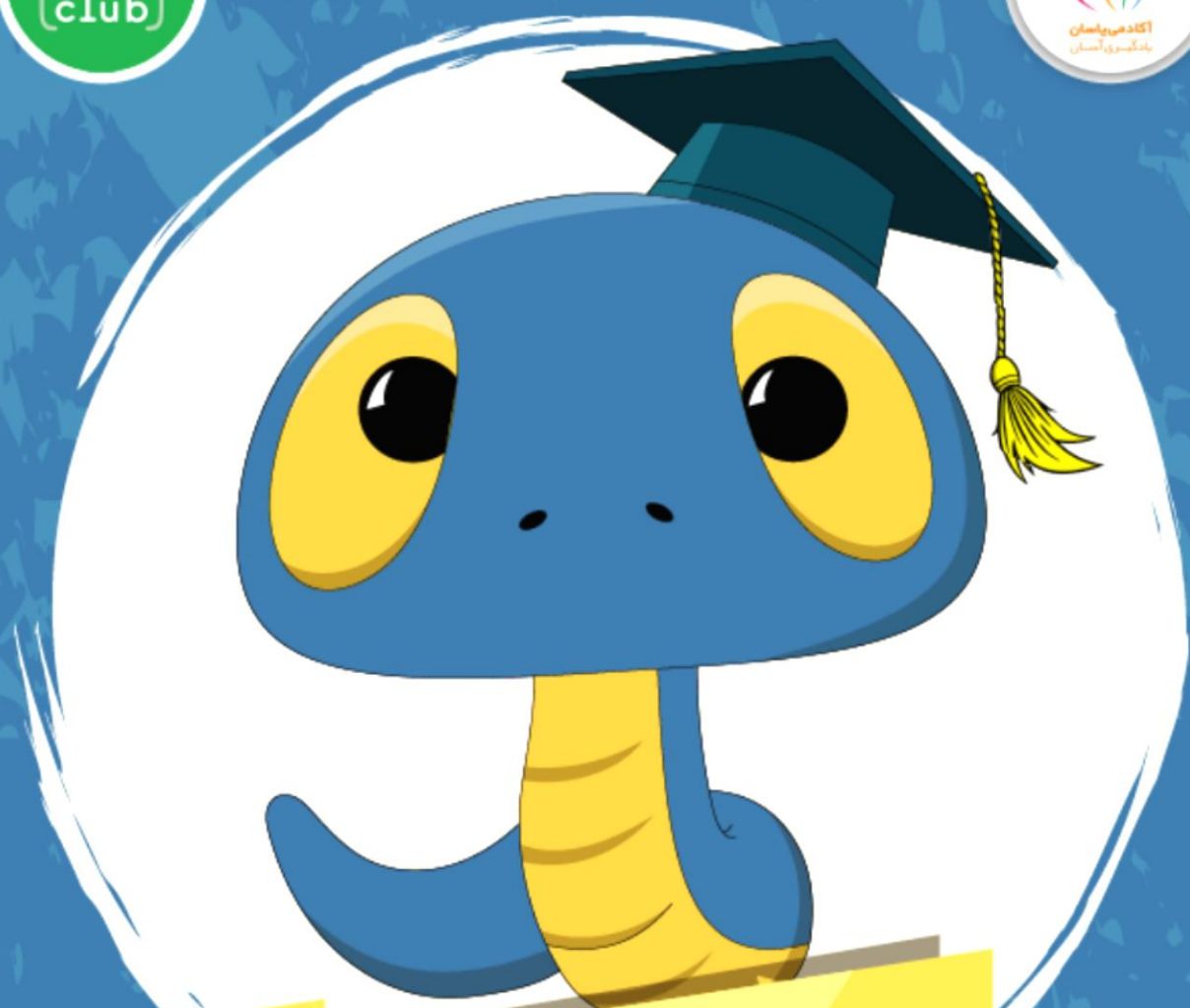




پایتون

مقدماتی

فهرست:

۱.....	آشنایی با پایتون و خصوصیات آن.....
۲.....	آشنایی با محیط IDLE.....
۲.....	کد نویسی تعاملی ::.....
۳.....	کد نویسی در حالت اسکریپتی ::.....
۴.....	مزایای استفاده از یک IDLE برای نوشتن برنامه:
۵.....	ذخیره و ایجاد یک برنامه:
۵.....	اجرای یک برنامه نوشته شده:
۵.....	باز نمودن برنامه‌های ذخیره شده:
۶.....	آشنایی با انواع داده یا Data Types ::.....
۶.....	داده‌های عددی صحیح:
۶.....	داده‌های عددی اعشاری:
۷.....	داده‌های متنی :
۷.....	آشنایی با عملگرها:.....
۷.....	عملگرهای ریاضی
۸.....	عملگرهای مقایسه‌ای:
۱۰.....	ترتیب انجام عملیات
۱۱.....	متغیر یا Variable ::.....
۱۱.....	تعریف متغیر
۱۱.....	نحوه مقدار دهی به متغیرها
۱۳.....	قوانین نام گذاری متغیرها
۱۴.....	استفاده از متغیرها
۱۵.....	دستور print().....
۱۶.....	خطاهای رایج
۱۶.....	نوشتن اولین برنامه.....
۱۷.....	نوشتن یک برنامه محاسباتی
۱۷.....	نوشتن دو عبارت با دستور print ().....
۱۸.....	آنچه در هفته بعد خواهید خواند.....



آشنایی با پایتون و خصوصیات آن

پایتون یکی از پرطرفدارترین زبان‌های برنامه‌نویسی حال حاضر موجود به شمار می‌آید؛ این زبان برنامه‌نویسی بخاطر خصوصیات جالب و توانایی‌هایش، هم میان مبتدیان و هم میان برنامه‌نویسان حرفه‌ای از محبوبیت زیادی برخوردار است. این توانایی‌ها در حدی هستند که شرکت‌هایی مانند Google و YouTube از این زبان برای نوشتن برنامه‌های خود استفاده می‌کنند. برخی از مهم‌ترین این خصوصیات عبارتند از:

- سادگی دستورات:

دستورات زبان پایتون بسیار ساده بوده و به راحتی قابل فهم است به شکلی که یک برنامه نوشته‌شده با زبان پایتون به یک متن انگلیسی ساده و روان بسیار نزدیک است. این خصوصیت بسیار مهم است چرا که یک برنامه‌نویس بجای پرداختن به جزئیات مربوط به نحوه‌ی نوشتن صحیح دستورات زبان برنامه‌نویسی، بیشتر حواس خود را بر روی حل مسئله متمرکز می‌سازد. کم بودن جزئیات دست و پاگیر در دستورات این زبان، خود باعث کاهش چشم‌گیر اشتباهات نوشتاری از طرف برنامه‌نویسان بخصوص برنامه‌نویسان مبتدی خواهد شد.

