فصل اول

مقدمات و ساختار برنامه در پایتون

پایتون یکی از زبان‌های برنامه‌سازی مدرن است که تمام کاستی‌های زبان‌های برنامه‌سازی قبل از خودش را پوشش داده است و به یک زبان ساده، قدرتمند و کارآمد تبدیل شده است. قابلیت‌های این زبان به گونه‌ای است که برای نوشتن اپلیکیشن‌های مختلف و در سکوهای گوناگونی مثل ویندوز و لینوکس مناسب است. زبان‌های قبل از آن، مثل C، جاوا و C++ هر کدام مشکلاتی داشتند که این زبان آن‌ها را برطرف کرد و به یک زبان کامل تبدیل شده است.

۱-۱. ویژگی‌های زبان پایتون

هر زبان برنامه‌سازی ویژگی‌هایی دارد که آن را از سایر زبان‌ها متمایز می‌سازد. در این بخش به بعضی از ویژگی‌های پایتون اشاره خواهیم کرد:

۱. متن‌باز[1] است: توزیع‌های مختلفی از این زبان به طور رایگان در اختیار قرار می‌گیرد و برنامه‌نویس به کد آن دسترسی دارد و می‌تواند تغییرات لازم را در آن انجام دهد و در برنامه‌های خود به کار ببرد.

۲. مفسری[2] است: پایتون یک زبان مفسری است. پایتون از طریق یک نرم‌افزار به نام مفسر پایتون اجرا می‌شود. مفسر پایتون یک دستور از برنامه را می‌گیرد، آن دستور را ارزیابی می‌کند و نتیجه‌ی دستور را گزارش می‌دهد. در حالی که مفسر می‌تواند به طور محاوره‌ای مورد استفاده قرار گیرد (مخصوصاً هنگام

1. **Open source**
2. **Interpreter**

اشکال‌زدایی)، برنامه‌نویس معمولاً مجموعه‌ای از دستورات را می‌نویسد و در فایلی ذخیره می‌کند که به نام **اسکریپت**[3] یا **کد منبع**[4] خوانده می‌شود. در پایتون، فایل منبع با پسوند py ذخیره می‌شود. وقتی پایتون را راه‌اندازی می‌کنید، در حالت محاوره‌ای قرار می‌گیرید. پایتون به همراه خود یک محیط مجتمع برای تایپ، ذخیره، اشکال‌زدایی و اجرای برنامه دارد.

۳. شیءگرا است: این زبان به راحتی می‌تواند اشیای دنیای واقعی را مدل‌سازی کند و به همین دلیل آن را زبان شیءگرا می‌نامند. برنامه‌نویسی به زبان‌های شیءگرا با طرز تفکر انسان‌ها درباره‌ی مسائل دنیای واقعی سازگاری کامل دارد. در پایتون همه چیز به عنوان یک شیء شناخته می‌شود.

۴. توسعه‌پذیر است: برای جبران پایین بودن سرعت به خاطر مفسری بودن زبان، این امکان در آن فراهم شده است که بتوان کدهای حیاتی‌تر را با زبان‌های دیگری مثل C نوشت و همراه با کدهای پایتون اجرا کرد.

۵. ساده است: این زبان دارای دستورالعمل‌ها و ساختارهای ساده است، به طوری که طول یک برنامه‌ی یکسان در این زبان با زبان‌هایی مثل C، C++ یا جاوا به مراتب کمتر است. به همین دلیل یادگیری آن آسان است.

۶. کتابخانه‌های غنی دارد: به دلیل اینکه هسته‌ی پایتون فاقد برخی قابلیت‌ها است، کتابخانه‌های قدرتمند و غنی برای آن نوشته شده است. دو کتابخانه‌ی معروف آن NumPy و SciPy هستند که اولی از واژه‌های Numerical Python و دومی از واژه‌های Science Python گرفته شده‌اند. این دو کتابخانه، قابلیت‌های کارهای علمی و محاسباتی در پایتون، به‌خصوص کار کردن با کلان‌داده‌ها را فراهم کرده‌اند.

---

3. Script
4. Source code

۷. زبان سطح بالا است: دستورالعمل‌ها و قابلیت‌های آن نزدیک به دستورالعمل‌ها و زبان محاوره‌ای است و به همین دلیل دارای سطح انتزاعی بالاتری نسبت به زبان‌های دیگر است.

۸. کاربردهای گسترده دارد: زبان پایتون قابلیت‌هایی دارد که حوزه‌ی کاربرد آن بسیار گسترده است. از جمله:

- بخش‌هایی از سرویس‌های یاهو و گوگل
- سرویس ابری Dropbox
- هوش مصنوعی و یادگیری عمیق
- وب‌سایت‌های پویا و داده‌کاوی

۹. برای اپلیکیشن‌های مختلف مناسب است: از جمله:

- ساخت اپلیکیشن‌های وب
- ساخت اپلیکیشن‌های علمی و محاسباتی
- طراحی واسط کاربری مبتنی بر تجربه‌ی کاربر
- نمونه‌سازی اولیه‌ی اپلیکیشن‌ها با سرعت بالا

۱-۲. نصب و اجرای پایتون

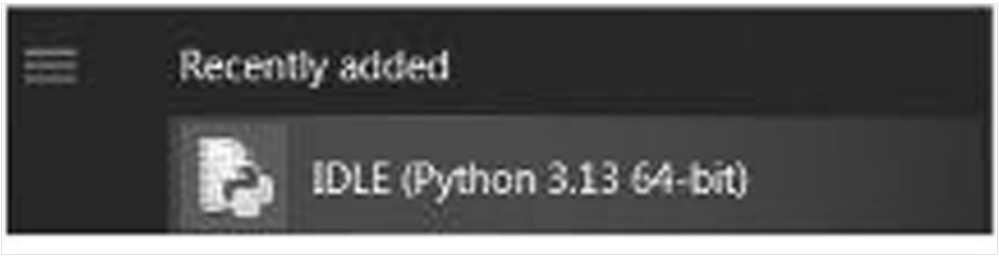
اکنون که به مرور پایتون چه ویژگی‌هایی دارد و با حوزه‌ی کاربردهای آن آشنا شدید، وقت آن رسیده است که کارمان را با این زبان شروع کنیم. برای شروع کار، باید پایتون را در سیستم خود نصب کنید. برای نصب پایتون مراحل زیر را انجام دهید:

۱. به سایت www.python.org بروید.

۲. در لینک Downloads، سیستم‌عامل موردنظرتان، مثلاً ویندوز را انتخاب کنید تا نسخه‌های تحت ویندوز پایتون ظاهر شوند. نسخه‌ی موردنظرتان را دانلود کنید.

۳. پس از دانلود کردن پایتون، بر روی فایل اجرایی آن کلیک مضاعف کنید تا مراحل نصب آن آغاز شود. جزئیات نصب را از پیوست کتاب ببینید. پس از نصب نرم‌افزار، برای راه‌اندازی آن مراحل زیر را دنبال کنید.

۴. در منوی Start ویندوز در کادر جست‌وجو، دو حرف py را تایپ کنید تا لیست نرم‌افزارهایی ظاهر شود که با py شروع شده‌اند (شکل ۱-۱).



