پایتون

به زبان ساده

بی شک این اثر، خالی از اشکال نیست و از شما خوانندگان عزیز می‌خواهم که با نظرات و پیشنهادات خود بنده را در تکمیل و رفع نواقص آن از طریق پست الکترونیکی younes.ebrahimi.1391@gmail.com یاری بفرمایید.

برای دریافت فایل‌ها و آپدیت‌های جدید این کتاب به سایت **www.w3-farsi.com** مراجعه فرمایید.

راه‌های ارتباط با نویسنده

وب سایت: www.w3-farsi.com

لینک تلگرام: https://telegram.me/ebrahimi_younes

ID تلگرام: @ebrahimi_younes

پست الکترونیکی: younes.ebrahimi.1391@gmail.com

پایتون چیست

پایتون (Python) یک زبان برنامه‌نویسی همه منظوره، شیءگرا و متن باز است که توسط خودو فان راسام (Guido van Rossum) در سال ۱۹۹۱ در کشور هلند طراحی شد. این زبان از زبان‌های برنامه‌نویسی مفسر بوده و به صورت کامل یک زبان شیءگرا است که به زبانهای تفسیری Perl و Ruby شباهت دارد و از مدیریت خودکار حافظه استفاده می‌کند.

پایتون، کد باز (Open Source) است، زبانی که گوگل و یا یاهو از آن به عنوان یکی از اصلی ترین ابزارهای توسعه استفاده می کنند. برنامه های پایتون مثل PHP قابل اجرا روی اغلب سیستم عامل هاست. پایتون، دستور زبانی شبیه گفتار ساده ی انگلیسی دارد و با دارا بودن ۳۳ کلمه کلیدی جزء ساده ترین زبان ها است.

سادگی و خوانایی از ویژگی‌های بارز زبان برنامه‌نویسی پایتون است، آنچنان ساده که حتی کودکان نیز قادر به آموختن آن هستند و قدرت در کنار این سادگی و خوانایی، معجزه پایتون می‌باشد. از نگاه هر برنامه‌نویسی، برنامه‌های پایتون مجموعه‌ای از کدهای زیبا هستند، بدون هیچ آشفتگی و پیچیدگی. جالب است بدانید مایکروسافت نیز این زبان را با نام IronPython در تکنولوژی .Net خود گنجانده است.

هم اکنون پایتون در شرکت ها و سازمان های بزرگی مثل ناسا و گوگل و یاهو و ... به صورت گسترده مورد استفاده قرار می گیرد. تا کنون نسخه های مختلفی از این زبان ارائه شده است که لیست آنها را در جدول زیر میشاهده می کنید:

نسخه	تاریخ پیدایش
Python 1.0	January 1994
Python 1.2	April 10, 1995
Python 1.3	October 12, 1995
Python 1.4	October 25, 1996
Python 1.5	December 31, 1997
Python 1.6	September 5, 2000
Python 2.0	October 16, 2000
Python 2.1	April 17, 2001
Python 2.2	December 21, 2001
Python 2.3	July 29, 2003
Python 2.4	November 30, 2004
Python 2.5	September 19, 2006
Python 2.6	October 1, 2008
Python 2.7	July 3, 2010
Python 3.0	December 3, 2008
Python 3.1	June 27, 2009
Python 3.2	February 20, 2011
Python 3.3	September 29, 2012
Python 3.4	March 16, 2014

Python 3.5	September 13, 2015
Python 3.6	December 23, 2016

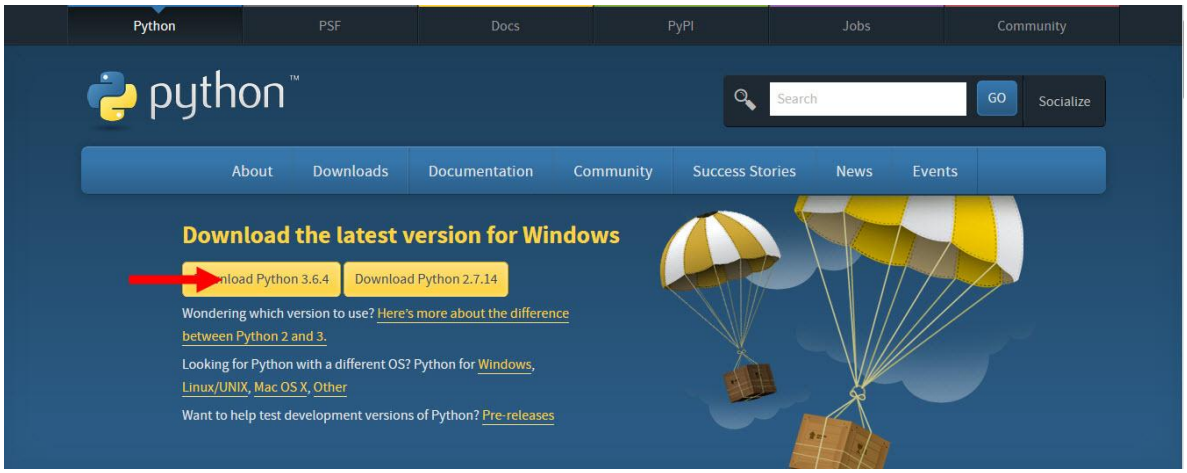
حال که با پایتون به طور مختصر آشنا شدید، در درس های بعد در مورد این زبان برنامه نویسی بیشتر توضیح می دهیم.

دانلود و نصب Python 3.6

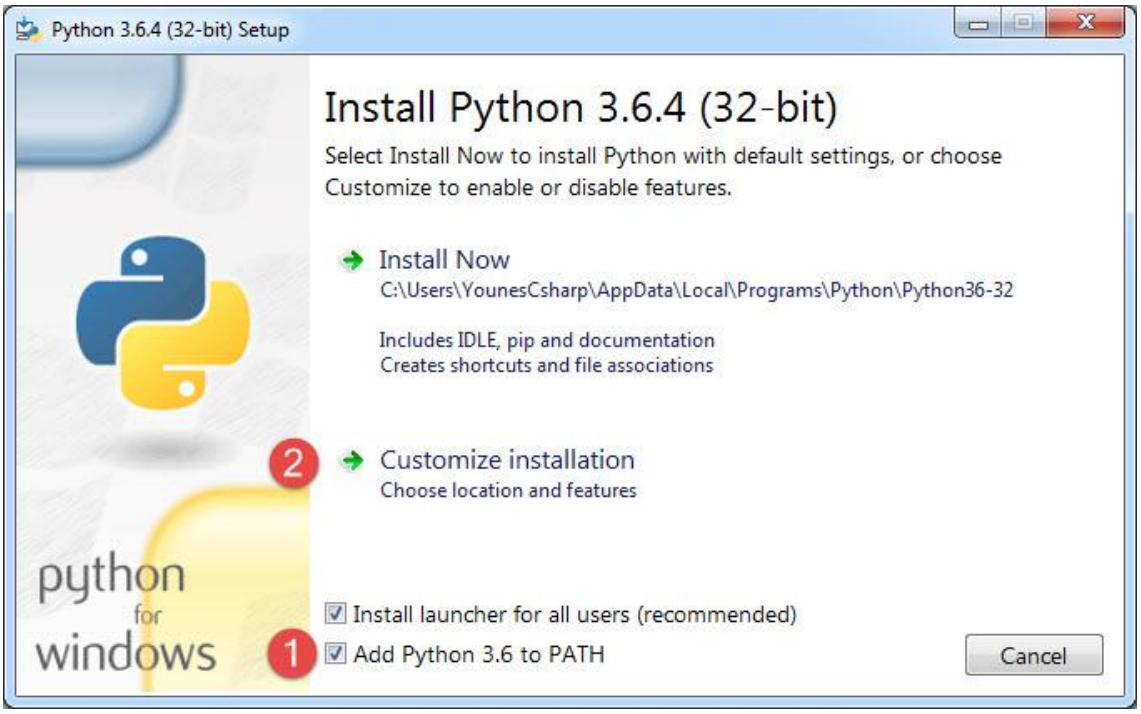
برای برنامه نویسی به زبان های مختلف محیط های توسعه ی یکپارچه یا IDE های مختلفی وجود دارند که به برنامه نویسان در نوشتن و ویرایش کدها، پیدا کردن خطاها، نمایش خروجی، و برخی موارد دیگر کمک می کنند. برای اجرای کدهای Python محیط های مختلفی وجود دارد که ساده ترین آنها IDEL می باشد. برای دانلود این محیط کدنویسی، بر روی لینک زیر کلیک کنید:

https://www.python.org/downloads

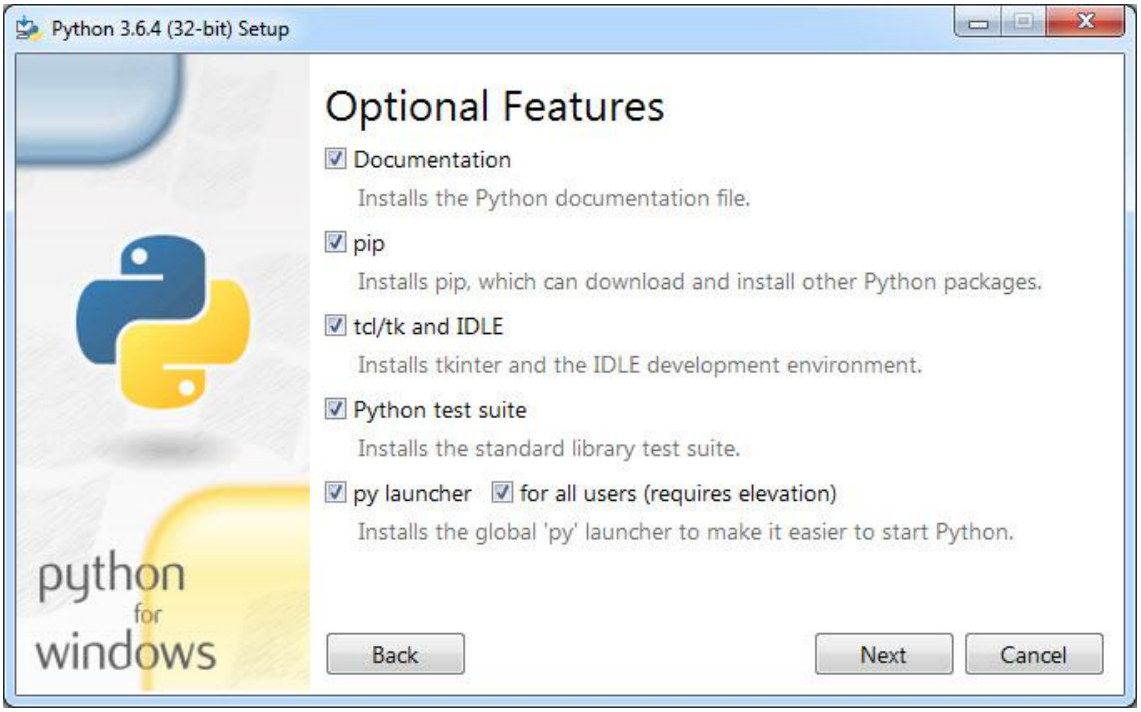
با کلیک بر روی لینک بالا، صفحه ای به صورت زیر باز می شود، که در این صفحه همانطور که در شکل زیر مشاهده می کنید، بر روی دکمه ای که با فلش نشان داده شده کلیک کرده، تا آخرین نسخه IDEL دانلود شود:



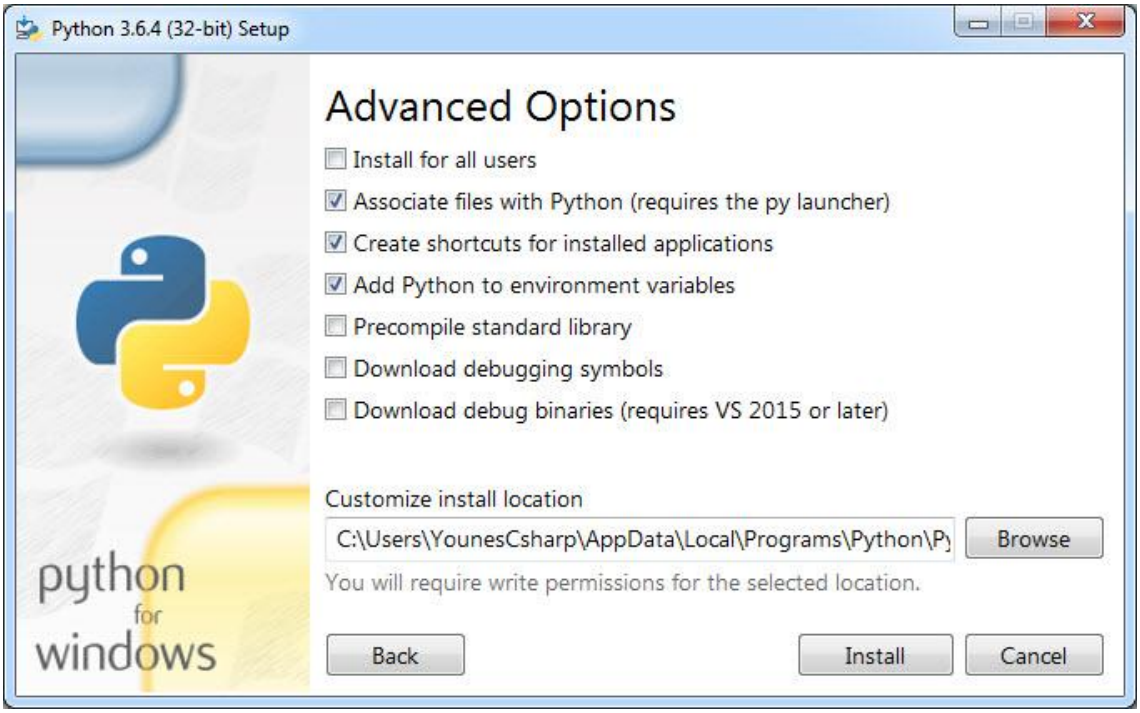
بعد از دانلود فایل مورد نظر به محل ذخیره آن رفته و بر روی فایل دو بار کلیک کنید. سپس در صفحه ای که به صورت زیر نمایش داده می شود، تیک مورد نظر را زده و سپس بر روی گزینه Customize installiation کلیک کنید:



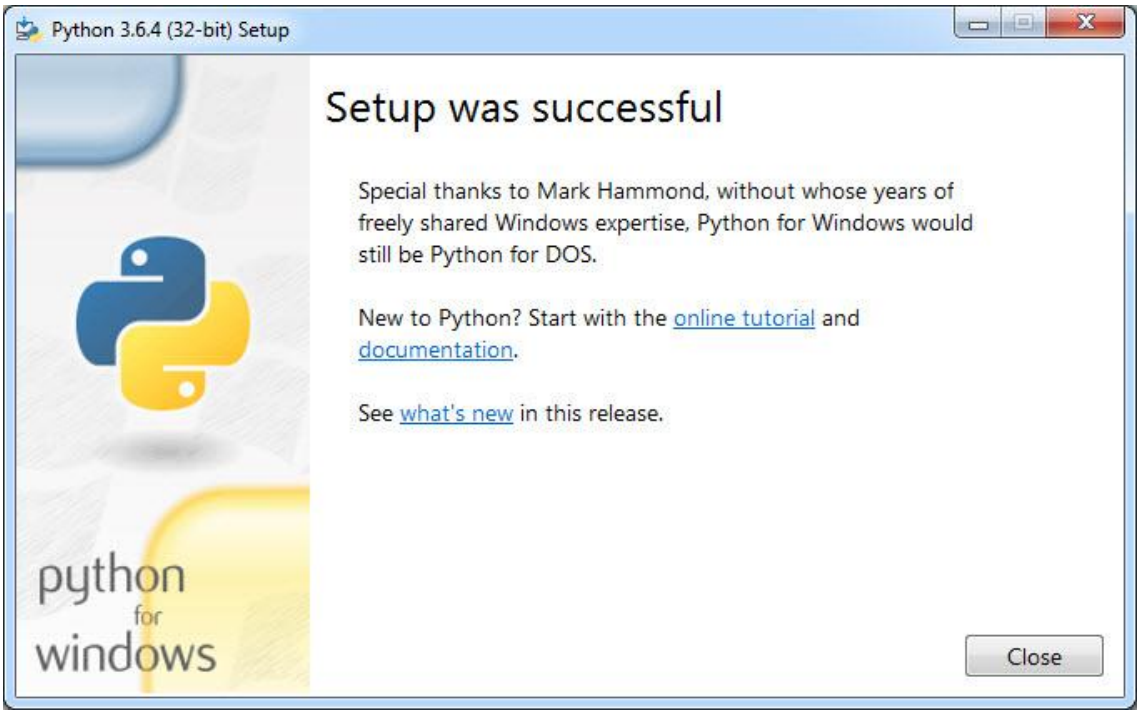
بعد از کلیک بر روی گزینه مذکور صفحه ای به صورت زیر نمایش داده می شود. در این صفحه بر روی دکمه `Next` کلیک کنید:



در صفحه بعد بر روی دکمه `Install` کلیک کرده و منتظر بمانید تا برنامه نصب شود:

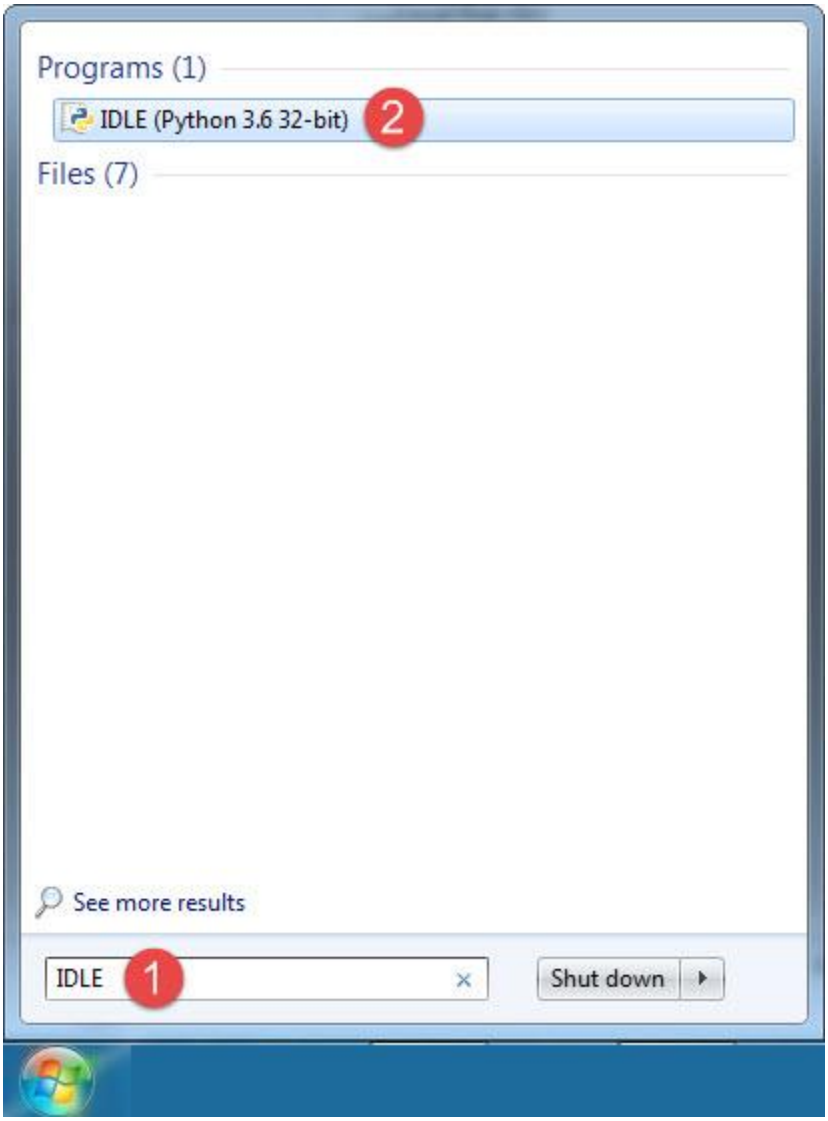


در آخر و بعد از نصب کامل برنامه پیغامی مبنی بر موفقیت آمیز بودن، نصب برنامه به شما نمایش داده می شود و شما می توانید دکمه `Close` را بزنید:

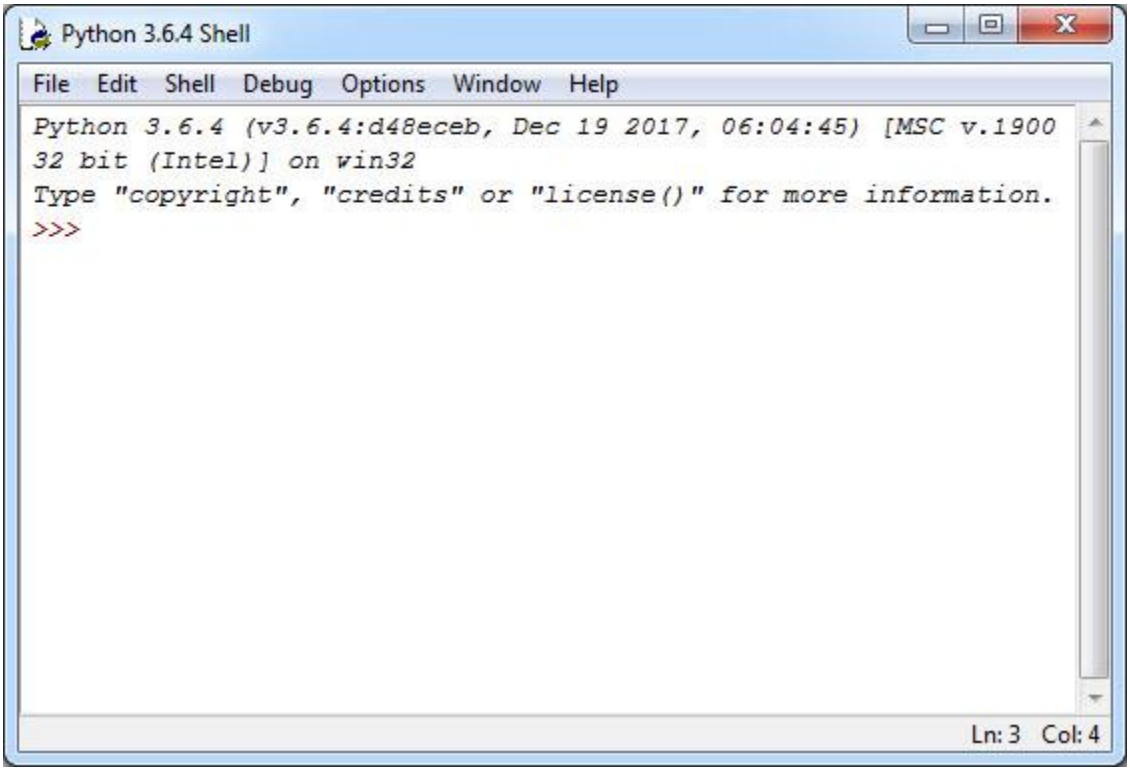


ساخت یک برنامه ساده

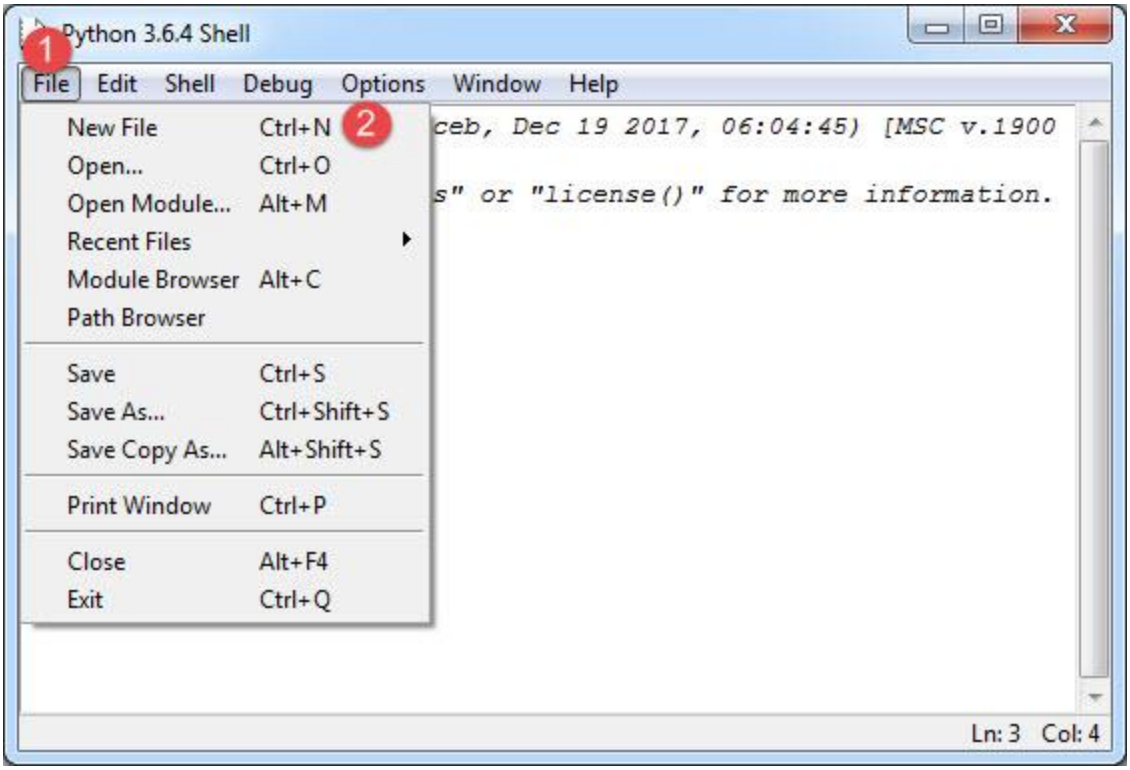
اجازه بدهید یک برنامه بسیار ساده به زبان پایتون بنویسیم .این برنامه یک پیغام را نمایش می‌دهد. از منوی Start محیط برنامه نویسی IDEL را به صورت زیر اجرا کنید:



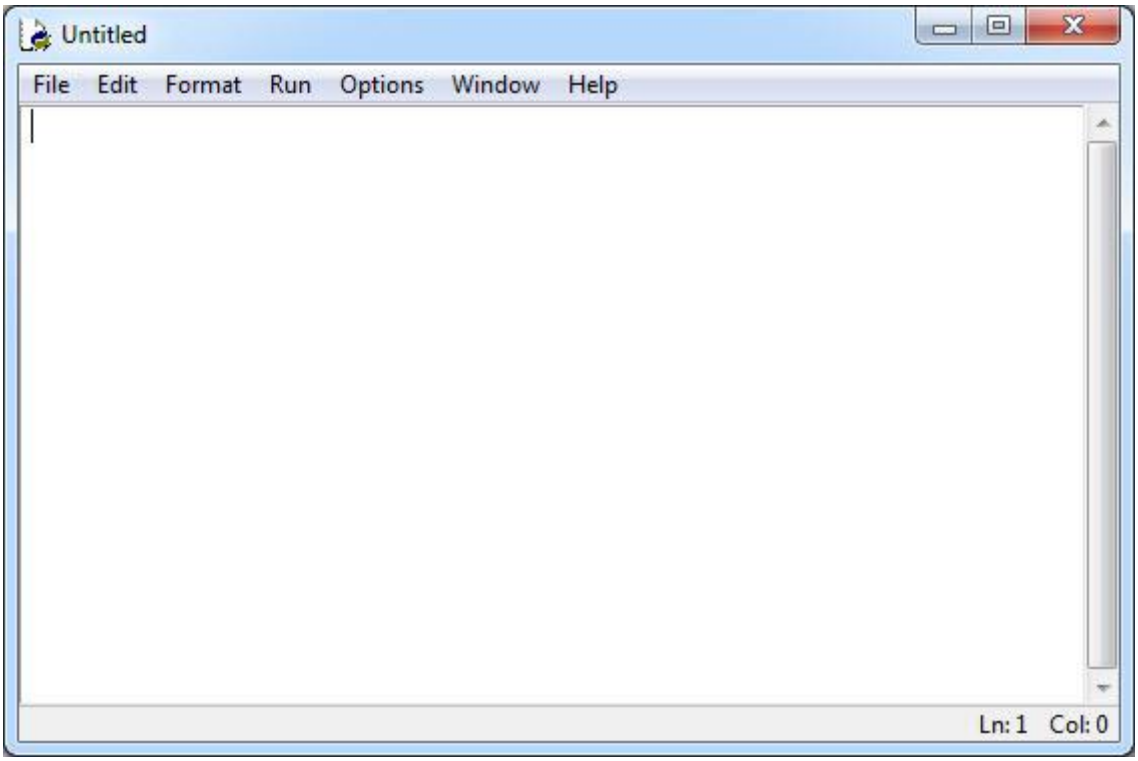
با کلیک بر روی IDEL صفحه ای به صورت زیر نمایش داده می شود:



در صفحه باز شده به صورت زیر بر روی منوی `File` و سپس گزینه `New File` کلیک کنید:

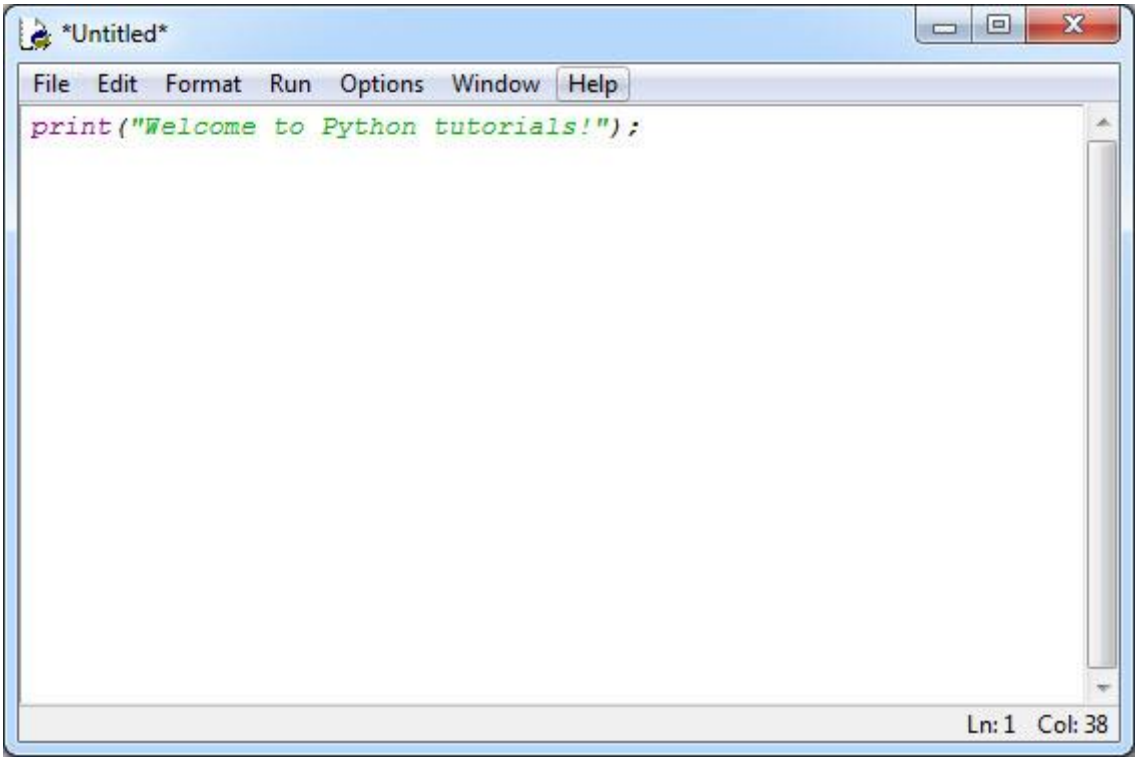


با کلیک بر روی گزینه `New File` صفحه ای به صورت زیر نمایش داده می شود که شما می توانید کدهای خود را در داخل آن بنویسید:

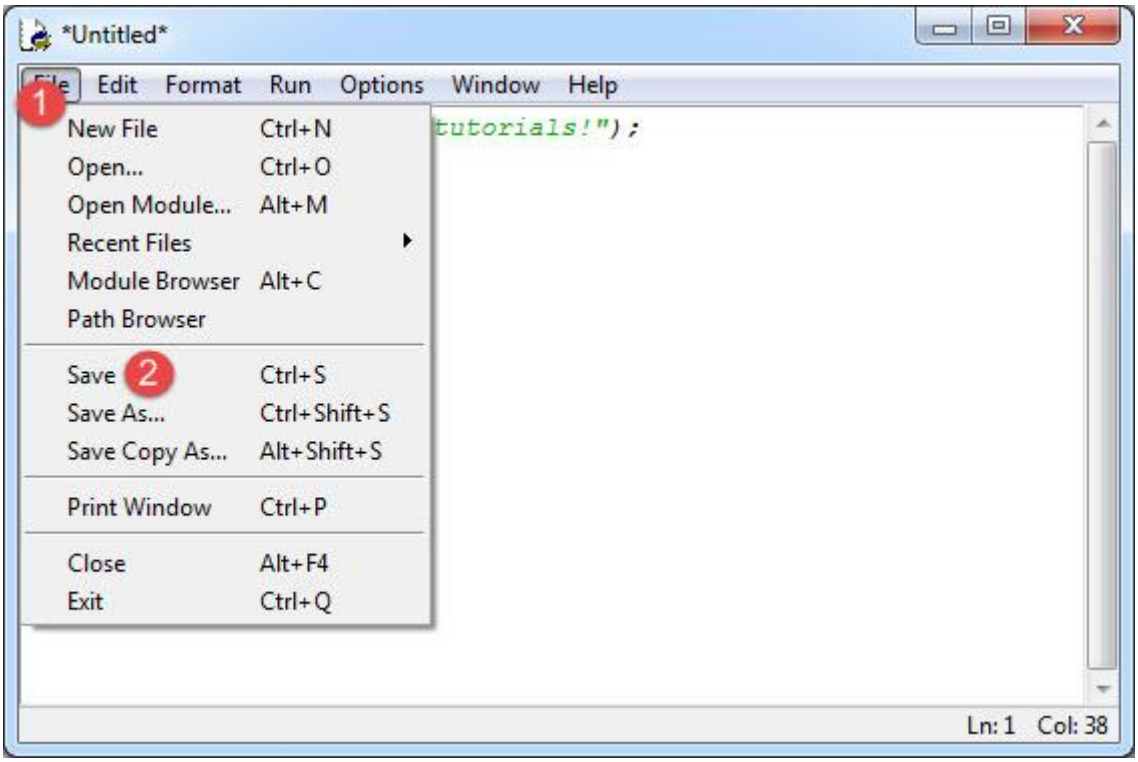


در پنجره بالا کدهای زیر را بنویسید:

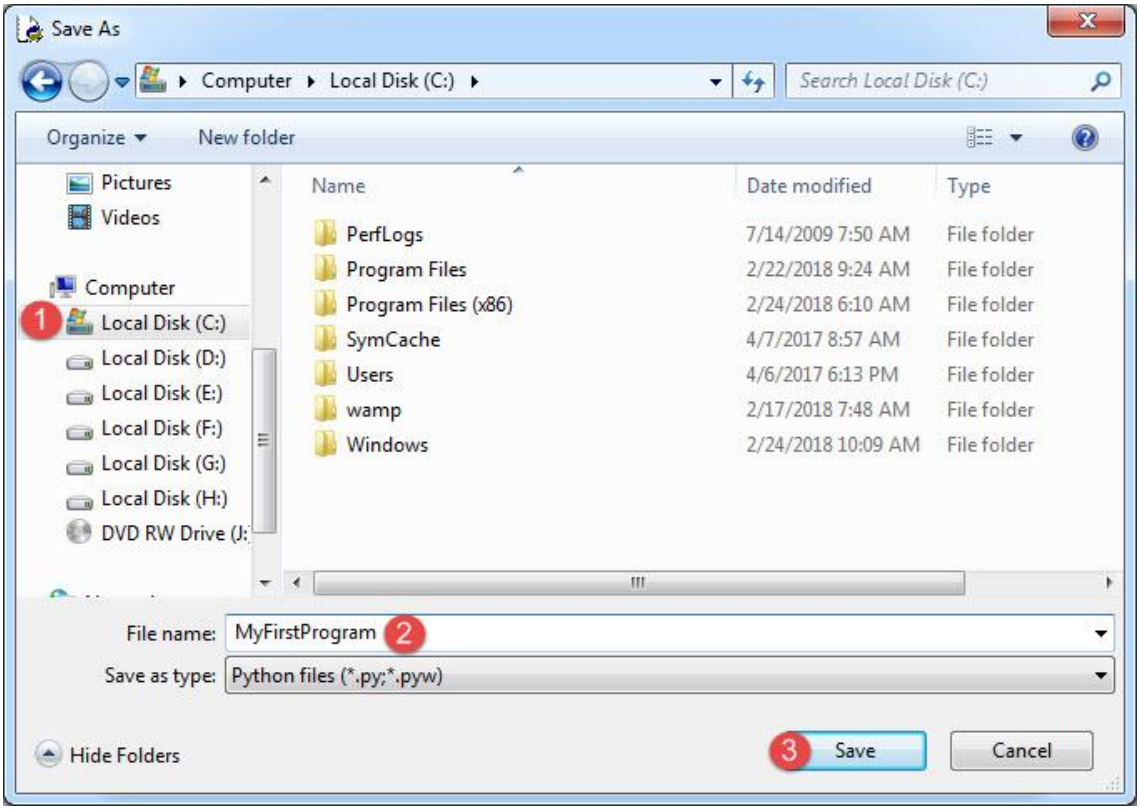
```
print("Welcome to Python Tutorials!");
```



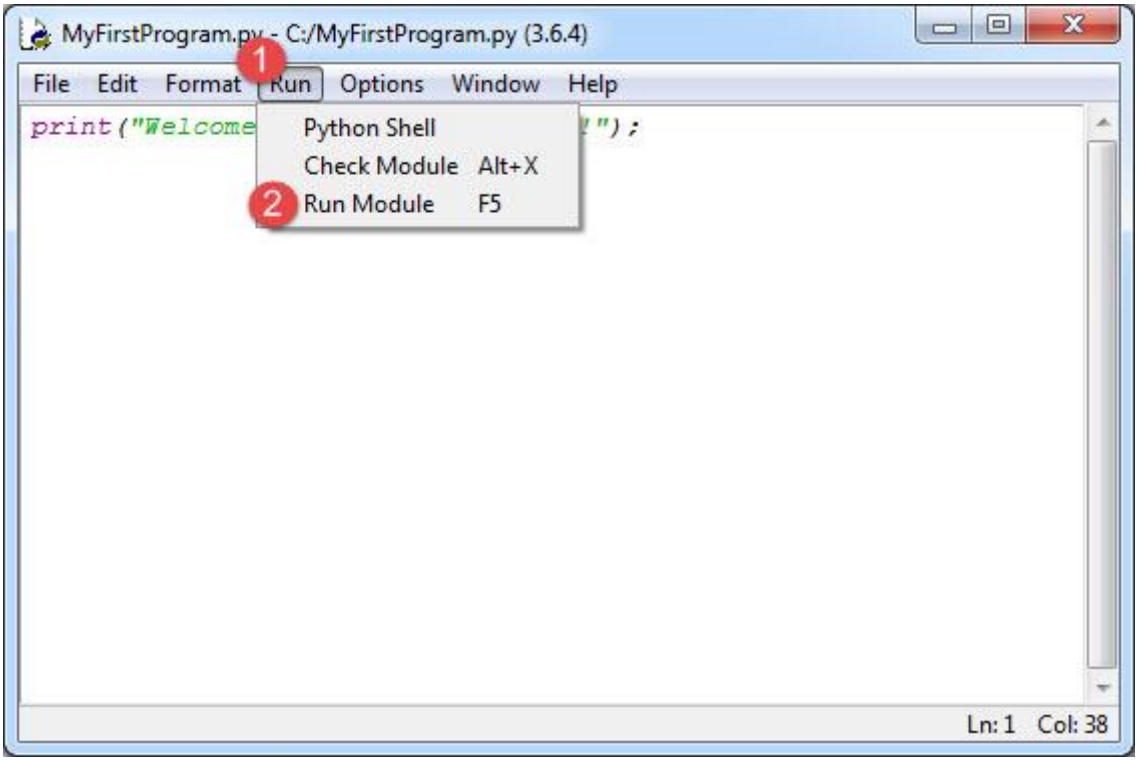
مانند شکل زیر از منوی `File` گزینه `Save` را بزنید:



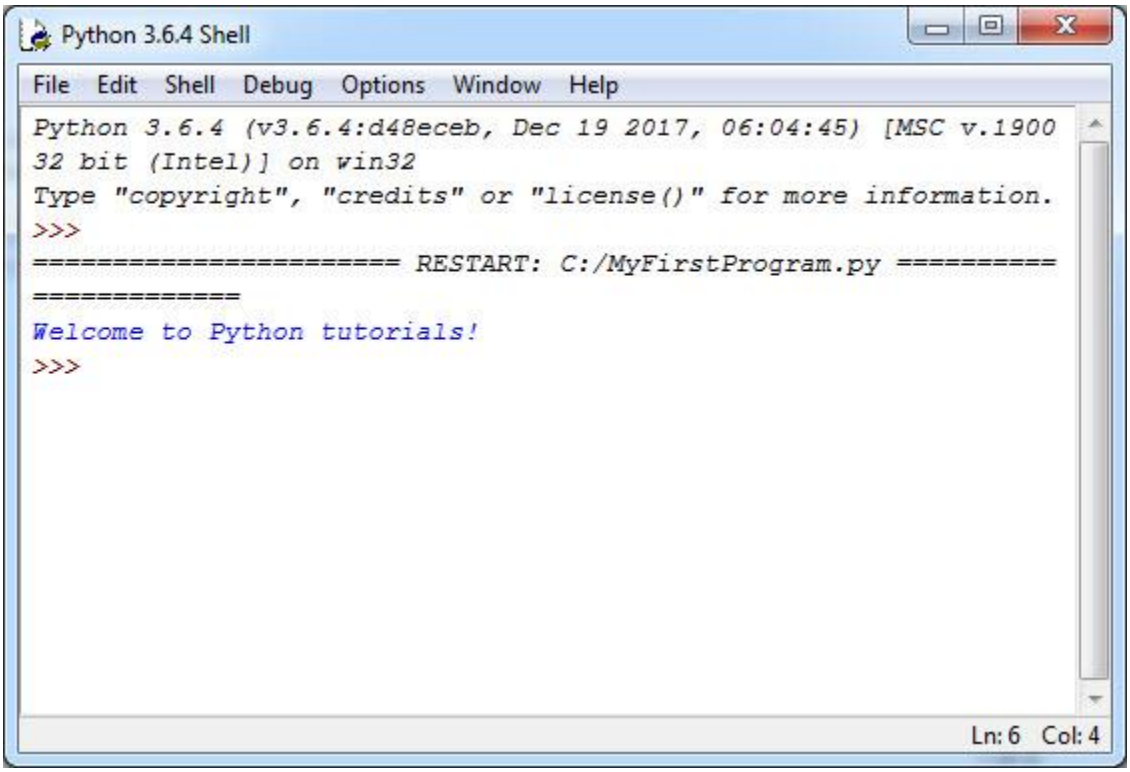
سپس یک مسیر برای ذخیره فایل انتخاب کنید. ما در شکل زیر فایل را در درایو `C` ذخیره کرده ایم:



بعد از ذخیره فایل به محیط کدنویسی بر گشته و از منوی `Run` گزینه `Run Module` و یا دکمه `F5` را بزنید:



مشاهده می کنید که برنامه اجرا شده و پیغام Welcome to Python tutorials! چاپ می شود:



مثال بالا ساده‌ترین برنامه ای است که شما می‌توانید در Python بنویسید. هدف در مثال بالا نمایش یک پیغام در صفحه نمایش است. هر زبان برنامه نویسی دارای قواعدی برای کدنویسی است. پایتون دارای متدهای از پیش تعریف شده ای است که هر کدام برای مقاصد خاصی به کار می روند. هر چند که در آینده در مورد متدها بیشتر توضیح می دهیم ولی در همین حد به توضیح متد بسنده می کنیم که متدها مجموعه ای از کدها هستند که دارای یک نام بوده و در جلوی نام آنها علامت () قرار می گیرد. یکی از این متدها، متد print() است. از متد print() برای چاپ یک رشته استفاده می‌شود. یک رشته گروهی از کاراکترها است، که به وسیله دابل کوتیشن (") محصور شده است. مانند : "Welcome to Python Tutorials!". یک کاراکتر می‌تواند یک حرف، عدد، علامت یا ... باشد. در کل مثال بالا نحوه استفاده از متد print() است. توضیحات بیشتر در درس‌های آینده آمده است. پایتون فضای خالی بالا را نادیده می‌گیرد و از کد زیر اشکال نمی‌گیرد:

```
print(
    "Welcome to Python Tutorials!");
```

