ** 20 = 100000000000000000000
10 // 20 = 0
```

خروجی ای که از برنامه بالا میگیریم بدین صورت است:

## عملگر های مقایسه ای:

از عملگر های مقایسه ای برا مقایسه مقادیر استفاده میشود. نتیجه این مقادیر یک مقدار بولی (منطقی) است. این عملگر ها اگر نتیجه مقایسه دو مقدار درست باشد مقدار True و اگر نتیجه مقایسه اشتباه باشد مقدار False را نشان میدهند.

== : خروجی در صورتی True است که مقدار عملوند ۱ با عملوند ۲ مساوی باشد و در غیر این صورت False است.

!= : خروجی در صورتی True است که مقدار عملوند ۱ با عملوند ۲ مساوی نباشد و در غیر این صورت False است.

< : خروجی در صورتی True است که مقدار عملوند ۱ از عملوند ۲ کوچکتر باشد و در غیر این صورت False است.

> : خروجی در صورتی True است که مقدار عملوند ۱ از عملوند ۲ بزرگتر باشد و در غیر این صورت False است.

<= : خروجی در صورتی True است که مقدار عملوند ۱ از عملوند ۲ کوچکتر یا مساوی باشد و در غیر این صورت False است.

>= : خروجی در صورتی True است که مقدار عملوند ۱ از عملوند ۲ بزرگتر یا مساوی باشد و در غیر این صورت False است.

مثال: محاسبات تمام حالت های عملگر های مقایسه ای با دو عملوند ۱۰ و ۲۰:

```
1 num1 = 10
2 num2 = 20
3
4 print(f"{num1} == {num2} =", num1==num2) # تساوی
5 print(f"{num1} != {num2} =", num1!=num2) # تضاد
6 print(f"{num1} < {num2} =", num1<num2) # کوچکتر
7 print(f"{num1} > {num2} =", num1>num2) # بزرگتر
8 print(f"{num1} <= {num2} =", num1<=num2) # کوچکتر مساوی
9 print(f"{num1} >= {num2} =", num1>=num2) # بزرگتر مساوی
```

```
10 == 20 = False
10 != 20 = True
10 < 20 = True
10 > 20 = False
10 <= 20 = True
10 >= 20 = False
```

خروجی ای که از برنامه بالا میگیریم بدین صورت است:

**عملگر های جایگزینی:** این عملگر ها نیز عملکرد مشابهی همچون عملگر های ریاضی دارند و تفاوت زیادی ندارند و صرفا برای ساده کردن محاسبات ریاضی نوشته میشوند به عنوان مثال به جای نوشتن  $2+5=2$  مینویسیم  $2+=5$  را مینویسیم و مستقیما ۷ بدست می آید.

برخی از عملگر های جایگزینی که از قواعد عملگر های ریاضی پیروی میکنند:  $=$ ,  $+=$ ,  $-=$ ,  $*$ ,  $/$ ,  $\%$ ,  $**$ ,  $//$

**تقدم عملگر ها:** اولویت عملگر های محاسباتی در پایتون به صورت زیر از چپ به راست است ( میتوان با پرانتز نقض کرد ) :

$**$ ,  $//$ ,  $\%$ ,  $/$ ,  $*$ ,  $-$ ,  $+$ ,  $!=$ ,  $==$ ,  $*$ ,  $**$ ,  $=$ ,  $+=$ ,  $-=$ ,  $//$ ,  $/$ ,  $\%$ ,  $=$

## دستورات شرطی:

دستورات شرطی در پایتون به برنامه نویسان اجازه میدهند تا با استفاده از یک عبارت شرطی، بخشی از کد را به اجرا در بیاورند یا آن را رد کنند. در پایتون سه نوع دستور شرطی وجود دارد: `if` , `elif` , `else`

### دستور `if`:

با استفاده از دستور `if` ، میتوان یک شرط را بررسی کرد و در صورت درستی آن، یک قسمتی از کد را اجرا کرد.

مثال: در این کد،  $x > 0$  بررسی میکند که آیا  $x$  مثبت است یا نه و اگر مثبت بود نتیجه را چاپ کند:



```
1 x = 5
2 if x > 0:
3     print("x is positive.")
```

در خروجی این کد اگر  $x$  عددی بیشتر از 0 باشد `x is positive.` چاپ میشود و اگر  $x$  عددی منفی یا کمتر از 0 باشد، خروجی خاصی دریافت نمیشود.

### دستور `else`:

دستور `else` نیز برای مواردی استفاده میشود که در صورت عدم برآورده شدن هیچ یک از شروط قبلی، قسمتی از کد اجرا شود.

مثال: در این کد،  $x > 0$  بررسی میکند که آیا  $x$  مثبت است و اگر مثبت بود نتیجه ی مربوط به مثبت را چاپ کند و بررسی میکند که اگر  $x$  منفی است نتیجه ی مربوط به منفی را چاپ کند:



```
1 x = -8
2 if x > 0:
3     print("x is positive.")
4 else:
5     print("x is not positive.")
```

در خروجی این کد اگر  $x$  عددی بیشتر از 0 باشد `x is positive.` چاپ میشود و اگر  $x$  عددی منفی باشد `x is not positive.` چاپ میشود و اگر  $x$  عدد 0 باشد خروجی خاصی دریافت نمیشود.

### دستور `elif`:

دستور `elif` نیز این امکان را میدهد که چندین شرط را در یک بلوک بررسی شود و در صورت برآورده نشدن هیچکدام از این شروط، `else` انجام شود.



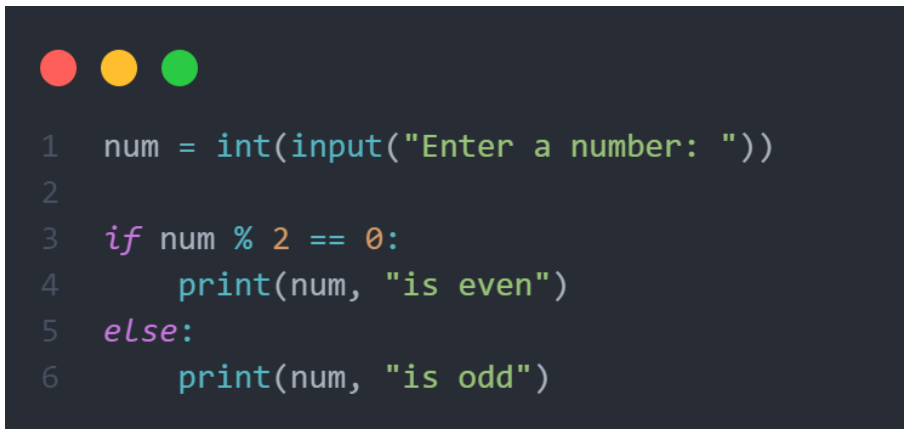
```
1 x = 0
2 if x > 0:
3     print("x is positive.")
4 elif x == 0:
5     print("x is zero(0).")
6 else:
7     print("x is not positive.")
```

مثال: در این کد،  $x > 0$  بررسی میکند که آیا  $x$  مثبت است و اگر مثبت بود نتیجه ی مربوط به مثبت را چاپ کند و `else` بررسی میکند که اگر  $x$  منفی است نتیجه ی مربوط به منفی را چاپ کند و `elif x == 0` بررسی میکند که  $x$  صفر است یا نه و نتیجه ی مربوط را چاپ کند:

مثال: با استفاده از اصول `if, else` برنامه ای بنویسید که اعداد زوج و فرد را تشخیص دهد:

در این برنامه ابتدا یک ورودی اعداد صحیح از کاربر میخواهیم سپس با استفاده از عملگر `%` که باقیمانده تقسیم را محاسبه میکند، باقیمانده تقسیم عدد ورودی بر ۲ را محاسبه میکنیم. طبق قوانین حاکم بر ریاضی اگر باقیمانده این تقسیم صفر شود عدد ورودی زوج است و اگر هرجوایی به جز صفر داشته باشد، عدد ورودی، عددی فرد است.

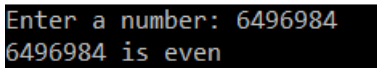
خروجی کد بالایی بدین صورت است:



```

1 num = int(input("Enter a number: "))
2
3 if num % 2 == 0:
4     print(num, "is even")
5 else:
6     print(num, "is odd")

```



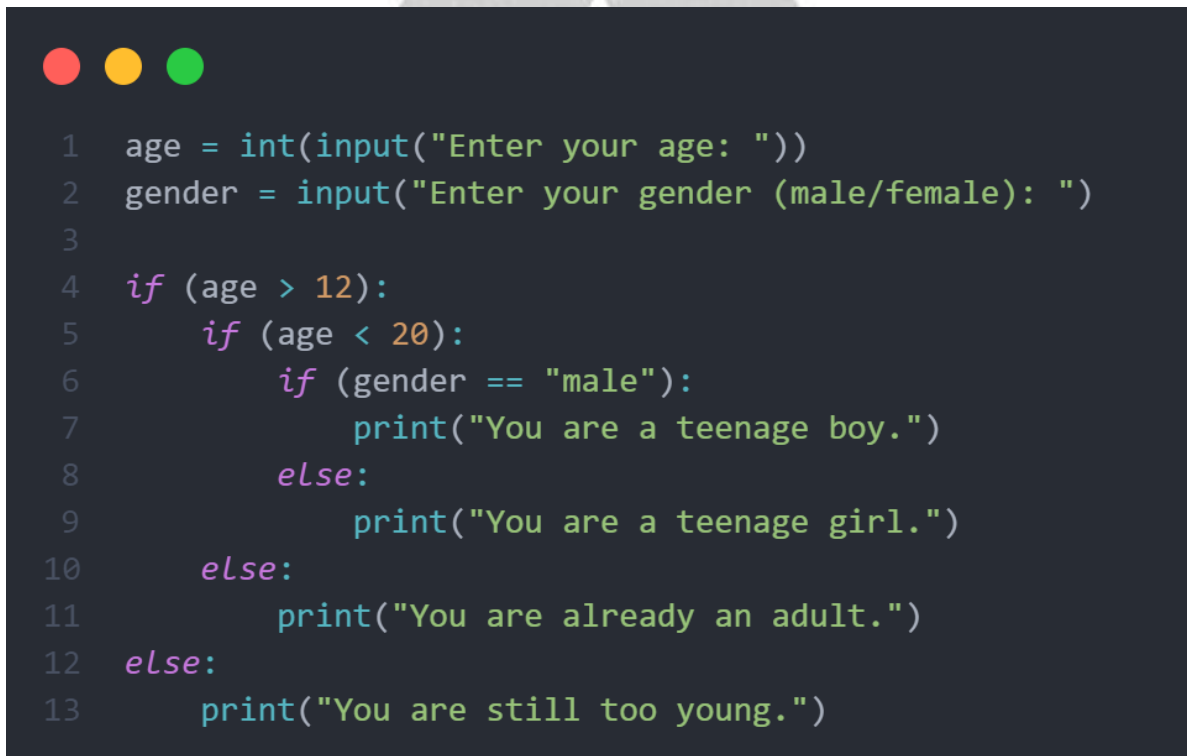
```

Enter a number: 6496984
6496984 is even

```

**دستور `if` تو در تو:** می توان از دستور `if` تو در تو در پایتون استفاده کرد. یک دستور ساده `if` در داخل دستور `if` دیگر.

با استفاده از یک مثال `if` تو در تو را توضیح میدهیم:



```

1 age = int(input("Enter your age: "))
2 gender = input("Enter your gender (male/female): ")
3
4 if (age > 12):
5     if (age < 20):
6         if (gender == "male"):
7             print("You are a teenage boy.")
8         else:
9             print("You are a teenage girl.")
10    else:
11        print("You are already an adult.")
12 else:
13    print("You are still too young.")

```

ابتدا برنامه از شما درباره سئال می کند (خط ۱). در خط ۲ درباره جنستان از شما سئال می کند. سپس به اولین دستور `if` می رسد (خط ۴). در این قسمت اگر سن شما بیشتر از ۱۲ سال باشد برنامه وارد بدنه دستور `if` می شود در غیر اینصورت وارد بلوک `else` (خط ۱۲) مربوط به همین دستور `if` می شود.

تهیه کننده: علیزاده

آموزش برنامه سازی با پایتون به زبان ساده

حال فرض کنیم که سن شما بیشتر از ۱۲ سال است و شما وارد بدنه اولین if شده‌اید. در بدنه اولین if دو دستور if دیگر را مشاهده می‌کنید. اگر سن کمتر ۲۰ باشد شما وارد بدنه if دوم می‌شوید و اگر نباشد به قسمت else متناظر با آن می‌روید (خط ۱۰). دوباره فرض می‌کنیم که سن شما کمتر از ۲۰ باشد، در اینصورت وارد بدنه if دوم شده و با یک if دیگر مواجه می‌شوید (خط ۶). در اینجا جنسیت شما مورد بررسی قرار می‌گیرد که اگر برابر "male" باشد، کدهای داخل بدنه سومین if اجرا می‌شود در غیر اینصورت قسمت else مربوط به این if اجرا می‌شود (خط ۸). پیشنهاد می‌شود که از if تو در تو در برنامه کمتر استفاده کنید چون خوانایی برنامه را پایین می‌آورد.

## تکرار (حلقه ها):

ساختار های تکرار به برنامه نویس اجازه میدهند که یک یا چند دستور کد را تا زمانی که یک شرط برقرار است تکرار کند. بدون ساختار های تکرار، برنامه نویسی عملی خسته کننده طلقی میشود چون برنامه نویس باید به همان تعداد تکرار، کد هارا تکرار کند. مثلا برای ۱۰ بار تکرار جمله Hello World! مجبور میشوید ۱۰ بار دستور آن را تایپ کنید:

```
1 print("Hello World!")
2 print("Hello World!")
3 print("Hello World!")
4 print("Hello World!")
5 print("Hello World!")
6 print("Hello World!")
7 print("Hello World!")
8 print("Hello World!")
9 print("Hello World!")
10 print("Hello World!")
```

البته برنامه نویس میتواند با کپی پیست کردن یک دستور، این کد را راحت تر بنویسد اما این کار کیفیت کلی کدنویسی را پایین می آورد. راهکاری که برای نوشتن کد های بالا استفاده میشود حلقه ها است.

در پایتون ۲ نوع حلقه داریم: ۱ - معین (for) ۲ - نامعین (while)

**حلقه while:** ابتدایی ترین ساختارهای تکرار در پایتون حلقه های While

هستند. ابتدا یک شرط را مورد بررسی قرار می‌دهد و تا زمانی که شرط برقرار باشد کدهای درون بلوک اجرا می‌شوند. ساختار حلقه های while به صورت مقابل است:

```
1 while(condition):
2     #code to loop;
```

**سازوکار حلقه های while:** ابتدا یک شرط را که نتیجه آن یک مقدار بولی (منطقی) است می‌نویسیم اگر نتیجه درست یا true باشد سپس کدهای داخل بلوک While اجرا می‌شوند. اگر شرط غلط یا false باشد وقتی که برنامه به حلقه While برسد هیچکدام از کدها را اجرا نمی‌کند. برای متوقف شدن حلقه باید مقادیر داخل حلقه While اصلاح شوند.

```
1 counter = 1;
2
3 while (counter <= 10):
4     print("Hello World!");
5     counter = counter + 1;
```

مثال: برنامه ای بنویسید که Hello World! را با استفاده از حلقه while، ۱۰ بار چاپ کند: ابتدا در خط ۱ یک متغیر تعریف و از آن به عنوان شمارنده حلقه استفاده شده است. سپس به آن مقدار ۱ را اختصاص می‌دهیم چون اگر مقدار نداشته باشد نمی‌توان در شرط از آن استفاده کرد.

تهیه کننده: علیزاده

آموزش برنامه سازی با پایتون به زبان ساده

```
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!  
Hello World!
```

در خط ۳ حلقه **while** را وارد می کنیم. در حلقه **while** ابتدا مقدار اولیه شمارنده با ۱۰ مقایسه می شود که آیا از ۱۰ کمتر است یا با آن برابر است. نتیجه هر بار مقایسه ورود به بدنه حلقه **While** و چاپ پیام است. همانطور که مشاهده می کنید بعد از هر بار مقایسه مقدار شمارنده یک واحد اضافه می شود (خط ۵). حلقه تا زمانی تکرار می شود که مقدار شمارنده از ۱۰ کمتر باشد. خروجی کد صفحه قبل بدین صورت است: اگر مقدار شمارنده ۱ بماند و آن را افزایش ندهیم و یا مقدار شرط هرگز **false** نشود یک حلقه بینهایت به وجود می آید. به این نکته توجه کنید که در شرط بالا به جای علامت **>** از **>=** استفاده شده است. اگر از

علامت **>** استفاده می کردیم کد ما ۹ بار تکرار می شد چون مقدار اولیه ۱ است و هنگامی که شرط به ۱۰ برسد **false** می شود چون  $10 > 10$  نیست.