شکل ۱-۱. نرم‌افزار پایتون در لیست برنامه‌های ویندوز

۵. نرم‌افزار IDLE (Python 3.x y-bit) را پیدا کرده، یک میانبر از آن را روی دسکتاپ ویندوز قرار دهید و آن را اجرا کنید (شکل ۱-۲). اکنون پایتون در حالت محاوره‌ای قرار می‌گیرد و علامت >>> نشان‌دهنده‌ی این محیط است. در این محیط هر دستوری از پایتون را که تایپ کنید و کلید Enter را فشار دهید، آن دستور ارزیابی و اجرا می‌شود و نتیجه‌ی آن گزارش داده می‌شود. اجازه دهید جلوتر برویم و دستورات پایتون را یاد بگیرید تا از این محیط استفاده کنید.

۶. برای اینکه برنامه‌ی کاملی بنویسید و در فایلی با پسوند py ذخیره کنید و سپس دستورات این برنامه را به طور یکجا اجرا کنید، باید در منوی File نرم‌افزار پایتون گزینه‌ی New File را انتخاب کنید تا یک ویرایشگر در اختیار شما قرار گیرد و

برنامه‌ی خود را در آن تایپ، ویرایش و ذخیره کنید. مراحل انجام این کار را در ادامه به شما می‌آموزیم. فعلاً پنجره‌ی پایتون را ببندید و از آن خارج شوید.

۱-۳. اجزای تشکیل‌دهنده‌ی برنامه پایتون

برنامه‌ی پایتون از اجزایی تشکیل شده است که برنامه‌نویس برای شروع برنامه‌نویسی در آن، باید این اجزا را بشناسد. در این بخش، اشاره‌ای به برخی از اجزای پایتون خواهیم داشت که آشنایی با آن‌ها برای شروع برنامه‌نویسی ضروری است.



شکل ۱-۲. اجرای پایتون و ورود به محیط محاوره‌ای

یادداشت: متن این نمونه از صفحات ۱۰ تا ۱۲ فایل اسکن‌شده، با بازخوانی انسانی و ویرایش تایپی استخراج شده است. تصویرهای محیط برنامه از منبع اصلی حفظ شده‌اند؛ نسخه‌ی کامل کتاب هنوز بازسازی و کنترل نشده است.

انواع داده‌ها[1]: هر برنامهٔ کامپیوتری با داده‌ها سروکار دارد؛ به طوری که داده‌ها را پردازش می‌کند و نتایجی را در اختیار کاربران قرار می‌دهد. پایتون نیز تعدادی از انواع داده‌ها دارد که باید مورد بررسی قرار گیرند. به عنوان مثال، مقادیری مثل ۱۲، ۱۷ و ۱۸۵ که نقطهٔ اعشار ندارند، به عنوان نوع دادهٔ صحیح و مقادیری مثل ۱۴٫۵ و ۱۹٫۷۵ به عنوان نوع دادهٔ اعشاری خوانده می‌شوند. انواع داده‌ها را در ادامه بررسی خواهیم کرد.

دستورالعمل‌ها[2]: هر برنامه در زبان پایتون از تعدادی دستورالعمل تشکیل می‌شود و هر دستورالعمل کاری را انجام می‌دهد؛ مثلاً داده‌ای را از ورودی می‌خواند یا محاسباتی را انجام می‌دهد یا اینکه نتایجی را به خروجی می‌برد.

عملگرها[3]: نمادهایی هستند که عمل خاصی را انجام می‌دهند؛ مثلاً نماد + عملگری است که عمل جمع دو یا چند مقدار را انجام می‌دهد.

توابع[4]: تابع قطعه‌کدی به زبان پایتون است که از قبل نوشته شده (یا برنامه‌نویس آن را می‌نویسد) و بخشی از مسئله را حل می‌کند. تابعی که از قبل نوشته شده و همراه پایتون قابل استفاده است، **تابع کتابخانه‌ای**[5] نامیده می‌شود. این توابع برای انجام کارهای متداول و تکراری نوشته شده‌اند. مثل تابع `sqrt()` برای محاسبهٔ جذر یک عدد و یا تابع `sin()` برای محاسبهٔ سینوس یک زاویه. تعداد زیادی از توابع به همراه پایتون وجود دارند که برنامه‌نویس می‌تواند از آنها استفاده کند. تابع یک نام دارد؛ مثل `sqrt()` و اطلاعاتی را می‌پذیرد که روی آنها عملی را انجام دهد که به آنها **پارامتر**[6] گفته می‌شود.

---

1. Data types
2. Statements
3. Operators
4. Functions
5. Library function
6. Parameter

به عنوان مثال، تابع `sqrt()` مقداری مثل عدد ۱۰۰ را به عنوان پارامتر دریافت می‌کند، جذر آن را محاسبه می‌کند و نتیجه را برمی‌گرداند. مثل `sqrt(100)` که حاصل آن ۱۰ است.

کلاس‌ها: کلاس موجودیتی است که شامل داده‌ها و توابعی می‌شود که توابع بر روی آن داده‌ها عمل می‌کنند. داده‌هایی که کلاس در بر می‌گیرد، **خصیصه‌ها**[7]ی کلاس نام دارند و توابعی که بر روی خصیصه‌ها عمل می‌کنند، **متدهای**[8] کلاس خوانده می‌شوند. به عبارت دیگر، هر کلاس دو نوع عضو دارد: خصیصه‌ها و متدها. تعداد زیادی از کلاس‌ها قبلاً نوشته شده‌اند و همراه زبان پایتون وجود دارند و برنامه‌نویس می‌تواند از آنها استفاده کند. علاوه بر این، برنامه‌نویس نیز در صورت نیاز می‌تواند کلاس‌هایی برای حل مسئله‌های خود بنویسد که چگونگی نوشتن کلاس‌ها را در پایتون به شما آموزش خواهیم داد.

کلاس مثل یک نقشهٔ ساختمان عمل می‌کند که با یک نقشه می‌توان صدها ساختمان درست کرد؛ یعنی نقشه به عنوان الگوی ساخت عمل می‌کند که ساختمان از آن ساخته می‌شود. هر کلاس به عنوان الگو عمل می‌کند که اشیای متعددی از آن ساخته می‌شوند. ساخت شیء از کلاس را **نمونه‌سازی**[9] می‌گویند. هر کلاس و نیز اشیای آن دارای نام هستند و از طریق نام به آنها مراجعه می‌شود.

هر کلاس یک متد ویژه دارد که وظیفه‌اش ساخت یک شیء از آن کلاس است و آن را **سازنده**[10]ی کلاس می‌نامند. نام این متد برای هر کلاس `__init__` است. هنگام ایجاد کلاس‌ها، روش نوشتن متدهای سازنده را هم خواهیم دید. فعلاً فقط اطلاع داشته باشید

---

7. Attributes
8. Methods
9. Instantiation
10. Constructor

که سازنده‌های هر کلاس، وظیفهٔ ساخت اشیا را بر عهده دارند. وقتی نام کلاس را در برنامه برای نمونه‌سازی مورد استفاده قرار دهید، متد سازندهٔ آن به طور خودکار فراخوانی می‌شود.

ماژول‌ها[11]: ماژول در واقع ابزاری برای سازمان‌دهی کلاس‌ها یا توابعی است که در پایتون وجود دارند. ماژول همچون یک فایل برنامه در پایتون است، با این تفاوت که محتوای آن، کلاس‌ها و توابعی هستند که کار خاصی را می‌توانند انجام دهند؛ مثلاً توابع و کلاس‌های مربوط به اعمال ریاضی در پایتون، در یک ماژول به نام `math` گنجانده شده‌اند.