همیشه به یاد داشته باشید که Python به بزرگی و کوچکی حروف حساس است. یعنی به طور مثال MAN و man در Python با هم فرق دارند. رشته‌ها و توضیحات از این قاعده مستثنی هستند که در درس‌های آینده توضیح خواهیم داد. مثلاً کدهای زیر با خطا مواجه می‌شوند و اجرا نمی‌شوند:

```
Print("Welcome to Python Tutorials!");
PRINT("Welcome to Python Tutorials!");
PrinT("Welcome to Python Tutorials!");
```

تغییر در بزرگی و کوچکی حروف از اجرای کدها جلوگیری می‌کند. اما کد زیر کاملاً بدون خطا است:

```
print("Welcome to Python tutorials!");
```

نکاتی در مورد کدنویسی در پایتون

در زبان هایی مثل جاوا و سی شارپ، از علامت آکولاد ({}) برای ایجاد یک بلاک کد

```
Block
{
    statement;
}
```

ولی در زبان پایتون از ترکیب علامت دو نقطه (:) و تو رفتگی برای اینکار استفاده می شود:

```
Block:
    statement;
```

ما با دونقطه به پایتون می‌گوییم که قصد داریم یک بلوک کد را آغاز کنیم و با تو رفتگی ابتدای خطوط دستورات آن بلوک را تعریف می‌کنیم. برای تورفتگی می‌توانیم از ۳ یا ۴ یا ۱۰ فضای خالی استفاده کنیم. میزان این فضای خالی تا زمانی که در تمام کد رعایت شود، اهمیتی ندارد. در کد زیر به اهمیت تو رفتگی ها پی می برید:

```
1   Block1:
2       statement;
3       statement;
4   Block2:
5       statement;
6       Block3:
7           statement;
8           statement;
9           Block4:
10              statement;
11      statement;
12  statement;
```

در کد بالا، بلاک اول (Block1) از خط ۱ تا ۳ را شامل می شود. به این نکته توجه کنید که خطوط بعد از علامت : حتما باید دارای تو رفتگی باشند. بلاک دوم (Block2) از خط ۴ شروع و به خط ۱۱ ختم می شود. نکته ای که باید در اینجا دوباره به آن اشاره کنیم این است که دستور یا بلاک هایی که دارای فاصله های برابر از سمت چپ هست جزو یک بلاک می باشند. مثلا در کد بالا خطوط 2و ۳ جز Block1 هستند چون تو رفتگی آنها از سمت چپ برابر است و اگر مثلا فاصله های خط ۳ از سمت چپ را حذف کنیم دیگر جز بلاک محسوب نمی شود. یک بلاک را می توان زیر مجموعه بلاک دیگر کرد. مثلا در خط ۶، Block3 را زیر مجموعه Block2 و در خط ۹، Block4

را زیر مجموعه Block3 کرده ایم. در نهایت خط ۱۲ جز هیچکدام از بلاک ها نیست و مستقل اجرا می شود. گاهی اوقات و هنگام کدنویسی، لازم است که رشته های طولانی را در چند خط بنویسید. برای اینکار در پایتون می توان از علامت \ به صورت زیر استفاده کنید:

```
print("Welcome \
to \
Python \

Tutorials!");
```

توضیحات

وقتی که کدی تایپ می کنید شاید بخواهید که متنی جهت یادآوری وظیفه آن کد به آن اضافه کنید. در Python (و بیشتر زبانهای برنامه نویسی) می توان این کار را با استفاده از توضیحات انجام داد. توضیحات متونی هستند که توسط مفسر نادیده گرفته می شوند و به عنوان بخشی از کد محسوب نمی شوند.

هدف اصلی از ایجاد توضیحات، بالا بردن خوانایی و تشخیص نقش کدهای نوشته شده توسط شما، برای دیگران است. فرض کنید که می خواهید در مورد یک کد خاص، توضیح بدهید، می توانید توضیحات را در بالای کد یا کنار آن بنویسید. از توضیحات برای مستند سازی برنامه هم استفاده می شود. در برنامه زیر نقش توضیحات نشان داده شده است:

```
#This line will print the message hello world
print("Hello World!");
```

در کد بالا، خط اول کد بالا یک توضیح درباره خط دوم است که به کاربر اعلام می کند که وظیفه خط دوم چیست ؟ با اجرای کد بالا فقط جمله Hello World چاپ شده و خط اول در خروجی نمایش داده نمی شود چون مفسر توضیحات را نادیده می گیرد. همانطور که مشاهده می کنید برای درج توضیحات در پایتون از علامت # استفاده می شود. برای توضیحات طولانی هم باید در ابتدای هر خط از توضیح این علامت درج شود:

```
#This line will print
#the message hello world
print("Hello World!");
```

کاراکترهای کنترلی

کاراکترهای کنترلی، کاراکترهای ترکیبی هستند که با یک بک اسلش (\) شروع می‌شوند و به دنبال آنها یک حرف یا عدد می‌آید و یک رشته را با فرمت خاص نمایش می‌دهند. برای مثال برای ایجاد یک خط جدید و قرار دادن رشته در آن می‌توان از کاراکتر کنترلی \n استفاده کرد:

```
print("Hello\nWorld!");
```

```
Hello
World
```

مشاهده کردید که مفسر بعد از مواجهه با کاراکتر کنترلی \n نشانگر ماوس را به خط بعد برده و بقیه رشته را در خط بعد نمایش می‌دهد.

جدول زیر لیست کاراکترهای کنترلی و کارکرد آنها را نشان می‌دهد:

عملکرد	کاراکتر کنترلی	عملکرد	کاراکتر کنترلی
Form Feed	\f	چاپ کوتیشن	\’
خط جدید	\n	چاپ دابل کوتیشن	\”
سر سطر رفتن	\r	چاپ بک اسلش	\\
حرکت به صورت افقی	\t	چاپ فضای خالی	\0
حرکت به صورت عمودی	\v	صدای بیپ	\a
چاپ کاراکتر یونیکد	\u	حرکت به عقب	\b

ما برای استفاده از کاراکترهای کنترلی، از بک اسلش (\) استفاده می‌کنیم. از آنجاییکه \ معنای خاصی به رشته‌ها می‌دهد برای چاپ بک اسلش (\) باید از (\\) استفاده کنیم:

```
print("We can print a \\ by using the \\\\ escape sequence.");
```

```
We can print a \ by using the \\ escape sequence.
```

یکی از موارد استفاده از \\، نشان دادن مسیر یک فایل در ویندوز است:

```
print("C:\\Program Files\\Some Directory\\SomeFile.txt");
```

```
C:\Program Files\Some Directory\SomeFile.txt
```

از آنجاییکه از دابل کوتیشن (“) برای نشان دادن رشته‌ها استفاده می‌کنیم برای چاپ آن از \” استفاده می‌کنیم:

```
print("I said, \"Motivate yourself!\".");
```

```
I said, "Motivate yourself!".
```

همچنین برای چاپ کوتیشن (‘) از \’ استفاده می‌کنیم:

```
print("The programmer\'s heaven.");
```

```
The programmer's heaven.
```

برای ایجاد فاصله بین حروف یا کلمات از \t استفاده می‌شود:

```
print("Left\tRight");
```

```
Left  Right
```

برای چاپ کاراکترهای یونیکد می‌توان از \u استفاده کرد. برای استفاده از \u، مقدار در مبنای ۱۶ کاراکتر را درست بعد از علامت \u قرار می‌دهیم. برای مثال اگر بخواهیم علامت کپی رایت (©) را چاپ کنیم، باید بعد از علامت \u مقدار 00A9 را قرار دهیم مانند:

```
print("\u00A9");
```

```
©
```

برای مشاهده لیست مقادیر مبنای ۱۶ برای کاراکترهای یونیکد به لینک زیر مراجعه نمایید:

http://www.ascii.cl/htmlcodes.htm

اگر مفسر به یک کاراکتر کنترلی غیر مجاز برخورد کند، برنامه پیغام خطا می‌دهد. بیشترین خطا زمانی اتفاق می افتد که برنامه نویس برای چاپ اسلش (\) از \\ استفاده می‌کند.

متغیر

متغیر مکانی از حافظه است که شما می‌توانید مقادیری را در آن ذخیره کنید. می‌توان آن را به عنوان یک ظرف تصور کرد که داده‌های خود را در آن قرار داده‌اید. محتویات این ظرف می‌تواند پاک شود یا تغییر کند. هر متغیر دارای یک نام نیز هست. که از طریق آن می‌توان متغیر را از دیگر متغیرها تشخیص داد و به مقدار آن دسترسی پیدا کرد. همچنین دارای یک مقدار می‌باشد که می‌تواند توسط کاربر انتخاب شده باشد یا نتیجه یک محاسبه باشد . مقدار متغیر می‌تواند تهی نیز باشد. متغیر دارای نوع نیز هست بدین معنی که نوع آن با نوع داده‌ای که در آن ذخیره می‌شود یکی است.

متغیر دارای عمر نیز هست که از روی آن می‌توان تشخیص داد که متغیر باید چقدر در طول برنامه مورد استفاده قرار گیرد. و در نهایت متغیر دارای محدوده استفاده نیز هست که به شما می‌گوید که متغیر در چه جای برنامه برای شما قابل دسترسی است. ما از متغیرها به

عنوان یک انبار موقتی برای ذخیره داده استفاده می‌کنیم. هنگامی که یک برنامه ایجاد می‌کنیم احتیاج به یک مکان برای ذخیره داده، مقادیر یا داده‌هایی که توسط کاربر وارد می‌شوند، داریم .این مکان، همان متغیر است .

برای این از کلمه متغیر استفاده می‌شود چون ما می‌توانیم بسته به نوع شرایط هر جا که لازم باشد، مقدار آن را تغییر دهیم. متغیرها موقتی هستند و فقط موقعی مورد استفاده قرار می‌گیرند که برنامه در حال اجراست و وقتی شما برنامه را می‌بندید محتویات متغیرها نیز پاک می‌شود. قبلاً ذکر شد که به وسیله نام متغیر می‌توان به آن دسترسی پیدا کرد. برای نامگذاری متغیرها باید قوانین زیر را رعایت کرد :

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

دو متغیر با نامهای myNumber و MyNumber دو متغیر مختلف محسوب می‌شوند چون یکی از آنها با حرف کوچک m و دیگری با حرف بزرگ M شروع می‌شود. متغیر دارای نوع هست که نوع داده‌ای را که در خود ذخیره می‌کند را نشان می‌دهد. در درس بعد در مورد انواع داده ها در پایتون توضیح می دهیم. لیست کلمات کلیدی پایتون، که نباید از آنها در نامگذاری متغیرها استفاده کرد در زیر آمده است:

```
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
```

انواع داده

انواع داده هایی که در پایتون وجود دارند عبارتند از:

نوع	توضیح	
عددی(Numeric)	integer	شامل اعداد مثبت و منفی صحیح می باشد.
	float	شامل اعداد اعشاری می باشد.
	boolean	شامل دو مقدار true یا false می باشد.

رشته ای(String)	به مجموعه ای از کاراکترها که بین دو علامت کوتیشن یا دابل کوتیشن قرار گرفته باشند، اطلاق می شود.
لیست(List)	مجموعه ای از آیتم ها هستند که بین دو علامت [] قرار گرفته و با علامت کاما (,) از هم جدا شده اند.
تاپل(Tuple)	مجموعه ای از آیتم ها هستند که بین دو علامت () قرار گرفته و با علامت کاما (,) از هم جدا شده اند.
دیکشنری(Dictionary)	مجموعه ای از آیتم ها هستند که به صورت کلید و مقدار بوده، بین دو علامت {} قرار گرفته، و با علامت کاما (,) از هم جدا شده اند.

در مورد انواع داده های بالا و نحوه استفاده از آنها در متغیرها، در درس بعد توضیح می دهیم.

استفاده از متغیرها

بر خلاف زبان هایی مثل جاوا و سی شارپ، که هنگام تعریف متغیر باید نوع متغیر را هم مشخص می کردیم، در پایتون کافیست که فقط نام متغیر را نوشته و به وسیله علامت مساوی یک مقدار به آن اختصاص دهیم:

```
variableName = Value;
```

در مثال زیر نحوه تعریف و مقداردهی متغیرها نمایش داده شده است:

```
intVar        = 10;
floatVar      = 12.5;
boolVar       = True;
StringVar     = "Hello World!";
listVar       = [1,5,8];
tupleVar      = ("Python","Programming","begginer");
dictionaryVar = {'Name': 'jack', 'family': 'Scalia', 'Age': 7}

print("intVar = {0}"           .format(intVar));
print("floatVar = {0}"      .format(floatVar));
print("boolVar = {0}"          .format(boolVar));
print("StringVar = {0}"     .format(StringVar));
print("listVar = {0}"          .format(listVar));
print("tupleVar = {0}"      .format(tupleVar));

print("dictionaryVar = {0}" .format(dictionaryVar));
```

```
intVar = 10
floatVar = 12.5
boolVar = True
StringVar = Hello World!
```

```
listVar = [1, 5, 8]
tupleVar = ('Python', 'Programming', 'begginer')
dictionaryVar = {'Name': 'jack', 'family': 'Scalia', 'Age': 7}
```

در خطوط ۷-۱، متغیر ها تعریف شده اند. اما نوع این متغیرها چیست؟ پایتون نوع متغیرها را بسته به مقداری که به آنها اختصاص داده می شود در نظر می گیرد. مثلا نوع متغیر StringVar در خط ۴ از نوع رشته است، چون یک مقدار رشته ای به آن اختصاص داده شده است. به خطوط ۵، ۶ و ۷ کد بالا توجه کنید. در خط ۵ یک متغیر تعریف شده است و نوع داده ای که به آن اختصاص داده شده است از نوع list است. همانطور که در درس قبل اشاره شد، برای تعریف list علامت [] به کار می رود و آیتم های داخل آن به وسیله کاما از هم جدا می شوند:

```
listVar = [1, 5, 8];
```

در خط ۶ هم یک متغیر تعریف شده است و یک مقدار از نوع tuple به آن اختصاص داده شده است. در تعریف tuple به جای علامت [] از () استفاده می شود. تفاوت بین این دو را در درس های آینده بیشتر توضیح می دهیم. و اما در خط ۷ یک نوع دیکشنری تعریف شده است. برای تعریف دیکشنری بین کلید و مقدار علامت : و بین کلید/مقدارها هم علامت , قرار می گیرد:

```
dictionaryVar = {Key1:Value1, Key2:Value2, Key3:Value3};
```