- دارا بودن کتابخانه‌های بسیار متنوع :

کتابخانه یا Library، به مجموعه کدهایی گفته می‌شود که برای انجام کار مشخصی، نوشته شده‌اند تا سایر برنامه‌نویسان برای انجام آن کارها دیگر نیازی به نوشتن مجدد آن کدها نداشته باشند و تنها با فراخوانی کتابخانه مرتبط در برنامه‌ی خود، آن کارها را انجام دهند.

به عبارت ساده‌تر، کتابخانه‌ها آن دسته از کدهایی هستند که توسط توسعه‌دهندگان و برنامه‌نویسان خبره نوشته شده و به محیط‌های برنامه‌نویسی اضافه شده‌اند تا برنامه‌نویسان دیگر در حین کدنویسی قادر به استفاده و به کارگیری آن‌ها باشند.

وجود این کتابخانه‌ها باعث شده است تا بتوان از زبان پایتون در بسیاری از موارد مانند هوش مصنوعی، رمزنگاری، مرورگرهای وب، ایمیل، بازی‌سازی، داده کاوی، توسعه‌ی وب‌سایت و ... به خوبی استفاده نمود. این امر باعث شده است تا از زبان پایتون در ساخت موتور جستجوی گوگل و یوتیوب، ساخت برنامه‌های علمی در سازمان فضایی ناسا، بخشی از سرویس ایمیل یاهو و نیز در تست سخت‌افزار در کمپانی‌های اینتل، آی‌بی‌ام و سیسکو استفاده شود.

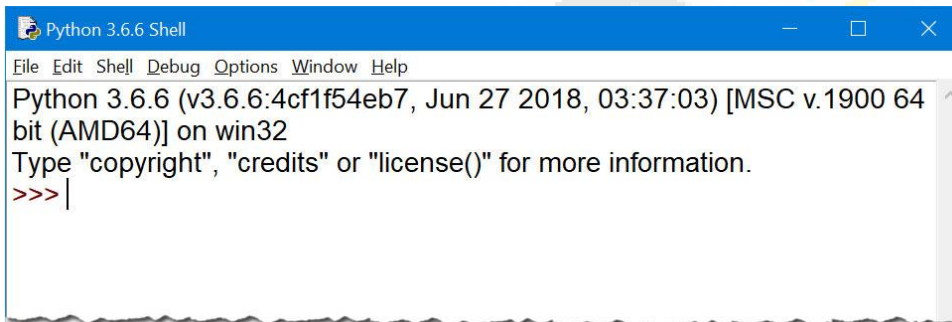


آشنایی با محیط IDLE

پس از نصب نرم افزار Python به روش‌های مختلفی می‌توان دستورات آن را نوشت و اجرا نمود. اما شاید در این مرحله استفاده از محیط IDLE مناسب‌ترین روش برای شروع به کار با این زبان برنامه نویسی باشد.

IDLE^۱ یا **محیط توسعه یکپارچه مخصوص آموزش** هم زمان با نصب برنامه پایتون در کامپیوتر نصب شده و به برنامه نویسان در نوشتن، ویرایش، پیدا کردن خطاها، اجرای برنامه و نمایش خروجی کمک می‌کند. برای اجرای این برنامه کافیست بعد از نصب پایتون روی سیستم، عبارت IDLE را در Start Menu جستجو و اجرا نمایید.

با اجرای این دستور پنجره‌ای مطابق شکل زیر ظاهر می‌شود که در بالای آن عبارت Shell نوشته شده است و به همین نام نیز خوانده می‌شود. در خطوط اول این پنجره، اطلاعاتی از



برنامه مانند شماره‌ی نگارش یا Version آن به کاربر نشان داده می‌شود.

کد نویسی تعاملی :

به دو روش مختلف می‌توان دستورات پایتون را اجرا نمود. یکی از آنها حالت تعاملی یا Interactive Mode است که در همین پنجره Shell انجام می‌شود. از این حالت زمانی استفاده می‌شود که بخواهید یک دستور یا مجموعه کد کوچکی را با سرعت اجرا کرده و نتیجه یا خروجی آن را در لحظه بررسی نمایید.

در این پنجره علامتی به شکل ">>>" در ابتدای خط دیده می‌شود که محل وارد کردن دستور توسط کاربر است. به این محل Command Prompt گفته می‌شود. پس از نوشتن دستور در Command Prompt، می‌توان آن را با فشردن کلید Enter اجرا و نتیجه‌ی اجرای آن را (در صورتی که خروجی داشته باشد) در سطر زیر آن مشاهده نمود.

در نوشتن و اجرای دستورات در پنجره Shell باید به نکات زیر توجه نمود:

¹ Integrated Development Learning Environment



- همانطور که قبلاً نیز گفته شد، دستوراتی که در این محیط نوشته می‌شوند بلافاصله اجرا می‌گردند. به عنوان مثال در این شکل با وارد کردن عبارت محاسباتی $2 + 2$ و فشردن