کتابخانه‌ها[12]: کتابخانه‌ها در پایتون بسیار گسترده‌اند و گسترهٔ وسیعی از امکانات را فراهم می‌آورند. کتابخانه شامل ماژول‌های آماده است که امکان دسترسی به عملکردهای سیستم مثل ورودی ـ خروجی فایل‌ها را فراهم می‌سازد و بدون آنها برنامه‌نویسان نمی‌توانند به فایل‌ها دسترسی داشته باشند. علاوه بر این، شامل ماژول‌هایی است که راه‌حل استانداردی برای بسیاری از مسئله‌هایی ارائه می‌دهند که روزانه با آنها سروکار داریم. کتابخانه برای توسعهٔ سریع برنامه‌ها مفید واقع می‌شود.

11. Modules
12. Libraries

۱-۴. مفهوم شناسه و متغیر

شناسه‌ها نام‌هایی هستند که برای شناسایی موجودیت‌های گوناگون در پایتون مورد استفاده قرار می‌گیرند. هر موجودیت مثل کلاس، تابع، ماژول، شیء و متغیر دارای یک نام است که به عنوان **شناسه**[13] شناخته می‌شود. بنابراین، برای نام‌گذاری این موجودیت‌ها از شناسه استفاده می‌کنیم. شناسه‌ها در پایتون دارای ویژگی‌های زیر هستند:

- با حروف a تا z یا A تا Z یا خط زیر (_) شروع می‌شوند که بعد از آن یک یا چند حرف، ارقام صفر تا نه یا خط زیر می‌تواند قرار گیرد؛ مثل math، text، ave_digit و money_1.
- طول شناسه محدودیتی ندارد و علائم نگارشی[14] مثل @، % و نقطه و فضای خالی نمی‌توانند در شناسه ظاهر شوند.
- در انتخاب شناسه بین حروف کوچک و بزرگ تفاوت است؛ به عنوان مثال شناسهٔ SUM با شناسهٔ sum متفاوت است و شناسهٔ Temperature با شناسهٔ temperature یکسان نیست.

برخی از شناسه‌های مجاز و غیرمجاز در پایتون را در جدول ۱-۱ مشاهده می‌کنید. برخی از نام‌ها در پایتون وجود دارند که به عنوان واژه‌های رزرو شده‌اند و نباید به عنوان شناسه انتخاب شوند. این واژه‌ها در جدول ۱-۲ نشان داده شده‌اند.

13. Identifier

14. Punctuation characters

شناسه	مجاز یا غیرمجاز	دلیل
sum	مجاز	رعایت قوانین نام‌گذاری شناسه
_ave	مجاز	رعایت قوانین نام‌گذاری شناسه
1_test	غیرمجاز	استفاده از رقم در ابتدای شناسه
mail@t	غیرمجاز	استفاده از علامت @ در شناسه
tem.x	غیرمجاز	استفاده از نقطه در شناسه

جدول ۱-۱. برخی از شناسه‌های مجاز و غیرمجاز

False	as	continue	else	from	in	not	yield
None	assert	except	global	is	or	try	while
True	break	del	finally	if	lambda	pass	with
and	class	elif	for	import	nonlocal	raise	

جدول ۱-۲. واژه‌های رزروشده در پایتون (مطابق کتاب اصلی)

متغیرها در پایتون برچسب‌هایی هستند که برای اشیا انتخاب می‌شوند. در واقع، متغیرها به عنوان یک **مرجع**[15] یا یک **اشاره‌گر**[16] به یک شیء عمل می‌کنند. متغیرها اشیا نیستند و به عنوان کانتینر یا ظرفی برای نگهداری اشیا محسوب نمی‌شوند. همه چیز در پایتون یک شیء محسوب می‌شود و متغیرها برای ارجاع به اشیا مورد استفاده قرار می‌گیرند. برای اینکه پایتون را وادار کنیم یک متغیر را برای ارجاع به یک شیء ایجاد کند، از علامت = به معنای علامت انتساب استفاده می‌کنیم. به عنوان مثال، اگر بخواهیم متغیری ایجاد کنیم که به شیء ۱۵ اشاره کند، به صورت زیر عمل می‌کنیم:

```
temp = 15
```

15. Reference

16. Pointer

می‌دانید که ۱۵ یک مقدار عددی **صحیح**[17] است که در پایتون به عنوان یک شیء تلقی می‌شود. با این دستور انتساب، حافظه‌ای که تخصیص می‌یابد به شرح شکل زیر است (دایره نشان‌دهندهٔ مرجع شیء و مستطیل نشان‌دهندهٔ شیء است):

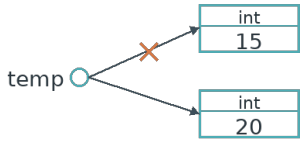


نمایش ارجاع متغیر `temp` به شیء صحیح ۱۵، مطابق شکل اصلی

اکنون دستور زیر را اجرا می‌کنیم:

```
temp = 20
```

با اجرای این دستور، حافظه به صورت شکل زیر درمی‌آید:



پس از انتساب جدید، `temp` به شیء صحیح ۲۰ اشاره می‌کند

یعنی با اجرای این دستور، مرجع `temp` به شیء جدید ۲۰ اشاره می‌کند و ارتباط آن با شیء ۱۵ قطع می‌شود. توجه کنید که ۱۵ یک شیء از نوع صحیح است و اشیای نوع صحیح در پایتون **تغییرناپذیر**[18] هستند.

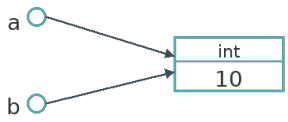
بنابراین، وقتی شیء جدیدی به مرجع شیء نوع صحیح نسبت داده می‌شود، آن مرجع دیگر به آن شیء اشاره نمی‌کند، بلکه به شیء جدید اشاره خواهد کرد.

---

17. Integer
18. Immutable

اکنون دستورات زیر و نتیجهٔ اجرای آنها را ببینید:

```
a = 10
b = a
```

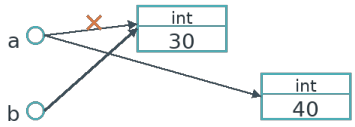


دو مرجع a و b به شیء ۱۰ اشاره می‌کنند

با اجرای دستور اول، مرجع a به شیء ۱۰ اشاره می‌کند و با اجرای دستور دوم، مرجع b هم به شیء ۱۰ اشاره می‌کند.

اکنون دستورات زیر را در نظر بگیرید:

```
a = 30
b = a
a = 40
```



قطع ارتباط a با شیء ۳۰ و اتصال آن به شیء ۴۰

با اجرای دستور اول، متغیر a به شیء ۳۰ اشاره می‌کند. با اجرای دستور دوم، متغیر b به همان شیء اشاره می‌کند که مرجع a به آن اشاره می‌کند و با اجرای دستور سوم، ارتباط متغیر a با شیء ۳۰ قطع می‌شود که با علامت × روی پیوند نشان داده شده است و متغیر a به شیء ۴۰ اشاره خواهد کرد. با این توضیحات باید متوجه شده باشید که بین شیء و مرجع شیء (متغیر) تفاوت وجود دارد و اشیای نوع صحیح تغییرناپذیر هستند.

