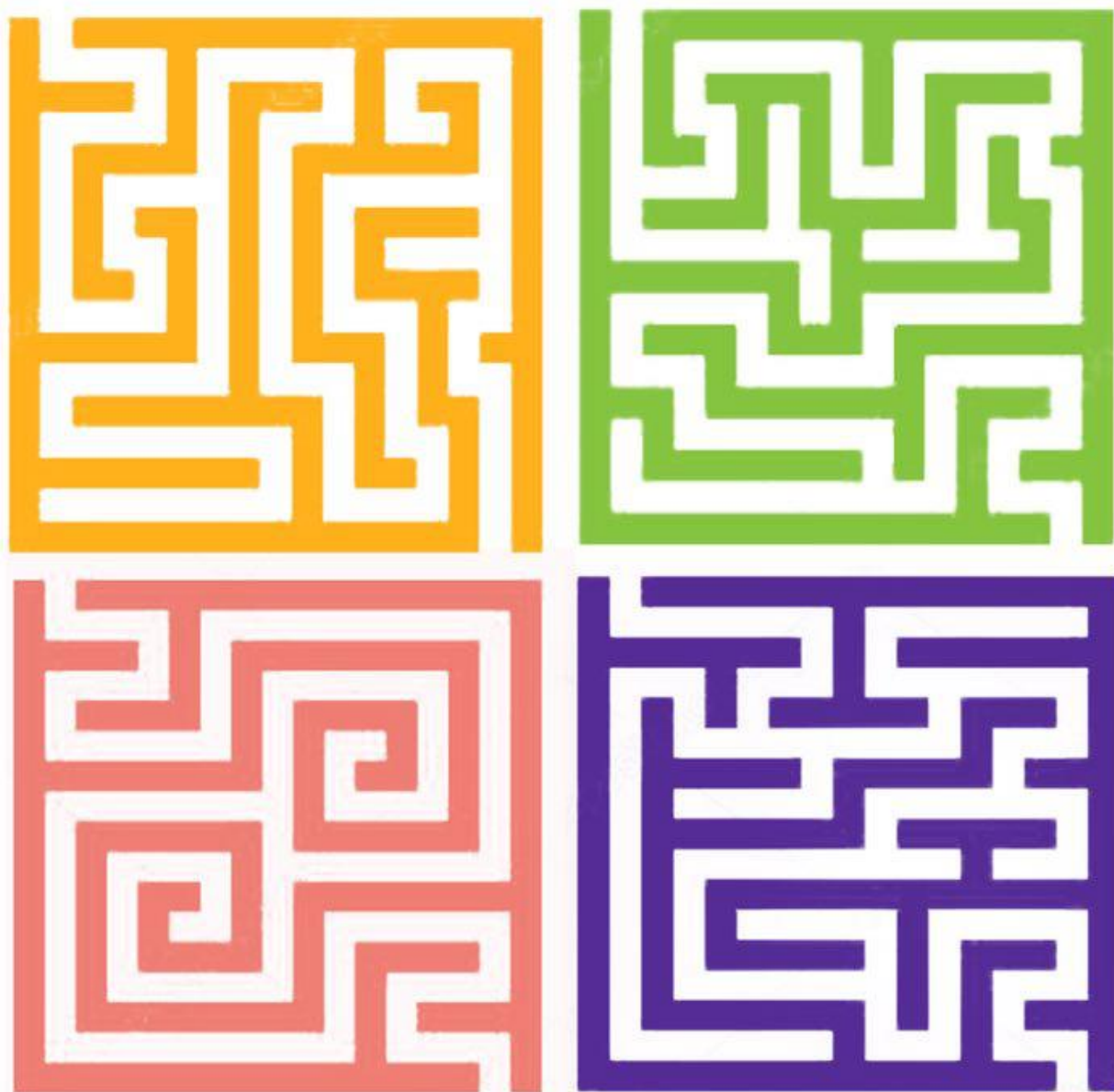




سلام پایتون



نگارنده: فرشید میدانی



آموزش زبان برنامه نویسی پایتون برای بچه‌های ایران

نسخه : Salam_Python_By_Farshid_Meidani_v1_17.docm

تاریخ آخرین آپدیت: یکشنبه، ۰۶ دسامبر ۲۰۲۰

نویسنده: فرشید میدانی

f.meidani@farsaran.com

سخنی با بچه‌ها

لطفا انگلیسی کلمات را تا حد ممکن یاد بگیرید چون در آینده شما کتاب‌های انگلیسی را خواهید خواند و خواهید دید که با دانستن این کلمات خواندن این کتاب‌ها چقدر ساده خواهد شد. خواهش می‌کنم که از چیزی نترسید. از کلمات انگلیسی نترسید. از برنامه‌های که متوجه نمی‌شوید نترسید. از چیزهای پیچیده نترسید.

هر وقت که ترسیدید یاد وقتی که خیلی کوچک بودید و هیچ کلمه‌ای را بلد نبودید بیافتید. ببیند که از آن زمان تا حالا هزاران کلمه را یاد گرفته‌اید بدون اینکه زحمت خاصی بکشید. برنامه نویسی هم همین‌طور است. حوصله کنید و نترسید. کم‌کم یاد می‌گیرد. بعضی از چیزها شاید ۱ ماه طول بکشد و برخی دیگر شاید بیشتر. اما بالاخره یادشان می‌گیرید. قول میدهم.

من سعی کرده‌ام که این کتاب تا حد ممکن ساده باشد و خودآموز. یعنی خودتان بتوانید بدون نیاز به معلم آنرا بخوانید و متوجه شوید. البته ممکن است که برخی از مطالب را در ابتدا خوب نفهمید. اشکالی ندارد و کاملاً در یادگیری برنامه نویسی این موضوع طبیعی است. اگر نکته‌ای را نفهمیدید از یکی از دوستانتان که او هم به کامپیوتر علاقمند است، بخواهید که آن مطلب را بخواند و سپس آنچه را که فهمیده است برای شما آنرا توضیح دهد. جالب است که بعضی از مواقع وقتی که فقط سوال را می‌پرسید، به صورت عجیبی باعث می‌شود که خودتان جواب را بیابید. بازهم جالب است که برخی از مواقع دوستانتان پاسخ اشتباهی را می‌دهد و شما متوجه اشتباه او می‌شود و همین کار باعث می‌شود که شما کم‌کم آن موضوع و تفسیرهای متفاوتش را یاد بگیرید. از معلم ریاضی و زبان کمک بگیرید.

در برخی از تمرین‌ها ممکن است که شما مفهومی را متوجه نشوید و یا معنای کلمه‌ای برای شما نامفهوم باشد. در این مواقع لطفاً کمک بگیرید. از معلم‌های ریاضی و زبان و یا سایر همکلاسی‌هایتان بپرسید تا آنها شما را راهنمایی کنند.

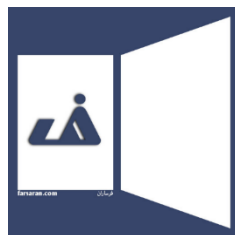
سلام پایتون

کتاب آموزش زبان برنامه نویسی پایتون

نسخه پیش نویس

نویسنده:

فرشید میدانی



موسسه فرساران تفکر

ایران، تهران

www.farsaran.com

طراحی جلد: بابک هوشمند

لطفاً با من تماس بگیرید و بگید:

که در مورد این کتاب چه فکر می‌کنید؟
چه جاهایی را خوب توضیح نداده‌ام؟
اشتباهات تایپی و املائی و .. را حتماً بگید.
به من بگید که از کجا این کتاب به دست شما رسید و
آیا تونستید با این کتاب اسکرچ رو یاد بگیرید؟



برای تماس با من به f.meidani@farsaran.com ایمیل بزنید. منتظر هستم.

حقوق مالکیت فکری و مادی و معنوی این اثر

کلیه حقوق مالکیت فکری، مادی و معنوی این اثر (کتاب سلام پایتون فرساران) متعلق به آقای فرشید میدانی است و این اثر تحت قانون حمایت حقوق مولفان و مصنفان و هنرمندان می‌باشد.

با رعایت شرایط زیر اجازه دانلود و خواندن این کتاب برای همگان آزاد و رایگان است.

- این اثر کاملاً رایگان و آزاد است و هیچ فردی (حقیقی و یا حقوقی) حق فروش آنرا به هر شکلی ندارد .
- اضافه کردن هر متن و یا لوگو و یا هر چیز دیگری بر روی قسمتی و یا کل این کتاب مجاز نیست .
- اجازه حذف و یا اضافه و یا ویرایش قسمتی و یا کل کتاب وجود ندارد و این اثر باید به همان صورت اولیه و همچنین کامل در اختیار سایرین قرار داده شود .
- استفاده از این اثر برای هر شکلی از تبلیغات و یا بازاریابی و یا جذب مشتری ممنوع است .
- پدر و مادرها و معلمین مجاز هستند که نسخه الکترونیکی و یا pdf این اثر را به بچه‌ها و یا دانش آموزان خود به رایگان بدهند به شرط آنکه این اثر را بر روی وب سایت و یا هرگونه فضای مجازی عمومی قرار ندهند.
- اجازه چاپ و یا تکثیر و یا پیرینت و یا کپی فیزیکی این اثر فقط برای مدارس دولتی و به شرط استفاده برای دانش آموزان همان مدرسه و یا کلاس هستند.
- پدر و مادرها و معلمین اسکرچ اجازه چاپ و یا تکثیر و یا پیرینت و یا کپی فیزیکی این اثر را برای ارائه به فرزندان خود و یا ارائه رایگان به آشنایان خود را دارند.
- هیچ فرد حقوقی و حقیقی اجازه چاپ و یا تکثیر و یا باز نشر و یا فروش جزئی و یا تمام این اثر را ندارد.
- وب سایت‌ها، وبلاگ‌ها و کانال‌های تلگرام و اینستاگرام و سامانه‌های آموزشی و هر شکلی از فضای مجازی اجازه باز نشر و یا share کردن این اثر را ندارند. بدیهی است که مجاز هستند که این اثر را معرفی نمایند و بایستی بازدیدکنندگان، مخاطبین و دانش آموزان را برای دانلود به صفحه آن به آدرس زیر ارجاع دهند.
- www.farsaran.com/pb
- مدارس غیر دولتی و یا خصوصی و سایر موسسات آموزشی برای تهیه نسخه چاپی این اثر باید از طریق صفحه تماس با ما وب سایت فرساران به آدرس www.farsaran.com/contact سفارش خود را اعلام نمایند.

1

فصل یک

قدم های اول

معرفی پایتون و هدف ما

قرار است ما یاد بگیریم که چطور با کامپیوترها حرف بزنیم تا آنها منظور ما را درک کنند و کاری را که خواسته‌ایم را انجام دهند. به اینکار، یعنی حرف زدن با کامپیوترها، «برنامه نویسی» می‌گویند.

دقیقا مانند آدم‌ها که با هم به زبان‌های مختلفی حرف می‌زنند، در دنیای کامپیوتر هم ما با کامپیوترها با زبان‌های مختلفی حرف می‌زنیم. نام برخی از مشهورتری زبانهای برنامه نویسی در دنیای کامپیوتر برای شما در زیر نوشته‌ام:

(۱) جاوا (۲) C++ (بخوانید سی پلاس پلاس) (۳) PHP (بخوانید پی اچ پی)

جالب است بدانید که بسیاری از اصول و دستورها این زبان‌ها با هم مشترک هستند و اگر یکی از آنها را خوب بلد باشیم، یادگیری زبان‌های دیگر اصلا سخت نیست و افراد زیادی هستند که چندین زبان برنامه‌نویسی را خوب می‌دانند.

حالا باید بگویم که قرار است باهم زبان برنامه نویسی «پایتون» را یاد بگیریم. شاید از خودتان بپرسید که چرا پایتون را انتخاب کرده‌ام. در پاسخ شما باید بگویم که پایتون یکی از زبان‌های ساده برای یادگیری است و در دنیا برای آموزش به بچه‌ها بکار می‌شود. در ضمن اینکه پایتون یکی از «مشهورترین» زبان‌های برنامه‌نویسی دنیا است و در آینده شما می‌توانید با آن حرفه‌ای کار کنید.

در ضمن آنکه با پایتون می‌توانید خیلی زود و سریع برنامه‌های جالبی و حتی بازی بنویسید و باعث می‌شود که از برنامه نویسی خسته نشوید و برای شما جذاب باشد.



چرا باید برنامه‌نویسی را یاد بگیرید

اولین پاسخ این است که نوشتن یک برنامه کار جالبی است و لذت بخش است. مخصوصا وقتی که یک بازی ساده و کوچک را خودتان می‌نویسید و می‌بینید که اجرا می‌شود، بسیار خوشحال خواهید شد.

نکته دوم آن است که ذهن و فکر شما قوی و قوی‌تر خواهد شد و ذهن قوی یکی از چیزهایی است که باعث می‌شود شما «فرصت زندگی بهتری» را در این دنیا داشته باشید.

اما نکته سوم و بسیار مهم این است که در آینده می‌توانید از دانش برنامه نویسی درآمد کسب کنید. حتی می‌توانید یک برنامه نویس شوید و حقوق خوبی بابت اینکار بگیرید.

توصیه‌های قبل از شروع

من در این جزوه سعی می‌کنم که اصول اولیه برنامه نویسی پایتون را به زبان بسیار بسیار ساده‌ای به شما یاد دهم. در ضمن اینکه مثال‌های جالبی را بزنم تا شما تشویق شوید تا برنامه نویسی را یاد بگیرید.

لطفا هر مثال را خودتان حتما تایپ و اجرا کنید و سپس آنرا به سلیقه خودتان تغییراتی بدهید و اجرا کنید تا ببینید که نتیجه آنها چه خواهد شد.

راستی در ابتدای راه ممکن است که کمی کلافه شوید، یعنی اینکه اشتباهات کوچکی را انجام دهید که برنامه اجرا نشود. در اینصورت از شما خواهش می‌کنم که اصلاً عصبانی و یا ناامید نشوید. حوصله کنید. برنامه را خط به خط و با حوصله بخوانید. خطایش را بخوانید و سعی کنید که خطایش را پیدا و اصلاح کنید. بدانید که در ابتدای راه خطاهای برنامه و این حالت کلافه کننده کاملاً طبیعی است.

اگر بازهم اشکال برنامه را نیافتید، به خودتان یک استراحت کوچک بدهید مثلاً بروید کمی قدم بزنید. جالب است که بدانید در بسیاری از موارد بعد از اینکار، می‌توانید به سادگی خطای برنامه را پیدا کنید.

چرا این کتاب نسخه پیش نویس است؟

از آنجایی که دیدم که یادگیری پایتون برای بچه‌های ایران کمی سنگین و سخت است، ادامه نوشتن این کتاب را رها کردم و زمان زیادی را برای آماده کردن کتاب، ترجمه و ویدئوهای «زبان برنامه نویسی اسکرچ» اختصاص دادم که نتیجه آن را در وب سایت فرساران می‌توانید ببیند. به همین دلیل در این کتاب ممکن است اشتباه‌ها و گاهی توضیحات ناقصی را ببینید و هر ابهام، نکته، اشتباه و ... را که دیدید، حتماً مرا از طریق ایمیل زیر مطلع کنید:

f.meidani@farsaran.com

نصب پایتون بر روی کامپیوترتان

خوشبختانه در سیستم عامل لینوکس، پایتون نصب است و نیازی ندارید که آنرا نصب کنید. اما اگر از سیستم عامل ویندوز استفاده می‌کنید باید پایتون را نصب کنید. نصب پایتون ساده و سریع است که من در پیوست الف (به انتهای همین کتاب مراجعه کنید) نحوه نصب آنرا توضیح داده‌ام.

اجرای پایتون

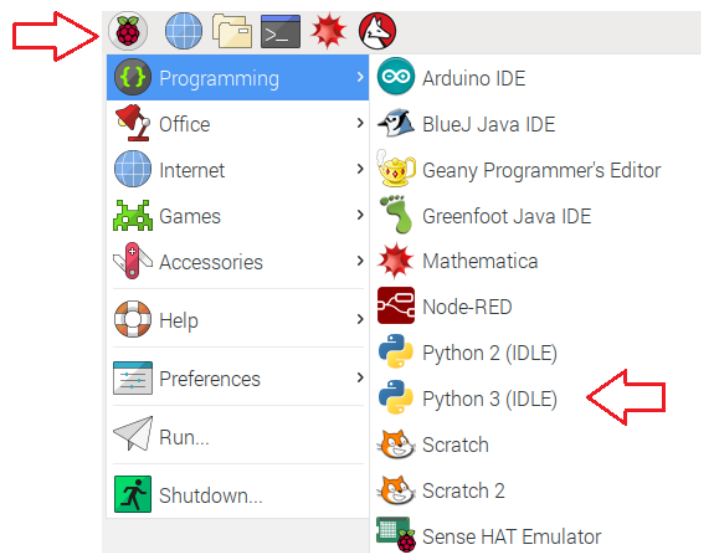
برای آنکه شما فوتبال بازی کنید به یک زمین فوتبال نیاز دارید و به همین ترتیب برای اینکه بتوانیم یک برنامه را بنویسیم به جایی نیاز داریم که برنامه‌هایمان را در آنجا تایپ و اجرا کنیم. برای اینکار ما از برنامه IDLE (بخوانید آی دی ال ئی) استفاده خواهیم کرد.

اگر پایتون بر روی کامپیوتر شما نصب باشد، برای باز کردن IDLE اینگونه عمل کنید:

در رسیبری پای:

منوی Start (که در سمت بالا سمت چپ است) را باز کنید و سپس از Programming روی گزینه (IDLE) Python 3 کلیک کنید تا پنجره‌ای باز شود.

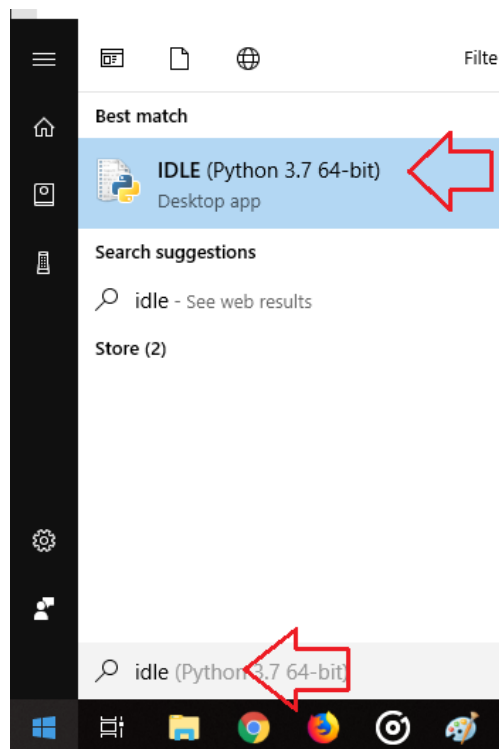
تصویر مسیر اجرای IDLE در Raspberry pi



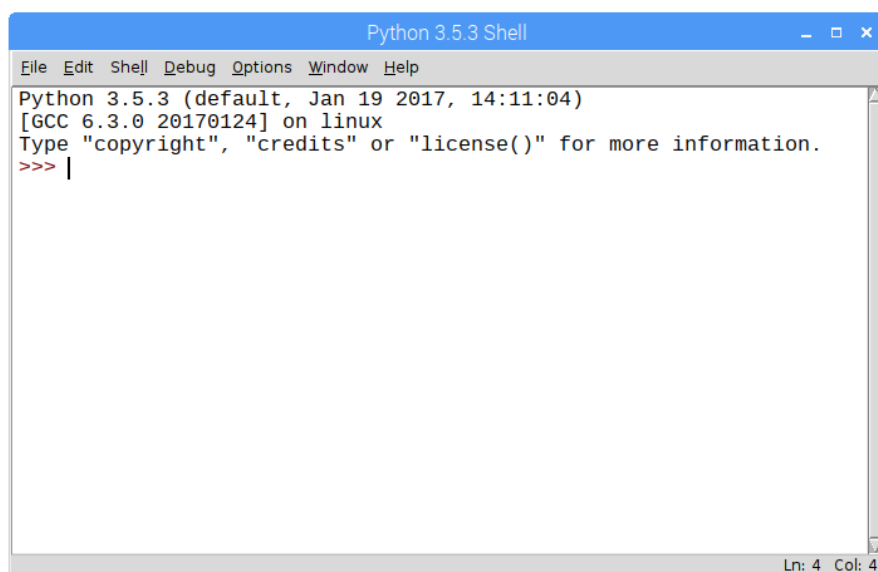
در ویندوز:

منوی Start ویندوز را باز کنید و سپس IDLE را تایپ کنید. ویندوز برای شما جستجو را انجام می‌دهد و گزینه‌ای شبیه (Python 3 IDLE) را می‌یابد. روی آن کلیک کنید تا برنامه اجرا شود.

تصویر اجرای IDLE بر روی ویندوز 10



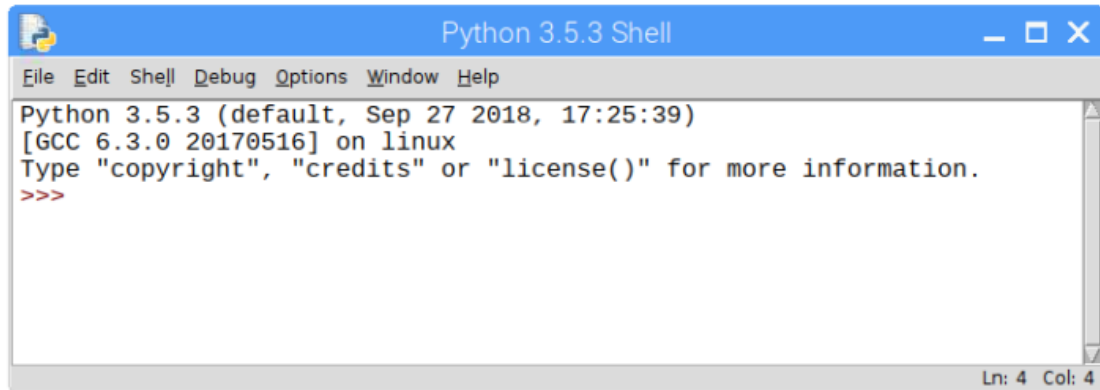
در تصویر زیر ظاهر IDLE را می‌بینید و در این محیط می‌توانید اولین برنامه‌های خود را به زبان پایتون بنویسید و آزمایش کنید.