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64
bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>>
```

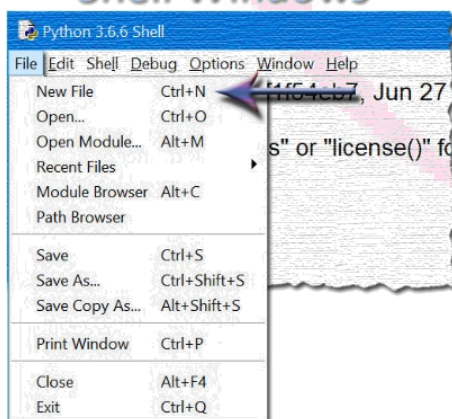
کلید Enter مقدار این عبارت محاسبه شده و عدد 4 که جواب آن است، ظاهر می‌شود.

- این دستورات پس از نوشته و اجرا شدن، دیگر قابل ویرایش و اجرای مجدد نیستند. البته می‌توان آنها را کپی و برای اجرای مجدد در مقابل Prompt جدیدی که ظاهر شده است Paste نمود.
- با بسته شدن این پنجره دیگر امکان دسترسی به دستورات آن وجود ندارد. یعنی دستوراتی که در این پنجره نوشته می‌شود قابل ذخیره و استفاده‌ی مجدد نیستند.

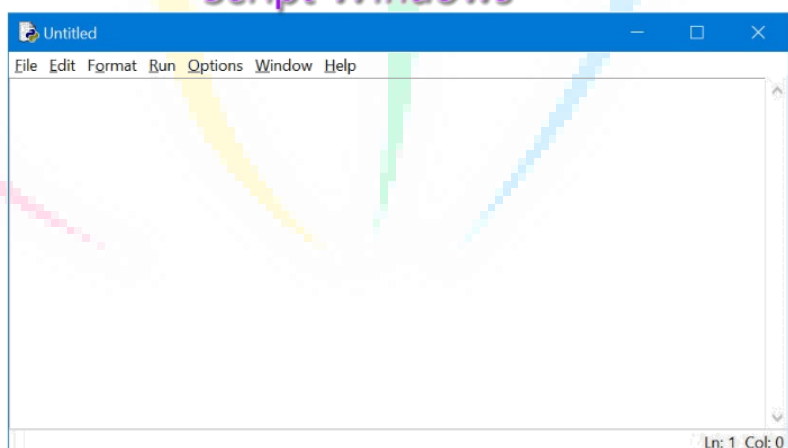
کد نویسی در حالت اسکرپتی :

با توجه به اینکه در پنجره Shell نمی‌توان چند خط کد را برای استفاده مجدد ذخیره نمود، پنجره دیگری به نام Edit Windows وجود دارد که به Script Windows نیز شناخته می‌شود. برای دسترسی به این پنجره لازم است از منوی File در پنجره Shell دستور New Windows را انتخاب نماییم. با انجام این عمل پنجره جدیدی باز می‌شود که در نوار عنوان آن عبارت Untitled نوشته شده است و اگر دقت کنید Menu bar آن با Menu bar پنجره Shell تفاوت دارد و علامت Command Prompt نیز در آن مشاهده نمی‌شود.

Shell Windows



Script Windows





در نوشتن و اجرای دستورات در این پنجره باید به این نکته توجه نمود که دستورات نوشته شده در این پنجره با فشار کلید Enter در انتهای آنها اجرا نشده و فشار این کلید تنها باعث رفتن به سطر بعد می‌شود.

مزایای استفاده از یک IDLE برای نوشتن برنامه:

شما می‌توانید دستورات برنامه خود را در برنامه‌های واژه پرداز ساده مثل Notepad نیز نوشته و ذخیره نمایید. اما استفاده از یک محیط IDLE مزایایی دارد که تعدادی از آنها عبارتند از:

- در محیط IDLE عباراتی که به عنوان مؤلفه‌های پایتون شناخته می‌شوند، با رنگ‌های مختلف نشان داده می‌شوند که در زمان تایپ برنامه و نیز در زمان رفع خطا برای کاربر بسیار مفید بوده و برنامه نویسی را برای او آسانتر و جذابتر می‌کند.

```
# Draw a Shape

from turtle import *
print("Academy Yasan")
for i in range(400):
    forward(i)
    left(91)
```

- در صورت وجود اشکال در دستورات نوشته شده در برنامه، در زمان اجرای برنامه آن اشکال با علامت قرمز به کاربر نشان داده می‌شود. به عنوان مثال همانطور که مشاهده می‌کنید، در تصویر مقابل یک حرف از یکی از دستورات برنامه تصویر بالا حذف شده است و IDLE در زمان اجرا یک پیغام خطا ظاهر کرده و عبارت غلط را نیز برای کاربر مشخص کرده است.

```
# Draw a Shape

from turtle import *
print("Academy yasan")
or i in range(400):
    forward(i)
    left(91)
```

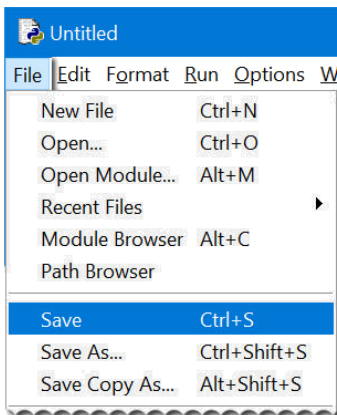
- بعضی از دستورات موجود در پایتون دارای پارامترهای مختلف و متعددی هستند که معمولاً در محیط‌های IDLE با رسیدن کاربر به این پارامترها، یک کادر کوچک

```
print(|
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

ظاهر می‌شود و کاربر را در وارد کردن مقادیر پارامترها راهنمایی می‌کند.



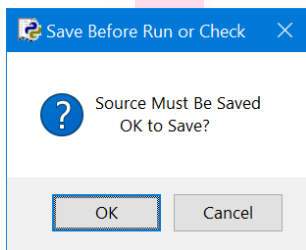
ذخیره و ایجاد یک برنامه:



مجموعه دستوراتی که در یک پنجره نوشته می‌شوند را می‌توان به صورت یک فایل ذخیره نمود که به آن برنامه می‌گویند. برای انجام این کار مانند هر نرم افزار دیگری از منوی File دستور Save را انتخاب می‌کنیم و در پنجره Save، نام فایل و محل ذخیره فایل را تعیین می‌کنیم. پسوندی که برای این فایل به صورت پیش فرض انتخاب می‌شود .py است.

اجرای یک برنامه نوشته شده:

برای اجرای دستوراتی که در پنجره Script نوشته می‌شود، باید از منوی Run گزینه Run Module را انتخاب کرد. البته روش ساده‌تر، استفاده از کلید F5 است که کلید میان‌بر آن بشمار می‌آید.



نکته اول: قبل از اجرای هر کدی که نوشته می‌شود ابتدا باید آن را Save نمود. در صورتی که قبل از اجرا، کد تغییر کرده باشد و Save نشده باشد، با فشردن کلید F5 ابتدا کادری ظاهر می‌گردد که از کاربر می‌خواهد برنامه را Save نماید و پس از ذخیره‌سازی، برنامه اجرا می‌شود.