۱-۵. انواع داده‌ها

قبل از اینکه مفهوم داده‌ها را در پایتون بیان کنیم، خوب است از فلسفهٔ وجودی آن آگاهی پیدا کنیم. برای این منظور به چند سؤال و پاسخ آنها توجه کنید:

سؤال: چرا برنامه می‌نویسیم؟

پاسخ: برنامه می‌نویسیم تا مسائل دنیای واقعی خود را با کامپیوتر حل کنیم، زیرا برنامه‌ها را کامپیوترها اجرا می‌کنند.

سؤال: نمونه‌هایی از مسائل دنیای واقعی که باید با برنامه‌نویسی حل شوند کدام‌اند؟

پاسخ: ثبت‌نام داوطلبان کنکور سراسری، ثبت‌نام دانشجویان در دانشگاه‌ها و انتخاب واحد، پرداخت حقوق و دستمزد کارکنان یک شرکت یا سازمان، مدیریت حساب‌های بانکی، حسابداری شرکت‌ها و سازمان‌ها و این لیست همچنان ادامه دارد.

سؤال: وظیفه‌های کلی برنامه‌ها چیست؟

پاسخ: تمام برنامه‌ها، چه کوچک و چه بزرگ، چه ساده و چه پیچیده، چه تجاری و چه علمی، سه وظیفهٔ کلی زیر را بر عهده دارند:

1. دریافت داده‌ها از دنیای واقعی
2. پردازش داده‌های دریافتی

3. ارائهٔ نتایج حاصل از پردازش داده‌ها

با این پرسش‌ها و پاسخ‌های آنها متوجه شدید که برنامه‌ها با داده‌ها سروکار دارند. حالا به سؤال دیگر و پاسخ آن توجه کنید.

سؤال: داده‌های دنیای واقعی چه شکلی‌اند؟

پاسخ: داده‌های دنیای واقعی به شکل‌های مختلفی ظاهر می‌شوند؛ به عنوان مثال، معدل دانشجو، موجودی حساب بانکی، شمارهٔ دانشجویی، نام دانشجو، عنوان رشتهٔ تحصیلی و شمارهٔ داوطلبی در کنکور سراسری، نمونه‌هایی از داده‌های دنیای واقعی هستند که هر برنامهٔ کامپیوتری باید بتواند آنها را پردازش کند.

برخی از داده‌هایی که نام بردیم، **مقادیر عددی**[19] دارند؛ مثل شمارهٔ دانشجویی، شمارهٔ داوطلبی، موجودی حساب بانکی و معدل دانشجو. از بین این مقادیر عددی برخی از آنها **مقادیر صحیح**[20] و بدون نقطهٔ اعشار هستند؛ مثل شمارهٔ داوطلبی و شمارهٔ دانشجویی و برخی دیگر **مقادیر اعشاری**[21] هستند و می‌توانند شامل نقطهٔ اعشار باشند؛ مثل معدل دانشجو و موجودی حساب بانکی.

علاوه بر این، انواع دیگری وجود دارند که از نوع عددی نیستند؛ مثل نام و نام خانوادگی و عنوان رشتهٔ تحصیلی دانشجو. این نوع داده‌ها هم باید در کامپیوتر ذخیره و پردازش شوند. به این نوع داده‌ها در اصطلاح زبان‌های برنامه‌سازی مثل پایتون، **نوع رشته‌ای**[22] گفته می‌شود.

---

19. Numeric values
20. Integer values
21. Floating point
22. String type

به طور کلی، **پایتون اصلی**[23] (پایتون بدون اضافه شدن کلاس‌های جدید) انواع داده‌ای را به عنوان انواع دادهٔ اولیه (**توکار**[24]) تعریف کرده است که عبارتند از:

- انواع عددی شامل نوع صحیح با `int` و نوع اعشاری با `float`
- نوع بولی با `bool`
- نوع رشته‌ای با `str`
- نوع لیست با `list`
- نوع مجموعه با `set`
- نوع مجموعهٔ غیرقابل‌تغییر با `frozenset`
- نوع تاپل (چندتایی) با `tuple`
- نوع دیکشنری با `dict`

در پایتون، هر یک از انواع به صورت کلاس‌ها تعریف شده‌اند که در جدول ۱-۳ مشاهده می‌شوند. بعضی از این کلاس‌ها **تغییرپذیر**[25] و بعضی دیگر **تغییرناپذیر**[26] هستند.

23. Core Python
24. Built-in
25. Mutable
26. Immutable

کلاس	توصیف	تغییرپذیر؟
`bool`	مقادیر بولی `True` و `False`	خیر
`int`	مقادیر صحیح با هر طولی	خیر
`float`	مقادیر ممیز شناور	خیر
`tuple`	دنباله‌ای از اشیای تغییرناپذیر	خیر
`str`	رشته‌های کاراکتری	خیر
`list`	دنباله‌ای از اشیای تغییرپذیر	بله
`set`	مجموعه‌های مرتب‌نشده از اشیا	بله
`frozenset`	شکل تغییرناپذیر کلاس `set`	خیر
`dict`	دیکشنری (نگاشت انجمنی)	بله
`complex`	اعداد مختلط را اداره می‌کند	خیر

جدول ۱-۳. کلاس‌های انواع داده در پایتون

کلاس را در صورتی **تغییرپذیر** گویند که وقتی شیءای از آن کلاس در هنگام ایجاد شدن مقدار گرفت، قابل تغییر باشد؛ مثل کلاس `list` که در جدول ۱-۳ مشاهده می‌کنید. کلاس را در صورتی تغییرناپذیر گویند که وقتی شیءای از آن کلاس در هنگام ایجاد شدن مقدار گرفت، دیگر قابل تغییر نباشد؛ مثل کلاس `int` در جدول ۱-۳ که توضیحاتی دربارهٔ آن را دیدید.

علاوه بر این کلاس‌ها، برای اداره کردن انواع دادهٔ مختلف، نوعی به نام **None** وجود دارد که یک شیء تهی است؛ یعنی وقتی این شیء را به یک متغیر (یا مرجع) نسبت می‌دهیم، یعنی آن شیء به جایی اشاره نمی‌کند. دستورات زیر را ببینید:

```
x = 10
x = None
```

۱-۵-۱. نوع دادهٔ صحیح

نوع دادهٔ صحیح در پایتون برای نمایش اعداد صحیح مثل ۱۵، ۱۲۵، ۱۷۸۷۱۹ و منفی ۱۴۷۸ به کار می‌رود که شامل اعداد منفی و مثبت بدون نقطهٔ اعشار است و با کلاس `int` مشخص می‌شود. در پایتون، اندازهٔ مقادیر صحیح فقط با حافظهٔ ماشین محدود می‌شود و با تعداد ثابتی از بایت‌ها مشخص نمی‌شود. به عنوان مثال، مقدار زیر یک مقدار صحیح است:

```
20162438375212211924 6
```

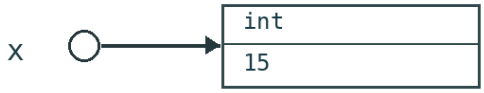
بنابراین، اعداد صحیح صدها رقمی را می‌توان ایجاد کرد، ولی ممکن است سرعت پردازش این اعداد کاهش یابد.