پرسش (با توجه به تصویر قبل، نسخه پایتون چند است؟ پاسخ: 3.5.3)
در توضیح بیشتر به این پرسش باید بگویم که مانند مدل‌های ماشین که هر چند وقت یکبار (مثلا سالیانه) جدید و به روز می‌شوند، نسخه‌های زبان برنامه نویسی پایتون هم به روز رسانی می‌شوند. در حال حاضر آخرین نسخه پایتون 3 است. توجه: از اینکه همه چیز انگلیسی است نگران نشوید. کم کم آنها را یاد می‌گیرید. 😊

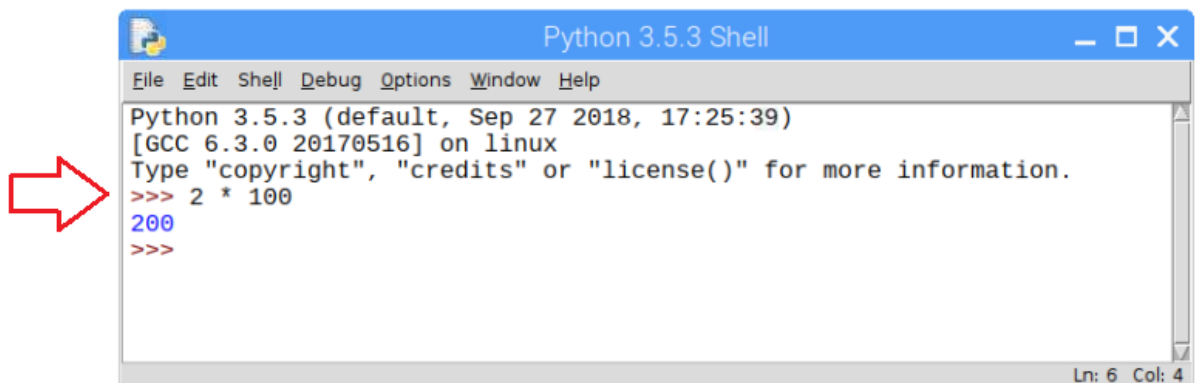
تایپ اولین دستورها

به پنجره Shell (بخوانید شِل) پایتون خوش آمدید. به تصویر زیر دقت کنید، علامت >>> را ببیند. در جلوی این علامت می‌توانید هر دستوری را که پایتون می‌فهمد را وارد کنید و سپس کلید Enter (بخوانید اینتر) را بزنید. پایتون برای شما آن دستور را بلافاصله اجرا می‌کند و نتیجه آن دستور را برای شما نشان خواهد داد.



```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
```

برای شروع ما می‌توانیم از پنجره شِل به عنوان یک ماشین حساب ساده استفاده کنیم، مثلاً تایپ کنید $2 * 100$ و Enter را بزنید:



```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Sep 27 2018, 17:25:39)
[GCC 6.3.0 20170516] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2 * 100
200
>>>
```

نکته ۱) توجه کنید نتیجه دستور به رنگ آبی است.

نکته ۲) در این کتاب و یا سایر کتاب‌ها هر جایی که علامت >>> را دیدی یعنی که در حالت Shell باید دستورها را وارد کنید. توجه داشته باشید که نباید این علامت‌ها را خودتان تایپ کنید، بلکه باید چیزی را که در جلوی آن وجود دارد را تایپ کنید. بنابراین اگر من در جایی نوشتم :

```
>>> 57+5+17
```

یعنی دستور ما $57+5+17$ است و فقط باید همین دستور را در پنجره Shell و جلوی علامت >>> تایپ نمایید و برای اجرای آن کلید Enter را بزنید.

نکته ۳) اگر دستوری را بنویسید که پایتون معنای آنرا متوجه نشود، به شما پیغام خطایی به رنگ قرمز نمایش داده می‌شود. در این پیغام به شما می‌گوید که چه خطایی در کدام دستور اتفاق افتاده است. فعلا با ترجمه و تفسیر این خطاها کاری نداریم. برای نمونه من نام خودم را تایپ می‌کنیم و چون پایتون متوجه این دستور نمی‌شود، پیغام خطا را نمایش می‌دهد:

```
>>> farshid
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    farshid
NameError: name 'farshid' is not defined
>>> |
```

Ln: 11 Col: 4

آشنایی با واژه Shell:

Shell به معنی پوسته که رابطه ما را با موتور پایتون برقرار می‌کند. Shell مانند یک پیشخدمت رستوران است که از ما سفارش غذا را می‌گیرد و به آشپزخانه می‌رود و غذا را سرآشپز می‌گیرد و برای ما می‌آورد.

آشنایی با محاسبات ساده

اگر در داخل IDLE نیستد، ابتدا آنرا اجرا نمایید و زیرا می‌خواهیم چند دستور را در آن تایپ و نتیجه آنرا بررسی کنیم. مثال ۱) دستور زیر را تایپ کنید و سپس Enter را بزنید.

```
>>> 2 * 10
```

خواهید دید که حاصلضرب عدد 2 در 10 چاپ می‌شود.

توجه ۱: علامت ضربدر در کامپیوتر * است که با کلید 8 + Shift باید آنرا تایپ کنید. یعنی باید کلید Shift (بخوانید شیفت) را پایین فشار دهید و در حالی که پایین نگه‌داشته‌اید، کلید 8 را بزنید.

مثال ۲) دستور زیر را تایپ کنید و سپس Enter را بزنید.

```
>>> 48 / 6
```

علامت تقسیم در کامپیوتر، علامت «/» است و نام این علامت «اسلش» است.

مثال ۳) دستور زیر را تایپ کنید و سپس Enter را بزنید.


```
>>> 10 + 20 + 30
```

خواهید دید که فاصله‌های اضافی مهم نیستند و اشکالی در اجرای محاسبه پیش نمی‌آید.

مثال ۴) دستور زیر را تایپ کنید و سپس Enter را بزنید.

```
>>> 20 % 8
```

علامت % برای محاسبه باقی مانده یک تقسیم بکار می‌رود. یعنی اگر عدد ۲۰ را بر ۸ تقسیم کنید، باقی‌مانده ما عدد ۴ می‌شود. برای تایپ علامت % کلید Shift را پایین نگه دارید و کلید 5 را بزنید.

$$\begin{array}{r|l} 20 & 8 \\ - 16 & 2 \\ \hline & 4 \end{array}$$


مثال ۵) دستور زیر را تایپ کنید و Enter را بزنید.

```
>>> 15.5 + 2.5
```

علامت ممیز در کامپیوتر «.» است. که آنرا «دات» می‌خواهیم.

مثال ۶) عمداً می‌خواهیم یک دستور اشتباه را تایپ کنیم تا ببینیم که واکنش پایتون به آن چیست. بنابراین دستور زیر را تایپ کنید و Enter را بزنید.

```
>>> 2 a 8
```

در واقع کامپیوتر نمی‌تواند معنی علامت a را در بین دو عدد تشخیص دهد و به همین دلیل پیغام خطای زیر را برای ما نشان می‌دهد:

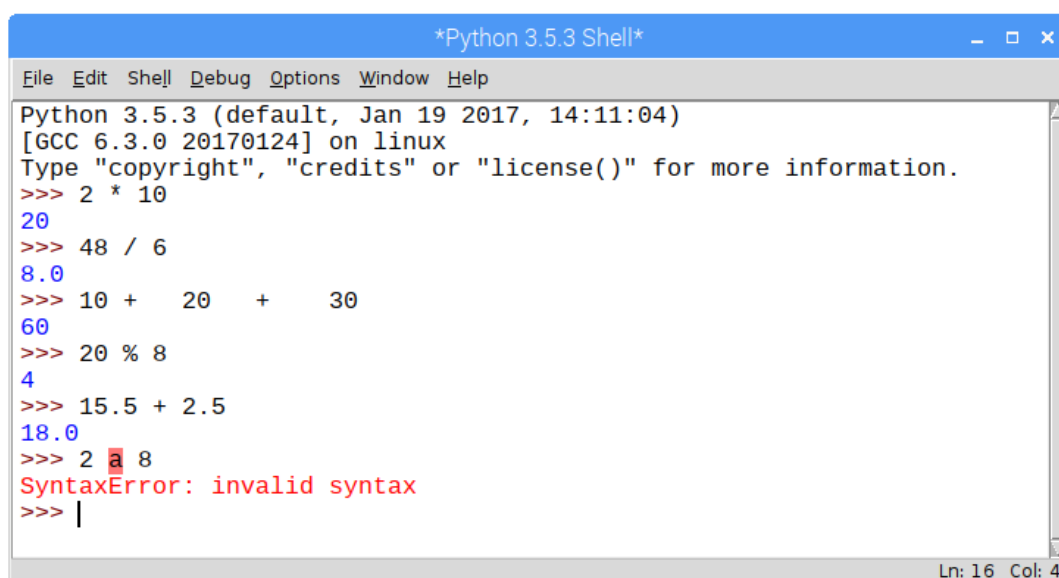
```
SyntaxError: invalid syntax
```

در این خطا کلمه Syntax (بخوانید سینتکس) یعنی نوشته‌ای که تایپ کرده اید و کلمه Error (بخوانید ارور) یعنی خطا. این پیغام به شما می‌گوید که "غلط املایی یا تایپی دارید". دقیقا مانند معلم شما که خطاهای املایی شما را پیدا می‌کرد.

در اینجا پایتون فهمید که حرف a اینجا اشتباه است اما واقعا نمی‌تواند آنرا به جای شما تصحیح کند. آخر از کجا بداند که جای آن باید چه علامتی از بین همه علامت‌های مجاز را قرار دهد. بنابراین شما باید این دستور را مجدد و به صورت صحیح تایپ کنید تا کامپیوتر آنرا بفهمد.

تذکر: لطفا از کلمات و خطاهای انگلیسی نترسید. به تدریج با معنای آنها آشنا خواهید شد.

در تصویر زیر تصویر تایپ شده همه این دستورها به همراه خروجی (نتیجه) آنها را مشاهده می‌کنید.



```
*Python 3.5.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2 * 10
20
>>> 48 / 6
8.0
>>> 10 + 20 + 30
60
>>> 20 % 8
4
>>> 15.5 + 2.5
18.0
>>> 2 a 8
SyntaxError: invalid syntax
>>> |
```

توجه: لازم نیست که برای هر وارد کردن هر دستور (یا فرمان) از Shell خارج شوید و مجدد آنرا باز کنید. هر دستور را تایپ کنید و Enter بزنید و سپس دستور بعدی را در جلوی علامت >>> وارد کنید.

بگذارید یک نکته دیگر را هم در اینجا بگویم. به علامت‌هایی مانند + ، * ، / و یا % در دنیای کامپیوتر اصطلاحاً Operator (بخوانید آپ، ری، تور) می‌گویند که در فارسی آنرا «عملگر» ترجمه می‌کنند.

یعنی هنگامی که دو مقدار داریم باید به کامپیوتر بگوییم که بر روی این دو مقدار چه عملیاتی را انجام دهد و اینکار توسط این «عملگرها» انجام می‌شود. در ادامه شما با Operator های دیگر زبان پایتون آشنا خواهید شد.

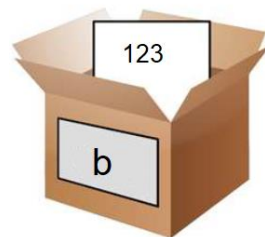
جعبه شیرینی

اگر بخواهید که از قنادی شیرینی بخرید، بدیهی است که آن شیرینی را در داخل جعبه‌ای می‌گذارند تا شما آنرا به خانه ببرید و هر وقت که دوست داشتید، آنرا بخورید. در واقع جعبه شیرینی محلی است که شیرین‌ها در آن نگهداری می‌شود تا بعداً مورد استفاده قرار گیرد.

دقیقاً مشابه همین کار را در دنیای کامپیوتر هم انجام می‌دهیم یعنی یک مقدار را در جایی ذخیره می‌کنیم تا بعداً بتوانیم از آن استفاده کنیم. برای اینکار دستور زیر را می‌نویسیم و Enter را می‌زنیم:

```
>>> b = 123
```

این دستور باعث می‌شود که در حافظه کامپیوتر فضایی (یا جایی) به نام b ساخته شود و سپس مقدار 123 در داخل این محل (که نامش b است) قرار خواهد گیرد. بگذارید فرض کنید که این فضا مانند همان جعبه شیرینی است که مقدار 123 در آن قرار دارد.



حال از این پس (یعنی بعد از اجرای دستور قبل) هر جایی که بنویسیم b ، کامپیوتر می‌رود و مقداری که در b ذخیره شده است را برای ما می‌آورد. مثلاً اگر بنویسیم:

```
>>> b * 100
```

مقدار 12300 برای ما چاپ خواهد شد. یعنی کامپیوتر مقدار b که عدد 123 است را در عدد 100 ضرب خواهد کرد و البته می‌توانیم بارها از مقدار b در سایر دستورها استفاده کنیم. به شکل زیر دقت کنید که از مقدار b در دو دستور استفاده شده است.

```
>>> b = 123
>>> b * 100
12300
>>> b / 10
12.3
>>> |
```

Ln: 10 Col: 4

لیوان آب یا شربت

اگر یک لیوان داشته باشیم، هم می‌شود در داخل آن شیر ریخت و هم آب و بعضی مواقع هم شربت. یعنی در لیوان آب می‌توان چیزهای مختلفی را بریزیم. به این لیوان در برنامه نویسی اصطلاحاً «متغیر» می‌گوییم زیرا چیزی که در آن می‌توانیم بریزیم، متغیر است و می‌تواند هر چیزی را در داخل خود نگه دارد. مثلاً اول در لیوان آب بریزیم و سپس آن آب را خالی کنیم و اینبار لیوان با شربت پُر کنیم.

حالا می‌توانم به شما بگویم که b در مثال قبلی یک «متغیر» است. یعنی در b می‌توان مقادیر مختلفی را ذخیره کرد. به عنوان مثال با دستور زیر مقدار 123 را در متغیر b قرار می‌دهیم:

```
>>> b = 123
```

و سپس اگر فرمول زیر را بنویسیم، مقدار 12300 نمایش داده می‌شود:

```
>>> b * 100
```

حال می‌توانیم یک عدد دیگری را در b قرار دهیم. مثلاً با دستور زیر مقدار b ، عدد 55 خواهد شد:

```
>>> b = 55
```

و از این به بعد مقدار b ، عدد 55 است و پاسخ دستور زیر عدد 5500 خواهد شد:

```
>>> b * 100
```

نکته! توجه داشته باشید که ما می‌توانیم در یک برنامه چندین متغیر داشته باشیم. دستورها زیر را در نظر بگیرید:

```
>>> x = 12
```

```
>>> y = 19
```

```
>>> x + y
```

ما یک متغیر به نام x ساختیم و سپس مقدار 12 را در آن قرار داریم و سپس یک متغیر دیگر به نام y ساختیم و مقدار 19 را در آن قرار دادیم و سپس مقدار دو متغیر x و y را با هم جمع زدیم.

نکته ۲) نام یک متغیر می‌تواند هر چیزی باشد. مثلاً فرض کنید نمره فارسی شما 13 شده است و نمره ریاضی شما 12 شده و بخواهید که میانگین نمره فارسی و ریاضی را محاسبه کنید، برنامه آن به شکل زیر خواهد شد:

```
>>> farsi = 13
>>> riazi = 12
>>> (farsi + riazi) / 2
12.5
```

توجه: عدد 12.5 خروجی دستور آخر است و نباید آنرا تایپ کنید. از این به بعد خروجی برنامه‌ها را به همین شکل برای شما خواهم نوشت.

نکته ۳) در دستور آخر باید ابتدا جمع نمرات را محاسبه کنید و سپس آنها را بر عدد 2 تقسیم کنید، به همین دلیل باید آنها را در داخل علامت پرانتز قرار دهید. در آینده در مورد این نکته بیشتر صحبت خواهیم کرد. !!!

نکته ۴) در انگلیسی به «متغیر» اصطلاحاً «Variable» (بخوانید وَری ای پِل) می‌گویند.

کار با کلمات و جمله‌ها در برنامه نویسی

تا اینجا ما فقط با اعداد کار کردیم. مثلاً روی آنها محاسباتی را انجام داده‌ایم و یا آنها را در متغیری را ذخیره کرده‌ایم. در برنامه نویسی ما لازم داریم که بتوانیم با کلمات و جملات کار کنیم. مثلاً بتوانیم جمله‌ی "باران جان تولد مبارک" را نمایش دهیم. کار با کلمات و جملات در پایتون بسیار ساده است.

قبل از شروع بگذارید یک واژه جدید را به شما یاد بدهیم. ما به صورت کلی به جملات، کلمه‌ها و متن‌ها در برنامه نویسی به انگلیسی String (بخوانید استرینگ) می‌گوییم و به فارسی این کلمه را «رشته» ترجمه می‌کنند. حال می‌خواهیم در یک متغیر به نام `n` اسم یک دانش آموز به نام باران را ذخیره کنیم. بنابراین خواهیم نوشت:

```
>>> n = 'baran'
```

توجه کنید که باید متن‌ها را در داخل علامت ' ' بگذاریم. به این علامت اصطلاحاً «کوته‌نویسی» می‌گویند. چون اگر متن را در داخل کوته‌نویسی نگذاریم، پایتون نمی‌تواند بفهمد که منظور ما یک متن است و خطا خواهد داد. حالا یک دستور جالب را اجرا می‌کنیم. بیایید دستور زیر را اجرا کنیم:

```
>>> 5 * n  
'baranbaranbaranbaranbaran'
```

می‌بینید که کلمه باران، پنج بار پشت سر هم تکرار می‌شود. جالب است که عملگر `*` در هنگام محاسبه بر روی اعداد باعث ضرب آنها می‌شود و همین علامت هنگامی که بر روی یک متن بکار می‌رود، باعث تکرار آن می‌شود. حالا می‌خواهیم که یک متغیر دیگر به نام `m` به شکل زیر تعریف کنیم:

```
>>> m = ' tavalodat mobarak.'
```

به فاصله‌ی خالی که در ابتدای این متن وجود دارد توجه کنید. این فاصله در داخل کوته‌نویسی‌ها قرار داد و جزیی از متن است.

حال می‌خواهیم که این دو متغیر را به هم بچسبانیم تا یک جمله کامل را تولید کنیم و برای اینکار دستور زیر را خواهیم نوشت:

```
>>> n + m  
'baran tavalodat mobarak.'
```

همانطور که می‌بینید عملگر `+` باعث می‌شود که متن‌ها به یکدیگر بچسبند (ترکیب شوند).

برنامه از شما سوالی بپرسد

بگذارید یک برنامه ساده بنویسیم که نام یکی از دوستانتان را بپرسد و سپس به او تولدش را تبریک بگوید. برای اینکه برنامه از ما سوالی کند و یک مقدار را بپرسد از دستور input استفاده می‌کنیم.

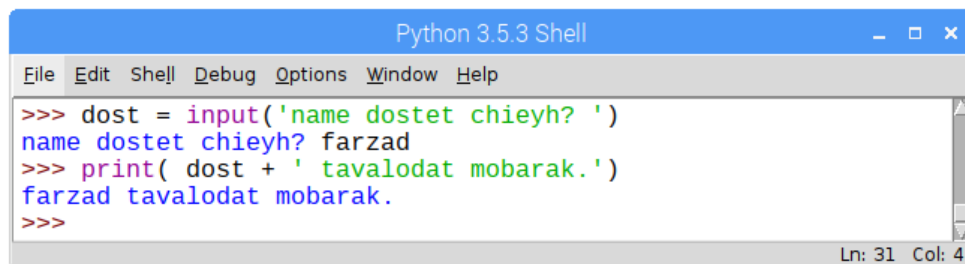
کافی است که این دستور را به شکل زیر تایپ کنیم:

```
>>> dost = input('name dostet chieyh? ')
```

همین که Enter را بزنید، پیغامی که در داخل پرانتز نوشته‌اید چاپ می‌شود و برنامه منتظر می‌ماند تا شما یک اسم را تایپ کنید و Enter را بزنید. توجه داشته باشید که پیغام داخل پرانتز کاملاً اختیاری است و هر چیزی می‌تواند باشد. بعد از آنکه شما اسمی را تایپ کردید و Enter را زدید، آن اسم در داخل متغیر dost ذخیره خواهد شد و شما می‌توانید با نوشتن دستور زیر تولد دوستانتان را تبریک بگویید.

```
>>> print( dost + ' tavalodat mobarak.')
```

در تصویر زیر کل مراحل وارد کردن دستورها و پیغام‌های چاپ شده را می‌توانید ببینید:



```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
>>> dost = input('name dostet chieyh? ')
name dostet chieyh? farzad
>>> print( dost + ' tavalodat mobarak.')
```

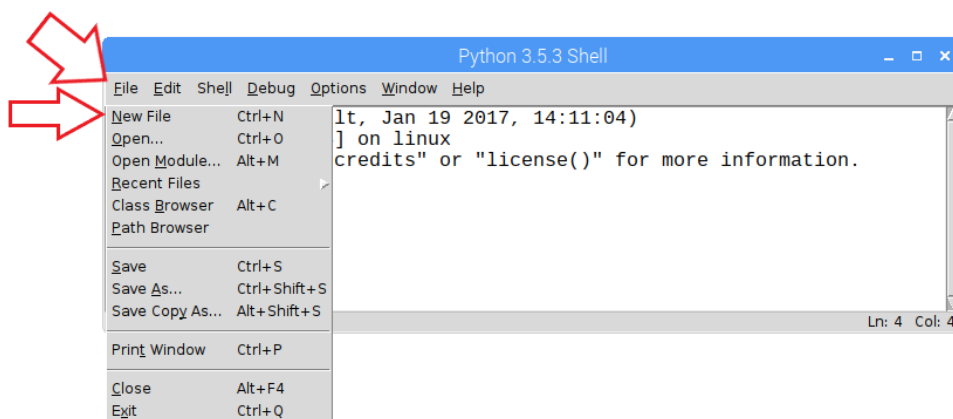
ذخیره یک برنامه

حال بیا ببینیم برنامه‌ای را بنویسیم که نام یک نفر را بپرسد و سپس پیغام تولدت مبارک را برای او چاپ کند. البته برای اینکار مشکلاتی داریم که من آنها را به شما توضیح می‌دهیم و راه حل آن را به شما خواهیم گفت.