مثلا در مثال بالا یک دیکشنری تعریف کرده ایم که سه آیتم یا کلید/مقدار دارد که بین آنها علامت کاما (,) قرار داده ایم. ولی بین یک کلید و مقدار مربوط به آن علامت : قرار گرفته است. برای اختصاص یک مقدار به چند متغیر می توان به صورت زیر عمل کرد:

```
identifier1 =  identifier2 = ... indentifierN = Value;
```

به مثال زیر توجه کنید:

```
num1 = num2 = num3 = num4 = num5 = 10;
message1 = message2 = message3 = "Hello World!";

print(num1);
print(num4);
print(message1);
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 :

```
identifier1, identifier2, ... indentifierN = Value1, Value2, ... ValueN;
```

به مثال زیر توجه کنید:

```
num1, num2, message1 = 10, 12.5, "Hello World!";

print(num1);
print(num2);
print(message1);
```

```
10
12.5
Hello World!
```

همانطور که در درس قبل هم اشاره کردیم، یک رشته در اصل یک مجموعه از کاراکترهاست که در داخل علامت "" یا ' ' قرار دارند. هر کدام از این کاراکترها دارای یک اندیس است که به وسیله آن اندیس قابل دسترسی هستند. اندیس کاراکترها در رشته از ۰ شروع می شود.

به رشته زیر توجه کنید:

```
message = "Hello World!";
```

در رشته بالا اندیس کاراکتر O برابر ۴ است. برای درک بهتر به شکل زیر توجه کنید:

```
H e l l o   W o r l d  !
0 1 2 3 4 5 6 7 8 9 10 11
```

حال برای چاپ یک کاراکتر (مثل W) از این رشته کافیست که به صورت زیر عمل کنیم:

```
message = "Hello World! ";

print(message[6]);
```

```
W
```

همانطور که در کد بالا مشاهده می کنید کافیست که نام متغیر را نوشته، در جلوی آن یک جفت کروشه و در داخل کروشه ها اندیس آن کاراکتری را که می خواهیم چاپ شود را بنویسیم. چاپ مقدار با استفاده از اندیس در مورد List و Tupleهم صدق می کند:

```
listVar  = [1, 5, 8];
tupleVar = ("Python", "Programming", "begginer");
```

```
print(listVar[2]);
print(tupleVar[1]);
```

```
8
Programming
```

و اما در مورد دیکشنری، شما باید نام کلید را بنویسید تا مقدار آن برای شما نمایش داده شود:

```
dictionaryVar = {'Name': 'jack', 'Family': 'Scalia', 'Age': 7}

print(dictionaryVar['Family']);
```

```
Scalia
```

نکته ای که بهتر است در همینجا به آن اشاره کنیم این است که کلید/مقدارها در دیکشنری می توانند از هر نوعی باشند و شما برای چاپ مقدار مربوط به یک کلید باید نام کلید را دقیق بنویسید. به مثال زیر توجه کنید:

```
dictionaryVar   = {1:'Jack', '2':'Scalia', 3:7}

print(dictionaryVar['2']);
```

```
Scalia
```

در مثال بالا ما مقدار کلید '۲' را چاپ کرده ایم. حال اگر به جای '۲' 'عدد ۲ را بنویسیم، یعنی علامت کوتیشن را نگذاریم با خطا مواجه می شویم:

```
dictionaryVar = {1:'Jack', '2':'Scalia', 3:7}

print(dictionaryVar[2]);
```

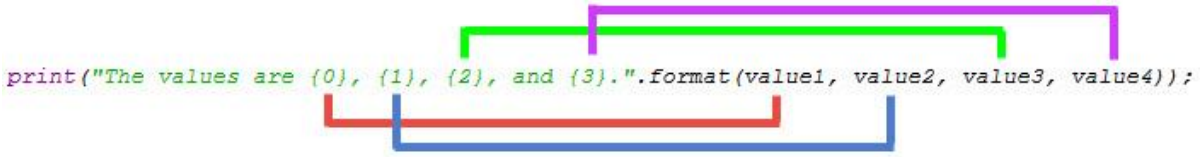
```
Traceback (most recent call last):
  File "C:/MyFirstProgram.py", line 3, in
    print(dictionaryVar[2]);
KeyError: 2
```

جانگهدار(Placeholders)

به متد print() در خطوط (۱۵-۹) توجه کنید. این متد به دو قسمت تقسیم شده است. قسمت اول یک رشته قالب بندی شده است و و قسمت دوم هم شامل متدی به نام format() است که دارای مقدار یا مقادیری است که توسط رشته قالب بندی شده مورد استفاده قرار می‌گیرند. اگر به دقت نگاه کنید رشته قالب بندی شده دارای عدد صفری است که در داخل دو آکولاد محصور شده است. البته عدد داخل دو آکولاد می‌تواند از صفر تا n باشد. به این اعداد جانگهدار می گویند. این اعداد بوسیله مقدار یا مقادیری که در داخل متد

format() هستند جایگزین می‌شوند. به عنوان مثال جانگهدار {0} به این معناست که اولین مقدار داخل متد format() در آن قرار می‌گیرد. برای روشن شدن مطلب به شکل زیر توجه کنید:

```
print("The values are {0}, {1}, {2}, and {3}.".format(value1, value2, value3, value4));
```



جانگهدارها از صفر شروع می‌شوند. تعداد جانگهدارها باید با تعداد مقادیری که در داخل متد format() آورده شده اند برابر باشد. برای مثال اگر شما چهار جانگهدار مثل بالا داشته باشید باید چهار مقدار هم برای آنها بعد از رشته قالب بندی شده در نظر بگیرید. اولین جانگهدار با اولین مقدار و دومین جا نگهدار با دومین مقدار جایگزین می‌شود. در ابتدا فهمیدن این مفهوم برای کسانی که تازه برنامه نویسی را شروع کرده‌اند سخت است اما در درسهای آینده مثالهای زیادی در این مورد مشاهده خواهید کرد.

عبارات و عملگرها

ابتدا با دو کلمه آشنا شوید:

- عملگر: نمادهایی هستند که اعمال خاص انجام می‌دهند.
- عملوند: مقادیری که عملگرها بر روی آنها عملی انجام می‌دهند.

مثلاً X+Y: یک عبارت است که در آن X و Y عملوند و علامت + عملگر به حساب می‌آیند.

زبانهای برنامه نویسی جدید دارای عملگرهایی هستند که از اجزاء معمول زبان به حساب می‌آیند. پایتون دارای عملگرهای مختلفی از جمله عملگرهای ریاضی، تخصیصی، مقایسه‌ای، منطقی و بیتی می‌باشد. از عملگرهای ساده ریاضی می‌توان به عملگر جمع و تفریق اشاره کرد.

سه نوع عملگر در پایتون وجود دارد :

یگانی - (Unary) به یک عملوند نیاز دارد

دودویی - (Binary) به دو عملوند نیاز دارد

سه تایی - (Ternary) به سه عملوند نیاز دارد

انواع مختلف عملگر که در این بخش مورد بحث قرار می‌گیرند، عبارتند از:

- عملگرهای ریاضی
- عملگرهای تخصیصی
- عملگرهای مقایسه‌ای

- عملگرهای منطقی
- عملگرهای بیتی

عملگرهای ریاضی

پایتون از عملگرهای ریاضی برای انجام محاسبات استفاده می‌کند. جدول زیر عملگرهای ریاضی پایتون را نشان می‌دهد:

عملگر	دسته	مثال	نتیجه
+	Binary	var1 = var2 + var3;	Var1 برابر است با حاصل جمع var2 و var3
-	Binary	var1 = var2 - var3;	Var1 برابر است با حاصل تفریق var2 و var3
*	Binary	var1 = var2 * var3;	Var1 برابر است با حاصلضرب var2 در var3
/	Binary	var1 = var2 / var3;	Var1 برابر است با حاصل تقسیم var2 بر var3
%	Binary	var1 = var2 % var3;	Var1 برابر است با باقیمانده تقسیم var2 و var3
**	Unary	var1 = var2 ** var3;	Var1 برابر است با مقدار var2به توان var3
//	Unary	var1 = var2 // var3;	Var1 برابر است با var2 تقسیم بر var3 (نتیجه به صورت صحیح نمایش داده می شود).

مثال بالا در از نوع عددی استفاده شده است. اما استفاده از عملگرهای ریاضی برای نوع رشته‌ای نتیجه متفاوتی دارد. اگر از عملگر + برای رشته‌ها استفاده کنیم دو رشته را با هم ترکیب کرده و به هم می‌چسباند. حال می‌توانیم با ایجاد یک برنامه نحوه عملکرد عملگرهای ریاضی در پایتون را یاد بگیریم:

```
#Assign test values
num1 = 5;
num2 = 3;

#Demonstrate use of mathematical operators
print("The sum of {0} and {1} is {2}."              .format(num1, num2, (num1 +  num2)));
print("The difference of {0} and {1} is {2}."       .format(num1, num2, (num1 -  num2)));
print("The product of {0} and {1} is {2}."          .format(num1, num2, (num1 *  num2)));
print("The quotient of {0} and {1} is {2:.2f}."     .format(num1, num2, (num1 /  num2)));
print("The remainder of {0} divided by {1} is {2}." .format(num1, num2, (num1 %  num2)));
print("The result of {0} power {1} is {2}."         .format(num1, num2, (num1 ** num2)));
print("The quotient of {0} and {1} is {2}."         .format(num1, num2, (num1 // num2)));

#Demonstrate concatenation on strings using the + operator
msg1 = "Hello ";
msg2 = "World!";
print(msg1 + msg2);
```

```
The sum of 5 and 3 is 8.
The difference of 5 and 3 is 2.
```

```
The product of 5 and 3 is 15.
The quotient of 5 and 3 is 1.67.
The remainder of 5 divided by 3 is 2.
The result of 5 power 3 is 125.
The quotient of 5 and 3 is 1.
Hello World!
```

برنامه بالا نتیجه هر عبارت را نشان می‌دهد. در این برنامه از متد print() برای نشان دادن نتایج در سطرهای متفاوت استفاده شده است. در خط ۹ برای اینکه ارقام کسری بعد از عدد حاصل دو رقم باشند از {2:.2f} استفاده می‌کنیم. {2:.2f} در این جا بدین معناست که عدد را تا دو رقم اعشار نمایش بده. پایتون خط جدید و فاصله و فضای خالی را نادیده می‌گیرد. در خط ۱۷ مشاهده می‌کنید که دو رشته به وسیله عملگر + به هم متصل شده‌اند. نتیجه استفاده از عملگر + برای چسباندن دو کلمه "Hello " و "World!" رشته "Hello World!" خواهد بود .به فاصله‌های خالی بعد از اولین کلمه توجه کنید اگر آنها را حذف کنید از خروجی برنامه نیز حذف می‌شوند.

عملگرهای تخصیصی (جایگزینی)

نوع دیگر از عملگرهای پایتون عملگرهای جایگزینی نام دارند .این عملگرها مقدار متغیر سمت راست خود را در متغیر سمت چپ قرار می‌دهند. جدول زیر انواع عملگرهای تخصیصی در پایتون را نشان می‌دهد:

عملگر	مثال	نتیجه
=	var1 = var2;	مقدار var1 برابر است با مقدار var2
+=	var1 += var2;	مقدار var1 برابر است با حاصل جمع var1 و var2
-=	var1 -= var2;	مقدار var1 برابر است با حاصل تفریق var1 و var2
*=	var1 *= var2;	مقدار var1 برابر است با حاصل ضرب var1 در var2
/=	var1 /= var2;	مقدار var1 برابر است با حاصل تقسیم var1 بر var2
%=	var1 %= var2;	مقدار var1 برابر است با باقیمانده تقسیم var1 بر var2
**=	var1 **= var2;	مقدار var1 برابر است با var1 به توان var2
//=	var1 //= var2;	مقدار var1 برابر است با حاصل تقسیم var1 بر var2

از عملگر =+ برای اتصال دو رشته نیز می‌توان استفاده کرد. استفاده از این نوع عملگرها در واقع یک نوع خلاصه نویسی در کد است. مثلاً شکل اصلی کد var1 += var2به صورت var1 = var1 + var2 می‌باشد. این حالت کدنویسی زمانی کارایی خود را نشان می‌دهد که نام متغیرها طولانی باشد. برنامه زیر چگونگی استفاده از عملگرهای تخصیصی و تأثیر آنها را بر متغیرها نشان می‌دهد:

```
print("Assigning 10 to number...");
number = 10;
print("Number = {0}" .format(number));

print("Adding 10 to number...");
number += 10;
print("Number = {0}" .format(number));

print("Subtracting 10 from number...");
number -= 10;
print("Number = {0}" .format(number));
```

```
Assigning 10 to number...
Number = 10
Adding 10 to number...
Number = 20
Subtracting 10 from number...
Number = 10
```

در برنامه از ۳ عملگر تخصیصی استفاده شده است. ابتدا یک متغیر و مقدار 10با استفاده از عملگر = به آن اختصاص داده شده است. سپس به آن با استفاده از عملگر =+ مقدار ۱۰ اضافه شده است. و در آخر به وسیله عملگر =- عدد ۱۰ از آن کم شده است.

عملگرهای مقایسه ای

از عملگرهای مقایسه‌ای برای مقایسه مقادیر استفاده می‌شود . نتیجه این مقادیر یک مقدار بولی (منطقی) است. این عملگرها اگر نتیجه مقایسه دو مقدار درست باشد مقدار ۱ و اگر نتیجه مقایسه اشتباه باشد مقدار ۰ را نشان می‌دهند. این عملگرها به طور معمول در دستورات شرطی به کار می‌روند به این ترتیب که باعث ادامه یا توقف دستور شرطی می‌شوند. جدول زیر عملگرهای مقایسه‌ای در پایتون را نشان می‌دهد:

عملگر	مثال	نتیجه
==	var1 = var2 == var3	var1 در صورتی True است که مقدار var2 با مقدار var3 برابر باشد در غیر اینصورت False است
!=	var1 = var2 != var3	var1 در صورتی True است که مقدار var2 با مقدار var3 برابر نباشد در غیر اینصورت False است
<>	var1 = var2 <> var3	var1 در صورتی True است که مقدار var2 با مقدار var3 برابر نباشد در غیر اینصورت False است

<	var1 = var2 < var3	var1 در صورتی True است که مقدار var2 کوچک‌تر از var3 مقدار باشد در غیر اینصورت False است
>	var1 = var2 > var3	var1 در صورتی True است که مقدار var2 بزرگ‌تر ازمقدار var3 باشد در غیر اینصورت False است
<=	var1 = var2 <= var3	var1 در صورتی True است که مقدار var2 کوچک‌تر یا مساوی مقدار var3 باشد در غیر اینصورت False است
>=	var1 = var2 >= var3	var1 در صورتی True است که مقدار var2 بزرگ‌تر یا مساوی var3 مقدار باشد در غیر اینصورت False است

برنامه زیر نحوه عملکرد این عملگرها را نشان می‌دهد:

```
num1 = 10;
num2 = 5;

print("{0} == {1} : {2}" .format(num1, num2, num1 == num2));
print("{0} != {1} : {2}" .format(num1, num2, num1 != num2));
print("{0} <> {1} : {2}" .format(num1, num2, num1 != num2));
print("{0} <  {1} : {2}" .format(num1, num2, num1 <  num2));
print("{0} >  {1} : {2}" .format(num1, num2, num1 >  num2));
print("{0} <= {1} : {2}" .format(num1, num2, num1 <= num2));
print("{0} >= {1} : {2}" .format(num1, num2, num1 >= num2));
```

```
10 == 5 : False
10 != 5 : True
10 <> 5 : True
10 <  5 : False
10 >  5 : True
10 <= 5 : False
10 >= 5 : True
```

در مثال بالا ابتدا دو متغیر را که می‌خواهیم با هم مقایسه کنیم را ایجاد کرده و به آنها مقادیری اختصاص می‌دهیم. سپس با استفاده از یک عملگر مقایسه‌ای آنها را با هم مقایسه کرده و نتیجه را چاپ می‌کنیم. به این نکته توجه کنید که هنگام مقایسه دو متغیر از عملگر == به جای عملگر = باید استفاده شود. عملگر = عملگر تخصیصی است و در عبارتی مانند x = y مقدار y را در به x اختصاص می‌دهد. عملگر == عملگر مقایسه‌ای است که دو مقدار را با هم مقایسه می‌کند مانند x==y و اینطور خوانده می‌شود x برابر است با y.

عملگرهای منطقی

عملگرهای منطقی بر روی عبارات منطقی عمل می کنند و نتیجه آنها نیز یک مقدار بولی است. از این عملگرها اغلب برای شرطهای پیچیده استفاده می شود. همانطور که قبلا یاد گرفتید مقادیر بولی می توانند False یا True باشند. فرض کنید که var2 و var3 دو مقدار بولی هستند.

عملگر	مثال
and	var1 = var2 and var3;
or	var1 = var2 or var3;
not	var1 = not (var1);

عملگر منطقی and

اگر مقادیر دو طرف عملگر and، True باشند عملگر and مقدار True را بر می گرداند. در غیر اینصورت اگر یکی از مقادیر یا هر دوی آنها False باشند مقدار False را بر می گرداند. در زیر جدول درستی عملگر and نشان داده شده است:

X and Y	Y	X
True	True	True
False	False	True
False	True	False
False	False	False

برای درک بهتر تاثیر عملگر and یاد آوری می کنم که این عملگر فقط در صورتی مقدار True را نشان می دهد که هر دو عملوند مقدارشان True باشد.در غیر اینصورت نتیجه تمام ترکیبهای بعدی False خواهد شد. استفاده از عملگر and مانند استفاده از عملگرهای مقایسه ای است. به عنوان مثال نتیجه عبارت زیر درست (True) است اگر سن (age) بزرگتر از ۱۸ و salary کوچکتر از ۱۰۰۰ باشد.

```
result = (age > 18) and (salary < 1000);
```

عملگر and زمانی کارامد است که ما با محدود خاصی از اعداد سرو کار داریم. مثلا عبارت 10 <= x <= 100بدین معنی است که x می تواند مقداری شامل اعداد ۱۰ تا ۱۰۰ را بگیرد. حال برای انتخاب اعداد خارج از این محدوده می توان از عملگر منطقی and به صورت زیر استفاده کرد.

```
inRange = (number <= 10) and  (number >= 100);
```

عملگر منطقی or

اگر یکی یا هر دو مقدار دو طرف عملگر or، درست (True) باشد، عملگر or مقدار True را بر می گرداند. جدول درستی عملگر or در زیر نشان داده شده است :

X	Y	X or Y
True	True	True
True	False	True
False	True	True
False	False	False

در جدول بالا مشاهده می کنید که عملگر or در صورتی مقدار False را بر میگرداند که مقادیر دو طرف آن False باشند. کد زیر را در نظر بگیرید.نتیجه این کد در صورتی درست (True) است که رتبه نهایی دانش آموز (finalGrade) بزرگتر از ۷۵ یا یا نمره نهایی امتحان آن ۱۰۰ باشد.

```
isPassed = (finalGrade >= 75) or (finalExam == 100);
```

عملگر منطقی not

برخلاف دو اپراتور or و and عملگر منطقی NOT یک عملگر یگانی است و فقط به یک عملوند نیاز دارد. این عملگر یک مقدار یا اصطلاح بولی را نفی می کند. مثلا اگر عبارت یا مقدار True باشد آنرا False و اگر False باشد آنرا True می کند. جدول زیر عملکرد اپراتور NOT را نشان می دهد:

X	not X
True	False
False	True

نتیجه کد زیر در صورتی درست است که age (سن) بزرگتر یا مساوی ۱۸ نباشد.

```
isMinor = not(age >= 18);
```

عملگرهای بیتی

عملگرهای بیتی به شما اجازه می‌دهند که شکل باینری انواع داده‌ها را دستکاری کنید. برای درک بهتر این درس توصیه می‌شود که شما سیستم باینری و نحوه تبدیل اعداد دهدهی به باینری را از لینک زیر یاد بگیرید:

http://www.w3-farsi.com/?p=5698

در سیستم باینری (دودویی) که کامپیوتر از آن استفاده می‌کند وضعیت هر چیز یا خاموش است یا روشن. برای نشان دادن حالت روشن از عدد ۱ و برای نشان دادن حالت خاموش از عدد ۰ استفاده می‌شود. بنابراین اعداد باینری فقط می‌توانند صفر یا یک باشند. اعداد باینری را اعداد در مبنای ۲ و اعداد اعشاری را اعداد در مبنای ۱۰ می گویند. یک بیت نشان دهنده یک رقم باینری است و هر بایت نشان دهنده ۸ بیت است. به عنوان مثال برای یک داده از نوع int به ۳۲ بیت یا ۴ بایت فضا برای ذخیره آن نیاز داریم، این بدین معناست که اعداد از ۳۲ رقم ۰ و ۱ برای ذخیره استفاده می‌کنند. برای مثال عدد ۱۰۰ وقتی به عنوان یک متغیر از نوع int ذخیره می‌شود در کامپیوتر به صورت زیر خوانده می‌شود:

```
00000000000000000000000001100100
```

عدد ۱۰۰ در مبنای ده معادل عدد ۱۱۰۰۱۰۰ در مبنای ۲ است. در اینجا ۷ رقم سمت راست نشان دهنده عدد ۱۰۰ در مبنای ۲ است و مابقی صفرهای سمت راست برای پر کردن بیتهایی است که عدد از نوع int نیاز دارد. به این نکته توجه کنید که اعداد باینری از سمت راست به چپ خوانده می‌شوند. عملگرهای بیتی پایتون در جدول زیر نشان داده شده‌اند:

عملگر	مثال
&	x = y & z;
|	x = y | z;
^	x = y ^ z;
~	x = ~y;
&=	x &= y;
|=	x |= y;
^=	x ^= y;

عملگر بیتی (&)AND

عملگر بیتی AND کاری شبیه عملگر منطقی AND انجام می‌دهد با این تفاوت که این عملگر بر روی بیتها کار می‌کند. اگر مقادیر دو طرف آن ۱ باشد مقدار ۱ را بر می‌گرداند و اگر یکی یا هر دو طرف آن صفر باشد مقدار صفر را بر می‌گرداند. جدول درستی عمگر بیتی AND در زیر آمده است:

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

در زیر نحوه استفاده از عملگر بیتی AND آمده است:

```
result = 5 & 3;
```

```
print(result);
```

```
1
```

همانطور که در مثال بالا مشاهده می‌کنید نتیجه عملکرد عملگر AND بر روی دو مقدار ۵ و ۳ عدد یک می‌شود. اجازه بدهید ببینیم که چطور این نتیجه را به دست می‌آید:

```
5:  00000000000000000000000000000101
3:  00000000000000000000000000000011
---------------------------------
1:  00000000000000000000000000000001
```

ابتدا دو عدد ۵ و ۳ به معادل باینری‌شان تبدیل می‌شوند. از آنجاییکه هر عدد صحیح (int) ۳۲ بیت است از صفر برای پر کردن بیتهای خالی استفاده می‌کنیم. با استفاده از جدول درستی عملگر بیتی AND می‌توان فهمید که چرا نتیجه عدد یک می‌شود.

عملگر بیتی OR(|)

اگر مقادیر دو طرف عملگر بیتی OR هر دو صفر باشند نتیجه صفر در غیر اینصورت ۱ خواهد شد. جدول درستی این عملگر در زیر آمده است:

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

نتیجه عملگر بیتی OR در صورتی صفر است که عملوندهای دو طرف آن صفر باشند. اگر فقط یکی از دو عملوند یک باشد نتیجه یک خواهد شد. به مثال زیر توجه کنید:

```
result = 7 | 9;
print(result);
```

```
15
```

وقتی که از عملگر بیتی OR برای دو مقدار در مثال بالا (۷ و ۹) استفاده می‌کنیم نتیجه ۱۵ می‌شود. حال بررسی می‌کنیم که چرا این نتیجه به دست آمده است؟

```
 7: 00000000000000000000000000000111
 9: 00000000000000000000000000001001
---------------------------------
15: 00000000000000000000000000001111
```

با استفاده از جدول درستی عملگر بیتی OR می‌توان نتیجه استفاده از این عملگر را تشخیص داد. عدد ۱۱۱۱ باینری معادل عدد ۱۵ صحیح است.

عملگر بیتی XOR(^)

جدول درستی این عملگر در زیر آمده است:

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

در صورتیکه عملوندهای دو طرف این عملگر هر دو صفر یا هر دو یک باشند نتیجه صفر در غیر اینصورت نتیجه یک می‌شود. در مثال زیر تأثیر عملگر بیتی XOR را بر روی دو مقدار مشاهده می‌کنید:

```
result = 5 ^ 7;

print(result);
```

```
2
```

در زیر معادل باینری اعداد بالا (۵ و ۷) نشان داده شده است.

```
5: 00000000000000000000000000000101
7: 00000000000000000000000000000111
-----------------------------------
2: 00000000000000000000000000000010
```

با نگاه کردن به جدول درستی عملگر بیتی XOR، می‌توان فهمید که چرا نتیجه عدد ۲ می‌شود.

عملگر بیتی NOT(~)

این عملگر یک عملگر یگانی است و فقط به یک عملوند نیاز دارد. در زیر جدول درستی این عملگر آمده است:

X	NOT X
1	0
0	1

عملگر بیتی NOT مقادیر بیتها را معکوس می‌کند. در زیر چگونگی استفاده از این عملگر آمده است:

```

```

```
result = 5 ^ 7;

print(result);
```

```
-8
```

به نمایش باینری مثال بالا که در زیر نشان داده شده است توجه نمایید.

```
7: 00000000000000000000000000000111
-----------------------------------
-8: 11111111111111111111111111111000
```

عملگر بیتی تغییر مکان(shift)

این نوع عملگرها به شما اجازه می‌دهند که بیتها را به سمت چپ یا راست جا به جا کنید. دو نوع عملگر بیتی تغییر مکان وجود دارد که هر کدام دو عملوند قبول می‌کنند. عملوند سمت چپ این عملگرها حالت باینری یک مقدار و عملوند سمت راست تعداد جابه جاییبیت ها را نشان می‌دهد.

عملگر	نام	مثال
>>	تغییر مکان به سمت چپ	x = y << 2;
<<	تغییر مکان به سمت راست	x = y >> 2;

عملگر تغییر مکان به سمت چپ

این عملگر بیتهای عملوند سمت چپ را به تعداد n مکان مشخص شده توسط عملوند سمت راست، به سمت چپ منتقل می‌کند. به عنوان مثال:

```
result = 10 << 2;

print(result);
```

```
40
```

در مثال بالا ما بیت‌های مقدار ۱۰ را دو مکان به سمت چپ منتقل کرده‌ایم، حال بیایید تأثیر این انتقال را بررسی کنیم:

```
10: 00000000000000000000000000001010
-----------------------------------
40: 00000000000000000000000000101000
```

مشاهده می‌کنید که همه بیت‌ها به اندازه دو واحد به سمت چپ منتقل شده‌اند. در این انتقال دو صفر از صفرهای سمت چپ کم می‌شود و در عوض دو صفر به سمت راست اضافه می‌شود.

عملگر تغییر مکان به سمت راست

این عملگر شبیه به عمگر تغییر مکان به سمت چپ است با این تفاوت که بیت‌ها را به سمت راست جا به جا می‌کند. به عنوان مثال:

```
result = 100 >> 4;

print(result);
```

```
6
```

با استفاده از عملگرتغییر مکان به سمت راست بیت‌های مقدار ۱۰۰ را به اندازه ۴ واحد به سمت چپ جا به جا می‌کنیم. اجازه بدهید تأثیر این جا به جایی را مورد بررسی قرار دهیم:

```
100: 00000000000000000000000001100100
-------------------------------------
  6: 00000000000000000000000000000110
```

هر بیت به اندازه ۴ واحد به سمت راست منتقل می‌شود، بنابراین ۴ بیت اول سمت راست حذف شده و چهار صفر به سمت چپ اضافه می‌شود.

عملگرهای خاص

علاوه بر عملگرهایی که تا کنون ذکر شد، پایتون دارای عملگرهای خاص زیر نیز می‌باشد:

- عملگرهای `membership` که بررسی می‌کنند آیا متغیر مورد نظر در یک مجموعه (sequence) همچون رشته، `list`یا `tuple`وجود دارد یا خیر.
- عملگرهای `Identity` که مکان‌های قرار گیری دو شیء را با هم مقایسه می‌کند (بررسی می‌کنند آیا دوشی با هم برابر هستند یا خیر).

در جدول زیر انواع عملگرهای `membership` و `Identity` ذکر شده‌اند:

عملگر		توضیح
`Memberships operator`	`in`	در صورت یافتن متغیر مورد نظر در مجموعه‌ی مشخص شده، `True` و در غیر این صورت `False` را برمی گرداند.

	not in	در صورت یافت نشدن متغیر مورد نظر در مجموعه‌ی مشخص شده، True و در غیر این صورت False را برمی گرداند.
Identity operator	is	اگر متغیرهای هر دو طرف عملگر به شی یکسان اشاره داشته باشند، True و در غیر این صورت False را برمی گرداند.
	is not	چنانچه متغیر در دو طرف عملگر به شی یکسان اشاره داشته باشد، False و در غیر این صورت True را بر می گرداند.

به مثال‌های زیر توجه کنید:

```
print(5 in     [3, 8, 5, 10]);
print(5 not in [3, 8, 5, 10]);
```

```
True

False
```

در خط اول کد بالا، چک می‌شود که آیا عدد ۵ در مجموعه [3, 8, 5, 10] وجود دارد یا نه؟ و چون وجود دارد مقدار True بر گردانده می‌شود. در خط دوم هم که کاملاً مشخص است که اگر عدد ۵ در مجموعه وجود نداشته باشد مقدار True بر گردانده می‌شود ولی چون عدد ۵ وجود دارد مقدار False برگردانده می‌شود.