علامت `0x` یا `0X` نشان می‌دهد که یک مقدار در مبنای شانزده است. به عنوان مثال، مقدار `0o4371` در مبنای هشت و مقدار `0xb7D` در مبنای شانزده است.

برای اینکه متغیری مثل `x` به شیئی از نوع صحیح اشاره کند، باید با دستور انتساب (=) یک مقدار صحیح را به آن نسبت داد. دستور زیر را ببینید:

```
x = 15
```

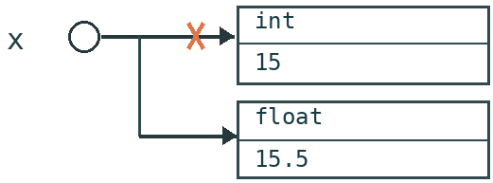
با اجرای این دستور، وضعیت حافظه به صورت زیر است:



متغیر x به شیء صحیح با مقدار 15 اشاره می‌کند

معنایش این است که متغیر یا مرجع `x` به مکان حافظه‌ای اشاره می‌کند که مقدارش ۱۵ و نوع آن مقدار `int` است؛ یعنی متغیر `x` فاقد نوع است، بلکه مقداری که `x` به آن اشاره می‌کند دارای نوع است. `int` نشان می‌دهد که در حال حاضر `x` به شیئی از نوع `int` اشاره می‌کند. اگر بعداً نوع دیگری مثل اعشاری به `x` نسبت داده شود، وضعیت فرق می‌کند. دستورات زیر را ببینید:

```
x = 15
x = 15.5
```



قطع ارجاع قبلی و تغییر مرجع x از شیء صحیح به شیء اعشاری

اجرای دستور اول موجب می‌شود که x به شیئی از نوع صحیح، یعنی ۱۵ اشاره کند، ولی اجرای دستور دوم باعث می‌شود که x از این لحظه به بعد به شیئی اعشاری اشاره کند. این تصاویر نشان می‌دهند که همراه هر مقدار، توصیفی از نوع آن وجود دارد که سیستم در هر لحظه متوجه می‌شود متغیر به چه نوعی اشاره دارد. به عنوان مثال، `int` و `float` در شکل بالا، توصیفگر نوع هستند.

هر کدام از کلاس‌های جدول ۱-۳ متد ویژه‌ای به نام **متد سازنده**[1] دارد که وظیفه‌اش ایجاد نمونه‌ای از آن کلاس است. برای ایجاد شیئی از یک کلاس، کافی است نام آن را همراه با پرانتز ذکر کنید. به عنوان مثال، کلاس `int` که برای اداره کردن انواع دادهٔ صحیح به کار می‌رود، متدی به نام `int()` دارد که اشیایی از نوع صحیح را ایجاد می‌کند. به عنوان مثال، دستور زیر را ببینید:

```
x = int(15.4)
```

در این دستور، متد سازندهٔ `int()` مقدار اعشاری ۱۵٫۴ را به ۱۵ تبدیل می‌کند و در x قرار می‌دهد.

۲-۵-۱. نوع دادهٔ اعشاری

نوع دادهٔ اعشاری شامل مقادیری با نقطهٔ اعشار می‌شود که به آن‌ها **ممیز شناور**[2] گفته می‌شود. مقادیری مثل `19.75` و `-14.25` به عنوان مقادیر اعشاری خوانده می‌شوند. یک روش نمایش اعداد به نام «نماد علمی» در صفحهٔ بعد توضیح داده می‌شود.

1. Constructor
2. Floating point

یک روش نمایش اعداد به نام **نماد علمی**[3] خوانده می‌شود و اعدادی مثل 7×10^{15} و 4×10^{-8} را می‌توان به ترتیب به صورت `7E15` (یا `7e15`) و `4e-8` (یا `4E-8`) نمایش داد. نوع دادهٔ اعشاری در پایتون با کلاس `float` مشخص می‌شود.

برای ایجاد متغیری که به شیئی از نوع اعشاریِ ممیز شناور اشاره کند، کافی است یک مقدار ممیز شناور را به آن متغیر نسبت دهیم. دستور زیر را ببینید:

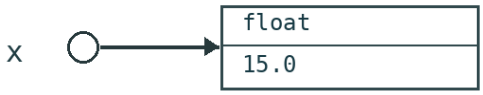
```
x = 17.75
```



مرجع `x` به یک شیء `float` با مقدار `17.75` اشاره دارد

کلاس `float` سازنده‌ای به نام `float()` دارد که یک شیء از نوع اعشاری ممیز شناور ایجاد می‌کند. دستور زیر را ببینید:

```
x = float(15)
```



خروجی سازندهٔ `float()` برای مقدار صحیح ۱۵

3. Scientific notation

۱-۵-۳. نوع دادهٔ مختلط

هر عدد مختلط دارای دو بخش است: بخش **حقیقی**[4] و بخش **موهومی**[5]. این نوع داده‌ها با کلاس `complex` در پایتون تعریف و اداره می‌شوند که در پایتون اصلی وجود نداشته است و بعداً به آن اضافه شد. این کلاس سازنده‌ای دارد که عدد مختلط را تولید می‌کند. دستور زیر را ببینید:

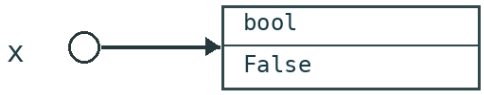
```
a = complex(3, 2)
```

این دستور عدد مختلط `3 + 2j` را ایجاد می‌کند.

۱-۵-۴. نوع دادهٔ بولی

این نوع داده در پایتون با کلاس `bool` مشخص می‌شود و فقط دو نمونه از این کلاس با مقادیر `True` و `False` ایجاد می‌شوند. این کلاس متد سازنده‌ای به نام `bool()` دارد که به طور پیش‌فرض، یک مقدار `False` را تولید می‌کند. دستور زیر یک مقدار `False` را به وجود می‌آورد:

```
a = bool()
```



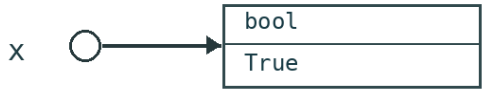
ارجاع به شیء بولی با مقدار `False`

اگر پارامتر `bool()` عبارت دیگری باشد، چنانچه آن عبارت به صفر یا تهی ارزیابی شود، مقدار `False` و در غیر این صورت به `True` ارزیابی می‌شود. دستور زیر را ببینید:

---

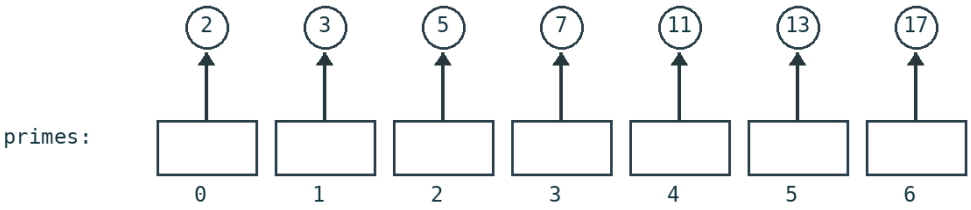
4. Real
5. Imaginary

```
a = bool(5)
```



ارجاع به شیء بولی با مقدار `True`

مقدار ۵ غیرصفر است و در نتیجه به مقدار `True` ارزیابی می‌شود.