```
1 while(True):  
2     #code to loop
```

اگر می خواهید یک حلقه بی نهایت ایجاد کنید که هیچگاه متوقف نشود باید یک شرط ایجاد کنید که همواره درست (**true**) باشد:

```
1 for iterator_var in sequence:  
2     #code to repeat;
```

**حلقه for**: یکی دیگر از ساختارهای تکرار مربوط به حلقه های **for** است. این حلقه ها عملی مشابه با حلقه **while** انجام می دهند. ساختار حلقه **for** به صورت مقابل است:

**iterator\_var** یک متغیر موقتی، **in** کلمه کلیدی و **sequence** هم یک سری مانند **list**، **tuple** و ... می باشد. می توان حلقه **for** را اینگونه ترجمه کرد، که به ازای یا به تعداد آیتم های موجود در سری، فلان کارها یا کدها را تکرار کن.

مثال: برنامه ای بنویسید که با استفاده از حلقه **for** لیستی از اعداد ۱ تا ۱۰ را به صورت طولی و با پیشوند **number** چاپ کند:

```
1 for i in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:  
2     print("Number", i);
```

برنامه بالا اعداد ۱ تا ۱۰ را با استفاده از حلقه **for** می شمارد. ابتدا یک متغیر موقتی (**i**)، سپس کلمه کلیدی **in** و در آخر یک سری از اعداد که در اینجا یک **list** می باشد، تعریف می کنیم. کد اجرا می شود. هر بار که حلقه اجرا می شود، ابتدا یکی از آیتم های **list** در متغیر **i** قرار گرفته و در خط بعد چاپ می شود. این کار تا چاپ آخرین آیتم ادامه می یابد. به جای **list** در کد بالا می توانید از **tuple** و **dictionary** هم استفاده کنید.

```
Number 1  
Number 2  
Number 3  
Number 4  
Number 5  
Number 6  
Number 7  
Number 8  
Number 9  
Number 10
```

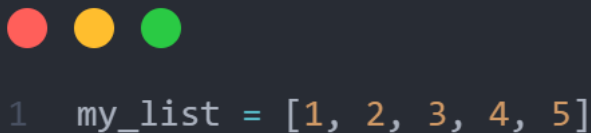
خروجی کد بالا به صورت مقابل است:



آرایه ها ( تک بعدی ):

در پایتون، آرایه های تک بعدی به عنوان لیست ها در نظر گرفته می شوند. لیست ها در پایتون مجموعه ای از اشیاء هستند که هر یک دارای یک شناسه (index) منحصر به فرد و یک مقدار است. شناسه هر عنصر در لیست از صفر شروع می شود و به صورت ترتیبی افزایش می یابد.

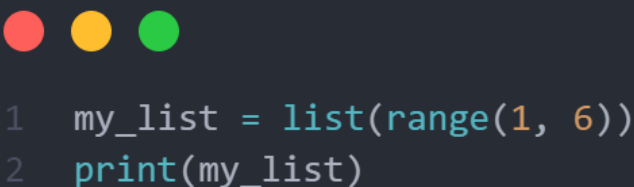
برای مثال، یک آرایه تک بعدی که شامل اعداد ۱ تا ۵ است به صورت زیر می تواند ایجاد شود:



```
1 my_list = [1, 2, 3, 4, 5]
```

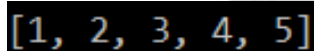
در این مثال، شناسه هر عضو در لیست از صفر شروع می شود. بنابراین، `my_list[0]` برابر با ۱، `my_list[1]` برابر با ۲ و به همین ترتیب ادامه می یابد.

همچنین، می توان با استفاده از حلقه ها و توابع مختلف، از جمله توابع `list()` و `range()`، آرایه های تک بعدی را ایجاد کرد و به آن ها دسترسی داشت. به عنوان مثال:



```
1 my_list = list(range(1, 6))
2 print(my_list)
```

در این مثال، با استفاده از تابع `range()`، یک آرایه تک بعدی شامل اعداد ۱ تا ۵ ایجاد شده و سپس با استفاده از تابع `list()`، آن آرایه به لیست تبدیل شده است. سپس، لیست حاوی عناصر [۱, ۲, ۳, ۴, ۵] چاپ می شود:



```
[1, 2, 3, 4, 5]
```