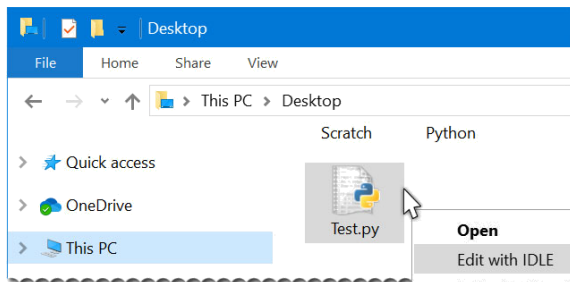
نکته دوم: نتیجه اجرای دستورات نوشته شده در پنجره Script در پنجره Shell مشاهده می‌شود. یعنی با فشردن کلید F5 پنجره Shell باز شده و خروجی یا نتیجه اجرای برنامه را به کاربر نشان می‌دهد.

نکته سوم: دستورات نوشته شده پنجره Script از بالا به پایین و به ترتیب اجرا می‌شوند.

باز نمودن برنامه‌های ذخیره شده:

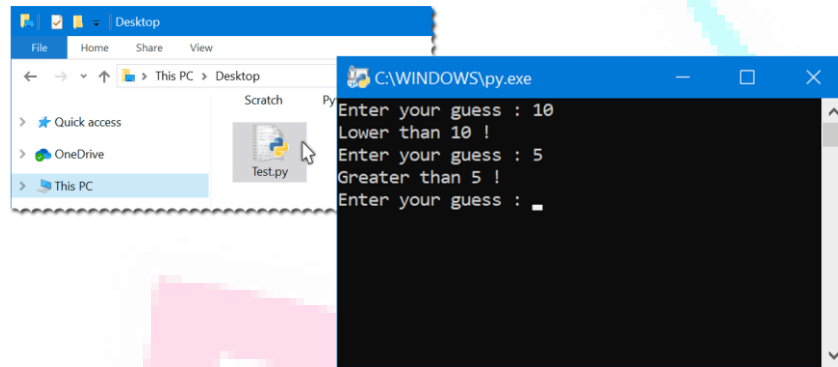
برای باز کردن یک فایل .py که در حافظه جانبی ذخیره شده است، به دو روش می‌توان عمل نمود.

- اگر یکی از پنجره‌های محیط IDLE باز باشد، از منوی File دستور Open را اجرا و فایل مورد نظر خود را انتخاب کنیم.



- در زمانی که پنجره‌ای از IDLE باز نیست، شاید بهترین راه برای باز کردن فایل‌های برنامه، انجام عمل کلیک راست بر روی فایل و انتخاب دستور Edit with IDLE از منوی باز شده باشد.

توجه داشته باشید که در این حالت انجام عمل Double Click بر روی این فایل‌ها آنها را در محیط IDLE باز نمی‌کند و تنها باعث اجرا شدن آنها می‌شود. البته نه در پنجره Shell بلکه در پنجره سیاه رنگی به نام ترمینال که بلافاصله بعد از خاتمه اجرای دستورات بسته می‌شود.



پس از این مقدمه و آشنایی با IDLE بعنوان محیط برنامه نویسی پایتون، وارد فضای برنامه‌نویسی شده و با مفاهیم برنامه‌نویسی آشنا می‌شویم.

آشنایی با انواع داده یا Data Types :

بطور کلی می‌توان گفت هر برنامه مجموعه‌ای از داده‌ها را دریافت کرده، مطابق با دستوراتی که در برنامه نوشته شده است، داده‌ها را پردازش کرده و نتایج بدست آمده را در اختیار کاربر قرار می‌دهد. به همین دلیل قبل از انجام هر عملی لازم است با انواع داده‌هایی که در برنامه نویسی از آنها استفاده می‌شود، آشنا شویم.

داده‌های عددی صحیح:

این داده‌ها همانطور که از اسمشان مشخص است، شامل اعداد صحیح مثبت، منفی و صفر می‌شوند. به این اعداد Integer نیز گفته می‌شود.

داده‌های عددی اعشاری:

این داده‌ها نیز شامل اعدادی می‌شوند که یک نقطه اعشار در آنها وجود داشته باشد. مانند 4.5 به این اعداد Float گفته می‌شود.



داده‌های متنی :

داده‌های متنی شامل حروف، کلمات و جملاتی هستند که ما برای بیان مطالب خود از آنها استفاده می‌کنیم. این نوع از داده‌ها برای کامپیوتر قابل درک نیستند. بنابراین به صورت دنباله‌ای از کاراکترها از طریق صفحه کلید به کامپیوتر داده می‌شود. به عنوان مثال یک برنامه در زمان اجرا ممکن است نام، نام خانوادگی، آدرس و اطلاعات دیگری از کاربر بخواهد که داده‌هایی از نوع متنی به شمار می‌آیند. یا ممکن است لازم باشد عبارتی را به کاربر نشان دهد مانند « **لطفا نام کاربر خود را وارد نمایید** ». به این نوع داده‌ها String نیز گفته می‌شود. اگر لازم باشد در داخل برنامه از این نوع داده‌ها استفاده شود باید توجه داشت که حتما این داده‌ها را در داخل دو علامت " (گیومه) قرار دهیم.

آشنایی با عملگرها:

در برنامه‌ها برای انجام عملی پردازش بر روی داده‌ها از عملگرها استفاده می‌شود مثل جمع، تفریق، ضرب، تقسیم و مقایسه. در این قسمت به دو گروه اصلی از عملگرها اشاره می‌کنیم.

عملگرهای ریاضی

شاید پرکاربردترین عملگرها، عملگرهای ریاضی باشند که عبارتند از:

جمع : با علامت « + » نشان داده می‌شود و داده‌های عددی دو طرف خود را با یکدیگر جمع می‌کند. به مثال‌هایی از نحوه عمل این عملگر در محیط Shell نگاه کنید.

همانطور که در شکل مقابل مشاهده می‌کنید اگر در محیط Shell و در مقابل Prompt یک

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> 2 + 3
5
>>> 2.5 + 1.5
4.0
>>> 2 + 2.5
4.5
Ln: 9 Col: 4
```

عمل ریاضی نوشته شود، پس از فشردن کلید Enter جواب آن عبارت محاسبه و چاپ خواهد شد.

- در مثال اول دو عدد صحیح با یکدیگر جمع شده و حاصل نیز یک عدد صحیح است.

- در مثال دوم دو عدد اعشاری با یکدیگر جمع شده و حاصل نیز یک عدد اعشاری است.
- در مثال سوم یک عدد صحیح با یک عدد اعشاری جمع شده و حاصل یک عدد اعشاری است.