شکل ۱-۳. نمایش داخلی لیست با ۷ عنصر در پایتون

۱-۵-۵. نوع دادهٔ لیست

نوع دادهٔ لیست که با کلاس `list` در پایتون ایجاد می‌شود، از نوع **دنباله**[6] است که مجموعه‌ای از مقادیر را نشان می‌دهد که ترتیب آن‌ها مهم است. در واقع، لیست ساختاری است که مجموعه‌ای از ارجاع‌ها به تعدادی عنصر دارد (شکل ۱-۳). عناصر لیست می‌توانند از هر نوعی باشند؛ یعنی نوع دادهٔ یک عنصر لیست می‌تواند با نوع دادهٔ عنصر دیگر لیست متفاوت باشد. لیست با علامت‌های `[]` مشخص می‌شود.

لیست شکل ۱-۳ به نام `primes` است و ۷ عنصر دارد که عناصر آن از ۰ تا ٦ شماره‌گذاری می‌شوند و هر عنصر با شمارهٔ اندیس خود مشخص می‌شود. مثلاً `primes[0]` به مقدار ۲ و `primes[1]` به مقدار ۳ اشاره می‌کند و غیره؛ یعنی هر عنصر لیست یک متغیر مرجع است که به یک شیء اشاره می‌کند. چنین لیستی را با دستور زیر می‌توان ایجاد کرد:

6. Sequence

```
primes = [2, 3, 5, 7, 11, 13, 17]
```

سازندهٔ کلاس لیست با `list()` مشخص می‌شود که یک لیست خالی ایجاد می‌کند. دستور زیر را ببینید:

```
x = list()
```

با این دستور، لیستی خالی ایجاد می‌شود که `x` به آن اشاره می‌کند. با جزئیات لیست‌ها در فصل پنجم آشنا خواهید شد.

۱-۵-۶. نوع دادهٔ تاپل

نوع دادهٔ تاپل با کلاس `tuple` در پایتون تعریف و اداره می‌شود. در واقع، نوع دادهٔ تاپل نسخهٔ تغییرناپذیر کلاس لیست است که ساختار داخلی آن خطی است. تاپل با پرانتزها مشخص می‌شود و تاپل خالی را با `()` نمایش می‌دهند. نکته‌ای که باید به آن توجه داشته باشید این است که در تاپلی با یک عنصر، باید بعد از آن عنصر یک کاما قرار گیرد وگرنه مفسر آن را با یک عبارت پرانتزدار اشتباه می‌گیرد. لذا، `(17,)` یک تاپل تفسیر می‌شود؛ درحالی‌که `(17)` یک عبارت پرانتزی در نظر گرفته می‌شود. کلاس تاپل سازنده‌ای به نام `tuple()` دارد که چندتایی خالی ایجاد می‌کند. دستورات زیر را ببینید:

```
t1 = tuple()
t2 = (7,)
t3 = (7, 8, 9, 10)
```

دستور اول تاپل خالی ایجاد می‌کند. دستور دوم تاپلی با یک عنصر ۷ و دستور سوم تاپلی با چهار عنصر ۷، ۸، ۹ و ۱۰ ایجاد می‌کند. برای دسترسی به عناصر تاپل از علامت‌های `[]` استفاده می‌شود. در تاپل `t3` برای دسترسی به عنصر ۷ از `t3[0]` و برای دسترسی به ۸ از `t3[1]` استفاده می‌کنیم و غیره. تاپل‌ها را به طور کامل در فصل پنجم بحث خواهیم کرد.

به نکات زیر دربارهٔ تاپل توجه کنید:

- آنچه تاپل را می‌سازد، کاما است نه پرانتز؛ یعنی تاپل را بدون استفاده از پرانتز هم می‌توان ایجاد کرد؛ مثل `t = 1, 2, 3, 4`.
- اعضای تاپل تغییرپذیر نیستند.
- تفاوت لیست و تاپل در این است که اعضای لیست قابل تغییر ولی اعضای تاپل غیرقابل تغییر هستند.
- اعضای تاپل مثل اعضای لیست می‌توانند هم‌نوع نباشند.
- تاپل همانند لیست، مجموعه‌ای از مرجع‌های شیء است که به مقادیر آن اشاره می‌کنند (مانند شکل ۱-۳ که دربارهٔ لیست‌ها توضیح داده شد).

۱-۵-۷. نوع دادهٔ رشته‌ای

کلاس `str` برای نمایش و اداره کردن نوع دادهٔ رشته‌ای مورد استفاده قرار می‌گیرد. رشته شامل دنباله‌ای تغییرناپذیر از کاراکترهای **یونیکد**[7] است. توجه داشته باشید که در پایتون، کلاس جداگانه‌ای برای کاراکترها وجود ندارد و کاراکتر، رشته‌ای به طول یک است. نمایش داخلی رشته در مقایسه با نمایش داخلی لیست‌ها و تاپل‌ها فشرده‌تر است.

مقادیر رشته‌ای می‌توانند در داخل کوتیشن یگانی یا کوتیشن دوتایی قرار گیرند. علاوه بر این، کلاس `str` متدی به نام `str()` دارد که اگر مقداری به آن داده نشود، یک رشتهٔ تهی (خالی) ایجاد می‌کند. دستورات زیر را ببینید:

```
s1 = "OlomRayaneh"
s2 = str("test")
```

7. Unicode

دستور اول متغیر (مرجع شیء) s1 را ایجاد می‌کند که به شیء "OlomRayaneh" اشاره دارد و دستور دوم با سازندهٔ str() متغیر s2 را ایجاد می‌کند که به شیء "test" اشاره می‌کند. رشته‌ها را در فصل ششم بررسی خواهیم کرد.

اینکه می‌توان رشته‌ها را هم با کوتیشن یگانی و هم با کوتیشن دوتایی نمایش داد، قابلیت خوبی است؛ زیرا می‌توان کاراکتر کوتیشن دیگر را در خروجی نمایش داد. مثلاً رشتهٔ "Don't worry" یک رشتهٔ معتبر است.

در ایجاد رشتهٔ "Don't worry" از کوتیشن دوتایی استفاده کردیم و در نتیجه کوتیشن یگانی به عنوان یک کاراکتر محسوب می‌شود. اگر بخواهیم همان علامت کوتیشن را به عنوان جداکنندهٔ رشته به کار ببریم، باید از کاراکتر کنترلی \ استفاده کنیم. همچنین اگر بخواهیم خود کاراکتر \ را در خروجی داشته باشیم، باید از دو علامت \\ استفاده کنیم. به عنوان مثال، خروجی حاصل از "I like \\python!" به صورت I like \python! است.

۸-۵-۱. نوع دادهٔ مجموعه

در پایتون دو نوع دادهٔ مجموعه‌ای وجود دارد که یکی از آن‌ها با کلاس set و دیگری با کلاس frozenset تعریف و اداره می‌شود. مجموعه‌ای که با set ایجاد می‌شود، تغییرپذیر و مجموعه‌ای که با frozenset ایجاد می‌شود، تغییرناپذیر است. نوع دادهٔ مجموعه برای نمایش مفهوم مجموعه در ریاضیات به کار می‌رود؛ یعنی عناصر آن تکراری نیستند و ترتیب خاصی برای آن‌ها تعریف نمی‌شود. اعضای مجموعه در داخل علامت‌های {} قرار می‌گیرند. متد set() برای ایجاد مجموعه‌های تغییرپذیر و متد frozenset() برای ایجاد مجموعه‌های تغییرناپذیر استفاده می‌شوند. دستورات زیر را ببینید:

```
a = set([3, 4, 5])
b = frozenset(['x', 2, 'y'])
c = {2, 4, 6}
```

دستور اول مجموعهٔ تغییرپذیر `a` را با اعضای `{3, 4, 5}` و دستور دوم مجموعهٔ تغییرناپذیر `b` را با اعضای `{'x', 2, 'y'}` و دستور سوم مجموعهٔ تغییرپذیر `c` را با اعضای `{2, 4, 6}` ایجاد می‌کند. مجموعه‌ها را در فصل ششم بحث خواهیم کرد.