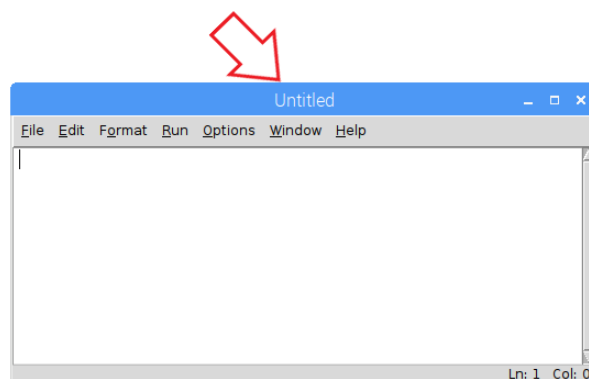
محدودیت مهمی که در «پنجره Shell» (یعنی همین حالتی که بلافاصله بعد از نوشتن یک دستور و زدن کلید Enter آن دستور اجرا می‌شود)، نمی‌توان یک برنامه را ذخیره کرد! متأسفانه اگر پنجره IDLE را ببندید، تمامی دستورها شما از بین می‌رود و مجبور می‌شوید که آنها را مجدد بنویسید و چون می‌خواهیم که این برنامه را برای دوستانمان که قرار است فردا آنها را ببینیم، هم اجرا کنیم، باید آنها را ذخیره کنیم.

راه حل این مشکل خیلی ساده است. ما باید یک فایل جدید بسازیم و برنامه خود را در «پنجره ویرایشگر» تایپ می‌کنیم و سپس برنامه را ذخیره کنیم.

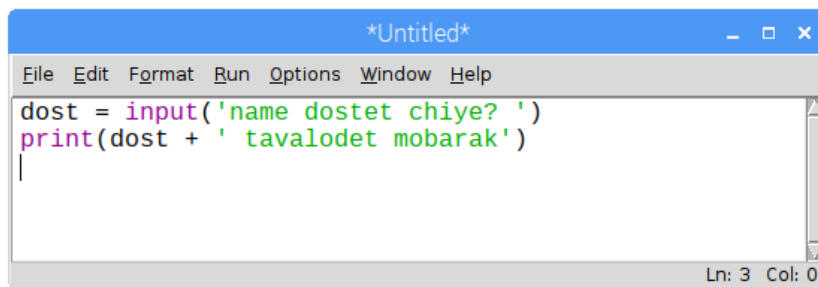
از منوی File گزینه New File را مانند شکل زیر انتخاب کنید



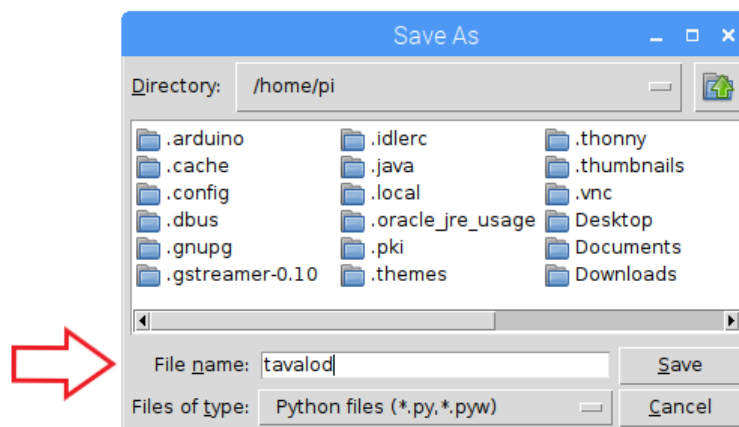
یک صفحه خالی برای نوشتن برنامه برای شما ایجاد خواهد شد و چون هنوز فایل را ذخیره نکرده‌اید، کلمه Untitled را مشاهده می‌کنید. (کلمه Title - بخوانید تایتل- یعنی عنوان و کلمه Untitled - بخوانید آن‌تای‌تلد- یعنی بدون عنوان).



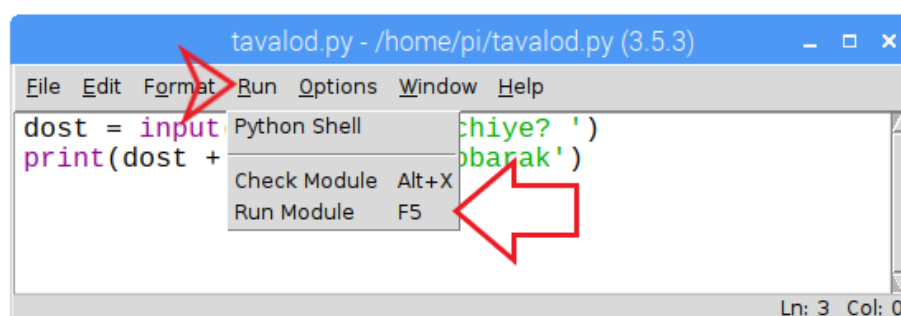
توجه) دقت کنید که خبری از علامت >>> نیست. یعنی در حالت Shell نیستیم و دستورها بلافاصله اجرا نخواهند شد. در این صفحه برنامه‌مان را خواهیم نوشت. بیا ببینیم فعلاً برنامه را بنویسیم و بعد از تکمیل شدن آنرا ذخیره خواهیم کرد.



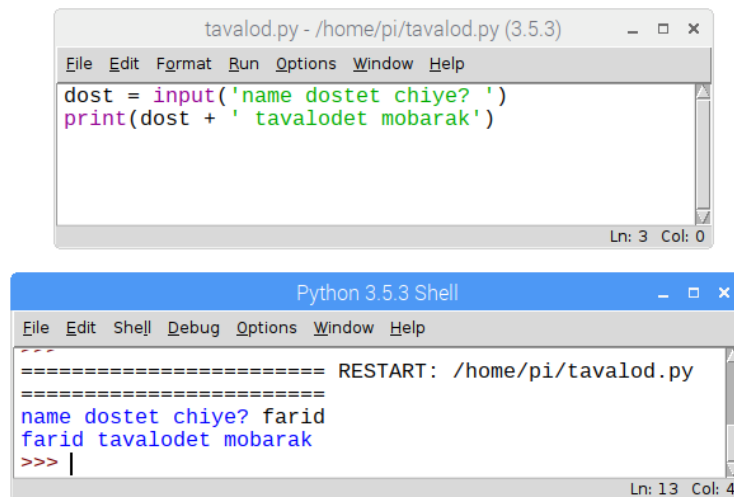
برنامه کامل شد و حال باید فایل را ذخیره کنیم. بنابراین از منوی File گزینه Save را میزنیم. بلافاصله یک پنجره باز می‌شود که از ما نام فایل و محلی که باید این فایل ذخیره شود را می‌پرسد. یک نام دلخواه مثلا tavalod را تایپ کنید و دکمه Save را بزنید.



حال وقت آن است که برنامه را اجرا کنیم. در «پنجره Shell» ما نیازی به اجرای برنامه نداشتیم و با زدن Enter در انتهای در دستور، آن دستور بلافاصله اجرا و نتیجه آن نمایش داده می‌شد. اما الان که برنامه را در یک فایل جدید نوشته‌ایم، باید برنامه را Run کنیم. برای اجرای برنامه از منوی Run گزینه Run Module (بخوانید ران ماژول) را بزنید.



برنامه شما اجرا خواهید شد و ابتدا نام دوست شما را خواهد پرسید و سپس پیغام را برای ما چاپ خواهد کرد. توجه داشته باشید که بعد از اجرای برنامه، پرسیدن نام دوست شما و چاپ شدن پیغام در همان «محیط تعاملی» اتفاق خواهد افتاد. به تصویر زیر دقت کنید.



```
tavalod.py - /home/pi/tavalod.py (3.5.3)
File Edit Format Run Options Window Help
dost = input('name dostet chiye? ')
print(dost + ' tavalodet mobarak')
Ln: 3 Col: 0

Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: /home/pi/tavalod.py =====
name dostet chiye? farid
farid tavalodet mobarak
>>> |
Ln: 13 Col: 4
```

توجه ۱) ما در «حالت تعاملی» می‌توانستیم دستور print را بنویسیم و برنامه بعد از زدن Enter خروجی را به ما

نمایش می‌داد، اما در اینجا باید حتما دستور print را برای چاپ خروجی بنویسیم.

توجه ۲) از این پس هرگاه گفتیم «فایل جدیدی بسازید»، منظور ما این است که:

۱) از File گزینه New را بزنید تا یک فایل جدید ساخته شود.

۲) خبری از علامت >>> نیست و دستورها بلافاصله بعد از زدن Enter اجرا نمی‌شوند.

۳) باید برنامه ابتدا ذخیره و سپس Run کنید.

اگر فراموش کردید که برنامه را ابتدا ذخیره کنید، نگران نباشید، خود برنامه به شما پیغام می‌دهد که ابتدا باید

فایل ذخیره شود.

پرسش) چرا باید برنامه‌ها را ذخیره کنیم؟

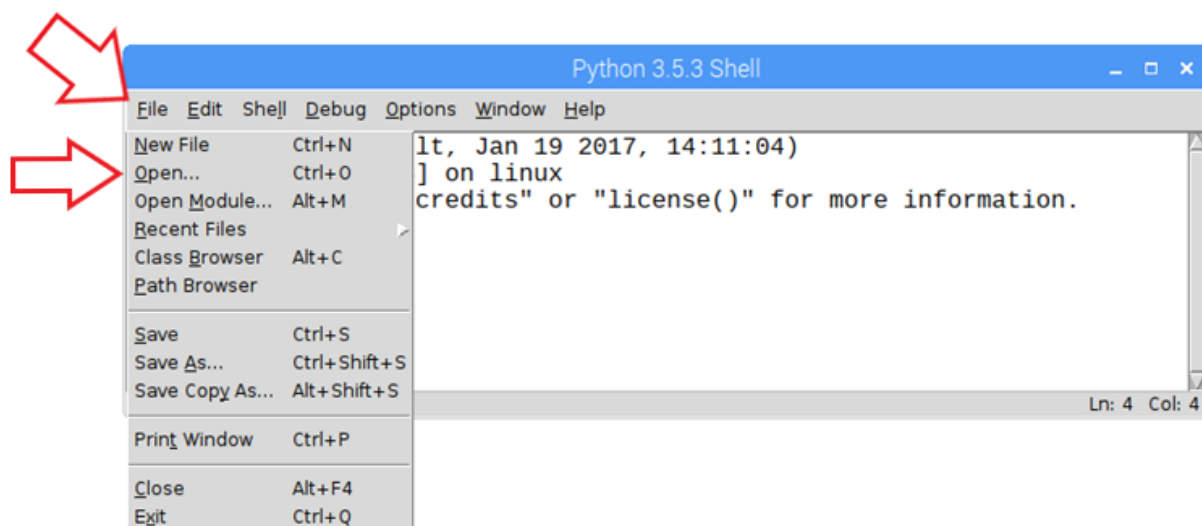
هر چیزی که در کامپیوتر اتفاق می‌افتد در حافظه‌ای به نام RAM (بخوانید رم) قرار دارد. چون RAM بسیار سریع است و می‌تواند با CPU که مغز کامپیوتر است، ارتباط سریعی پیدا کند. و RAM با برق کار می‌کند و به همین دلیل است که خیلی سریع است اما مشکلش آنجاست که اگر کامپیوتر خاموش شود، اطلاعاتش پاک می‌شود و به همین دلیل اطلاعات موجود در RAM را بر روی یک حافظه دیگر که دائمی است، ذخیره کرد.

انتقال اطلاعات از RAM به حافظه دائمی را اصطلاحاً «Save کردن» می‌گوییم و برعکس اینکار یعنی چیزی را از روی حافظه دائمی به حافظه RAM انتقال دادن را Open (بخوانید اُپن) کردن می‌گویند.

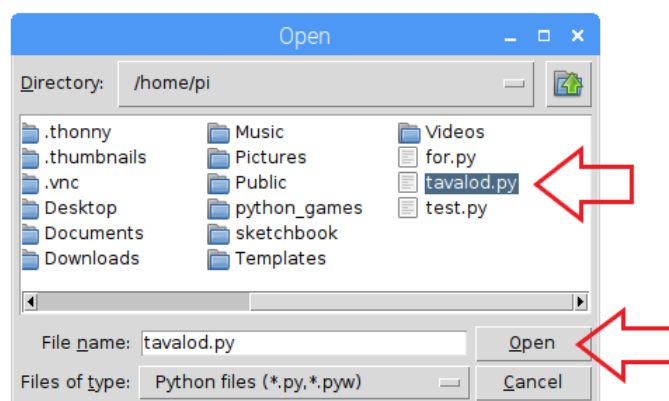
در ضمن به اطلاعاتی که در حافظه دائمی ذخیره شده است، اصطلاحاً «File» (بخوانید فایل) می‌گویند.

باز کردن یک فایل

برای باز کردن یک فایل که قبلاً ذخیره شده است، از منوی File گزینه Open را کلیک کنید.



یک پنجره برای شما نمایش داده می‌شود که در این پنجره باید بر روی نام فایل خود کلیک کنید و سپس دکمه Open را بزنید.



نکته: اگر به دستورهای که تایپ کرده‌اید دقت کنید، متوجه می‌شوید که برخی از کلمات آن رنگی می‌شوند. رنگی شدن کلمات به ما کمک می‌کنند تا درک و خواندن برنامه ساده‌تر شود. من در زیر نام این رنگ‌ها به همراه توضیح مختصری از آنها را برای شما می‌نویسم.

■ ■ ■ رنگ بنفش : دستورها پایتون مانند دستور print بنفش خواهند شد.

■ ■ ■ رنگ آبی : خروجی و یا نتیجه دستورها با این رنگ مشخص می‌شوند.

■ ■ ■ رنگ نارنجی : دستورها ویژه زبان پایتون مانند if و else به این رنگ در می‌آیند. به این دستورها ویژه اصطلاحاً Keyword (بخوانید کی ورد) گفته می‌شود.

■ ■ ■ رنگ سیاه: رنگ سایر دستورها برنامه مانند 2+2 است.

■ ■ ■ رنگ قرمز: اگر نتیجه یا خروجی یک دستور خطا باشد، با این رنگ نمایش داده می‌شود.

■ ■ ■ رنگ سبز: رنگ Stringها (رشته‌ها) که در داخل کوتیشن‌ها قرار گرفته‌اند.

2

فصل دوم،

آشنایی با توابع

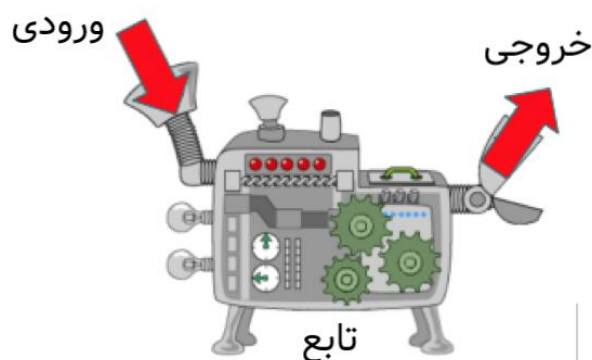
تابع چیست

اگر از شما بپرسم که شباهت قابلمه و ماشین لباسشویی و گوسفند چیست، چه پاسخی می‌دهید. احتمالا فکر می‌کنید که این یک شوخی است و هیچ شباهتی بین قابلمه و ماشین لباسشویی و گوسفند وجود ندارد.

اتفاقا این سه چیز به صورت کلی کاملا شبیه هم هستند. آنها از ما چیزهایی را می‌گیرند و سپس به ما چیزی را می‌دهند:

- ما به قابلمه پیاز، نمک و گوشت و سیب زمینی و حرارت می‌دهیم و در آخر قابلمه به ما آبگوشت می‌دهد.
- ما به ماشین لباسشویی برق و پودر و لباس کثیف می‌دهیم و ماشین لباسشویی به ما لباس تمیز می‌دهد.
- ما به گوسفند آب و علف می‌دهیم و گوسفند به ما شیر می‌دهد.

ما می‌توانیم بگوییم که این سه چیز، تابع هستند. تابع یعنی چیزی که از ما ورودیهایی را بگیرد و سپس به ما خروجی را بدهد. در عکس زیر این مفهوم را به صورت کلی می‌توانید ببیند.



در دنیای کامپیوتر هر تابعی یک اسمی دارد و ورودی‌های تابع در داخل یک پرانتز گذاشته می‌شوند و با علامت کاما از هم جدا می‌شوند. مثلا فرض کنید که اگر پایتون یک تابع به نام «قابلمه» داشت، باید آنرا اینطوری تایپ می‌کردیم:

(حرارت , سیب زمینی , نمک , پیاز) قابلمه >>>

و بلافاصله بعد از زدن Enter به ما آبگوشت را می‌داد. 🍖

معرفی چند تابع ساده

مثال ۱) چاپ کردن یک متن

ما قبلا از فرمان print استفاده کرده‌ایم و حالا می‌خواهیم که به شما بگوییم print یک «تابع» است که کارش این است که چیزی را از ما بگیرد و سپس آنرا در خروجی چاپ کند. به همین سادگی.

```
>>> print('iran')
iran
```

البته می‌توانیم به تابع print چندین ورودی را بدهیم که آنها را برای ما با اضافه کردن یک فاصله بینشان چاپ خواهد کرد.

```
>>> print('salam','iran')
salam iran
```

توجه: به علامت کاما که بین دو متن 'salam' و 'iran' وجود دارد، دقت کنید.

مثال ۲) اسم شما چند حرف دارد؟

برای شمارش تعداد حروف یک متن تابعی به نام len داریم. این تابع یک متن را از ما می‌گیرد و سپس به ما می‌گوید که آن متن از چند حرف تشکیل شده است. بنابراین کافی است که دستور زیر را در «محیط تعاملی» تایپ کنید و بلافاصله بعد زدن Enter عدد 5 که تعداد حروف اسم kaveh است، برای شما چاپ می‌شود.

```
>>> len('kaveh')
5
```

مثال ۳) تابع pow

دستور زیر را تایپ کنید و Enter را بزنید. خروجی آن چی عددی است؟

```
>>> pow(2,4)
```

تابع pow همان توان است و عدد 2 را به توان 4 می‌رساند که عدد 16 خواهد شد.

نکته: ما برای به توان رساندن یک عملگر اختصاصی در زبان پایتون داریم که «**» است. بنابراین به جای آنکه از تابع pow استفاده کنیم، می‌توانید دستور زیر را بنویسید که نتیجه آن یکسان است.

```
>>> 2**4
16
```

مثال ۴) حذف اعشار یک عدد

ما تابعی داریم به نام `int` که اگر عددی اعشار داشته باشد، اعشار آنرا حذف می‌کند و به ما فقط قسمت صحیح آنرا می‌دهد.

```
>>> int(17.5)
17
```

نکته: کلمه `int` مخفف کلمه `integer` است (بخوانید اینتجر) و یعنی عددی که اعشار ندارد که به این اعداد در دنیای کامپیوتر اعداد صحیح گفته می‌شود.

تبدیل یک متن به عدد صحیح

مثال ۵) برنامه‌ای بنویسید که شعاع یک دایره را از کاربر بگیرد و سپس محیط دایره را محاسبه کند. این برنامه یک نکته بسیار مهمی را دارد که باید به شما دقیق توضیح دهم. فرمان `input` که قبلاً با آن آشنا شده‌اید در حقیقت یک تابع است که یک مقدار را از کاربر دریافت می‌کند و به عنوان خروجی، آن مقدار را به ما می‌دهد. اما نکته در خروجی تابع `input` است. خروجی تابع همیشه یک متن (رشته) است. فرض کنید که برنامه زیر را نوشته‌اید و بعد از اجرا مقدار 10 را به عنوان ورودی تایپ می‌کنید.

```
>>> r = input('yk addad vared konid? ')
yek addad vared konid? 10
```

حال اگر مقدار `r` را چاپ کنیم، خواهید دید که 10 در بین دو علامت `' '` قرار دارد و این یعنی این مقدار یک عدد نیست و یک متن است.

```
>>> r
'10'
```

ممکن است که کمی گیج شده باشید که کاملاً حق باشماست زیرا تا به حال با این موضوع برخورد نکرده بودید. بگذارید کمی دقیق‌تر توضیح دهم. در دنیای کامپیوتر بین اعداد و متن‌ها تفاوت جدی وجود دارد و نحوه ذخیره سازی آنها در

حافظه کاملاً متفاوت است. شما می‌توانید مقدار 10 را به کامپیوتر بدهید و از کامپیوتر بخواهید که آنرا در حافظه مانند یک متن (نه یک عدد) ذخیره کند و کامپیوتر اینکار را انجام می‌دهد.