```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> 5 - 3.5
1.5
>>> 2 * 4
8
>>> 10 / 3
3.3333333333333333
>>> 2 ** 3
8
Ln: 9 Col: 4
```

تفریق : با علامت « - » نشان داده می‌شود و داده‌ی سمت راست را از داده‌ی سمت چپ کم می‌کند.

ضرب : با علامت « * » نشان داده می‌شود و داده‌های دو طرف خود را در یکدیگر ضرب می‌کند.

تقسیم : با علامت « / » نشان داده می‌شود

و داده‌ی سمت چپ را بر داده‌ی سمت راست تقسیم می‌کند. لازم به ذکر است که نتیجه عمل تقسیم، همواره یک عدد اعشاری است و نقطه اعشار دارد. حتی اگر این تقسیم باقیمانده‌ای نداشته باشد. برای مثال نتیجه تقسیم 8 بر 2، برابر 4.0 است نه عدد 4.

توان : با علامت « * * » نشان داده می‌شود و داده‌ی سمت چپ را به توان داده‌ی سمت راست می‌رساند. یعنی عدد سمت چپ را به تعداد عدد سمت راست در خودش ضرب می‌کند. به عنوان مثال اگر بخواهیم عدد ۲ را سه بار در خودش ضرب کنیم بجای نوشتن $2 \times 2 \times 2$ عبارت 2^3 را در ریاضیات می‌نویسم و در پایتون آن را بصورت $2 ** 3$ نشان می‌دهیم.

یک نکته جالب : از عملگر ضرب می‌توان برای ضرب یک عدد در یک عبارت متنی نیز استفاده

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> 2 * "Bye "
'Bye Bye '
Ln: 56 Col: 4
```

کرد و در این حالت عبارت متنی به تعداد عددی که در آن ضرب شده است پشت سر هم تکرار می‌شود. همانطور که در شکل مقابل مشاهده می‌کنید، عدد ۲ در عبارت

Bye ضرب شده است و در نتیجه آن عبارت Bye Bye به وجود آمده است. به این نکته توجه نمایید که عبارت Bye چون یک داده از نوع متنی است در داخل دو علامت " نوشته شده است.

عملگرهای مقایسه‌ای:

در بسیاری از موارد لازم است در زمان پردازش، داده‌ها را با یکدیگر مقایسه کرد. به عنوان مثال می‌خواهیم بدانیم آیا دو عدد با یکدیگر برابر هستند؟ آیا عدد اول بزرگتر از عدد دوم است؟ و ... برای انجام این نوع پردازش‌ها از عملگرهای مقایسه‌ای استفاده می‌شود.



```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> 2 == 2
True
>>> 2+3 == 5
True
>>> 4 == 3
False
>>> 3-1 == 4
False
Ln: 33 Col: 4
```

تساوی : با علامت « == » نشان داده می‌شود و برابری عبارات دو طرف خود را بررسی می‌نماید و در صورت برابری آنها، عبارت **True** به معنی **درست** و در صورت برابر نبودن آنها، عبارت **False** به معنی **نادرست** نشان داده می‌شود.

بعنوان مثال در این شکل به علت آنکه نتیجه عبارت 2+3 با عدد 5 برابر است، عبارت **True** به دلیل آنکه 4 با 3 برابر نیست عبارت **False** نشان داده شده است.

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> 2 = 2
SyntaxError: invalid syntax
Ln: 56 Col: 4
```

نکته : باید توجه کرد در عملگرهایی که از دو کاراکتر ساخته شده‌اند نباید بین این دو کاراکتر فاصله یا Space وجود داشته باشد چرا که در صورت وجود فاصله، مطابق شکل

فوق پیغام خطای ^۲Syntax ظاهر می‌شود. همان طور که مشاهده می‌کنید در این حالت با فشردن کلید Enter علاوه بر ظاهر شدن پیغام خطا، قسمتی از دستور که در آنجا اشکال وجود دارد، با رنگ قرمز مشخص می‌شود.

نامساوی : با علامت « != » نشان داده می‌شود و **برابر نبودن** عبارات دو طرف خود را با یکدیگر بررسی می‌نماید. برخلاف عملگر تساوی، اگر دو عبارت با یکدیگر برابر نباشند عبارت **True** و اگر برابر باشند عبارت **False** نشان داده می‌شود.

به عنوان مثال در شکل مقابل و در مثال اول پایتون می‌پرسد آیا 2 مخالف 2 است؟ مسلماً

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> 2 != 2
False
>>> 2+3 != 5
False
>>> 4 != 3
True
>>> 3-1 != 4
True
Ln: 45 Col: 4
```

پاسخ **False** باید باشد چون مخالف یکدیگر نیستند. در مثال دوم می‌پرسد آیا نتیجه 2+3 مخالف 5 است؟ باز هم پاسخ **False** باید باشد چون مخالف یکدیگر نیستند. اما در مثال سوم که می‌پرسد آیا 4 مخالف 3 است؟ پاسخ باید **True** باشد چون مخالف یکدیگر هستند. در مثال چهارم هم همینطور. چون نتیجه 3-1 مخالف 4 است، پاسخ باید **True** باشد.

^۲ خطای سینتکس یا خطای نحوی، به معنی خطا در نحوه نگارش دستور زبان است.



```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help

>>> 2+3 < 6
True

>>> 8 < 2 ** 2
False

>>> 5 <= 5
True
```

کوچکتر : با علامت « < » نشان داده می‌شود و کوچکتر بودن عبارت سمت چپ را با عبارت سمت راست مقایسه می‌کند و در صورت کوچکتر بودن عبارت **True** و در غیر این صورت عبارت **False** ظاهر می‌شود.

کوچکتر و مساوی : با علامت « <= » نشان داده می‌شود و کوچکتر یا مساوی بودن عبارت سمت چپ را با عبارت سمت راست مقایسه می‌کند.

بزرگتر : با علامت « > » نشان داده می‌شود و بزرگتر بودن عبارت سمت چپ را با عبارت سمت راست مقایسه می‌کند و در صورت بزرگتر بودن عبارت **True** و در غیر این صورت عبارت **False** ظاهر می‌شود.

بزرگتر و مساوی : با علامت « >= » نشان داده می‌شود و بزرگتر یا مساوی بودن عبارت سمت چپ را با عبارت سمت راست مقایسه می‌کند.