۹-۵-۱. نوع دادهٔ دیکشنری

نوع دادهٔ **دیکشنری**[8] با کلاس `dict` تعریف و اداره می‌شود. نام دیگر دیکشنری، **نگاشت**[9] است. دیکشنری مجموعه‌ای از کلیدهای مجزا و مقادیر مربوط به آن‌ها است. به عنوان مثال، ممکن است یک دیکشنری، نگاشتی از شمارهٔ دانشجویی یکتا به رکوردهای طولانی‌تر دانشجویان، مثل نام، آدرس و نمرات درس‌های او باشد. در پایتون کلاس `dict` با همان روش کلاس `set` پیاده‌سازی می‌شود، با این تفاوت که حافظه‌ای برای مقادیر کلیدها در نظر می‌گیرد. سازندهٔ `dict()` برای ایجاد دیکشنری استفاده می‌شود. دستورات زیر را در نظر بگیرید:

```
a = dict({'c': 20, 'd': 40})
b = {'Iran': 'ir', 'Germany': 'gr'}
```

دستور اول دیکشنری `a` را با استفاده از سازندهٔ `dict()` و با دو عضو ایجاد می‌کند که کلید `'c'` دارای مقدار ۲۰ و کلید `'d'` دارای مقدار ۴۰ است. دستور دوم دیکشنری `b` را ایجاد می‌کند که کلید `'Iran'` دارای مقدار `'ir'` و کلید `'Germany'` دارای مقدار `'gr'` است.

---

8. Dictionary
9. Mapping

اکنون دستور زیر را ببینید:

```
c = {}
```

به نظر شما این دستور یک دیکشنری خالی یا یک مجموعهٔ خالی ایجاد می‌کند؟ از آنجا که در پایتون نوع دادهٔ دیکشنری قبل از نوع دادهٔ مجموعه تعریف و پیاده‌سازی شده است، دستور فوق یک دیکشنری خالی به نام c ایجاد می‌کند. بحث کامل دربارهٔ دیکشنری‌ها در فصل ششم ارائه خواهد شد.

۱-۶. سیستم نوع در پایتون

پایتون زبانی با **سیستم نوع پویا**[10] است؛ یعنی نوع متغیرها نه در زمان نوشتن کد، بلکه در زمان اجرا تعیین و بررسی می‌شود. هنگام تعریف یک متغیر نیازی به مشخص کردن نوع آن نیست و مفسر پایتون با توجه به مقدار اختصاص‌یافته به متغیر، نوع آن را برای آن لحظه تعیین می‌کند. بعداً در ادامهٔ برنامه ممکن است نوع متغیر تغییر کند. دستورات زیر را ببینید که در یک برنامه تایپ شده‌اند:

10. Dynamic type system

```
1  x = 5
2  ...
3  x = "Hello"
4  ...
5  x = 15.5
6  ...
7  x = complex(3, 2)
8  ...
9  x = [2, 3, 4]
10 ...
11 x = (7, 8, 9)
12 ...
13 x = {1, 3, 5}
14 ...
15 x = {'Iran': 'ir', 'green': 'gr'}
```

- در زمان اجرای برنامه، مفسر پایتون با دیدن دستور ۱، متغیر x را از نوع عددی «صحیح» در نظر می‌گیرد.
- خط ۲ به معنای چند دستور بعد است.
- در ادامهٔ اجرای برنامه، در خط ۳ به متغیر x مقدار رشته‌ای "Hello" نسبت داده می‌شود و در نتیجه از اینجا نوع متغیر x رشته‌ای منظور خواهد شد.
- با اجرای دستور ۵، متغیر x به نوع float تغییر می‌کند و از نوع رشته‌ای نخواهد بود.
- با اجرای دستور ۷، متغیر x در همان برنامه از این لحظه به بعد از نوع مختلط خواهد بود.
- با اجرای دستور ۹، متغیر x از اینجا به بعد از نوع لیست خواهد بود.
- با اجرای دستور ۱۱، متغیر x از اینجا به بعد از نوع تاپل خواهد بود.

- با اجرای دستور ۱۳ متغیر x از اینجا به بعد از نوع مجموعه خواهد بود.
- با اجرای دستور ۱۵ متغیر x از اینجا به بعد از نوع دیکشنری خواهد بود.

از آنجا که با انتساب یک مقدار جدید به یک متغیر، نوع آن به طور مناسب تغییر می‌کند، می‌گوییم سیستم نوع در پایتون، پویا است.

۱-۷. اعلان نوع پیشنهادی برای متغیرها

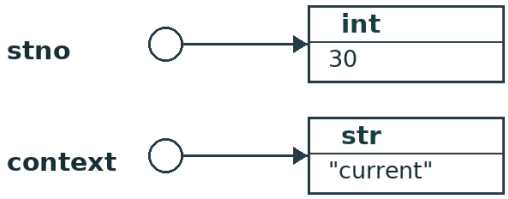
در بخش قبل با این واقعیت آشنا شدید که پایتون زبانی با سیستم نوع پویا است؛ یعنی نوع متغیرها در زمان اجرا تعیین می‌شود. این ویژگی گرچه موجب ساده‌نویسی و انعطاف بیشتر می‌شود، اما در پروژه‌های بزرگ و کدهای پیچیده می‌تواند چالش‌برانگیز باشد؛ به طوری که کدی که به درستی اجرا می‌شود، ممکن است در آینده با یک تغییر ساده دچار خطای نوع شود. آن هم نه در زمان نوشتن کد، بلکه درست در زمان اجرا. همچنین در غیاب تعیین نوع صریح، خواندن و درک کد توسط دیگران (حتی خود برنامه‌نویس با گذشت زمان) دشوارتر می‌شود.

برای رفع این چالش، از نسخهٔ 3.5 به بعد، پایتون قابلیتی به نام **تعیین نوع پیشنهادی**[1] را معرفی کرد. این قابلیت اجازه می‌دهد نوع متغیرها، پارامترها و خروجی‌های تابع را به صورت پیشنهادی و غیرالزام‌آور مشخص کنیم. در واقع، همان‌طور که از نامش پیداست، فقط جهت یادآوری و جلوگیری از اشتباهات احتمالی در پروژه‌های بزرگ، قابل استفاده است و برای پایتون هیچ الزامی ایجاد نمی‌کند. دستورات زیر را ببینید:

```
stno: int = 30
context: str = "current"
```

1. Type hinting

دستور اول اعلان می‌کند که متغیر `stno` از نوع `int` است که مقدار `30` به آن نسبت داده می‌شود. دستور دوم اعلان می‌کند که متغیر `context` از نوع رشته‌ای (`str`) است و مقدار `"current"` به آن نسبت داده می‌شود:



نمایش ارجاع متغیرها به مقادیر صحیح و رشته‌ای

اما باز هم از ادامه می‌توان انواع دیگری را به این متغیرها نسبت داد و پایتون مانع این کار نمی‌شود، مثلاً:

```
stno = "30"
name = 150
```

با اجرای این دستورات، متغیر `stno` از نوع رشته‌ای خواهد بود. تعیین نوع پیشنهادی فقط جنبهٔ یادآوری دارد و استفاده از آن در پروژه‌های بزرگ توصیه می‌شود.