و از این به بعد این 10 که در حافظه به مانند یک «متن» ذخیره شده است، خاصیت عددی ندارد یعنی نمی‌توان روی آن محاسبات انجام داد.

. مثلاً اگر بعد از اجرای دستورها بالا بخواهیم که فرمول زیر را بنویسیم، کامپیوتر متغیر ۲ را یک متن فرض می‌کند و گفته بودیم که عملگر ستاره بر روی یک متن، باعث تکرار آن می‌شود و بنابراین 10 ، چهار بار تکرار می‌شود:

```
>>> r * 4
10101010
```

و یا اگر بخواهیم که مقدار ۲ را بر عدد 2 تقسیم کنیم، خطای زیر را خواهیم دید:

```
>>> x / 2
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    x/2
TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

خط آخر این خطا به ما مشکل را توضیح داده است که بکار بردن عملگر «/» بر روی دو مقدار که یکی عدد و دیگری یک متن، نشدنی است.

اگر بخواهیم که بر روی ۲ محاسبات ریاضی را انجام دهیم، باید آنرا ابتدا به یک عدد تبدیل کنیم، که توسط تابع `int` اینکار به سادگی قابل انجام است:

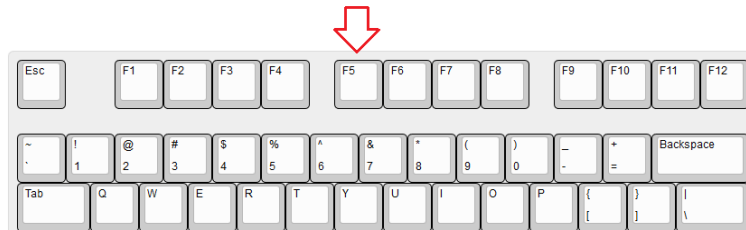
```
>>> int(r) * 4
40
```

به نظر می‌رسد که همه نکته‌ها گفته شد و وقت آن است که خود برنامه را در یک فایل جدید بنویسیم. از File با زدن گزینه New یک صفحه خالی بسازید و برنامه زیر را تایپ کنید.

```
r = input('yk addad vared konid? ')
p = 2 * 3.14 * int(r)
print('mohit = ' , p)
```

سپس فایل را ذخیره کنید و آنگاه با زدن کلید F5 برنامه را اجرا کنید. سپس فایل را ذخیره کنید و آنگاه با زدن کلید F5 برنامه را اجرا کنید. برنامه از شما یک عدد می‌خواهد که آنرا تایپ کنید و سپس محیط دایره را برای شما چاپ می‌کند. نکته ۱: کلید F5 میانبر برای Run کردن یک برنامه است. یعنی به جای آنکه برای اجرای برنامه از منوی Run گزینه Run را بزنید، می‌توانید این کلید را بزنید.

نکته ۲: کلید F12 ، ... ، F2 ، F1 در هر برنامه نقش متفاوتی را دارند. مثلاً در یک بازی کامپیوتری ممکن است که F5 برای شلیک گلوله باشد و در یک برنامه دیگر کار دیگری را انجام دهد.



نکته ۳: اگر مقدار شعاع را یک عدد اعشاری بدهید، تابع `int` قسمت اعشار آنرا حذف خواهد کرد. یعنی محیط دو دایره با شعاع 10 و 10.99 را یک عدد نمایش می‌دهد.

مثال ۶) معرفی تابع `float`

تابع `int` می‌توانید یک مقدار متنی را به یک عدد صحیح تبدیل کند و البته اعشار آنرا حذف خواهد کرد. حال اگر بخواهیم که برنامه محاسبه محیط دایره را طوری تغییر دهیم که شعاع یک عدد اعشاری باشد، باید از تابع `float` (بخوانید فلوت) استفاده کنیم. تابع `float` یک مقدار را به یک عدد اعشاری تبدیل می‌کند.

```
r = input('yk addad vared konid? ')
p = 2 * 3.14 * float(r)
print('mohit =' , p)
```

حال می‌توانید یک عدد اعشاری را به عنوان ورودی بدهید و محیط دقیق دایره را محاسبه کنید.

نکته مهم: در زبان برنامه نویسی پایتون یک عدد یا صحیح است و یا دارای اعشار. به اعداد صحیح `integer` و به اعداد اعشاری `float` می‌گوییم.

مثال ۷) تبدیل یک عدد به متن

تا اینجا دیدیم که با تابع `int` و `float` می‌توان یک متن را به عدد تبدیل کرد. گاهی نیاز داریم که برعکس این کار را انجام دهیم. یعنی یک عدد را به یک متن تبدیل کنیم. اینکار با تابع `str` انجام می‌شود.

```
>>> x= 2
```

```
>>> str(x)
'2'
>>> str(x) * 10
'2222222222'
```

حتما با خودتان می‌پرسید که این کار چه فایده‌ای دارد، خوب اجازه دهید فعلا آنها را یاد بگیریم و در برنامه‌های بزرگتر کاربرد این توابع را بهتر متوجه خواهیم شد. فعلا حوصله کنید.

3

فصل سوم - آشنایی با

ماژول‌ها

ماژول چیست؟

فرض کنید که می‌خواهید برنج درست کنید و برای اینکار وسایل زیر را لازم دارید:

(۱) برنج (۲) نمک (۳) روغن (۴) آب (۵) قابلمه ۷ (اجاق گاز

دقت کنید که هیچ کدام از این وسایل را خود شما نساخته‌اید و همه آنها را به صورت آماده دارید. یعنی شما نه برنج را خودتان کاشته‌اید و نه نمک را از دریا خودتان گرفتید و نه روغن را تولید کرده‌اید و نه آب را تصفیه کرده‌اید. در واقع این وسایل از قبل توسط دیگران برای شما آماده و ساخته شده است و تنها کاری که شما باید انجام دهید استفاده از این وسایل برای پختن برنج است.

در زبان پایتون به این چیزهایی که از قبل آماده شده است، «ماژول» می‌گوییم. یعنی لازم نیست همه برنامه‌ها را خودمان از اول بنویسیم، بلکه برخی از برنامه‌ها برای ما از قبل نوشته شده است و ما می‌توانیم از این برنامه‌های آماده - که به آنها ماژول می‌گوییم- استفاده کنیم. استفاده از ماژول‌ها کار ما را بسیار ساده‌تر و سریع‌تر خواهند کرد.

ماژول به انگلیسی اینگونه نوشته می‌شود: module

معرفی ماژول math

با استفاده از ماژول math می‌توانیم کارهای ریاضی را به سادگی انجام دهید. مثلاً به سادگی می‌توانیم بزرگترین مقسوم علیه مشترک (ب م م) دو عدد را پیدا کنیم.

قبل از استفاده از یک ماژول، ابتدا باید به پایتون بگویید که قصد دارید از آن استفاده کنید و برای اینکار از دستور import (بخوانید ایمپورت و به معنی وارد کردن) استفاده می‌کنیم. و دستور زیر را باید تایپ کنیم:

```
>>> import math
```

بعد از تایپ این دستور می‌توانید از توابع ماژول math استفاده کنید و تابع gcd برای شما ب م م دو عدد را می‌یابد. دستور زیر را برای یافتن ب م م دو عدد 12 و 16 تایپ کنید.

```
>>> math.gcd(12 ,16)
8
```

نکته ۱) به علامت نقطه که بین نام ماژول و نام تابع است دقت کنید. در ضمن ما نام تابع را به تنهایی ننوشتیم و قبل از آن نام ماژول را هم اضافه کردیم تا پایتون بتواند بفهمد که این تابع را از کدام ماژول باید پیدا کند.

نکته ۲) اگر مایلید که بدانید در ماژول math چه توابع دیگری وجود دارد، دستور زیر را تایپ کنید:

```
>>> help(math)
```

نکته ۳) کلمه math مخفف کلمه mathematical (بخوانید مَتَمَتیکال) است و معنی آن «ریاضی» می‌شود.

همچنین می‌توانیم با تابع sqrt، جذر یک عدد را محاسبه کنیم:

```
>>> math.sqrt(25)
5.0
```

معرفی ماژول random

یکی از کارهای ماژول random (بخوانید رندوم) تولید اعداد شانس برای ما است و در بازی‌های کامپیوتری از اعداد شانس زیاد استفاده می‌شود.

ابتدا به پایتون می‌گوییم که قرار است از ماژول random استفاده کنیم:

```
>>> import random
```

و سپس از تابع randint برای تولید یک عدد شانس بین 0 تا 100 استفاده می‌کنیم:

```
>>> random.randint(0,100)
```

84

این عدد چون یک عدد شانس است برای شما چیز دیگری نمایش داده می‌شود.

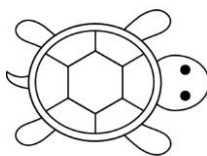
نکته ۱) هر بار که دستور قبل را اجرا کنید یک عدد شانس جدیدی برای شما تولید و نمایش داده می‌شود.

نکته ۲) برای اجرای مجدد یک دستور که قبلاً در «حالت تعاملی» تایپ شده است، می‌توانید کلید ALT+p را بزنید. (یعنی

کلید ALT را پایین نگه دارید و سپس کلید p را بزنید تا آن دستور برای شما مجدد نوشته شود).

معرفی ماژول turtle

یکی از جالب‌ترین ماژول‌ها برای شما turtle (بخوانید تِرْتل) است. این ماژول برای آموزش برنامه‌نویسی به بچه‌ها تهیه شده است. ماژول turtle یک لاکپشت است که ما به او فرمان‌هایی را می‌دهیم و آن لاکپشت فرمان‌های ما را اجرا می‌کند و ما می‌توانیم لاکپشت و نتیجه فرمان را به صورت یک تصویر متحرک ببینیم.



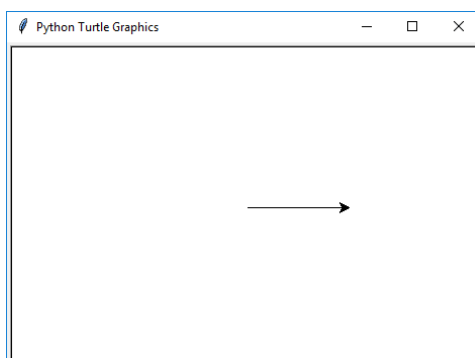
ابتدا به پایتون بگویید که می‌خواهید از ماژول turtle استفاده کنید:

```
>>> import turtle
```

و سپس به لاکپشت بگویید که به اندازه 100 تا حرکت کند:

```
>>> turtle.fd(100)
```

بلافاصله یک پنجره سفید رنگ برای شما باز می‌شود که در آن لاکپشت به شکل یک فلش نمایش داده شده است و به اندازه 100 واحد حرکت خواهد کرد و لاکپشت برای ما با مدادی که دارد، مسیری که حرکت کرده است را مشخص می‌کند.



نکته ۱) برای اندازه‌گیری یک میز و یا قد یک دانش‌آموز از واحد سانتی متر استفاده می‌کنیم. اما در کامپیوتر و یا روی مانیتور، برای اندازه‌گیری تصاویر از واحدی به نام پیکسل استفاده می‌شود. چون لاکپشت قرار است که روی مانیتور حرکت کند، ما به او گفتیم که 100 پیکسل حرکت کن.

مثال ۱) رسم یک مربع با لاکپشت

برای رسم مربع به لاکپشت می‌گوییم دستورها زیر را به ترتیب انجام دهد.

- 100 پیکسل به جلو حرکت کن
- 90 درجه به سمت چپ بچرخ
- 100 پیکسل به جلو حرکت کن
- 90 درجه به سمت چپ بچرخ
- 100 پیکسل به جلو حرکت کن
- 90 درجه به سمت چپ بچرخ
- 100 پیکسل به جلو حرکت کن

از File گزینه New را می‌زنیم تا به حالت اسکریپت برویم. حال دستورها زیر را تایپ می‌کنیم:

```
import turtle
turtle.fd(100)
turtle.left(90)
turtle.fd(100)
turtle.left(90)
turtle.fd(100)
turtle.left(90)
turtle.fd(100)
```

حال برنامه را ذخیره کنید و سپس با زدن F5 آن را اجرا کنید. در پنجره لاکپشت برای شما یک مربع را رسم می‌کند.

حالا بیایید این برنامه را کمی تغییر دهیم تا جالب تر شود.

- شکل فلش را به یک لاکپشت واقعی تغییر دهیم. از دستور shape (بخوانید شی‌پ) استفاده می‌کنیم.
- لاکپشت کمی آهسته تر حرکت کند. از دستور speed (بخوانید اسپید) استفاده می‌کنیم.

برنامه ما به شکل زیر خواهد شد:

```
import turtle
turtle.shape('turtle')
turtle.speed(1)
turtle.fd(100)
turtle.left(90)
turtle.fd(100)
turtle.left(90)
```

```
turtle.fd(100)
turtle.left(90)
turtle.fd(100)
```

نکته: معنی کلمه shape یعنی «شکل» و کلمه Speed یعنی «سرعت».

مثال ۲) تغییر رنگ و ضخامت خطها

گفتیم که در دست این لاکپشت یک مداد است و لاکپشت هنگام حرکت، با این مداد مسیرش را مشخص می‌کند. حال می‌خواهیم که رنگ این مداد و ضخامتش را نیز تنظیم کنیم.

برای تغییر رنگ مداد لاکپشت از فرمان pencolor و برای تغییر ضخامت نوک مداد از فرمان pensize استفاده می‌کنیم. (کلمه pen یعنی «مداد» و color (بخوانید کالر) یعنی «رنگ» و کلمه size (بخوانید سایز) یعنی «اندازه») بیاید یک مثلث متساوی الاضلاع (یعنی سه ضلع آن برابر باشد) بکشیم و که ضلع آن با یک ضخامت و یک رنگ باشد.

```
import turtle
```

```
turtle.pencolor('red')
turtle.pensize(2)
turtle.fd(100)
```

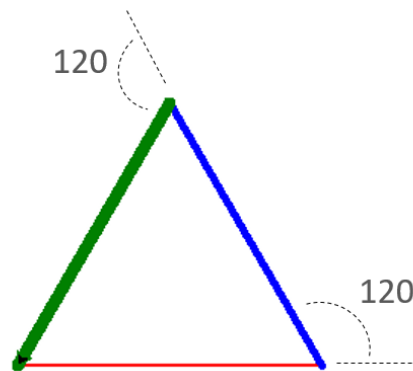
```
turtle.pencolor('blue')
turtle.pensize(5)
turtle.left(120)
turtle.fd(100)
```

بین سطرهایی که باعث رسم هر ضلع می‌شوند، یک فاصله خالی قرار داریم تا کد خواناتر باشد. گذاشتن این فاصله اختیاری است.

```
turtle.pencolor('green')
turtle.pensize(8)
turtle.left(120)
turtle.fd(100)
```

توجه ۱) همانطور که در این برنامه می‌بینید من هیچ علامت >>> را نگذاشته‌ام. یعنی این برنامه را در داخل یک فایل جدید باید تایپ و سپس ذخیره و سپس اجرا کنید.

توجه ۲) من برای شما زاویه‌های چرخش لاکپشت را در شکل زیر نشان داده‌ام تا بهتر بتوانید آن را درک کنید. اگر باز هم متوجه نشدید که چرخش‌ها چگونه محاسبه می‌شود از معلم ریاضیات کمک بگیرید.



آشنایی با مفهوم مُتد

در برنامه نویسی ما با مُتدها سر و کار داریم. معنی متد چیز اصلا پیچیده‌ای نیست و الان که لاکپشت را می‌شناسید می‌توانیم به سادگی به شما این معنا را بگوییم.

دستورهای مانند fd و left همگی متد هستند. یعنی به لاکپشت «دستور» می‌دهیم که کاری را انجام دهد. مثلا اگر بنویسیم turtle.fd(100) یعنی لاکپشت به مقدار 100 پیکسل «برو جلو». و اصلاحا می‌گوییم که fd یک متد است و بر روی چیزی به نام لاکپشت اجرا می‌شود.

امیدوارم که از این مفاهیم و یا واژه‌های جدید خسته نشوید زیرا من می‌خواهم که شما را با دنیای برنامه نویسی آشناتر کنم تا برای خواندن کتاب‌های دیگر آماده شوید و اگر این واژه‌ها را دیدید، نترسید و گیج نشوید. از این به بعد هر جا که واژه «متد» را گفتم، یعنی «دستور» یا «فرمانی» که قرار است روی چیزی اجرا شود. نکته) واژه «متد» یک کلمه انگلیسی است که اینگونه نوشته می‌شود: Method

مثال ۳) متد circle برای رسم دایره

لاکپشت ما یک متدی دارد به نام circle (بخوانید سیرکل) که باعث می‌شود لاکپشت برای ما یک دایره رسم کند. بنابراین در «مد تعاملی» ابتدا به پایتون می‌گوییم که قرار است از ماژول turtle استفاده کنیم و سپس با متد circle یک دایره رسم می‌کنیم.

```
>>> import turtle
>>> turtle.circle(150)
```

نکته ۱) اگر یکبار در «مد تعاملی» ماژولی را import کرده باشیم، دیگر لازم نیست آن ماژول را مجدد import کنیم. البته در «مد اسکریپت» (یعنی وقتی که یک File جدید می‌سازید) اینگونه نیست و در هر فایل جدیدی که می‌سازید، باید دستور import را بنویسید.

4

فصل چهارم - حلقه‌ها

می‌خواهیم که برنامه‌های جالب‌تری بنویسیم و به همین دلیل باید دستورها مهم پایتون را یاد بگیریم. یکی از این دستورها مهم، ساخت حلقه است.

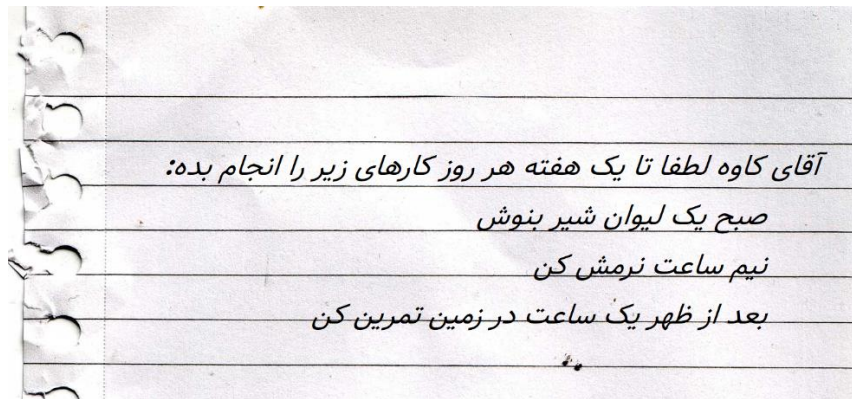
بگذارید یک مثال ساده از حلقه‌ها بزنم تا منظورم را دقیقاً متوجه شوید. هر روز خورشید از مشرق بیرون می‌آید و در مغرب غروب می‌کند. هر روز زمین به دور خودش یک دور کامل می‌چرخد. زمین سال‌هاست که به دور خورشید می‌چرخد. در همه این مثال‌ها یک چیز مشترک وجود دارد. آن چیز اتفاقی که با یک نظم خاص بارها و بارها تکرار می‌شود که در کامپیوتر به این تکرارهای پیاپی اصطلاحاً «حلقه» می‌گوییم.

ما در برنامه نویسی هم لازم داریم که دستوری بارها و بارها تکرار شود و اینکار با حلقه‌ها به سادگی قابل انجام است. مثلاً می‌خواهیم صد دایره رسم کنیم و برای اینکار یک حلقه با صد بار تکرار می‌سازیم و در آن حلقه دستور رسم دایره را می‌دهیم بنابراین در هر باری که تکرار می‌شود، برای ما یک دایره رسم کند و در آخر کار ما صد دایره داریم.

لطفاً این فصل را ابتدا یکبار تا آخر بخوانید و سپس برگردید و تمرین‌های آن را انجام دهید زیرا در تایپ دستورها نکاتی ظریفی وجود دارد که به تدریج به شما آموزش خواهیم داد.

شکل کلی حلقه در پایتون

اجازه دهید با یک مثال خیلی ساده منظورم را بگویم. شما برای شرکت در مسابقات فوتبال مدرسه انتخاب شده‌اید و مربی می‌خواهد که شما برای مسابقه کاملاً آماده باشید و این دستورها را بر شما می‌نویسد.



اگر خوب دقت کنید می‌بینید که :

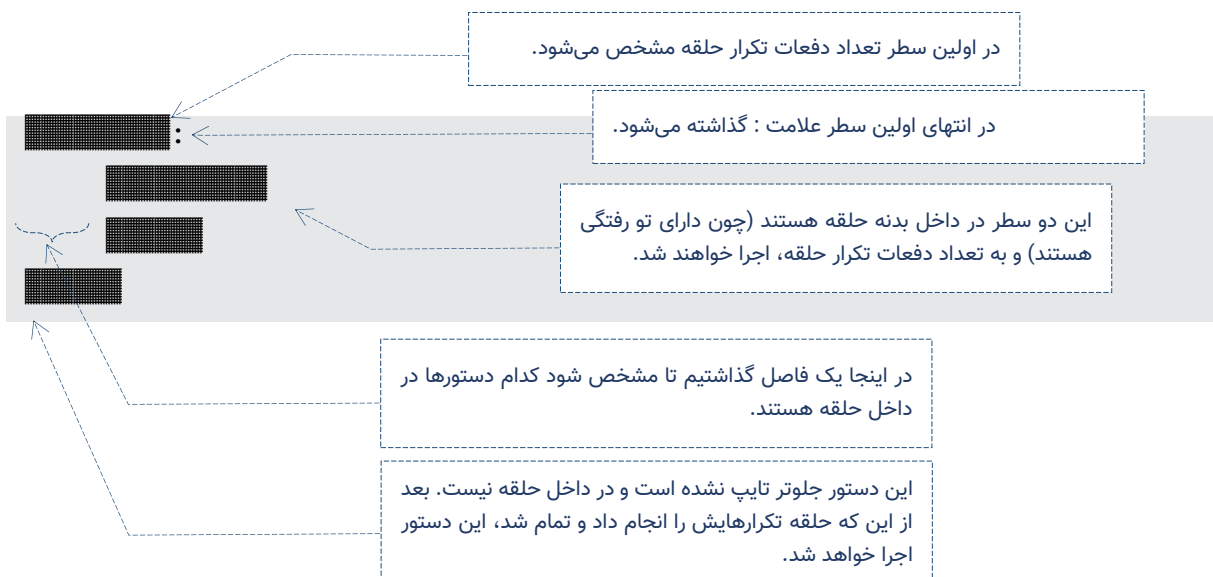
(۱) مربی از کاوه خواسته که کارهایی را ۷ روز انجام دهد. مربی در انتهای سطر اول، علامت دو نقطه گذاشته است و این علامت نشان می‌دهد که این سطر ادامه دار است.

(۲) جملات سطرهای دوم، سوم و چهارم کمی جلوتر نوشته شده‌اند و نشان دهنده کارهایی است که کاوه باید هر روز انجام دهد. این شکل نوشتن دستورها ترکیبی زیبا است و ما به شکل واضحی می‌فهمیم که منظور مربی انجام چه کارهایی است.

حلقه‌ها در پایتون دستورها ترکیبی هستند:

(۱) یعنی به پایتون می‌گوییم که باید چند بار باید حلقه تکرار شود. در انتهای این سطر علامت : را می‌گذاریم.
(۲) مشخص می‌کنیم که دستورهای که باید در هر بار تکرار حلقه انجام شوند. این دستورها را کمی جلوتر تایپ می‌کنیم. (به این کمی جلوتر تایپ کردن، اصطلاحاً «تو رفتگی» می‌گویند)

دستورها فرضی زیر، صورت کلی ایجاد حلقه‌ها را در پایتون به شما نشان می‌دهد:



آشنایی با دستور for

در زبان پایتون یک حلقه با دستور for (بخوانید فور) ساخته می‌شود. بیایید برنامه ساده زیر را بنویسیم تا نتیجه آنرا ببینیم و من توضیحات آنرا بعد از مشاهده نتیجه به شما خواهم داد.

تعداد دفعات تکرار حلقه 5 بار است.