ترتیب انجام عملیات

زمانی که در یک عبارت ریاضی چند عملگر وجود داشته باشد، مسلم است که عملیات ریاضی همیشه از سمت چپ به راست انجام می‌شود. اما کار به این سادگی‌ها هم نیست چرا که اولویت اجرای عملگرها نسبت به یکدیگر متفاوت است.

$$4 + 3 * 2$$

Diagram showing operator precedence for the expression $4 + 3 * 2$. The multiplication operation ($3 * 2$) is grouped together with a bracket labeled '1', indicating it is performed first. The result of this operation (6) is then added to 4, indicated by a bracket labeled '2'.

به عنوان مثال در عبارت $4 + 3 * 2$ ابتدا عدد 3 در 2 ضرب شده و جواب آن یعنی عدد 6، با 4 جمع می‌شود که نتیجه آن عدد 10 است.

یعنی می‌توان نتیجه گرفت ضرب نسبت به جمع از اولویت بالاتری برخوردار است و با وجود آنکه در سمت راست قرار گرفته، اما اول انجام می‌شود.

در جدول زیر انواع عملیات به ترتیب اولویت نوشته شده اند:

۱- پرانتز : ()

۲- توان : **

۳- ضرب و تقسیم : * / (نسبت به هم اولویتی ندارند و هرکدام که سمت چپ قرار داشته باشد اول انجام می‌شود.)

۴- جمع و تفریق : + - (نسبت به هم اولویتی ندارند و هرکدام که سمت چپ قرار داشته باشد اول انجام می‌شود.)



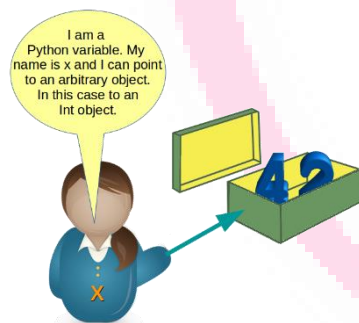
چند مثال :

عبارت ریاضی	معادل در برنامه نویسی	نتیجه	توضیحات
$2 + 3^2$	$ \begin{array}{c} 2 + 3^{**} 2 \\ \quad \quad \quad \boxed{1} \\ \quad \quad \quad \quad \quad \boxed{2} \end{array} $	11	ابتدا عمل توان و بعد از آن عمل جمع انجام می‌شود
$2 \times 3 + \frac{4}{2}$	$ \begin{array}{c} 2 * 3 + 4 / 2 \\ \quad \quad \quad \boxed{1} \quad \quad \quad \boxed{2} \\ \quad \quad \quad \quad \quad \quad \quad \quad \boxed{3} \end{array} $	8.0	ابتدا عمل ضرب بعد از آن عمل تقسیم و در انتها عمل جمع انجام می‌شود.
$\frac{12 + 3}{5 - 2}$	$ \begin{array}{c} (12 + 3) / (5 - 2) \\ \quad \quad \quad \boxed{1} \quad \quad \quad \boxed{2} \\ \quad \quad \quad \quad \quad \quad \quad \quad \boxed{3} \end{array} $	5.0	ابتدا پرانتز سمت چپ بعد از آن پرانتز سمت راست و در انتها عمل تقسیم انجام می‌شود.
$\frac{8 \times 5}{5 \times 2}$	$ \begin{array}{c} 8 * 5 / (5 * 2) \\ \quad \quad \quad \boxed{2} \quad \quad \quad \boxed{1} \\ \quad \quad \quad \quad \quad \quad \quad \quad \boxed{3} \end{array} $	4.0	ابتدا پرانتز بعد از آن عمل ضرب و در انتها عمل تقسیم انجام می‌شود.
$\frac{8 \times 5}{5} \times 2$	$ \begin{array}{c} 8 * 5 / 5 * 2 \\ \quad \quad \quad \boxed{1} \quad \quad \quad \boxed{2} \\ \quad \quad \quad \quad \quad \quad \quad \quad \boxed{3} \end{array} $	16.0	ابتدا عمل ضرب سمت چپ بعد عمل تقسیم و در انتها عمل ضرب سمت راست انجام می‌شود.

متغیر یا Variable :

در زمان اجرای برنامه معمولاً لازم است داده‌ها و نتیجه حاصل از پردازش داده‌ها به شکلی نگهداری شود تا در روند اجرای برنامه مورد استفاده قرار گیرند. برای انجام این کار از مفهومی به نام متغیر استفاده می‌شود.

تعریف متغیر



متغیر در حقیقت خانه‌ای از حافظه است که می‌تواند داده‌ها را در خود نگهداری نماید. هر متغیر دارای یک نام مشخص است که در برنامه از طریق این نام مورد استفاده قرار می‌گیرد.

نحوه مقدار دهی به متغیرها

برای قرار دادن داده‌ای در یک متغیر از علامت « = » استفاده می‌شود که به آن « **عملگر انتساب** » نیز گفته می‌شود. در سمت چپ این عملگر، نام متغیر و در سمت راست آن داده‌ای نوشته می‌شود که می‌خواهیم در متغیر قرار گیرد.



```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> a = 2
>>> b = 2 + 3
>>> c = a - 1
>>> d = "Python"
```

در شکل مقابل، چهار مثال از نحوه مقدار دهی به متغیرها نشان داده شده است.

در مثال اول عدد 2 در متغیر a قرار می‌گیرد. یعنی از این به بعد در هر قسمتی از برنامه اگر از متغیر a استفاده شود، برنامه عدد 2 را برای آن در نظر خواهد گرفت.

در مثال دوم در سمت راست علامت « = » یک عبارت ریاضی نوشته شده است. در این حالت ابتدا عبارت سمت راست محاسبه شده و نتیجه آن که در این مثال عدد 5 است در متغیر b قرار می‌گیرد.

در مثال سوم در سمت راست علامت « = » یک عبارت ریاضی نوشته شده است که در آن از یک متغیر دیگر استفاده شده. در این حالت برنامه پایتون برای محاسبه عبارت ریاضی از مقدار متغیر فراخوانی شده استفاده می‌کند که در اینجا مقدار a برابر با 2 است و در نتیجه حاصل تفریق یعنی عدد ۱، در متغیر c قرار می‌گیرد.

در مثال چهارم، یک عبارت متنی یعنی کلمه Python در متغیر d قرار داده شده است. توجه می‌کنید که کلمه Python در این مثال در داخل دو علامت « " » قرار گرفته است تا برنامه متوجه شود این عبارت، یک عبارت متنی بوده و نام یک متغیر دیگر نیست. یعنی اگر بنویسیم d = Python منظورمان این است که متغیر d برابر است با مقدار متغیری به نام Python.