```
number1 = 5;
number2 = 6;

print(number1 is     number2);
print(number1 is not number2);
```

```
True

False
```

در مثال بالا و در اولین مقایسه گفته شده است که آیا number1 همان number2است و چون چنین نیست مقدار False و در مقایسه دوم هم گفته شده است که آیا number1 برابر number2 نیست؟ و چون برابر نیستند مقدار True برگردانده شده است.

گرفتن ورودی از کاربر

پایتون متد input() را برای گرفتن ورودی از کاربر، در اختیار شما قرار می‌دهد. همانطور که از نام این متد پیداست، تمام کاراکترهایی را که شما در محیط برنامه نویسی تایپ می‌کنید تا زمانی که دکمه enterرا می‌زنید، می‌خواند. به برنامه زیر توجه کنید:

```
1  name   = input("Enter your name: ");
2  age    = input("Enter your age: ");
```

```
height = input("Enter your height: ");

#Print a blank line
print();

#Show the details you typed
print("Name is {0}.".format(name));
print("Age is {0}.".format(age));
print("Height is {0}.".format(height));
```

```
Enter your name: John
Enter your age: 18
Enter your height: 160.5

Name is John.
Age is 18.
Height is 160.5.
```

ابتدا ۳ متغیر را برای ذخیره داده در برنامه تعریف می‌کنیم (خطوط ۱ و ۲ و ۳). برنامه از کاربر می‌خواهد که نام خود را وارد کند (خط ۱). در خط ۲ شما به عنوان کاربر نام خود را وارد می‌کنید. سپس برنامه از ما سن را سؤال می‌کند (خط ۳). در خط ۶ هم یک خط فاصله به وسیله متد `print()` ایجاد کرده ایم تا بین ورودی های شما و خروجی فاصله ای جهت تفکیک ایجاد شود. حال برنامه را اجرا کرده و با وارد کردن مقادیر مورد نظر نتیجه را مشاهده کنید.

ساختارهای تصمیم

تقریبا همه زبانهای برنامه نویسی به شما اجازه اجرای کد را در شرایط مطمئن می دهند. حال تصور کنید که یک برنامه دارای ساختار تصمیم گیری نباشد و همه کدها را اجرا کند. این حالت شاید فقط برای چاپ یک پیغام در صفحه مناسب باشد ولی فرض کنید که شما بخواهید اگر مقدار یک متغیر با یک عدد برابر باشد سپس یک پیغام چاپ شود آن وقت با مشکل مواجه خواهید شد. پایتون راه های مختلفی برای رفع این نوع مشکلات ارائه می دهد. در این بخش با مطالب زیر آشنا خواهید شد:

- دستور`if`
- دستور`if…else`
- عملگر سه تایی
- دستور `if` چندگانه
- دستور `if` تو در تو
- عملگرهای منطقی

دستور if

می‌توان با استفاده از دستور if و یک شرط خاص که باعث ایجاد یک کد می‌شود یک منطق به برنامه خود اضافه کنید. دستور if ساده‌ترین دستور شرطی است که برنامه می‌گوید اگر شرطی برقرار است کد معینی را انجام بده. ساختار دستور if به صورت زیر است:

```
if (condition):

   code to execute
```

قبل از اجرای دستور if ابتدا شرط بررسی می‌شود. اگر شرط برقرار باشد یعنی درست باشد سپس کد اجرا می‌شود. شرط یک عبارت مقایسه‌ای است. می‌توان از عملگرهای مقایسه‌ای برای تست درست یا اشتباه بودن شرط استفاده کرد. اجازه بدهید که نگاهی به نحوه استفاده از دستور if در داخل برنامه بیندازیم. برنامه زیر پیغام Hello World را اگر مقدار number کمتر از ۱۰ و Goodbye World را اگر مقدار number از ۱۰ بزرگ‌تر باشد در صفحه نمایش می‌دهد:

```
1  #Declare a variable and set it a value less than 10
2  number = 5;
3
4  #If the value of number is less than 10
5  if (number < 10):
6      print("Hello World.");
7
8  #Change the value of a number to a value which is greater than 10
9  number = 15;
10
11 #If the value of number is greater than 10
12 if (number > 10):
13     print("Goodbye World.");
```

```
Hello World.
Goodbye World.
```

در خط ۲ یک متغیر با نام number تعریف و مقدار ۵ به آن اختصاص داده شده است. وقتی به اولین دستور if در خط ۲ می‌رسیم برنامه تشخیص می‌دهد که مقدار number از ۱۰ کمتر است یعنی ۵ کوچک‌تر از ۱۰ است.

منطقی است که نتیجه مقایسه درست می‌باشد، بنابراین دستور if دستور را اجرا می‌کند (خط ۶) و پیغام Hello World چاپ می‌شود. حال مقدار number را به ۱۵ تغییر می‌دهیم (خط ۹). وقتی به دومین دستور if در خط ۱۲ می‌رسیم برنامه مقدار number را با ۱۰ مقایسه می‌کند و چون مقدار number یعنی ۱۵ از ۱۰ بزرگ‌تر است برنامه پیغام Goodbye World را چاپ می‌کند (خط ۱۳). به این نکته توجه کنید که دستور if را می‌توان در یک خط نوشت:

```
if (number > 10): print("Goodbye World.");
```

شما می‌توانید چندین دستور را در داخل دستور if بنویسید. کافیست که حواستان به تو رفتگی کدها باشد. نحوه تعریف چند دستور در داخل بدنه if به صورت زیر است:

```
if (condition)
    statement1;
    statement2;
    .
    .
    .
    statementN;
```

این هم یک مثال ساده:

```
x = 15;

if (x > 10):
   print("x is greater than 10.");
   print("This is still part of the if statement.");
```

در مثال بالا اگر مقدار x از ۱۰ بزرگ‌تر باشد دو پیغام چاپ می‌شود. حال اگر به عنوان مثال، تو رفتگی خط آخر را حذف کنیم و مقدار x از ۱۰ بزرگ‌تر نباشد مانند کد زیر:

```
x = 5;

if (x > 10):
   print("x is greater than 10.");
print("This is not part of the if statement.");
```

کد بالا در صورتی بهتر خوانده می‌شود که بین دستورات فاصله بگذاریم:

```
x = 5;

if (x > 10):
   print("x is greater than 10.");

print("This is not part of the if statement.");
```

می‌بیند که دستور آخر در مثال بالا، جز دستور if نیست. اینجاست که چون ما فرض را بر این گذاشته‌ایم که مقدار x از ۱۰ کوچک‌تر است پس خط This is not part of the if statement. (Really?) چاپ می‌شود. در نتیجه اهمیت وجود تو رفتگی مشخص می‌شود. به عنوان تمرین همیشه حتی اگر فقط یک دستور در بدنه if داشتید برای آن یک تو رفتگی ایجاد کنید. فراموش نکنید که از قلم انداختن یک تو رفتگی باعث به وجود آمدن خطا شده و یافتن آن را سخت می‌کند. مثالی دیگر در مورد دستور if:

```
firstNumber  = input("Enter a number: ");
secondNumber = input("Enter another number: ");

if (firstNumber == secondNumber):
    print("{0} == {1}".format(firstNumber, secondNumber));

if (firstNumber != secondNumber):
```

```
    print("{0} != {1}".format(firstNumber, secondNumber));

if (firstNumber < secondNumber):
    print("{0} < {1}".format(firstNumber, secondNumber));

if (firstNumber > secondNumber):
    print("{0} > {1}".format(firstNumber, secondNumber));

if (firstNumber <= secondNumber):
    print("{0} <= {1}".format(firstNumber, secondNumber));

if (firstNumber >= secondNumber):
    print("{0} >= {1}".format(firstNumber, secondNumber));
```

```
Enter a number: 2
Enter another number: 5
2 != 5
2 < 5
2 <= 5
Enter a number: 10
Enter another number: 3
10 != 3
10 > 3
10 >= 3
Enter a number: 5
Enter another number: 5
5 == 5
5 <= 5
5 >= 5
```

ما از عملگرهای مقایسه‌ای در دستور `if` استفاده کرده‌ایم. ابتدا دو عدد که قرار است با هم مقایسه شوند را به عنوان ورودی از کاربر می‌گیریم. اعداد با هم مقایسه می‌شوند و اگر شرط درست بود پیغامی چاپ می‌شود. به این نکته توجه داشته باشید که شرطها مقادیر بولی هستند، یعنی دارای دو مقدار `True` یا `Flase` می باشند.

دستور if…else

دستور `if` فقط برای اجرای یک حالت خاص به کار می‌رود یعنی اگر حالتی برقرار بود کار خاصی انجام شود. اما زمانی که شما بخواهید اگر شرط خاصی برقرار شد یک دستور و اگر برقرار نبود دستور دیگر اجرا شود باید از دستور `if else` استفاده کنید. ساختار دستور `if else` در زیر آمده است:

```
if (condition):

   code to execute if condition is true

else:

   code to execute if condition is false;
```

از کلمه کلیدی else نمی‌توان به تنهایی استفاده کرد بلکه حتماً باید با if به کار برده شود. اگر فقط یک کد اجرایی در داخل بدنه if و بدنه else دارید استفاده از آکولاد اختیاری است. کد داخل بلوک else فقط در صورتی اجرا می‌شود که شرط داخل دستور if نادرست باشد. در زیر نحوه استفاده از دستور if…else آمده است:

```
1  number = 5;
2
3  #Test the condition
4  if (number < 10):
5      print("The number is less than 10.");
6  else:
7      print("The number is either greater than or equal to 10.");
8
9  #Modify value of number
10 number = 15;
11
12 #Repeat the test to yield a different result
13 if (number < 10):
14     print("The number is less than 10.");
15 else:
16     print("The number is either greater than or equal to 10.");
```

```
The number is less than 10.
The number is either greater than or equal to 10.
```

در خط ۱ یک متغیر به نام number تعریف کرده‌ایم و در خط ۴ تست می‌کنیم که آیا مقدار متغیر number از ۱۰ کمتر است یا نه و چون کمتر است در نتیجه کد داخل بلوک if اجرا می‌شود (خط ۷) و اگر مقدار number را تغییر دهیم و به مقداری بزرگتر از ۱۰ تغییر دهیم (خط ۱۰)، شرط نادرست می‌شود (خط ۱۳) و کد داخل بلوک else اجرا می‌شود (خط ۱۶).

دستور if…elif…else

اگر بخواهید چند شرط را بررسی کنید چکار می‌کنید؟ می‌توانید از چندین دستور if استفاده کنید و بهتر است که این دستورات if را به صورت زیر بنویسید:

```
if (condition):
   code to execute;
else:
   if (condition):
      code to execute;
   else:
      if (condition):
         code to execute;
      else:
```

```
        code to execute;
```

خواندن کد بالا سخت است. بهتر است دستورات را به صورت تو رفتگی در داخل بلوک else بنویسید. می‌توانید کد بالا را ساده‌تر کنید :

```
if (condition):
   code to execute;
elif (condition):
   code to execute;
elif (condition):
   code to execute;
else:
   code to execute;
```

حال که نحوه استفاده از دستور if else را یاد گرفتید باید بدانید که مانند else، elif نیز به دستور if وابسته است. دستور elif وقتی اجرا می‌شود که اولین دستور if اشتباه باشد حال اگر elif اشتباه باشد دستور elif بعدی اجرا می‌شود. و اگر آن نیز اجرا نشود در نهایت دستور else اجرا می‌شود. برنامه زیر نحوه استفاده از دستورelif را نشان می‌دهد:

```
print("What's your favorite color?");
print("[1] Black");
print("[2] White");
print("[3] Blue");
print("[4] Red");
print("[5] Yellown");

choice = int(input("Enter your choice: "));

if (choice == 1):
    print("You might like my black t-shirt.");
elif (choice == 2):
    print("You might be a clean and tidy person.");
elif (choice == 3):
    print("You might be sad today.");
elif (choice == 4):
    print("You might be inlove right now.");
elif (choice == 5):
    print("Lemon might be your favorite fruit.");
else:
    print("Sorry, your favorite color is not in the choices above.");
```

```
What's your favorite color?
[1] Black
[2] White
[3] Blue
```

```
[4] Red
[5] Yellow

Enter your choice: 1
You might like my black t-shirt.
What's your favorite color?
[1] Black
[2] White
[3] Blue
[4] Red
[5] Yellow

Enter your choice: 999
Sorry, your favorite color is not in the choices above.
```

خروجی برنامه بالا به متغیر choice وابسته است. بسته به اینکه شما چه چیزی انتخاب می‌کنید پیغامهای مختلفی چاپ می‌شود. اگر عددی که شما تایپ می‌کنید در داخل حالتهای انتخاب نباشد کد مربوط به بلوک else اجرا می‌شود.

دستور if تو در تو

می‌توان از دستور if تو در تو در پایتون استفاده کرد. یک دستور ساده if در داخل دستور if دیگر:

```
if (condition):
    code to execute;
    if (condition):
         code to execute;
    elif (condition):
        if (condition):
           code to execute;
else:
    if (condition):
       code to execute;
```

اجازه بدهید که نحوه استفاده از دستور if تو در تو را نشان دهیم:

```
age = int(input("Enter your age: "));
gender = input("Enter your gender (male/female): ");

if (age > 12):
    if (age < 20):
        if (gender == "male"):
            print("You are a teenage boy.");
        else:
            print("You are a teenage girl.");
    else:
        print("You are already an adult.");
else:
```

```
13      print("You are still too young.");
```

```
Enter your age: 18
Enter your gender: male
You are a teenage boy.
Enter your age: 12
Enter your gender: female
You are still too young.
```

اجازه بدهید که برنامه را کالبد شکافی کنیم. ابتدا برنامه از شما درباره سنتان سؤال می‌کند (خط ۱). در خط ۲ درباره جنستان از شما سؤال می‌کند. سپس به اولین دستور `if` می‌رسد (خط ۴). در این قسمت اگر سن شما بیشتر از ۱۲ سال باشد برنامه وارد بدنه دستور `if` می‌شود در غیر اینصورت وارد بلوک `else` (خط ۱۲) مربوط به همین دستور `if` می‌شود.

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.

استفاده از عملگرهای منطقی

عملگرهای منطقی به شما اجازه می‌دهند که چندین شرط را با هم ترکیب کنید. این عملگرها حداقل دو شرط را در گیر می‌کنند و در آخر یک مقدار بولی را بر می‌گردانند. در جدول زیر برخی از عملگرهای منطقی آمده است :

عملگر	مثال	تأثیر
`and`	`z = (x > 2) and (y < 10)`	مقدار `Z` در صورتی `True` است که هر دو شرط دو طرف عملگر مقدارشان `True` باشد. اگر فقط مقدار یکی از شروط `False` باشد مقدار `False` ، `z`خواهد شد.
`or`	`z = (x > 2) or (y < 10)`	مقدار `Z` در صورتی `True` است که یکی از دو شرط دو طرف عملگر مقدارشان `True` باشد. اگر هر دو شرط مقدارشان `False` باشد مقدار `False` ، `z`خواهد شد .
`not`	`z = not(x > 2)`	مقدار `Z` در صورتی `True` است که مقدار شرط `False` باشد و در صورتی `False` است که مقدار شرط `True` باشد

به عنوان مثال جمله `z = (x > 2) and (y < 10)` را به این صورت بخوانید: "در صورتی مقدار z برابر True است که مقدار x بزرگ‌تر از 2و مقدار y کوچک‌تر از ۱۰ باشد در غیر اینصورت False است". این جمله بدین معناست که برای اینکه مقدار کل دستور True باشد باید مقدار همه شروط True باشد. عملگر منطقی or تأثیر متفاوتی نسبت به عملگر منطقی and دارد. نتیجه عملگر منطقی or برابر True است اگر فقط مقدار یکی از شروط True باشد. و اگر مقدار هیچ یک از شروط True نباشد نتیجه False خواهد شد. می‌توان عملگرهای منطقی and و or را با هم ترکیب کرده و در یک عبارت به کار برد مانند:

```
if ( (x == 1) and ( (y > 3) or z < 10) ) ):

   #do something here
```

در اینجا استفاده از پرانتز مهم است چون از آن در گروه بندی شرطها استفاده می‌کنیم. در اینجا ابتدا عبارت `(y > 3) or (z < 10)` مورد بررسی قرار می‌گیرد (به علت تقدم عملگرها). سپس نتیجه آن بوسیله عملگر and با نتیجه `(x == 1)` مقایسه می‌شود. حال بیایید نحوه استفاده از عملگرهای منطقی در برنامه را مورد بررسی قرار دهیم:

```
1   age = int(input("Enter your age: "));
2   gender = input("Enter your gender (male/female): ");
3
4   if (age > 12 and age < 20):
5       if (gender == "male"):
6           print("You are a teenage boy.");
7       else:
8           print("You are a teenage girl.");
9   else:
10      print("You are not a teenager.");
```

```
Enter your age: 18
Enter your gender (male/female): female
You are a teenage girl.
Enter you age: 10
Enter your gender (male/female): male
You are not a teenager.
```

برنامه بالا نحوه استفاده از عملگر منطقی and را نشان می‌دهد (خط ۴). وقتی به دستور if می‌رسید (خط ۴) برنامه سن شما را چک می‌کند. اگر سن شما بزرگتر از ۱۲ و کوچکتر از ۲۰ باشد (سنتان بین ۱۲ و ۲۰ باشد) یعنی مقدار هر دو True باشد سپس کدهای داخل بلوک if اجرا می‌شوند. اگر نتیجه یکی از شروط False باشد کدهای داخل بلوک else اجرا می‌شود. عملگر and عملوند سمت چپ را مورد بررسی قرار می‌دهد. اگر مقدار آن False باشد دیگر عملوند سمت راست را بررسی نمی‌کند و مقدار False را بر می‌گرداند. بر عکس عملگر or عملوند سمت چپ را مورد بررسی قرار می‌دهد و اگر مقدار آن True باشد سپس عملوند سمت راست را نادیده می‌گیرد و مقدار True را بر می‌گرداند.

```
if (x == 2 and y == 3):
   #Some code here
```

```
if (x == 2 or y == 3)
   #Some code here
```

نکته مهم اینجاست که شما می‌توانید از عملگرهای and و or به عنوان عملگر بیتی استفاده کنید. تفاوت جزئی این عملگرها وقتی که به عنوان عملگر بیتی به کار می‌روند این است که دو عملوند را بدون در نظر گرفتن مقدار عملوند سمت چپ مورد بررسی قرار می‌دهند. به عنوان مثال حتی اگر مقدار عملوند سمت چپ False باشد عملوند سمت چپ به وسیله عملگر بیتی and ارزیابی می‌شود. اگر شرطها را در برنامه ترکیب کنید استفاده از عملگرهای منطقی and و or به جای عملگرهای بیتی and و or بهتر خواهد بود. یکی دیگر از عملگرهای منطقی عملگر not است که نتیجه یک عبارت را خنثی یا منفی می‌کند. به مثال زیر توجه کنید:

```
if (not(x == 2))
   print("x is not equal to 2.");
```

اگر نتیجه عبارت x == 2 برابر False باشد عملگر not آن را True می‌کند.

عملگر شرطی

عملگر شرطی در پایتون مانند دستور شرطی if…else عمل می‌کند. در زیر نحوه استفاده از این عملگر آمده است:

```
condition_is_true if condition else condition_is_false
```

عملگر شرطی تنها عملگر سه تایی پایتون است که نیاز به سه عملوند دارد، یک مقدار زمانی که شرط درست باشد، شرط و یک مقدار زمانی که شرط نادرست باشد. اجازه بدهید که نحوه استفاده این عملگر را در داخل برنامه مورد بررسی قرار دهیم:

```
1  pet1 = "puppy";
2  pet2 = "kitten";
3
4  type1 = "dog" if (pet1 == "puppy" ) else "cat";
5  type2 = "cat" if (pet2 == "kitten") else "dog";
6
7  print(type1);
8  print(type2);
```

```
dog
cat
```

برنامه بالا نحوه استفاده از این عملگر شرطی را نشان می‌دهد. خط ۴ به این صورت ترجمه می‌شود که مقدار dog را در متغیر type1 قرار بده اگر مقدار pet1 برابر با puppy بود در غیر این صورت مقدار cat را type1 قرار بده. خط ۵ به این صورت ترجمه می‌شود که مقدار cat را در type2 قرار بده اگر مقدار pet2 برابر با kitten بود در غیر این صورت مقدار dog. حال برنامه بالا را با استفاده از دستور if else می‌نویسیم:

```
if (pet1 == "puppy"):
   type1 = "dog";
else:
   type1 = "cat";
```

هنگامی که چندین دستور در داخل یک بلوک if یا else دارید از عملگر شرطی استفاده نکنید چون خوانایی برنامه را پایین می‌آورد.

تکرار

ساختارهای تکرار به شما اجازه می‌دهند که یک یا چند دستور کد را تا زمانی که یک شرط برقرار است تکرار کنید. بدون ساختارهای تکرار شما مجبورید همان تعداد کدها را بنویسید که بسیار خسته کننده است. مثلاً شما مجبورید ۱۰ بار جمله "Hello World!" را تایپ کنید مانند مثال زیر:

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

البته شما می‌توانید با کپی کردن این تعداد کد را راحت بنویسید ولی این کار در کل کیفیت کدنویسی را پایین می‌آورد. راه بهتر برای نوشتن کدهای بالا استفاده از حلقه‌ها است. حلقه‌ها در پایتون عبارتند از:

- while
- for

حلقه While

ابتدایی‌ترین ساختار تکرار در پایتون حلقه While است. ابتدا یک شرط را مورد بررسی قرار می‌دهد و تا زمانیکه شرط برقرار باشد کدهای درون بلوک اجرا می‌شوند. ساختار حلقه While به صورت زیر است:

```
while(condition):

    code to loop;
```

می‌بینید که ساختار While مانند ساختار if بسیار ساده است. ابتدا یک شرط را که نتیجه آن یک مقدار بولی 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.

به یک متغیر شمارنده در داخل بدنه حلقه نیاز داریم. این شمارنده برای آزمایش شرط مورد استفاده قرار می‌گیرد و ادامه یا توقف حلقه به نوعی به آن وابسته است. این شمارنده را در داخل بدنه باید کاهش یا افزایش دهیم. در برنامه زیر نحوه استفاده از حلقه While آمده است:

```
counter = 1;

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

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

برنامه بالا ۱۰ بار پیغام Hello World! را چاپ می‌کند. اگر از حلقه در مثال بالا استفاده نمی‌کردیم مجبور بودیم تمام ۱۰ خط را تایپ کنیم. اجازه دهید که نگاهی به کدهای برنامه فوق بیندازیم. 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.

در خط ۳ حلقه while را وارد می‌کنیم. در حلقه while ابتدا مقدار اولیه شمارنده با ۱۰ مقایسه می‌شود که آیا از ۱۰ کمتر است یا با آن برابر است. نتیجه هر بار مقایسه ورود به بدنه حلقه While و چاپ پیغام است. همانطور که مشاهده می‌کنید بعد از هر بار مقایسه مقدار شمارنده یک واحد اضافه می‌شود (خط ۵). حلقه تا زمانی تکرار می‌شود که مقدار شمارنده از ۱۰ کمتر باشد.

اگر مقدار شمارنده یک بماند و آن را افزایش ندهیم و یا مقدار شرط هرگز false نشود یک حلقه بینهایت به وجود می‌آید. به این نکته توجه کنید که در شرط بالا به جای علامت > از => استفاده شده است. اگر از علامت > استفاده می‌کردیم کد ما ۹ بار تکرار می‌شد چون مقدار اولیه ۱ است و هنگامی که شرط به ۱۰ برسد false می‌شود چون ۱۰ > ۱۰ نیست. اگر می‌خواهید یک حلقه بی نهایت ایجاد کنید که هیچگاه متوقف نشود باید یک شرط ایجاد کنید که همواره درست (true) باشد:

```
while(True):

   #code to loop
```

این تکنیک در برخی موارد کارایی دارد و آن زمانی است که شما بخواهید با استفاده از دستورات break و return که در آینده توضیح خواهیم داد از حلقه خارج شوید.

حلقه for

یکی دیگر از ساختارهای تکرار حلقه for است. این حلقه عملی شبیه به حلقه while انجام می‌دهد. ساختار حلقه for به صورت زیر است:

```
for iterator_var in sequence:
   code to repeat;
```

iterator_var یک متغیر موقتی، in کلمه کلیدی و sequence هم یک سری مانند tuple، list 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. در زیر یک مثال از حلقهfor آمده است:

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

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

برنامه بالا اعداد ۱ تا ۱۰ را با استفاده از حلقه for می‌شمارد. ابتدا یک متغیر موقتی (i)، سپس کلمه کلیدی in و در آخر یک سری از اعداد که در اینجا یک list می باشد، تعریف می‌کنیم. کد اجرا می‌شود. 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:

```
for i in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10):
```

یا

```
for i in {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}:
```

حلقه های تو در تو (Nested Loops)

پایتون به شما اجازه می‌دهد که از حلقه‌ها به صورت تو در تو استفاده کنید. اگر یک حلقه در داخل حلقه دیگر قرار بگیرد، به آن حلقه تو در تو گفته می‌شود. در این نوع حلقه‌ها، به ازای اجرای یک بار حلقه بیرونی، حلقه داخلی به طور کامل اجرا می‌شود. در زیر نحوه ایجاد حلقه تو در تو آمده است:

```
for iterator_var in sequence:

    for iterator_var in sequence:

        statements(s)

        statements(s)
```

```
while expression:

    while expression:

        statement(s)

        statement(s)
```

نکته‌ای که در مورد حلقه‌های تو در تو وجود دارد این است که، می‌توان از یک نوع حلقه در داخل نوع دیگر استفاده کرد. مثلاً می‌توان از حلقه for در داخل حلقه while استفاده نمود. در مثال زیر نحوه استفاده از این حلقه‌ها ذکر شده است. فرض کنید که می‌خواهید یک مستطیل با ۳ سطر و ۵ ستون ایجاد کنید:

```
1  for i in (1, 2, 3, 4):
2      for j in (1, 2, 3, 4, 5):
3          print("*", end=' ');
4      print();
```

```
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
```

در کد بالا به ازای یک بار اجرای حلقه for اول (خط ۱)، حلقه for دوم (۳-۲) به طور کامل اجرا می‌شود. یعنی وقتی مقدار i برابر عدد ۱ می‌شود، علامت * توسط حلقه دوم ۵ بار چاپ می‌شود، وقتی i برابر ۲ می‌شود، دوباره علامت * پنج بار چاپ می‌شود و ... . در کل منظور از دو حلقه for این است که در ۴ سطر علامت * در ۵ ستون چاپ شود یا ۴ سطر ایجاد شود و در هر سطر ۵ بار علامت * چاپ شود. خط ۴ هم برای ایجاد خط جدید است. یعنی وقتی حلقه داخلی به طور کامل اجرا شد، یک خط جدید ایجاد می‌شود و علامت‌های * در خطوط جدید چاپ می‌شوند.

خارج شدن از حلقه با استفاده از continue،break و pass

گاهی اوقات با وجود درست بودن شرط می‌خواهیم حلقه متوقف شود. سؤال اینجاست که چطور این کار را انجام دهید؟ با استفاده از کلمه کلیدی break حلقه را متوقف کرده و با استفاده از کلمه کلیدی continue می‌توان بخشی از حلقه را رد کرد و به مرحله بعد رفت.

برنامه زیر نحوه استفاده از continue، breakو pass را نشان می‌دهد:

```
1  print("Demonstrating the use of break\n");
2
3  for x in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10):
4      if (x == 5):
5          break;
6
7      print("Number ", x);
8
9  print("\nDemonstrating the use of continue\n");
10
11 for x in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10):
12     if (x == 5):
13         continue;
14
15     print("Number ", x);
16
17 print("\nDemonstrating the use of pass\n");
18
19 for x in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10):
20     if (x == 5):
21         pass;
```

```
Demonstrating the use of break

Number  1
Number  2
Number  3
Number  4

Demonstrating the use of continue

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

Demonstrating the use of pass
```

در این برنامه از حلقه for برای نشان دادن کاربرد دو کلمه کلیدی فوق استفاده شده است اگر به جای for از حلقه while استفاده می‌شد نتیجه یکسانی به دست می‌آمد. همانطور که در شرط برنامه (خط ۴) آمده است، وقتی که مقدار x به عدد ۵ برسد، سپس دستور break اجرا (خط ۵) و حلقه بلافاصله متوقف می‌شود، حتی اگر شرط x < 10 برقرار باشد. از طرف دیگر در خط ۲۱ حلقه for فقط برای یک تکرار

خاص متوقف شده و سپس ادامه می‌یابد. وقتی مقدار x برابر ۵ شود حلقه از ۵ رد شده و مقدار ۵ را چاپ نمی‌کند و بقیه مقادیر چاپ می‌شوند.

ممکن است این سوال برایتان پیش آمده باشد که کاربرد کلمه pass چیست؟ از این دستور زمانی استفاده می کنیم که در شرایطی خاص نیاز به انجام هیچ کاری نباشد! مثلا برای تعریف یک تابع خالی تا بعدا کدهای آن نوشته شود. یا زمانی که بخواهیم همانند مثلا بالا، کدهای بدنه یک دستور شرطی و یا حلقه را بعدا بنویسیم، به کار می رود. حال شما برای درک بهتر، کلمه pass را از کد بالا حذف کرده و کد را اجرا کنید. مشاهده می کنید که به شما پیغام خطا نمایش داده می شود و از شما می خواهد که بدنه دستور if و for را مشخص کنید ولی اگر کلمه pass را دوباره بنویسید، این خطا نادیده گرفته و کد اجرا می شود.

برای مشاهده سایر مطالب کتاب به آدرس زیر مراجعه فرمایید

W3-farsi.com