```
>>> for i in range(5):  
    print('hi')
```

دقت کنید که علامت : در اینجا وجود دارد و باید آنرا تایپ کنید.

این تو رفتگی (فاصله خالی جلوی دستورها) به صورت اتوماتیک ایجاد می‌شود و نشان دهنده دستورهایی است که متعلق به حلقه است و این دستورها چندین بار تکرار خواهد شد.

بعد از نوشتن این دستور، باید 2 بار کلید Enter را بزنید تا از دستور ترکیبی for خارج شوید و دستور شما اجرا شود.

نتیجه این برنامه در تصویر زیر قابل مشاهده است:

```
>>> for i in range(5):  
    print('hi')  
  
hi  
hi  
hi  
hi  
hi  
>>>
```

چون اولین بار است که یک حلقه را می‌بینید، لطفاً به توضیحاتی که می‌دهم دقت کنید.

با دستور `for i in range(5)` ما یک حلقه می‌سازیم که قرار است 5 بار اجرا شود. بعد از نوشتن این سطر باید بگوییم که چه دستورهایی قرار است در این حلقه قرار بگیرد و اگر دقت کنید بعد از علامت تایپ «:» و زدن Enter، می‌بینیم که به صورت خودکار یک فاصله خالی (تو رفتگی) برای ما ایجاد می‌شود. این فاصله خالی مشخص می‌کند که بدنه حلقه ما شامل چه دستورهایی می‌باشد و هر چیزی که در این بدنه تایپ شود، 5 بار اجرا خواهد شد.

ما دستور `print('hi')` را در این بدنه نوشتیم و بعد از اینکه Enter را زدیم، به ابتدای سطر بعدی رفتیم و مجدد Enter را باید بزنیم تا از بدنه حلقه for خارج شویم و دستور ما اجرا شود. نتیجه این حلقه این است که دستور `print('hi')` پنج بار اجرا شد و ما پنج پیغام hi را مشاهده می‌کنیم.

توجه) لطفاً عجله نکنید چون با مثال‌های بیشتری به شما کمک می‌کنم تا دستور for را بهتر یاد بگیرید.

مثال ۲) رسم یک مستطیل

می‌خواهیم که با علامت x و o ، یک مستطیل به شکل زیر ایجاد کنیم.

```
XXXXXXXXXXXXXXXXXXXXX
OOOOOOOOOOOOOOOOOOO
XXXXXXXXXXXXXXXXXXXXX
OOOOOOOOOOOOOOOOOOO
XXXXXXXXXXXXXXXXXXXXX
OOOOOOOOOOOOOOOOOOO
```

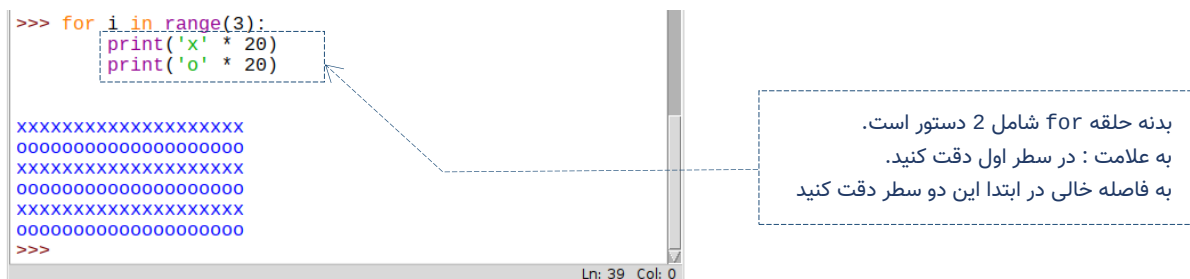
یادآوری) قبلاً گفته بودیم که اگر یک متن را در یک عدد ضرب کنیم، حاصل تکرار آن متن است مثلاً :

```
>>> 'x' * 20
XXXXXXXXXXXXXXXXXXXXX
```

حال برنامه رسم این مستطیل به شکل زیر خواهد بود:

```
>>> for i in range(3):
    print('x' * 20)
    print('o' * 20)
```

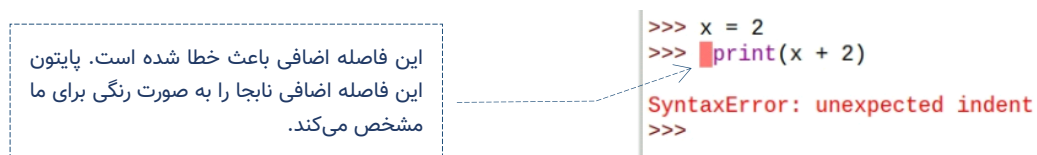
در این مثال حلقه ما که با دستور for ساخته شده است دارای 2 دستور است و این دو دستور سه مرتبه اجرا خواهند شد و خروجی آن شکلی است که خواسته بودیم.



نکته) ما بعد از نوشتن دستور دوم یعنی `print('o' * 20)` باید 2 بار Enter را بزنیم تا از بدنه حلقه خارج شویم.

در مورد تو رفتگی‌ها بیشتر بدانیم

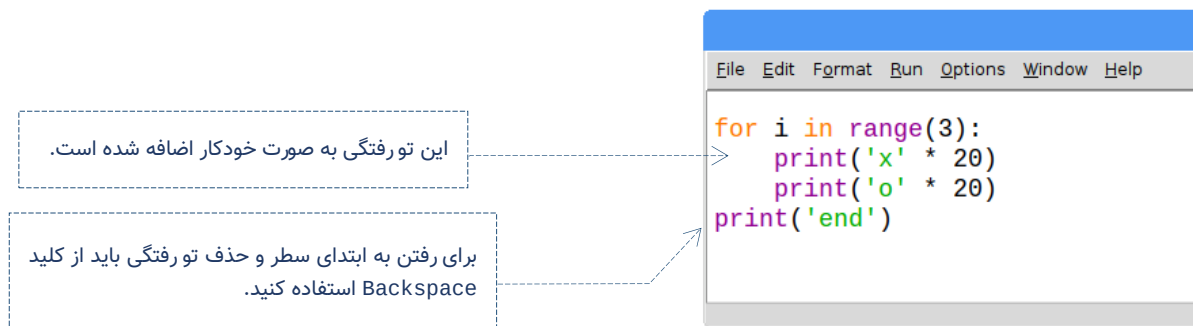
در پایتون فاصله یا فاصله‌هایی که در ابتدای یک سطر اضافه می‌کنیم دارای معنا است و اضافه یا کم گذاشتن آن باعث خطا در اجرای برنامه خواهد شد. به تصویر زیر دقت کنید که قبل از `print` ما یک فاصله اضافه قرار داده‌ایم و چون این فاصله در اینجا معنایی ندارد، پایتون به ما خطا می‌دهد:



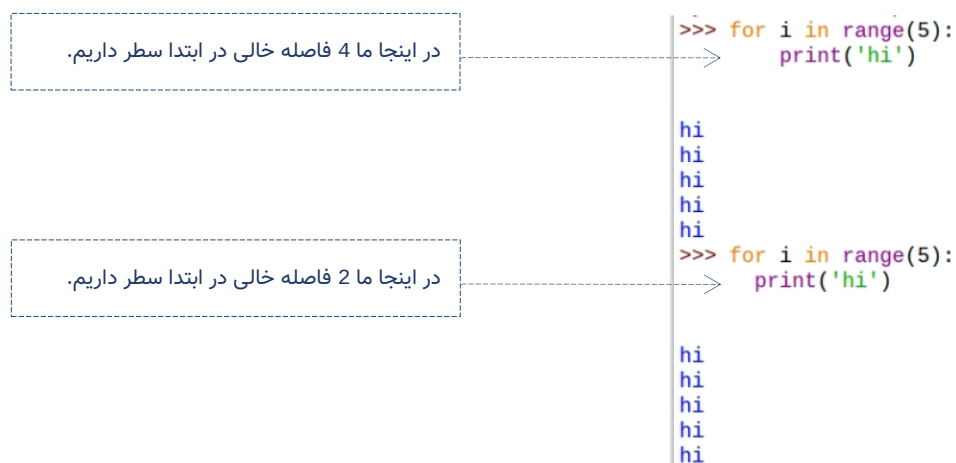
اگر به خطا توجه کنید SyntaxError است یعنی یک خطای املايي دارید و دليل اين خطا را پايتون به شما گفته است. indent (بخوانيد اين دنت) يعني تو رفتگي و unexpected (بخوانيد آن اِکس پِکْتِد) يعني غير منتظره و معنای آن اين است که یک تورفتگی اضافه وجود دارد.

دستورهای که در داخل حلقه for قرار می گیرند باید دارای تو رفتگی باشند یعنی کمی جلوتر تایپ شوند تا پايتون متوجه شود که اين دستورها را باید چندین بار اجرا کند.

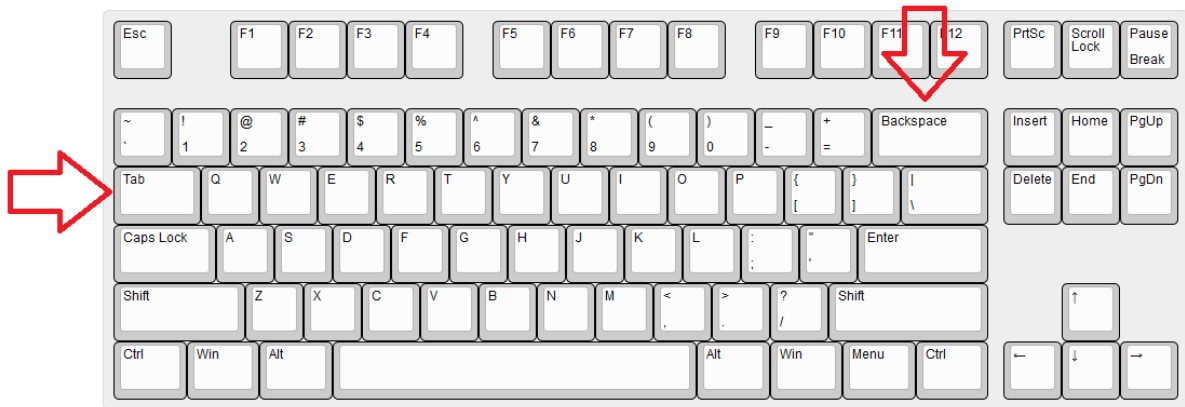
خوشبختانه در IDLE بعد از نوشتن دستور :for i in range(5) و زدن Enter ، به صورت خودکار برای ما تو رفتگی ایجاد می شود.



شاید بپرسید که برای ایجاد تو رفتگی چند فاصله باید در ابتدای سطر قرار دهیم؟ پاسخ ساده است، هر چند تا که دلتان می خواهد. اما به صورت استاندارد معمولا 4 فاصله قرار داده می شود. در تصویر زیر می بینید که یکبار تو رفتگی با 4 فاصله و یکبار با 2 فاصله ایجاد شده است و هر دو دستور بدون خطا اجرا شده اند:



نکته ۱: اگر کلید Tab (بخوانید تب) را بزنید به صورت خودکار برای شما یک تورفتگی به اندازه 4 فاصله ایجاد می شود. در ضمن اگر بخواهیم که از یک تو رفتگی را حذف کنیم باید از کلید Backspace (بخوانید بَک اسپیس) استفاده کنیم. در شکل زیر جای کلید Tab و Backspace بر روی کیبورد را می توانید ببینید:



نکته ۲) تعداد فاصله‌های خالی باید در یک دستور ترکیبی یکسان باشد در غیر اینصورت پایتون به ما خطا می‌دهد.

```
>>> for i in range(3):
    print('x' * 20)
    print('o' * 20)
SyntaxError: unexpected indent
>>>
```

آشنایی دستور range

هنوز نکات مهمی در مورد دستور for وجود دارد که باید با آن نکات هم آشنا شوید. بگذارید به اولین برنامه‌ای که نوشتیم، نگاهی دقیق‌تر کنیم:

```
>>> for i in range(5):
    print('hi')
```

دیدیم که دستور `print('hi')`، پنج بار اجرا می‌شود اما چیزهای دیگری هم در این دستور وجود دارد که هنوز آنها را نمی‌دانیم، مثلاً منظور از دستور `range(5)` چیست؟

باید بگوییم که دستور `range` هم یکی از دستورها مهم زبان پایتون است. دستور `range(5)` باعث می‌شود تا پایتون از عدد 0 تا 4 برای ما به ترتیب تولید کند. بنابراین این دستور اعداد 0, 1, 2, 3, 4 را تولید خواهد کرد.

نکته ۱) دقت کنید که خود عدد 5 را نداریم بلکه «تا» عدد 5 را داریم.

نکته ۲) در زبان پایتون شمارش‌ها از عدد 0 شروع می‌شوند.

قبلاً با متغیرها هم آشنا شدیم و حالا می‌توانیم این اعداد را به ترتیب در یک متغیر (به نام دلخواه) قرار دهیم که در این دستور نام متغیر ما `i` است که :

- در اولین بار، که حلقه اجرا می‌شود این متغیر `i` برابر عدد 0 است.
- سپس در دومین تکرار حلقه، متغیر `i` برابر عدد 1 است.

- سپس در سومین تکرار حلقه، متغیر `i` برابر عدد 2 است.
- سپس در چهارمین تکرار حلقه، متغیر `i` برابر عدد 3 است.
- سپس در پنجمین تکرار حلقه، متغیر `i` برابر عدد 4 است.

بنابراین به سادگی می‌توانیم با تغییر این برنامه به شکل زیر مقدار متغیر `i` را چاپ کنیم:

```
>>> for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

مقدار متغیر `i` چاپ می‌شود.

مثال ۳) چاپ توان 2 اعداد 0 تا 100

برای اینکار توسط `for` اعداد 0 تا 100 را تولید می‌کنیم و سپس از متغیری که این اعداد را نگهداری می‌کند، برای محاسباتمان استفاده می‌کنیم. این برنامه به شکل زیر است:

```
>>> for x in range(100):
    print(x, x**2)
```

```
0    0
1    1
2    4
```

نام متغیر دلخواه است و در این برنامه نام متغیر `x` گذاشته ایم.

مثال ۴) چاپ اعداد 2 رقمی

یکی از قابلیت‌های خوب دستور `range` این است که ما می‌توانیم شروع و پایانی را برایش مشخص کنیم تا اعدادی در آن بازه برای ما تولید شود. برای تولید اعداد دو رقمی (یعنی از عدد 10 تا 99) از دستور `range` به شکل زیر استفاده می‌کنیم:

```
>>> for x in range(10,100):
    print(x)
```

اولین عدد ما 10 خواهد بود.

مثال ۵) چاپ اعداد زوج بین 0 تا 100

در حالت عادی دستور range برای ما اعداد را به ترتیب ایجاد می‌کند. اگر بخواهیم که اعداد به ترتیب نباشند و مثلاً اعداد 0, 2, 4, 6, 8 تولید شوند، یعنی هر عدد از عدد قبلیش 2 واحد زیادتر باشد (در ریاضی اصطلاحاً می‌گوییم قدر نسبت 2)، دستور range به شکل زیر بکار می‌رود:

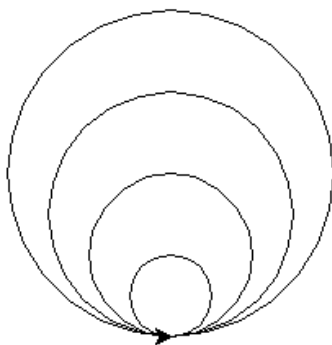
```
>>> for x in range(0,100,2):  
    print(x)
```

مثال ۶) کشیدن دایره‌های تو در تو

می‌خواهیم که چهار دایره تو در تو با turtle بکشیم. برای اینکار برنامه زیر را در یک «فایل» بنویسید و آنرا Run کنید.

```
import turtle  
for r in range(5,21,5):  
    turtle.circle(r*5)
```

بباید برنامه را بررسی کنیم، دستور range(5, 21, 5)، از عدد 5 شروع می‌کند و با قدر نسبت 5، برای ما تا عدد 21، اعدادی را تولید می‌کند که این اعداد عبارتند از 5, 10, 15, 20. اولین عدد یعنی 5 در متغیر r قرار می‌گیرد و سپس دستور به لاکپشت می‌گوییم که یک دایره با شعاع r رسم کن، بنابراین اولین دایره (به شعاع 5) رسم می‌شود. سپس عدد بعدی یعنی 10 در متغیر r قرار می‌گیرد و به لاکپشت می‌گوییم که یک دایره به شعاع r رسم کن و برای ما دومین دایره (به شعاع 10) رسم می‌شود. و سپس نوبت عدد 15 می‌شود و همین کارها برای آن تکرار می‌شود و بعد از آن نوبت عدد 20 می‌شود که آخرین دایره رسم خواهد شد. تمرین) بچه‌ها لطفاً اعداد بالا را خودتان چندین بار عوض کنید و برنامه را اجرا کنید و لذت ببرید.



مثال ۷) برنامه رسم مربع

می‌خواهیم به لاکپشت بگوییم که یک مربع ضلعی را رسم کند.

بنابراین لاکپشت باید چهار بار دستورها زیر را انجام دهد:

به اندازه 100 پیکسل به جلو حرکت کن.

به اندازه 90 درجه به سمت چپ بچرخ

و برنامه آن به شکل زیر خواهد شد:

```
import turtle
for x in range(4):
    turtle.fd(100)
    turtle.left(90)
```

مثال ۸) برنامه رسم یک چند ضلعی منتظم

می‌خواهیم کمی برنامه سخت‌تری را بنویسیم. قرار است که برنامه از ما تعداد اضلاع یک چند ضلعی را بپرسید و سپس

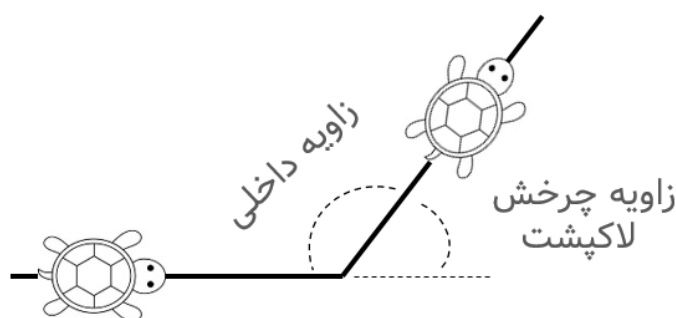
لاکپشت برای ما آن را رسم کند. تقریباً این برنامه شبیه رسم مربع است با این تفاوت که باید به جای چهار بار تکرار، به

تعداد بیشتری تکرار انجام شود و لاکپشت به جای چرخش 90 درجه‌ای، به اندازه دیگری بچرخد.

باید اینجا کمی به ریاضی مراجعه کنیم و ریاضیات به ما می‌گوید:

$$\text{زاویه داخلی یک } n \text{ ضلعی} = (n-2) \times 180^\circ / n$$

$$\text{زاویه داخلی} = 180 - \text{زاویه چرخش لاکپشت}$$



این برنامه به شکل زیر خواهد شد:

```
import turtle

n = input('tedad azla ra vared konid ')
n=int(n)
z = 180 - (n-2) * 180/n

for x in range(n):
    turtle.fd(50)
    turtle.left(z)
```

شرح دقیق این برنامه:

اولین سطر: ماژول turtle را import می‌کنیم.

سطر دوم: از کاربر یک مقدار را می‌گیریم و سپس آنرا در متغیر n ذخیره می‌کنیم.

سطر سوم: می‌دانیم تابع input به ما یک «متن» می‌دهد بنابراین آنرا باید به یک عدد تبدیل کنیم.