نکته: در صورتی که در سمت **راست** علامت « = » از متغیری استفاده شده باشد، حتماً آن متغیر باید در خطوط بالاتر مقدار دریافت کرده باشد در غیر این صورت، اجراء برنامه با رسیدن به متغیری که مقدار ندارد، متوقف شده و یک پیغام خطا مطابق شکل زیر ظاهر می‌گردد و معنی آن این است که متغیری با این نام تعریف یا **define** نشده است. در مثال زیر برای متغیر m در خطوط بالاتر مقداری تعیین نشده است.

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> c = m - 1
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    c = m - 1
NameError: name 'm' is not defined
Ln: 56 Col: 4
```



قوانین نام گذاری متغیرها

- در نام گذاری متغیرها معمولاً از حروف کوچک استفاده می‌شود.
- نام هر متغیری حتماً باید با یک حرف یا علامت underline (_) شروع شود.
- در نام گذاری متغیرها می‌توان از اعداد نیز استفاده نمود به شرط آنکه در ابتدای نام متغیر قرار نگیرند.
- در داخل نام یک متغیر نمی‌توان از فاصله یا Space استفاده نمود. (به همین صورت نمی‌توان از علائمی مانند @، #، \$، %، ! استفاده نمود)
- اگر نام متغیر از دو کلمه تشکیل شده باشد و با توجه به اینکه امکان استفاده از Space برای جدا کردن دو کلمه از یکدیگر وجود ندارد، می‌توان برای خوانا شدن نام متغیر، بین دو کلمه از علامت « _ » که یک کاراکتر مجاز در نام گذاری متغیرها است استفاده نمود یا حروف اول هر کلمه (بجز کلمه اول) را به صورت بزرگ نوشت. مثلاً اگر بخواهیم نام یک متغیر را Age Number بگذاریم، می‌توان آن را به یکی از دو روش زیر نوشت.

age_number
ageNumber

نکته : یک قرارداد بین برنامه‌نویسان دنیا وجود دارد که در نوشتن نام متغیرها، معمولاً حروف اول همه کلمات بجز کلمه اول، با حروف بزرگ نوشته می‌شود. به این شکل نوشتن اصطلاحاً نگارش شتری یا Camel Case گفته می‌شود.



- در Python هم مانند سایر زبان‌ها، تعدادی کلمات کلیدی از پیش تعریف شده وجود دارد که به آنها **keywords** گفته می‌شود و برای زبان شناخته شده است. برای همین برنامه‌نویس نمی‌تواند از این کلمات، به عنوان نام متغیر استفاده کند. بعضی از این کلمات عبارتند از : True , False , not , if , and , ...

نمونه‌ای از نام گذاری‌های صحیح :

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> i = 2
>>> Color2 = 255
>>> ID_Number = 123456
>>> UserName = "Yasan"
>>> mobileNumber = 12345
Ln: 45 Col: 4
```



نمونه‌ای از نام گذاری‌های غلط :

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> 1a
>>> Id.Number
>>> Name&family
>>> my email
>>> not
```

- 1a غلط است چون با عدد شروع می‌شود.
- Id.Number، Name&family و my email به دلیل استفاده از کاراکترهای غیر مجاز غلط هستند.
- not از کلمات کلیدی موجود در Python است.

استفاده از متغیرها

- اگر متغیری مقدار دریافت کرده باشد، با نوشتن نام آن متغیر در مقابل علامت >>> در پنجره Shell مقدار آن متغیر نشان داده می‌شود. به عنوان مثال در شکل زیر در سطر اول عدد 18 در متغیر age قرار داده می‌شود و نوشتن عبارت age در سطر دوم باعث ظاهر شدن مقدار آن یعنی عدد 18 در سطر بعد می‌شود.

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> age = 18
>>> age
18
>>> age = 21.5
>>> age
21.5
>>> age = " Five "
>>> age
' Five '
```

- اگر متغیری از قبل مقدار گرفته باشد، قرار دادن یک مقدار جدید در آن، باعث جایگزین شدن مقدار جدید با مقدار قبلی خواهد شد یعنی مقدار اول از داخل متغیر پاک شده و مقدار دوم بجای آن قرار خواهد گرفت. مثل عبارات نشان داده شده در شکل مقابل که ابتدا عدد 18 بعد عدد 21.5 و در انتها عبارت Five در متغیر age قرار داده می‌شود.

- هر متغیری در پایتون به راحتی می‌تواند هر نوع داده‌ای را ذخیره نماید. مثل نمونه بالا که متغیر age ابتدا یک داده‌ی عددی صحیح، در مرحله بعد یک داده‌ی عددی اعشاری و در انتها یک داده‌ی متنی را دریافت و ذخیره نموده است.
- در ریاضیات اگر متغیری به نام a داشته باشیم و بخواهیم مقدار آن را ۳ برابر کنیم، آن را به صورت 3a می‌نویسیم. اما در زبان‌های برنامه نویسی حتماً باید عبارت را به صورت a*3 نوشت. یعنی باید بین ضریب و متغیر یک عملگر ضرب قرار داده شود.



دستور print()

در مثال‌هایی که تا بحال در پنجره Shell نوشتیم، مشاهده کردیم که :

- اگر در مقابل علامت `>>>` یک عبارت متنی نوشته شود، فشردن کلید Enter باعث ظاهر شدن آن عبارت در سطر بعد می‌شود.
- اگر در مقابل علامت `>>>` یک عبارت ریاضی نوشته شود، فشردن کلید Enter باعث چاپ نتیجه آن عبارت ریاضی پس از محاسبه آن خواهد شد.
- اگر در مقابل علامت `>>>` نام متغیری نوشته شود که قبلاً مقدار دریافت کرده است، فشردن کلید Enter باعث چاپ مقدار آن متغیر خواهد شد.

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help

>>> " Python "
' Python '

>>> 20 + 45 / 9
25.0

>>> a = 12
>>> a
12
Ln: 14 Col: 4
```

نکته: این روش چاپ تنها در پنجره Shell انجام می‌شود و در داخل یک برنامه که در پنجره Edit یا Script نوشته می‌شود، قابل استفاده نیست. پس برای انجام عمل چاپ در یک برنامه چه باید کرد؟

در زبان برنامه نویسی پایتون دستوری برای چاپ وجود دارد به نام `print ( )`. این دستور هم در پنجره Edit و هم در پنجره Shell قابل استفاده است. ابتدا در پنجره Shell آن را بررسی می‌کنیم.

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help

>>> print("Python")
Python

>>> print(20 + 45 / 9 )
25.0

>>> a=2
>>> print (a)
2
Ln: 14 Col: 4
```

- هر چیزی که قرار است توسط دستور `print ( )` چاپ شود، باید در داخل یک جفت پرانتز در مقابل این دستور قرار گیرد.
- نوشتن یک عبارت متنی داخل دستور `print ( )` باعث چاپ همان عبارت خواهد شد.
- نوشتن یک عبارت ریاضی باعث چاپ نتیجه آن عبارت ریاضی پس از محاسبه آن خواهد شد.
- نوشتن نام یک متغیر، باعث چاپ مقدار آن متغیر خواهد شد.



خطاهای رایج

در صورتی که هنگام نوشتن دستور **print ()** پرانتز آن را فراموش نماییم، یک پیغام خطا

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> print 2 + 4
SyntaxError: Missing parentheses in call to 'print'.
Did you mean print(2 + 4)?
>>>
```

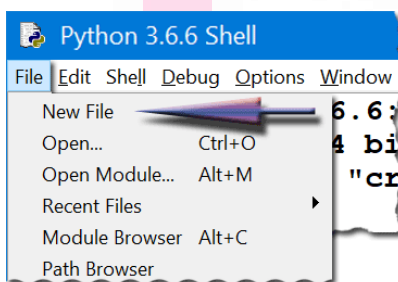
ظاهر می‌شود که محل خطا را نشان می‌دهد و حتی دستور درست را نیز پیشنهاد می‌کند!

در زبان پایتون برای نوشتن دستورات، تنها می‌توان از حروف کوچک استفاده نمود و استفاده

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> Print("Python")
Traceback (most recent call last):
  File "<pyshell#20>", line 1, in <module>
    Print("Python")
NameError: name 'Print' is not defined
>>>
```

از حروف بزرگ باعث ظاهر شدن خطا می‌شود. به عنوان مثال در شکل زیر حرف اول دستور **Print ()** با حرف بزرگ نوشته شده است که برای پایتون قابل قبول نیست.

نوشتن اولین برنامه



همانطور که قبلاً نیز گفته شد با انتخاب گزینه New File از منوی File در پنجره Shell، می‌توان به Edit Windows یا همان Script Windows دسترسی پیدا کرد.

می‌خواهیم برنامه‌ای بنویسیم که عبارت « **hello word** » را برایمان چاپ نماید. برای انجام این کار در پنجره Edit دستور **print ("hello world")** را می‌نویسیم و پس از ذخیره آن در حافظه جانبی، با فشردن کلید **F5** آن را اجرا می‌کنیم. این عمل باعث باز شدن پنجره Shell می‌شود. مشاهده می‌کنید که نتیجه اجرای دستور ما یعنی عبارات **hello world** در این پنجره چاپ شده است.

```
01.py
File Edit Format Run Options Window Help
print ("hello world")
Ln: 4 Col: 19

Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>>
hello world
>>>
Ln: 6 Col: 4
```



نوشتن یک برنامه محاسباتی

$$20 + \frac{a}{9}$$

می‌خواهیم برنامه‌ای بنویسیم که عدد 45 را در متغیر a قرار داده و حاصل عبارت مقابل را محاسبه نماید.

The screenshot shows two windows. The left window, titled '02.py', contains the following code:

```
a=45
print( 20 + a / 9 )
```

A red arrow labeled 'F5' points from the code to the right window. The right window, titled 'Python 3.6.6 Shell', shows the execution results:

```
>>>
25.0
>>>
```

نوشتن دو عبارت با دستور `print ( )`

اگر بخواهیم با استفاده از دستور `Print ( )` دو عبارت را چاپ نماییم، کافیست آنها را با استفاده از علامت کاما « , » از یکدیگر جدا کنیم.

The screenshot shows a window titled '03.py' with the following code:

```
print ( " 12 * 3 = " , 12 * 3 )
```

Callouts explain the code:

- A blue callout points to the code: "محاسبه شده و جواب آن چاپ می‌شود"
- A green callout points to the string: "به دلیل آنکه در علامت \" \" نقل قول قرار دارد به صورت یک عبارت متنی یعنی بدون تغییر چاپ می‌شود"
- An orange callout points to the comma: "دو عبارت را از یکدیگر جدا می‌کند"

The right window, titled 'Python 3.6.6 Shell', shows the output:

```
12 * 3 = 36
>>>
```

دست کرمی :

The screenshot shows a window titled '04.py' with the following code:

```
a = 2
b = " Python "
print( a * b )
```

به نظر شما خروجی برنامه روبرو چیست؟ آیا می‌توانیم این برنامه را اجرا کنیم یا با پیغام خطا مواجه خواهیم شد؟



آنچه در هفته بعد خواهید خواند

دستور () input

با استفاده از دستور () **input** می‌توان در زمان اجرای برنامه، داده‌هایی را از کاربر دریافت نمود و در زمان اجرای برنامه از آن داده‌ها استفاده کرد.

- برنامه در زمان اجرا با رسیدن به دستور () **input** منتظر می‌ماند تا کاربر داده‌ای را وارد نماید.
- با توجه به اینکه این دستور در زمان اجرای برنامه داده‌ای را از کاربر دریافت می‌نماید، به متغیری نیاز دارد تا داده دریافت شده در آن نگهداری شود.
- این دستور مانند دستور () **print** می‌تواند یک عبارت متنی را قبل از دریافت داده از کاربر بعنوان راهنما به او نشان دهد تا کاربر در زمان اجرای برنامه بداند چه چیزی را باید به عنوان داده وارد کامپیوتر نماید. با توجه به اینکه این عبارت، یک عبارت متنی است، مسلماً باید داخل دو علامت گیومه یا نقل قول (" ") قرار گیرد.

متغیری که قرار است داده دریافت شده در آن ذخیره شود

عبارتی که قبل از دریافت داده از کاربر به او نشان داده می‌شود

```
name = input("Enter your Name : ")
print( " Hi " , name )
```

Ln: 4 Col: 20



داده وارد شده توسط کاربر که در متغیر name ذخیره می‌شود

دست گرمی رو انجام دادم این چاپ شد. درسته؟

Python Python