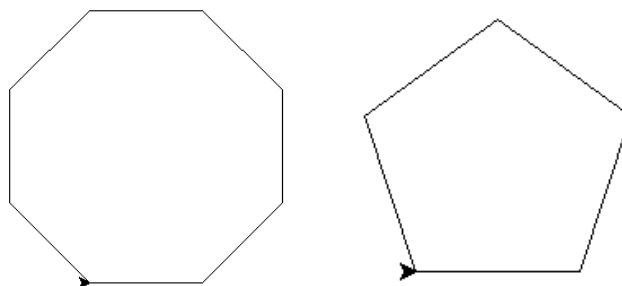
این اولین بار است که دستوری مانند `n=int(n)` را می‌بینید. به شما باید بگویم که همیشه دستورها سمت راست علامت مساوی اول اجرا می‌شوند بنابراین پایتون ابتدا `int(n)` را محاسبه می‌کند که پاسخ آن یک عدد خواهد شد و سپس این عدد را در متغیر n ذخیره می‌کند.

سطر چهارم: عدد Z زاویه چرخش لاکپشت است که با توجه به فرمول‌هایی که گفته شده است محاسبه می‌شود. اگر متوجه این فرمول‌ها نشدید، حتماً از معلم ریاضی یا یکی از همکلاسی‌های خود کمک بگیرید تا به شما توضیح دهد.

سطر پنجم: یک حلقه با دستور for ساخته می‌شود که این حلقه به تعداد n بار اجرا خواهد شد.

سطر ششم: این دستور در داخل حلقه است و باعث می‌شود که لاکپشت به مقدار 50 پیکسل جلو برود.

سطر هفتم: این دستور داخل حلقه است و باعث می‌شود که لاکپشت به اندازه زاویه Z به سمت چپ چرخش کند. در زیر یک پنج ضلعی و یک هشت ضلعی را که این لاکپشت رسم کرده است، می‌توانید ببینید:



مثال ۹) سفر لاکپشت

تا امروز لاکپشت ما فقط از یک جا شروع به حرکت می‌کرد اما امروز می‌خواهیم برنامه‌ای بنویسیم تا لاکپشت به صورت شانس‌ی به 10 جای متفاوت سفر کند و در نقطه‌ای که رسید یک دایره با شعاعی متفاوت (شانسی) بکشد. برای اینکه لاکپشت به نقطه‌ای برود از متد goto استفاده می‌کنیم. مثلاً اگر بخواهیم که لاکپشت به نقطه 100, 100 برود، می‌نویسیم:

```
>>> turtle.goto(100,100)
```

همچنین اگر خاطرتان باشد با ماژول random می‌توانستیم اعداد شانس‌ی تولید کنیم. برای تولید مختصات x و y که قرار است لاکپشت به آنجا بروید و تولید شعاع دایره از تابع randint این ماژول استفاده خواهیم کرد و برنامه ما به شکل زیر خواهد شد.

```
import turtle
import random

for i in range(10):
    x = random.randint(-200,200)
    y = random.randint(-200,200)
    r = random.randint(5,100)
    turtle.goto(x, y)
    turtle.circle(r)
```

شرح برنامه:

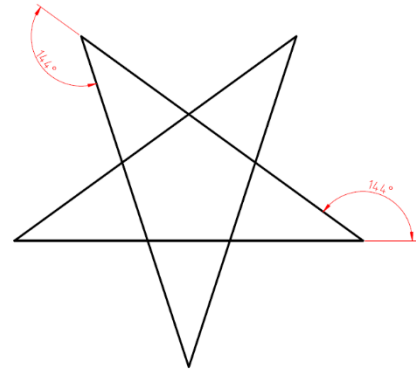
در این برنامه هم از ماژول turtle استفاده کرده‌ایم و هم از ماژول random. یک حلقه for ساختیم که قرار است دستورها داخل آن 10 بار تکرار شوند. در سطر چهارم، تابع randint یک عددی شانس‌ی تولید می‌کند و آنرا در متغیر x ذخیره کردیم. در سطرهای پنجم و ششم هم همین کار تکرار شد. حال به لاکپشت می‌گوییم که به نقطه x, y برود و سپس یک دایره به شعاع r که عددی شانس‌ی است، برای ما رسم کند.

مثال ۱۰) رسم یک ستاره

در این برنامه می‌خواهیم یک ستاره بکشیم که البته کار سختی نیست اما این برنامه یک نکته دیگری هم دارد، می‌خواهیم برای این لاکپشت یک اسم بگذارم و او را با نامش صدا بزنیم.

همانطور که می‌توان یک عدد را در یک متغیر ذخیره کرد، می‌توانیم لاکپشت را هم در یک متغیر ذخیره کنیم! تعجب نکنید در دنیای کامپیوتر می‌شود هر چیزی را در داخل یک متغیر ذخیره‌اش کرد و هیچ محدودیتی نداریم. ما اسم لاکپشتمان را در این برنامه loki می‌گذاریم و برنامه ما به شکل زیر خواهد شد:

```
import turtle
loki = turtle
for i in range(5):
    loki.fd(100)
    loki.left(144)
```



جالب شد! ما تقریباً برنامه‌مان را به شکلی که فارسی حرف می‌زنیم، نوشته‌ایم و اگر آنرا بخوانید می‌شود "لاکی برو جلو"، "لاکی بچرخ به چپ"! .

5

فصل پنجم - شرط‌ها

ما هر روزه بارها و بارها با «اگر و شرط» برخورد می‌کنیم. مثلاً:
"به شرط آنکه هوا آفتابی باشد مسابقه را برگزار می‌کنیم"
"اگر بازهم دروغ بگویی دیگر با تو صحبت نمی‌کنم"
"اگر پاسخ سوال را درست دادی آنگاه یک امتیاز می‌گیری"

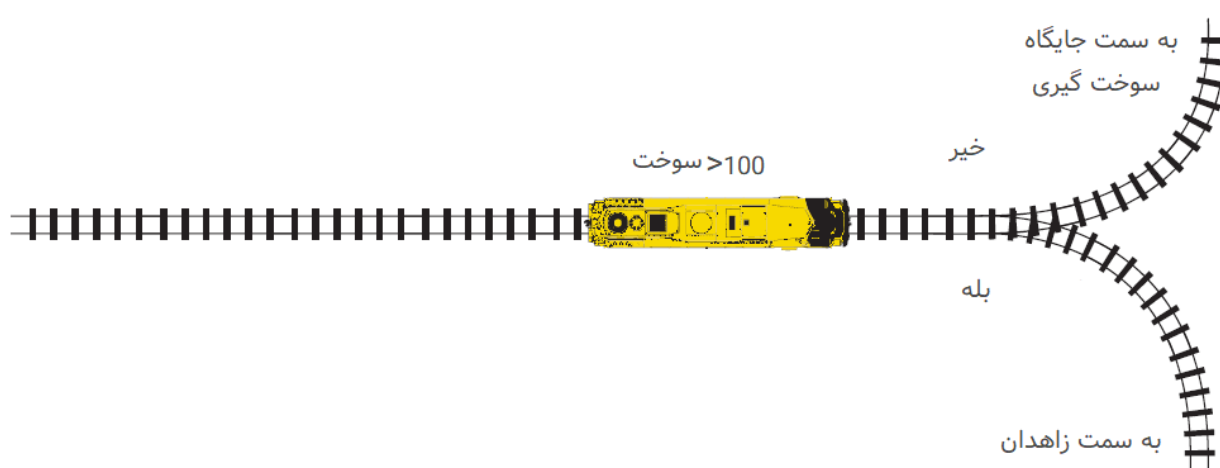
" اگر تیم فوتبال ایران به مسابقات جهانی رفت آنگاه من برای کل کلاس شیرینی می‌خرم "

در دنیای کامپیوتر هم دقیقا همین اتفاق می‌افتد مثلا به کامپیوتر می‌گوییم که یک عدد را بگیر و اگر آن عدد بزرگتر از صفر بود آنگاه جذر آنرا محاسبه کن و گرنه یک پیغام خطا نمایش بده. جالب بدانید که اکثر بازی‌های کامپیوتری بر اساس همین شرطها کار می‌کنند، مثلا اگر گلوله شما به دشمن برخورد کرد، شما یک امتیاز می‌گیرید.

نحوه نوشتن دستورها شرطی

بگذارید قبل از آنکه وارد برنامه نویسی شویم، باید یک نکته مهم را درمورد دستورها شرطی بدانیم. یک جمله شرطی با واژه «اگر» شروع می‌شود و بعد از آن «شرط» نوشته می‌شود که این شرط یا «درست» است و یا اینکه «غلط» و در ادامه جمله مشخص می‌کنیم که اگر شرط «درست» بود چه اتفاقی باید رخ دهد و اگر شرط «غلط» از آب درآمد، چه اتفاقی باید روی دهد.

بگذارید مثالی بزنم، راننده قطاری به نزدیک یک دو راهی می‌رسد و اگر «بیشتر از 100 لیتر سوخت داشت»، می‌تواند به «سمت زاهدان» برود و در غیر اینصورت (یعنی اگر سوخت کافی نداشته باشد) باید به «طرف جایگاه سوخت گیری» حرکت کند.



در این مثال قسمت شرط، اگر «سوخت بیشتر از 100 باشد»، است. حال اگر این شرط برقرار باشد (یعنی سوخت بیش از 100 باشد)، ما به سمت زاهدان خواهیم رفت و اگر این شرط را نداشته باشیم، آنگاه باید به طرف جایگاه سوخت گیری حرکت کنیم.

این دستور را به شکل کامپیوتری برای شما می‌نویسم:

اگر سوخت بزرگتر از 100 بود:

حرکت به سمت زاهدان

وگرنه :

حرکت به سمت جایگاه سوخت گیری

برای آنکه در پایتون یک شرط را بنویسیم از دستور if-else استفاده می‌کنیم. واژه «اگر» به انگلیسی if (بخوانید ایف) و واژه «وگرنه» به انگلیسی else (بخوانید الز) می‌شود و باید این دستورها را به شکل زیر در پایتون بنویسیم:

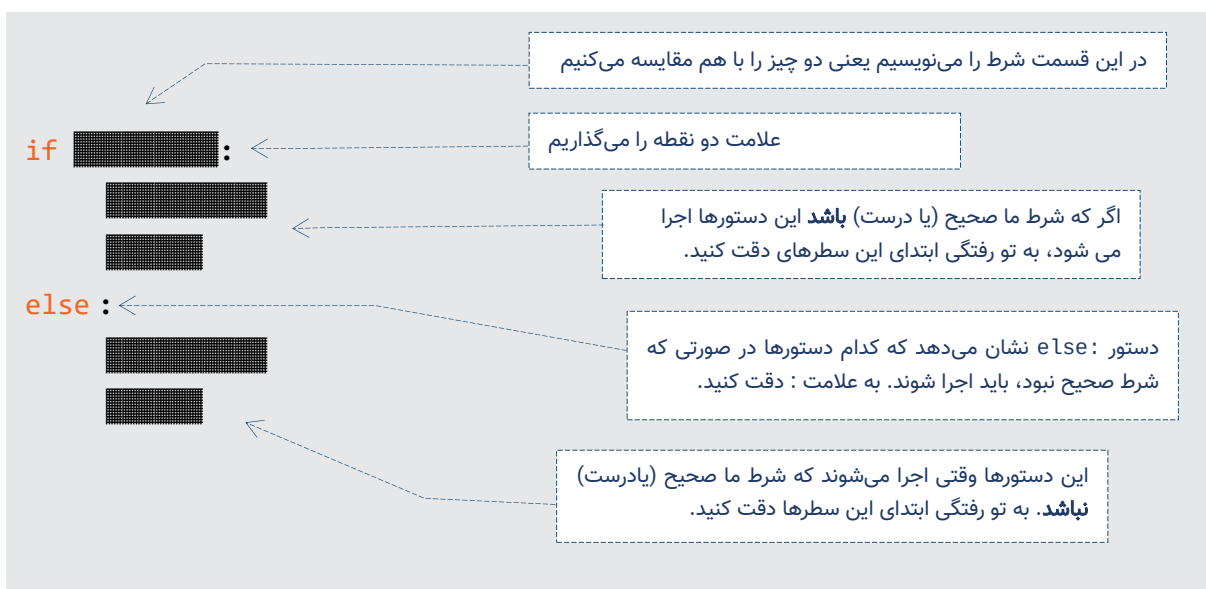
```
if > 100: سوخت
```

حرکت به سمت زاهدان

```
else:
```

حرکت به سمت جایگاه سوخت گیری

اجازه دهید که نگاه دقیق تری به شکل کلی دستور if-else در پایتون بیندازیم:



تذکر ۱) در انتهای سطر دستور if و همینطور دستور else باید از علامت دو نقطه استفاده کنید که به معنای این است که این دستورها ترکیبی هستند و ادامه دارند.

تذکر ۲) به تو رفتگی ابتدای سطر دوم و چهارم دقت کنید. همانطور که در دستور for دید، این تورفتگی‌ها در دستورها ترکیبی اجباری است و اگر فراموش شوند، SyntaxError (پیغام خطای املایی) به شما داده می‌شود و برنامه اجرا نخواهد شد.

حال بیا ببینیم چند برنامه ساده بنویسیم تا با دستور if در پایتون در عمل بیشتر آشنا شوید.

مثال ۱) بلیط مجانی

فرض کنید که می‌خواهیم برای فروش بلیط سینما یک برنامه بنویسیم که سن یک نفر را بپرسد و اگر سن او کمتر از 4 سال بود پیغام دهد که «شما بلیط لازم ندارید» و در غیر اینصورت (یعنی سن بیشتر از 4 سال)، پیغام بدهد که «باید بلیط بخرید».

نکته ۱) در زبان پایتون می‌توانیم پیغام‌ها را فارسی بنویسیم، اما در این کتاب ما پیغام‌ها را به صورت فینگلیش (یعنی فارسی که با حرف انگلیسی نوشته می‌شود) خواهیم نوشت.

```
sen = float( input('shoma chand sal darid? ') )
if sen < 4:
    print('shoma blit lazem nadarid')
else:
    print('bayad blit bekharid')
```

شرح سطر اول:

ما سن یک فرد را در متغیری به نام `sen` ذخیره خواهیم کرد و توسط تابع `input` ما از کاربر این مقدار را خواهیم پرسید. اما همانطور که قبلاً هم گفتیم، تابع `input` به ما یک متن (رشته) خواهد داد و ما قادر نیستیم که عملیات‌های ریاضی را روی آن متن انجام دهیم. به همین دلیل باید با `int` و یا `float` این متن را به یک عدد تبدیل می‌کنیم. به دلیل اینکه ممکن است یک نفر عددی اعشاری را وارد کند و ما نمی‌خواستیم که اعشار آن را حذف کنیم، از تابع `float` استفاده کردیم.

هنگامی که در داخل یک دستور، دستوری دیگری را می‌نویسیم، کامپیوتر ابتدا دستور داخلی‌تر را اجرا می‌کند بنابراین در سطر اول این برنامه ابتدا دستور `input` اجرا می‌شود و یک متن از کاربر دریافت می‌شود، سپس این متن توسط تابع `float` به یک عدد تبدیل خواهد شد و در آخر هم این عدد در داخل متغیر `sen` ذخیره می‌شود.

شرح سطر دوم:

گفتیم که دستورها شرطی در پایتون با `if` شروع می‌شود و سپس باید خود شرط را بنویسیم. در اینجا شرط ما این است که سن فرد بزرگتر از 10 باشد، بنابراین خواهیم نوشت `if sen > 10`. باید بدانید که در زبان پایتون ما عملگرهای مقایسه‌ای هم داریم و در این مثال از عملگر `>` (بزرگتر) استفاده کرده‌ایم. به زودی با سایر عملگرهای مقایسه آشنا خواهید شد.

نکته ۲) ما جمله‌های ریاضی را از سمت چپ به راست می‌خوانیم. بنابراین عبارت `20 > 10` را باید به فارسی بخوانیم که «20 بزرگتر از 10 است». خواندن این عبارت به شکل «10 کوچکتر از 20 است» غلط است. توجه) مجدد تاکید می‌کنم که به علامت‌های دو نقطه و همچنین تو رفتگی‌ها دقت کنید. در شکل زیر، جاهایی که احتمال تایپ اشتباه وجود دارد را با 8 فلش مشخص کرده‌ام. به آنها دقت کنید.

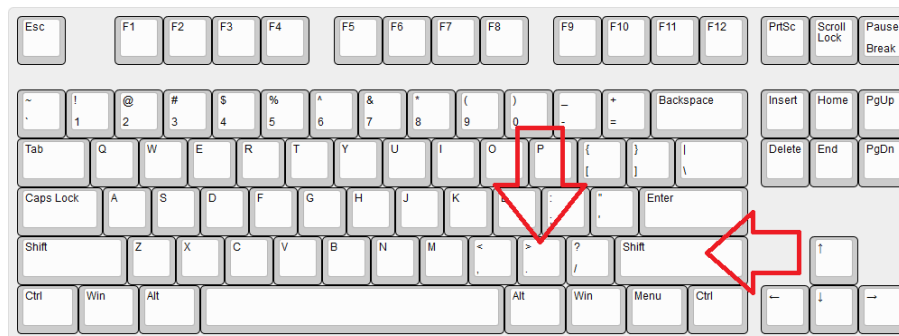
```

        ↓      ↓      ↓
sen = float( input('shoma chand sal darid? ') )

if sen < 4:
    print('shoma blit لازم ندارید')
else:
    print('bayad blit bekharid')

```

نکته ۳) برای تایپ علامت > بر روی کیبورد، باید کلید Shift را پایین نگه دارید و کلید > را بزنید.



مثال ۲) حدس زدن

می‌خواهیم برنامه‌ای بنویسیم که کامپیوتر یک عدد شانسی بین 1 تا 10 (یا اصطلاحاً تصادفی) تولید کند و سپس از شما بخواهد که آن عدد را حدس بزنید، اگر حدس شما درست بود، پیغام دهد که شما برنده شده‌اید و در غیر اینصورت بگوید که حدس شما غلط بوده است.

```

import random
addad = random.randint(1,10)
hads = int( input('hads khod ra vared konid? '))

if addad == hads:
    print('afarin, barandeh shodid')
else:
    print('hads shoma ghalat boodeh')

```

توجه) عملگر مساوی بودن در پایتون علامت == است. (دو علامت مساوی)

مثال ۳) شما بزرگتر هستید و یا دوستان

فرض کنید که سن شما 14 سال است. برنامه‌ای بنویسید که سن دوستان را بگیرد و سپس بگوید که او چند سال از شما بزرگتر و یا کوچکتر است.

```
dost = float( input('sen dost ? ') )
e = 14 - dost
if e > 0:
    print( str(e) + ' sal az shoma kochehtart ast')
else:
    print( str(abs(e)) + ' sal az shoma bozorgtar ast')
```

شرح برنامه:

در این برنامه هم نکته‌های جدیدی را یاد می‌گیرید. در سطر دوم اختلاف سن شما و دوستان را حساب کردیم و در متغیری به نام e ذخیره کردیم. نکته ۱) قبلاً دیدیم که در پایتون می‌توانیم بنویسیم $n + m$ و این دستور اگر هر دو متغیر m و n از نوع متنی باشند، صحیح است و اگر یکی از آنها عددی و دیگر متن، پایتون به شما خطای زیر را خواهد داد:

```
TypeError: must be str, not int
```

کلمه `Type` (بخوانید تایپ) یعنی «نوع» و خطای `TypeError` یعنی نوع داده‌های شما برای اینکاری که می‌خواهید انجام بدهید، مناسب نیست. در ادامه پیغام دقیقاً نوشته است که باید نوع داده شما از نوع `str` (یعنی متن) باشد و نه `int` (عدد صحیح).

به همین دلیل در سطر چهارم در دستور `print`، ما با تابع `str` مقدار متغیر e را به یک متن تبدیل می‌کنیم تا بتوانیم آنرا به متن پیغام بعدی بچسبانیم.

نکته ۲) اگر سن شما کمتر از سن دوستان باشد، مقدار e عددی منفی خواهد شد و چون نمی‌خواستیم در پیغام عدد منفی را ببینیم، به همین دلیل ابتدا با تابع `abs`، عدد منفی را به مثبت تبدیل کردیم و سپس آنرا به `str` تبدیل کردیم تا بتوانیم پیغام مناسب را چاپ کنیم. (در ریاضی به مثبت کردن اعداد منفی، قدر مطلق گفته می‌شود که در ریاضی دبیرستان با آن آشنا می‌شوید).

نکته ۳) فرض کنید که سن دوست شما هم دقیقاً ۱۴ سال است، این برنامه را اجرا کنید. خواهید دید که پیغام اشتباهی چاپ می‌شود. یعنی این برنامه اشکال منطقی داد زیرا ما پیش بینی نکرده‌ایم که اگر e برابر صفر شد، باید پیغام بدهد که «شما هم سن هستید». برای همین منظور باید دستور `if` را کمی کامل‌تر به شکل زیر بنویسیم:

```
dost = float( input('sen dost ? ') )
e = 14 - dost
if e > 0:
    print( str(e) + ' sal az shoma kochehtart ast')
elif e < 0:
    print( str(abs(e)) + ' sal az shoma bozorgtar ast')
```

```
else:  
    print( 'shoma ham sen hastind')
```

می‌بینید که یک `elif` اضافه شده است. این دستور مخفف کلمه `else if` به معنای در «در غیر اینصورت اگر» می‌باشد.

بگذارید «نحوه عملکرد» کلی این برنامه را به صورت فارسی برای شما شرح دهم:
سطر یکم) سن دوست را بپرس و سپس آنرا به یک عدد اعشاری تبدیل کن و سپس آنرا در متغیر `dost` ذخیره کن.
سطر دوم) مقدار `14 - dost` را محاسبه کن و در متغیر `e` ذخیره کن. (بنابراین متغیر `e` نشان می‌دهد که شما و دوستتان چند سال اختلاف سن دارید)
سطر سوم) اگر مقدار متغیر `e` بزرگتر از 0 بود آنگاه:
سطر چهارم) بنویس که `e` سال از شما کوچکتر است.
سطر پنجم) در غیر اینصورت (یعنی اگر `e` بزرگتر از 0 نبود) اگر `e` کمتر از 0 است، آنگاه:
سطر ششم) بنویس که `e` سال از شما بزرگتر است.
سطر هفتم) در غیر اینصورت (یعنی اگر نه بزرگتر از 0 بود و نه کوچکتر از 0) آنگاه:
سطر هشتم) بنویس که شما هم سن هستید.

آشنایی با واژه الگوریتم:
در مثال قبلی ما به فارسی برای شما «نحوه عملکرد کلی» برنامه را نوشتیم. اصطلاحاً به اینکار یعنی «توصیف نحوه عملکرد کلی یک برنامه»، الگوریتم گفته می‌شود.

پیوست‌ها

نصب پایتون بر روی ویندوز

قبل از نصب چند نکته را بهتر است بدانید:

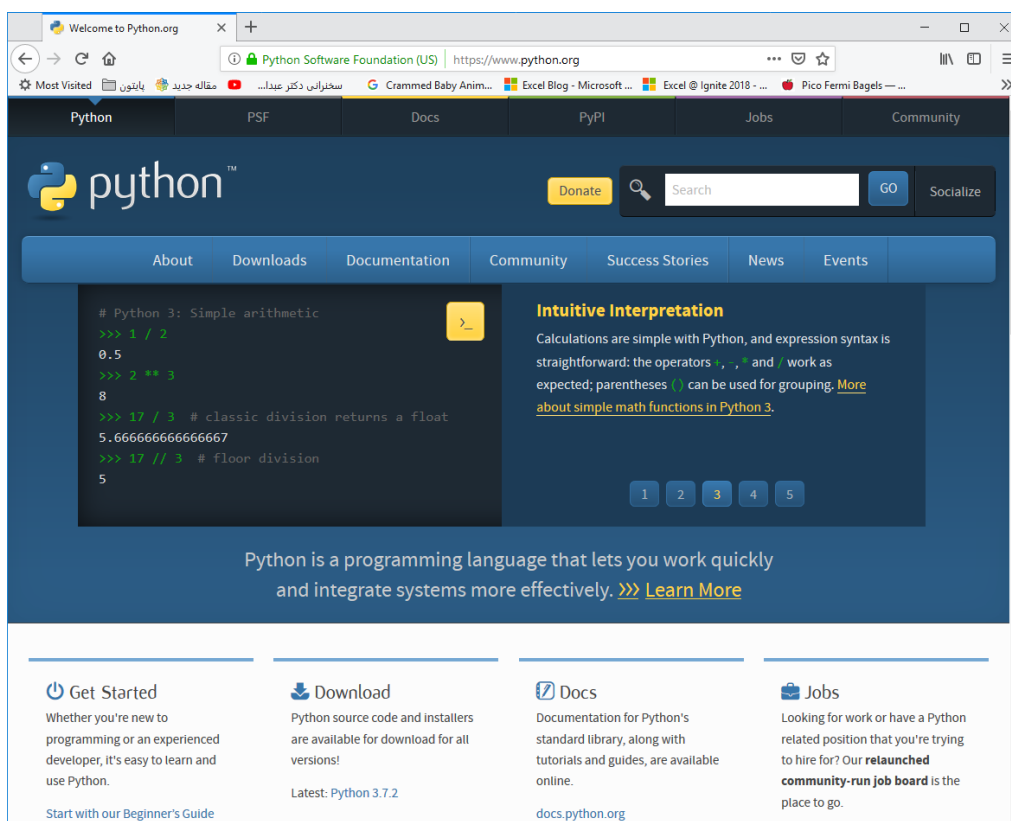
(۱) پایتون یک زبان رایگان است و از سایت اصلی آن می‌توانید آن را بدون هیچ محدودیتی و کامل دریافت کنید.
(۲) حجم دانلود پایتون کمتر از 30MB (۳۰ مگا بایت) است و دانلود بین ۱ تا ۵ دقیقه طول خواهد کشید. اگر به اینترنت دسترسی ندارید، از یکی از آشنایان/دوستان بخواهید که آنرا برای شما دانلود و بر روی یک فلش یا cd کپی کند تا بتوانید آنرا نصب کنید.

(۳) نصب پایتون بسیار ساده است و نیاز به تنظیمات خاصی ندارد و در حدود ۱ دقیقه نصب آن کامل می‌شود.
برای نصب پایتون بر روی ویندوز قدم‌های زیر را طی کنید.

قدم ۱)

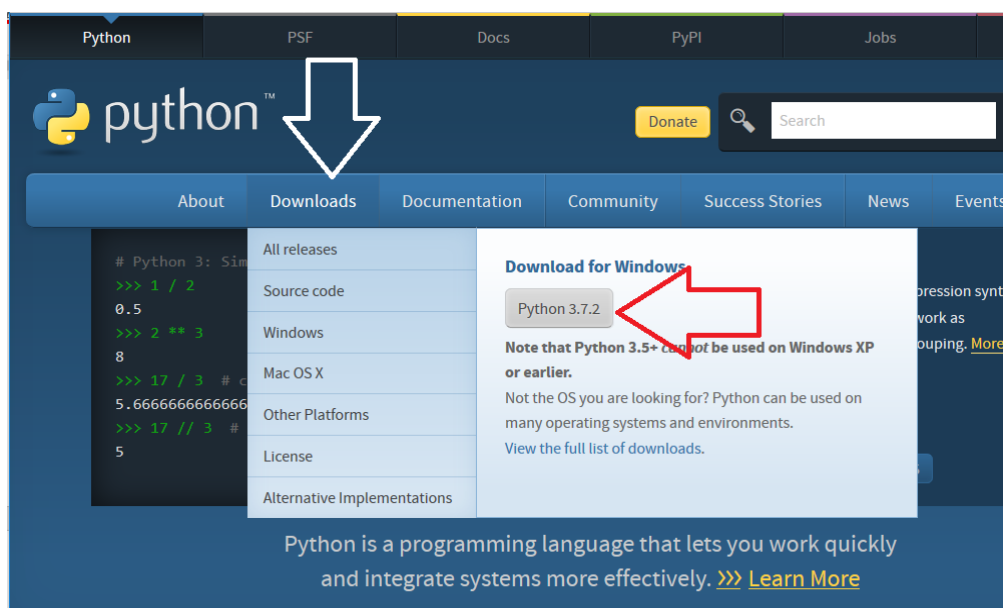
وارد سایت www.python.org شوید. ظاهر این سایت شبیه تصویر زیر است.

نکته: در هنگام تایپ آدرس در مرورگر یک سایت لازم نیست .www را بگذارید.



قدم ۲)

به منوهای بالای سایت دقت کنید. روی گزینه Download کلیک کنید و سپس روی دکمه Python 3.7.2 کلیک کنید. بلافاصله دانلود شما شروع خواهد شد.



نکته ۱) در این پنجره نسخه Python 3.7.2 را می بینید و از آنجایی که پایتون همواره بروز می شود، ممکن است شما عددی متفاوت را ببینید که نشان دهند نسخه جدیدتری از پایتون است بنابراین باید آن نسخه را دانلود کنید. نکته ۱۲۲ اگر بخواهید سایر نسخه های پایتون را دانلود کنید (مثلا 64 یا 32 بیت)، روی گزینه All releases کلیک کنید تا لیست کاملی از نسخه های مختلف را برای دانلود ببابید.

قدم ۳)

بعد از دانلود شما یک فایل به نام python-3.7.2.exe خواهد داشت. روی آن D-Click (یعنی دوبار کلیک کنید) تا نصب پایتون شروع شود. اولین پنجره‌ای که می‌بینید به شکل زیر است:

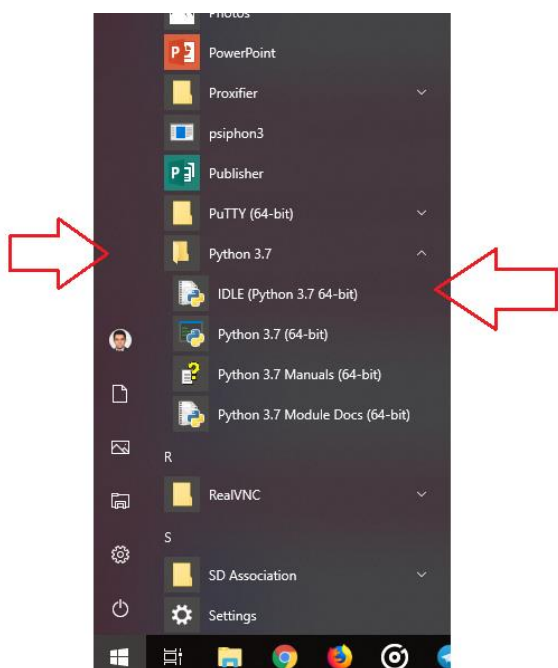


برای شروع نصب روی گزینه Install Now کلیک کنید تا نصب شروع شود. (اگر پنجره و یا سوالی پرسید، Ok را بزنید.) انتظار داریم که در حدود ۱ دقیقه نصب به صورت کامل انجام شود.

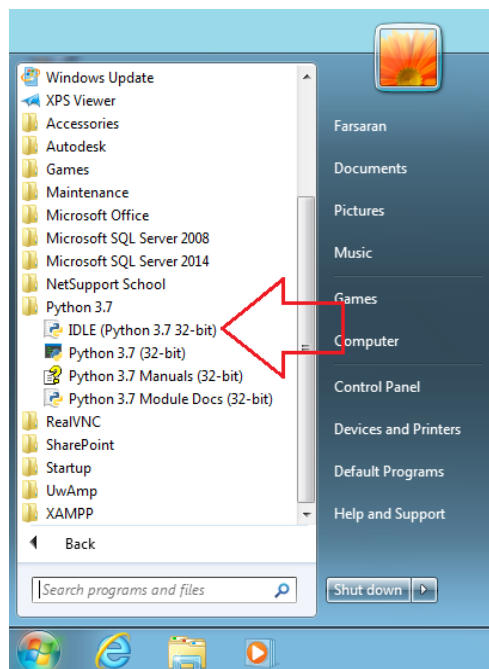
قدم ۴)

برای اجرای پایتون در منوی Start ویندوز به دنبال Python بگردید و سپس گزینه (Python 3.7) IDLE را کلیک کنید تا وارد محیط برنامه نویسی پایتون شوید. به تصاویر زیر دقت کنید:

تصویر اجرای پایتون در ویندوز 10



تصویر اجرا پایتون در ویندوز 7



پیوست ب

آشنایی با متدها و توابع

ماژول Turtle

آشنایی با متدها و توابع ماژول Turtle

ماژول Turtle یکی از روش‌های متداول آموزش برنامه نویسی به نوجوان است. در اینجا من فهرستی از دستورها (یا دقیق‌تر بگوییم متدهای) آنرا برای شما شرح می‌دهم.

نکته ۱: برخی از دستورها دارای نام‌های متعددی هستند. برای نمونه دستور forward و fd هر دو یک چیز هستند و برای حرکت لاکپشت به سمت جلو استفاده می‌شوند. بنابراین چه بنویسیم forward(100) و چه بنویسیم fd(100) فرقی نخواهد کرد.

نکته ۲: turtle یک ماژول است و باید در اولین خط برنامه با دستور زیر به پایتون بگوییم که قصد داریم از این ماژول استفاده کنیم و باید بنویسیم:

```
import turtle
```

در غیر اینصورت برنامه با خطا روبرو می‌شود و اجرا نخواهد شد.

توجه داشته باشید که در هر برنامه فقط یکبار لازم است که این دستور را بنویسید.

نکته ۳: بسیاری از دستورها از شما پارامتر می‌خواهند. پارامتر یعنی ورودی که به دستور می‌دهد و آن دستور بر طبق آن عمل می‌کند. مثلاً باید به دستور forward یک پارامتر بدهید که به عنوان مسافتی است که باید لاکپشت روبه جلو برود.

بنابراین برای آنکه به لاکپشت بگوییم که به اندازه 100 واحد رو به جلو حرکت کن باید بنویسیم:

```
import turtle
turtle.forward(100)
```

* پارامترها را در داخل پرانتز می‌گذاریم.

* اگر دستوری چند پارامتر می‌خواست، باید پارامترها را با علامت کاما یعنی « , » از هم جدا کنیم.

نکته ۴) توجه داشته باشید که برخی از دستورها مانند رفتن و نقاشی کشیدن بر روی هر لاکپشتی قابل انجام است. اما برخی از دستورها مربوط به چیزهای دیگری است. مثلاً اگر بخواهیم کل زمینی که لاکپشت بر روی آن حرکت می‌کند را رنگی یا بزرگتر کنیم، این دستور مربوط به زمین بازی است نه مربوط به لاکپشت. ما ابتدا دستورها لاکپشت را یاد می‌گیریم و سپس دستورها مربوط به تنظیمات زمین بازی که لاکپشت بر روی آن راه می‌رود را خواهیم آموخت.

- دستورها لاکپشت

- حرکت و نقاشی
- اطلاع از مکان لاکپشت
- تنظیمات مداد لاکپشت

۱) دستورها لاکپشت - حرکت و نقاشی

توجه: کلیه این دستورها بر روی هر لاکپشتی قابل اجرا هستند.

fd یا forward

لاکپشت را به اندازه مسافت داده شده رو به جلو جابجا می‌کند.

نحوه نگارش:

```
turtle.forward(مسافت)
```

پارامتر:

مسافت/ distance: عددی صحیح یا اعشاری است که میزان جابجایی لاکپشت را مشخص می‌کند.

مثال:

```
import turtle  
turtle.forward(100)
```

bk یا backward یا back

لاکپشت را به اندازه مسافت داده شده رو به عقب جابجا می‌کند.

نحوه نگارش:

```
turtle.backward(مسافت)
```

پارامتر:

مسافت/ distance: عددی صحیح یا اعشاری است که میزان جابجایی لاکپشت را مشخص می‌کند.

مثال:

```
import turtle  
turtle.backward(100)
```

rt یا right

لاکپشت به اندازه زاویه داده شده به سمت راست می‌چرخد.

نحوه نگارش:

```
turtle.right(زاویه)
```

پارمتر:

زاویه/ angle: عددی صحیح و یا اعشاری است که میزان چرخش را بر حسب «درجه» مشخص می‌کند.

مثال:

```
import turtle
turtle.right(90)
turtle.fd(100)
turtle.right(45)
turtle.fd(50)
```

left یا lt

لاکپشت به اندازه زاویه داده شده به سمت چپ می‌چرخد.

نحوه نگارش:

```
turtle.left(زاویه)
```

پارمتر:

زاویه/angle: عددی صحیح و یا اعشاری است که میزان چرخش را بر حسب «درجه» مشخص می‌کند.

مثال:

```
import turtle
turtle.fd(100)
turtle.left(90)
turtle.fd(50)
```

goto یا setpos یا setposition

لاکپشت را به نقطه مختصات داده شده می‌برد و اگر pendown باشد (یعنی لاکپشت مداد را روی صفحه گذاشته باشد)، خطی کشیده می‌شود.

نحوه نگارش:

```
turtle.goto(x , y)
```

پارامترها:

x : یک عدد است و مختصات نقطه‌ای روی محور x ها است.

y : یک عدد است و مختصات نقطه‌ای روی محور y ها است.

مثال: رسم یک مثلث متساوی الساقین

```
import turtle
turtle.goto(100 , 0)
turtle.goto(50 , 150)
turtle.goto(0 , 0)
```

نکته: می‌توانیم پارمتر x را یک زوج عدد (بردار) ، در داخل پرانتز بدهیم و پارمتر y را ندهیم.

```
turtle.goto( (10,20) )
```

setx

بدون تغییر مختصات y ، باعث رفتن لاکپشت به نقطه x بر روی صفحه می‌شود.

نحوه نگارش:

```
turtle.setx( x )
```

پارامترها:

x : یک عدد است.

مثال:

```
import turtle  
turtle.setx(100)
```

sety

بدون تغییر مختصات x ، باعث رفتن لاکپشت به نقطه y بر روی صفحه می‌شود.

نحوه نگارش:

```
turtle.sety( y )
```

پارامترها:

y : یک عدد است.

مثال: رسم یک مستطیل

```
import turtle  
turtle.setx(100)  
turtle.sety(50)  
turtle.setx(0)  
turtle.sety(0)
```

home

لاکپشت را به خانه یعنی مختصات $0, 0$ می‌برد و سر لاکپشت به سمت راست خواهد شد.

نحوه نگارش:

```
turtle.home()
```

پارامترها: این دستور پارامتری ندارد.

مثال:

```
import turtle  
turtle.fd(100)  
turtle.left(45)  
turtle.fd(100)  
turtle.home()
```


circle

با شعاع داده شد برای ما یک دایره رسم می‌شود که مرکز آن دایره در سمت چپ لاکپشت خواهد بود.
نحوه نگارش:

```
turtle.circle( شعاع دایره , محدوده , دقت )
```

پارمترها:

شعاع/radius: عددی مثبت و یا منفی است. اگر عدد مثبت باشد دایره در جهت خلاف عقربه‌ها ساعت رسم

می‌شود. اگر منفی باشد، دایره در جهت عقربه‌های ساعت خواهد بود.

محدوده/extent: یک عدد است و اختیاری است و اگر داده نشود، یک دایره کامل رسم می‌شود. این عدد

مشخص می‌کند که چه زاویه‌ای از دایره رسم شود.

دقت/Steps: یک عدد است و اختیاری است. دایره در حقیقت یک چند ضلعی با اضلاع بسیار زیاد است.

پارامتر دقت تعداد این اضلاع را مشخص می‌کند و اگر داده نشود به صورت خودکار توسط نرم افزار این تعداد

محاسبه و تعیین خواهد شد.

مثال ۱) رسم دو دایره قرینه زیر هم

```
import turtle
turtle.circle(100)
turtle.circle(-100)
```

مثال ۲) رسم یک نیم دایره

```
import turtle
turtle.circle(100, 180)
```

یادآوری: یک دایره ۳۶۰ درجه است. بنابراین نیم دایره ۱۸۰ درجه خواهد بود.

مثال ۳) رسم یک ۱۳ ضلعی

```
import turtle
turtle.circle(100, 360 , 13)
```

نکته: برای رسم چند ضلعی‌های منتظم از دستور circle می‌توانید استفاده کنید.

مثال ۴) رسم سه ضلعی تا 15 ضلعی منتظم

```
import turtle
for i in range(3,16):
    turtle.circle(100 , 360 , i)
```

توجه: هر چه تعداد اضلاع بیشتر می‌شود، شکل به دایره نزدیکتر خواهد شد.

نکته: برای آنکه یک شش ضلعی کامل رسم کنیم می‌توانید پارامتر دوم را ننویسید و پارامتر سوم را با نامش مقدار دهی

کنید:

```
turtle.circle(100, steps=6)
```

speed

سرعت حرکت لاکپشت را تنظیم می‌کند.

نحوه نگارش:

`turtle.speed(سرعت)`

پارامترها:

سرعت/speed: اختیاری است و عددی صحیح بین 0 تا 10 است که سرعت حرکت لاکپشت را مشخص می‌کند.

اگر این عدد بیشتر از 10 و یا کمتر از 0.5 باشد، سرعت 0 در نظر گرفته می‌شود.

نکته ۱) سرعت 0 یعنی که حرکت لاکپشت دیده نمی‌شود و به نظر می‌آید که جای لاکپشت به یکباره تغییر می‌کند.

نکته ۲) به جای عدد می‌توان از مقادیر زیر برای تنظیم سرعت استفاده کرد.

'fastest': 0 'fast': 10 'normal': 6 'slow': 3 'slowest': 1

نکته ۳) اگر پارامتر سرعت تنظیم نشود، سرعت فعلی لاکپشت برای ما نمایش داده می‌شود.

مثال:

```
import turtle
turtle.speed(10)
turtle.goto(0,100)
turtle.speed('slow')
turtle.goto(100,100)
turtle.speed(0)
turtle.goto(0,0)
```

۲) دستورها لاکپشت - اطلاع از مکان لاکپشت

توسط این دستورها می‌توانیم از مختصات که اکنون لاکپشت در آن مطلع شویم.

pos یا position

موقعیت فعلی لاکپشت به صورت یک زوج عدد (بردار) را به ما می‌دهد.

پارامترها: این دستور پارامتری ندارد.

مثال: به عنوان مثال اگر لاکپشت در نقطه‌ای به مختصات $x = 10$ و $y = 20$ باشد، دستور زیر $(10, 20)$ را نمایش

می‌دهد.

```
import turtle
print( turtle.pos() )
```

towards

زاویه خطی که بین موقعیت فعلی سر لاکپشت و یک نقطه می‌توان رسم کرد را برای ما محاسبه می‌کند.

شکل نگارش:

```
turtle.towards(x , y)
```

پارامترها:

x : یک عدد است.

y : یک عدد است. این پارامتر اختیاری است و اگر داده نشود، پارامتر x باید به صورت یک زوج عدد (بردار)

مشخص شود.

مثال) کوهی در مختصات 123, 246 وجود دارد. لاکپشت باید سرش را چند زاویه بچرخاند تا آن کوه را ببیند؟

```
import turtle  
print( turtle.towards(123,246) )
```

distance

فاصله بین لاکپشت تا یک نقطه را محاسبه می‌کند.

```
turtle.distance(x , y)
```

پارامترها:

x : یک عدد است.

y : یک عدد است. این پارامتر اختیاری است و اگر داده نشود، پارامتر x باید به صورت یک زوج عدد (بردار)

مشخص شود.

مثال) کوهی در مختصات 123, 246 وجود دارد. لاکپشت باید چقدر راه برود تا به آن کوه برسد؟

```
import turtle  
print( turtle.distance(123,246) )
```

xcor

مختصات x محل لاکپشت را به ما می‌گوید.

شکل نگارش:

```
turtle.xcor()
```

پارامتر: این تابع پارامتری ندارد.

مثال:

```
import turtle  
print( turtle.xcor() )
```

مختصات y محل لاکپشت را به ما می‌گوید و کاملاً شبیه xcor است.

۳) دستورها لاکپشت – مداد

لاکپشت یک مداد دارد و هنگام حرکت آن مداد را روی صفحه قرار می‌دهد و اشکالی را رسم می‌کند. در این قسمت با تنظیمات این مداد آشنا خواهید شد.

pendown یا pd یا down

مداد را روی صفحه قرار می‌دهد و باعث رسم اشکالی هنگام حرکتش می‌شود.
شکل نگارش:

```
turtle.pendown()
```

پارامترها: این دستور پارامتری ندارد.

penup یا pu یا up

لاکپشت مداد را از روی صفحه بر می‌دارد و چیزی هنگام حرکتش رسم نمی‌شود.
شکل نگارش:

```
turtle.penup()
```

پارامترها: این دستور پارامتری ندارد.

مثال: برنامه زیر یک خط چین برای ما رسم می‌کند.

```
import turtle
import turtle
for i in range(30):
    if i % 2 == 0:
        turtle.pendown()
    else:
        turtle.penup()
    turtle.fd(10)
```

pensize یا width

ضخامت نوک مداد را تعیین می‌کند.
شکل نگارش:

```
turtle.pensize( ضخامت )
```

پارامترها:

ضخامت/width: یک عدد مثبت است و ضخامت مداد را مشخص می‌کند و اختیاری است. در صورتی که این پارامتر خالی باشد، مقدار ضخامت برای ما برگردانده می‌شود.

مثال:

```
import turtle
turtle.fd(100)
turtle.pensize(10)
turtle.fd(100)
turtle.pensize(30)
turtle.fd(100)
```

pencolor

برای تنظیم رنگ مداد لاکپشت استفاده می‌شود.

شکل نگارش:

```
turtle.pencolor(رنگ)
```

پارامتر:

رنگ / cmode: یکی از کدهای رنگ قابل فهم توسط لاکپشت

آشنایی با کد رنگ‌ها:

در دنیای کامپیوتر چندین روش برای مشخص کردن یک رنگ وجود دارد و در ماژول turtle هم ما به صورت ۴ روش برای مشخص کردن یک رنگ داریم.

ساده ترین روش استفاده از نام رنگ است. مانند:

```
pencolor('red')
pencolor('green')
```

روش دیگر آن است که میزان ترکیب سه رنگ «قرمز - سبز - آبی» را به ترتیب مشخص کنیم. به این روش اصطلاحاً rgb می‌گویند که مخفف کلمات Red-Green-Blue است. در این روش میزان استفاده از هر رنگ را با عددی بین 0 تا 1 مشخص می‌کنیم. به مثال‌های زیر دقت کنید:

pencolor(1, 0, 0)	قرمز خالص
pencolor(0, 1, 0)	سبز خالص
pencolor(0, 0, 1)	آبی خالص
pencolor(0.5, 0, 0.5)	نصف سطل قرمز + نصف سطل آبی = رنگ بنفش

البته این روش که بین 0 تا 1 عددی را مشخص می‌کنیم فقط در دنیای لاکپشت وجود دارد و در شکل استاندارد آن باید عددی بین 0 تا 255 را بدهیم. برای آنکه بتوانیم از سیستم رنگ rgb استاندارد استفاده کنیم، باید سیستم رنگ لاکپشت را با دستور زیر به استاندارد تغییر دهیم:

```
turtle.colormode(255)
```

حال می‌توانیم برای رنگ قرمز بنویسیم:

```
pencolor(255, 0, 0)
```

سیستم دیگر رنگ‌ها به نام Hex (بخوانید هگز) شناخته می‌شود که به صورت کدهایی مانند #ff6464 است.

```
pencolor('#ff6464')
```

نکته ۱: ما می‌توانیم رنگ را به صورت یک تاپل یعنی سه عدد در داخل پرانتز به این دستور بدهیم. مانند:

```
pencolor( (1,1,1) )
```

نکته ۲: اگر این دستور را بدون پارامتر بنویسیم، رنگ جاری را برمی‌گرداند.

مثال:

```
import turtle
turtle.pencolor('red')
turtle.setx(100)
turtle.pencolor('green')
turtle.sety(100)
turtle.colormode(255)
turtle.pencolor(128,0, 255)
turtle.home()
```

خلاصه‌ای از نام رنگ‌هایی که در ماژول turtle می‌توانید استفاده کنید:

alice blue, antique white, aquamarine, azure, beige, bisque, black, blue, blue1, blue2, blue3, blue4, brown, burlywood, cadet blue, chartreuse, chocolate, coral, cyan, dark blue, deep pink, dim gray, dodger blue, firebrick, floral white, forest green, gainsboro, ghost white, gold, gray, green, honeydew, hot pink, indian red, ivory, khaki, lavender, lawn green, lemon chiffon, light blue, lime green, linen, magenta, maroon, medium aquamarine, midnight blue, mint cream, misty rose, moccasin, navajo white, old lace, olive drab, orange, orchid, pale goldenrod, papaya whip, peach puff, peru, pink, plum, powder blue, purple, red, rosy brown, royal blue, saddle brown, salmon, sandy brown, sea green, sienna, sky blue, slate blue, snow, spring green, steel blue, tan, thistle, tomato, turquoise, violet, wheat, white, yellow ,grey1, grey2, grey3, grey100

fillcolor

رنگ داخلی اشکال را مشخص می‌کند.

پارامترهای این دستور شبیه دستور pencolor هستند و از ذکر دوباره آنها خودداری می‌شود.

نکته ۲: اگر این دستور را بدون پارامتر بنویسیم، رنگ داخل جاری را برمی‌گرداند.

color

این دستور برای تنظیم pencolor و fillcolor بکار می‌رود.

نحوه نگارش:

(رنگ داخل , رنگ قلم) color

پارامترها:

رنگ قلم : کد رنگ قابل فهم توسط لاکپشت (به دستور pencolor نگاه کنید).

رنگ داخل: کد رنگ قابل فهم توسط لاکپشت (به دستور pencolor نگاه کنید).

نکته ۲: اگر این دستور را بدون پارامتر بنویسیم، رنگ جاری را برمی‌گرداند.

begin_fill

قبل از رنگ کردن یک شکل باید این دستور را بنویسیم.

شکل نگارش:

```
turtle.begin_fill()
```

پارامتر: این دستور پارامتری ندارد.

end_fill

باعث رنگ شدن تمام اشکالی می‌شود که بعد از دستور begin_fill رسم شده‌اند.

شکل نگارش:

```
turtle.end_fill()
```

پارامتر: این دستور پارامتری ندارد.

مثال:

```
import turtle
turtle.color('red', 'yellow')
turtle.pensize(5)
turtle.begin_fill()
turtle.circle(100)
turtle.circle(-100, 360, 6)
turtle.end_fill()
```

hideturtle یا ht

باعث مخفی شدن لاکپشت در حین رسم شکل‌ها می‌شود و برای رسم «سریع» شکل‌های پیچیده کاربرد دارد.

شکل نگارش:

```
turtle.hideturtle()
```

پارامتر: این دستور پارامتری ندارد.

showturtle یا st

باعث نمایش لاکپشتی که با دستور hideturtle مخفی شده است، خواهد شد.

شکل نگارش:

```
turtle.showturtle()
```

پارامتر: این دستور پارامتری ندارد.

reset

تمام شکل‌هایی که لاکپشت کشیده است را از روی صفحه پاک می‌کند و لاکپشت و سایر تنظیمات به حالت اولیه (Default) باز گردانده می‌شوند.

شکل نگارش:

```
turtle.reset()
```

پارامتر: این دستور پارامتری ندارد.

clear

تمام شکل‌هایی که لاکپشت کشیده است را از روی صفحه پاک می‌کند. جای لاکپشت و سایر تنظیمات آن و همچنین سایر لاکپشت‌ها تغییری نمی‌کند.

شکل نگارش:

```
turtle.clear()
```

پارامتر: این دستور پارامتری ندارد.

write

لاکپشت متنی را برای ما روی صفحه می‌نویسد.

شکل نگارش:

```
turtle.write( متن , جابجایی , تراز متن , فونت )
```

پارامترها:

متن: نوشته‌ای است که می‌خواهیم روی صفحه نمایش داده شود.

جابجایی/move: اگر جابجایی را True باشد، جای لاکپشت پس از نوشتن من به سمت راست پایین متن تغییر می‌کند. این گزینه به صورت پیش فرض False است یعنی جای لاکپشت بعد از نوشتن متن تغییری نمی‌کند.

تراز متن/align: جای متن نسبت به جایی که لاکپشت قرار دارد را مشخص می‌کند. این گزینه در حال پیش فرض left است. یعنی جای لاکپشت سمت چپ نوشته است. گزینه center و right به ترتیب تراز وسط و سمت راست هستند.

فونت/font: تنظیم فونت (قلم) نوشتاری لاکپشت است. این پارامتر باید شامل سه چیز باشند که به ترتیب عبارتند از نام قلم، اندازه قلم، تنظیم قلم که باید در داخل یک پرانتز نوشته شوند.

مثال ۱) نمایش پیغام salam

```
import turtle  
turtle.write('salam')
```

مثال ۲) پیغام I Love Iran را بنویسید که هر کلمه دارای یک رنگ باشد.

```
import turtle as t
t.hideturtle()
t.penup()
t.pencolor('red')
t.write('I ',True)
t.pencolor('green')
t.write('Love ',True)
t.pencolor('blue')
t.write('Iran')
```

نکته: در سطر اول ما برای ماژول turtle یک نام مستعار (alias) تعریف کردیم و به جای آنکه بنویسیم turtle می‌توانیم از نام مستعار t استفاده کنیم.

تمرین: یکبار پارامترهای True را حذف کنید و برنامه را اجرا کنید تا نتیجه ثابت ماندن جای لاکپشت را ببیند.

مثال ۳) در مرکز صفحه پیغام GAME OVER! را به رنگ قرمز با فونت Arial و اندازه 30 و به صورت توپر (Bold) چاپ کنید.

```
import turtle as t
t.pencolor('red')
t.write('GAME OVER!', False, 'Center', ('arial',30,'bold'))
```

تمرین ۴) برنامه ای بنویسید که لاکپشت نام شما را با فونتی دلخواه به اندازه ۳۰ در صفحه نمایش دهد.

نکته: در این مثال ما پارامترهای «جابجایی» و «تراز متن» را نمی‌خواهیم تغییر دهیم و فقط پارامتر «فونت» را می‌خواهیم تنظیم کنیم. توجه داشته باشید که پارامتر فونت چهارمین پارامتر این دستور است و به همین دلیل (چون پارامتر دوم و سوم را نمی‌خواهیم بنویسیم) باید حتما نام پارامتر font به همراه علامت = مانند زیر نوشته شود.

```
import turtle as t
t.write('Farshid',font = ('arial',30,'normal'))
```

توجه: کلمه font و bold باید حروف کوچک باشند. در غیر اینصورت اجرای این دستور با خطا مواجه خواهد شد.

۳) تنظیمات صفحه لاکپشت

منظور از صفحه، یعنی همان پنجره‌ای که بر روی آن لاکپشت نمایش داده می‌شود. به این صفحه اصطلاحاً screen گفته می‌شود.

bgcolor

برای تغییر دادن (و همچنین برگرداندن) رنگ بک گراند (background) صفحه لاکپشت بکار می‌رود.
نکته ۱: bg مخفف background است.

نکته ۲: به صفحه یا پنجره‌ای که لاکپشت روی آن ظاهر می‌شود اصطلاحاً screen هم می‌گویند.
نحوه نگارش:

```
turtle.bgcolor( رنگ )
```

پارمترها:

رنگ داخل: کد رنگ قابل فهم توسط لاکپشت (به دستور pencolor نگاه کنید).

مثال ۱) برنامه ای بنویسید که رنگ صفحه به رنگ سبز زمین فوتبال تبدیل شود.

```
import turtle
turtle.bgcolor('green4')
```

مثال ۲: برنامه زیر صفحه را به آرامی از سیاه به سفید (شب به روز) رنگ می‌کند.

```
import turtle
import time
for i in range(101):
    turtle.bgcolor( 'grey' + str(i) )
    time.sleep(0.05)
```

bgpic

توسط این دستور می‌توانیم یک عکس/تصویر را به عنوان بک گراند (background) صفحه لاکپشت قرار دهیم.
نکته : pic مخفف کلمه picture به معنی تصویر/عکس است.

نحوه نگارش:

```
turtle.bgpic( نام فایل )
```

پارمترها:

نام فایل/picname : نام فایل عکس است.

نکته ۱: حتما باید فرمت فایل یکی gif و با png باشد و سایر فایل‌ها مانند jpg قابل قبول نیست.

نکته ۱: اگر این فایل دقیقا در همان مسیری باشد که برنامه شما ذخیره شده است، می‌توانید فقط نام فایل را بنویسید و لزومی به نوشتن مسیر کامل فایل نیست. اما اگر فایل در جای دیگری باشد، باید مسیر کامل فایل را بنویسید.
نکته ۲: اگر این پارامتری nopic باشد، تصویر بک گراند را حذف می‌کند.

مثال ۱)

```
import turtle
```

```
turtle.bgpic('sky.png')
```

مثال ۲) اشاره به یک عکس در ویندوز که در مسیر `d:\ax\sky.gif` قرار دارد.

```
import turtle
```

```
turtle.bgpic( r'd:\ax\sky.gif' )
```

نکته: به `r` در خط بالا دقت کنید. علامت `\` در متن‌ها دارای مفهومی است و به همین دلیل ما از تابع `r` در ابتدای این مسیر استفاده کردیم تا پایتون از مفهوم این علامت صرف‌نظر کند و آنرا به عنوان `\` تفسیر کند.

البته می‌توانیم برای اشاره به یک مسیر در ویندوز از علامت `\\` به جای `\` و بدون `r` استفاده کنیم:

```
turtle.bgpic( d:\\ax\\sky.gif' )
```

نکته: مسیرها عکس در رزبری پای و لینوکس با علامت `/` مشخص می‌شوند.

```
turtle.bgpic('/home/pi/Documents/sky.png')
```

`clear` | `clearscreen`

`reset` | `resetscreen`

screensize

در صورتی که پارامتری نداشته باشد، اندازه پهنا و ارتفاع صفحه جاری را بر می‌گرداند در غیر این صورت اندازه صفحه را تغییر می‌دهد.

شکل نگارش:

```
turtle.screensize( رنگ , ارتفاع صفحه , پهناى صفحه )
```

پارامترها:

پهناى صفحه/canvas width: عدد صحیح مثبت که پهنا/عرض صفحه را بر حسب پیکسل مشخص می‌کند.

ارتفاع صفحه/canvas height: عدد صحیح مثبت که ارتفاع صفحه را بر حسب پیکسل مشخص می‌کند.

رنگ/background: رنگ زمینه صفحه . این پارامتر اختیاری است.

مثال:

```
import turtle
```

```
turtle.screensize(1000,1500)
```

textinput

یک پنجره برای گرفتن یک متن از کاربر نمایش داده می‌شود.

شکل نگارش:

```
turtle.textinput( عنوان پنجره , پیغام )
```

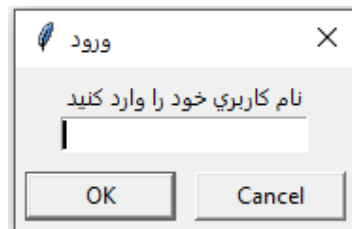
پارامترها:

عنوان پنجره/title: یک متن است که در عنوان پنجره (title bar) نمایش داده می‌شود.

پیغام/prompt: متنی پیغامی است که به کاربر نمایش داده می‌شود.

مثال:

```
import turtle
turtle.textinput( 'نام کاربری خود را وارد کنید' , 'ورود' )
```



numinput

یک پنجره برای گرفتن یک عدد از کاربر نمایش داده می‌شود.

شکل نگارش:

```
turtle.numinput( مقدار مجاز حداکثر , مقدار مجاز حداقل , مقدار پیش فرض , پیغام , عنوان پنجره )
```

پارامترها:

عنوان پنجره/title: یک متن است که در عنوان پنجره (title bar) نمایش داده می‌شود.

پیغام/prompt: متنی پیغامی است که به کاربر نمایش داده می‌شود.

مقدار پیش فرض/default: عددی است که به عنوان مقدار پیش فرض در پنجره نمایش داده می‌شود. (این

پارامتر اختیاری است)

مقدار مجاز حداقل/minimum value: کمترین مقداری که کاربر مجاز است وارد کند را مشخص می‌کند و

اگر عدد وارد شده از این مقدار کمتر باشد، پیغام خطایی نمایش داده خواهد شد. (این پارامتر اختیاری است)

مقدار مجاز حداکثر/maximum value: بیشترین مقداری که کاربر مجاز است وارد کند را مشخص می‌کند و

اگر عدد وارد شده از مقدار بیشتر باشد، پیغام خطایی نمایش داده خواهد شد. (این پارامتر اختیاری است)

مثال ۱: برنامه زیر سن یک فرد که باید بین ۱۰ تا ۲۰ باشد را می‌پرسد و در متغیر age ذخیره می‌کند. سن پیش فرض ۱۵

سال است.

```
import turtle
age = turtle.numinput('user info', 'Enter age ?', 15 , 10 , 20)
```

سخنی با آموزگاران/ برنامه نویسان حرفه‌ای / مسئولین

البته در بازار کتاب‌های زیادی است اما برای هدف و منظور من آن کتاب‌ها اصلا مناسب نبودند. اولین مشکل این کتاب‌ها آن است که مترجم و یا نویسنده آنها مدرس و استاد حرفه‌ای نیست. یعنی نویسنده سال‌ها تدریس نکرده است و ایده‌ای از اینکه چطور ذهن یک فرد و مخصوصا یک نوجوان ممکن است به سادگی بهم ریخته شود را ندارد. واقعا درکی از اینکه چطور ممکن است یک فرد از یادگیری بترسد را ندارند! (بله واقعا وقتی که شما بارها در کلاس درس می‌دهید ، خواهید دید که افراد از دیدن و شنیدن چیزهای جدید می‌ترسند ، و به تجربه یاد می‌گیرد که قبل از یاد دادن یک چیز جدید، باید مقدمه‌هایی بگویید که از ترسیدن فراگیران جلوگیری کنید)

من مشکلاتی را می‌بینم که شرح آنها به قرار زیر است:

۱) متأسفانه در مدارس ایران زبان C# تدریس می‌شود و به نظر من این آسیب جدی است. یعنی بچه‌ها به جای لذت بردن از برنامه نویسی‌های جالب و هیجان‌انگیز، با یک محیط جدی و خشک مواجه می‌شوند و حتی نمی‌توانند یک دایره به صورت گرافیکی بکشند. هیچ مثال هیجان‌انگیزی وجود ندارد. در کل کتاب‌های درسی یک بازی ساده وجود ندارد!

جالب آن است که حتی در تایپ کتاب‌های درسی فنی و حرفه‌ای نیز بی‌سلیقه‌ی بزرگی به چشم هر فرد حرفه‌ای در اولین نگاه به چشم می‌خورد. فونت برنامه‌ها کاملاً فونت اشتباهی است و Mono Space نیست! این نکته کوچکی نیست! برنامه‌نویسان حرفه‌ای کاملاً فونت و محیط برنامه نویسی برای آنها مهم است.

دوم اینکه ترجمه‌ها گویا نیستند و حتی یک فرد حرفه‌ای که آنها را مطالعه می‌کند گاهی از درک مطلب عاجز می‌ماند. امیدوارم که معلمینی که به برنامه نویسی علاقه دارند، در مورد زبان پایتون تحقیق کنند و هنگامی که مطمئن شدند که یکی از بهترین زبان‌های این سیاره است، خودشان آن را یاد بگیرند و آهسته آهسته به بچه‌ها آنرا درس دهند.

به نظر می‌آید که ترجمه کتاب‌های انگلیسی برای بچه‌های ایران کارساز نیست و حتماً باید کتابی تالیفی و ویژه بچه‌های ایران نوشت. واقعیت آن است که یک کتاب انگلیسی که برای بچه‌های انگلیسی نوشته شده است با یک فرض صحیح نوشته می‌شود و آن این است که بچه‌ها با واژه‌هایی از کامپیوتر کاملاً آشنا هستند و در زندگی روزمرشان هزاران بار آنها را بکار می‌برند. مثلاً فکر نکنم که واژه yield و یا import چیز سختی برای بچه‌های انگلیسی زبان باشد اما برای بچه‌های ایران نمی‌توان این واژه‌ها را بکار برد و باید این واژه‌ها هم به آنها تدریس شود. این موضوع امری است که معمولاً مترجمین کتاب‌های کامپیوتری (که معمولاً مهندس هستند نه مترجم حرفه‌ای) به طور چشمگیری از آن مغفول مانده‌اند و من ندیده‌ام که بر رفع آن اهتمامی بورزند.

در پایان اشاره می‌کنم که من آموزش پایتون را برای بچه‌های ایران کمی سنگین دیدم و به همین دلیل ادامه نگارش این کتاب را رها کردم و بر روی زبان برنامه نویسی اسکرچ کار کردم که نتیجه آن را می‌توانید در سایت فرساران ببیند. البته امیدوارم فرصتی و همتی دست دهد تا این کتاب را تکمیل و به سرانجام برسانم.

فرشید میدانی

f.meidani@farsaran.com