۸-۱. عملگرها

عملگرها[2] نمادهایی هستند که اعمال خاصی را انجام می‌دهند. به عنوان مثال نماد `+` عملگری است که عمل جمع را انجام می‌دهد. دستور زیر را ببینید:

```
a = 5 + 10
```

2. Operators

در این دستور، مجموع اشیای 5 و 10 محاسبه می‌شود و متغیر a به مکانی اشاره می‌کند که حاوی 15 است:



متغیر a به شیئی از نوع int با مقدار 15 اشاره می‌کند

توجه داشته باشید که 5 و 10 را **عملوندهای**[3] عملگر + می‌نامند. عملگرها در پایتون عبارت‌اند از:

- عملگرهای محاسباتی
- عملگرهای رابطه‌ای یا مقایسه‌ای
- عملگرهای منطقی یا بولی
- عملگرهای ترکیبی
- عملگرهای بیتی

۱-۸-۱. عملگرهای محاسباتی

این عملگرها، اعمال محاسباتی را روی عملوندهای خود انجام می‌دهند و در جدول ۱-۴ نشان داده شده‌اند. به نکات زیر در مورد عملگرهای محاسباتی توجه کنید:

- تفاوت بین عملگرهای / و // این است که عملگر / تقسیم اعشاری و عملگر // تقسیم صحیح را انجام می‌دهد. به عنوان مثال، حاصل 4 / 15 برابر با 3.75 است،

3. Operand

درحالی‌که حاصل 15 // 4 برابر با 3 است؛ زیرا خارج‌قسمت تقسیم ۱۵ بر ۴ بدون نقطهٔ اعشار گزارش می‌شود.

- عملگر % دو عملوند خود را تقسیم صحیح می‌کند و باقی‌ماندهٔ آن تقسیم را برمی‌گرداند. به عنوان مثال، حاصل 15 % 4 برابر با 3 است؛ زیرا ۱۵ بر ۴ تقسیم می‌شود که خارج‌قسمت آن ۳ و باقی‌ماندهٔ آن ۳ است.

ترکیبی از عملگرها و عملوندهای مختلف را یک **عبارت**[4] می‌گویند؛ مثلاً x + y * 2 یا 3 + 4 / 3 * 7.

عملگر	نام	مثال	نتیجه	شرح
+	جمع	5 + 3	۸	دو عملوند را جمع می‌کند.
-	تفریق	4 - 1	۳	تفریق دو عملوند را انجام می‌دهد.
*	ضرب	3 * 4	۱۲	دو عملوند را ضرب می‌کند.
/	تقسیم	12 / 4	۳	تقسیم دو عملوند را انجام می‌دهد.
%	باقی‌ماندهٔ تقسیم	14 % 3	۲	باقی‌مانده را محاسبه می‌کند.
//	تقسیم صحیح	15 // 6	۲	خارج‌قسمتِ تقسیم صحیح را به دست می‌آورد.
**	توان	2 ** 3	۸	مقدار ۲ را به توان ۳ می‌رساند.

جدول ۱-۴. عملگرهای محاسباتی

عملگر	نام	مثال	نتیجه	شرح
>	بزرگ‌تر از	5 > 2	True	۵ بزرگ‌تر از ۲ است.
>=	بزرگ‌تر یا مساوی	7 >= 9	False	۷ بزرگ‌تر یا مساوی ۹ نیست.
<	کوچک‌تر از	7 < 8	True	۷ کوچک‌تر از ۸ است.
<=	کوچک‌تر یا مساوی	9 <= 7	False	۹ کوچک‌تر یا مساوی ۷ نیست.
!=	نامساوی	4 != 3	True	۴ و ۳ برابر نیستند.
==	تساوی	6 == 7	False	۶ و ۷ برابر نیستند.

جدول ۱-۵. عملگرهای رابطه‌ای

4. Expression

۱-۸-۲. عملگرهای رابطه‌ای

این عملگرها که نام دیگرشان عملگرهای مقایسه‌ای است، برای مقایسهٔ دو عملوند به کار می‌روند. نتیجهٔ این مقایسه، یک مقدار `True` به معنای «درستی» یا یک مقدار `False` به معنای «نادرستی» است. عملگرهای رابطه‌ای را در جدول ۱-۵ مشاهده می‌کنید.

در مورد عملگرهای رابطه‌ای به موارد زیر توجه کنید:

- علامت == برای بررسی تساوی دو عملوند به کار می‌رود. به عنوان مثال `9 == 7` به معنی این است که آیا ۷ با ۹ برابر است یا خیر.
- عملگر == را با عملگر = اشتباه نگیرید. عملگر = به نام عملگر انتساب خوانده می‌شود که شیئی را به یک متغیر یا مرجع نسبت می‌دهد. در مورد قابلیت‌های عملگر انتساب در ادامه بحث خواهیم کرد.

۱-۸-۳. عملگرهای ترکیبی

تعدادی از عملگرها وجود دارند که از ترکیب عملگرهای محاسباتی و عملگر انتساب (=) به دست می‌آیند و آن‌ها را عملگرهای ترکیبی می‌نامیم. این عملگرها را در جدول ۱-۶ مشاهده می‌کنید.

عملگر	نام	مثال	نتیجه	شرح
`+=`	انتساب جمع	`x += y`	`x = x + y`	حاصل جمع x و y را در x قرار می‌دهد.
`-=`	انتساب تفریق	`x -= y`	`x = x - y`	حاصل تفریق x از y را در x قرار می‌دهد.
`*=`	انتساب ضرب	`x *= y`	`x = x * y`	حاصل ضرب x و y را در x قرار می‌دهد.
`/=`	انتساب تقسیم	`x /= y`	`x = x / y`	حاصل تقسیم x بر y را در x قرار می‌دهد.
`%=`	انتساب باقی‌مانده	`x %= y`	`x = x % y`	باقی‌ماندهٔ تقسیم x بر y را در x می‌گذارد.
`**=`	انتساب توان	`x **= y`	`x = x ** y`	حاصل x به توان y را در x قرار می‌دهد.
`//=`	انتساب تقسیم صحیح	`x //= y`	`x = x // y`	حاصل تقسیم صحیح x بر y را در x قرار می‌دهد.

جدول ۱-۶. عملگرهای ترکیبی (x و y دو متغیر هستند)

not y	not x	x or y	x and y	y	x
True	True	False	False	False	False
True	False	True	False	False	True
False	True	True	False	True	False
False	False	True	True	True	True

جدول ۱-۷. عملگرهای منطقی

۱-۸-۴. عملگرهای منطقی

وقتی این عملگرها روی عملوندهای خود عمل می‌کنند، نتایج منطقی `True` یا `False` را تولید خواهند کرد. جدول ۱-۷ عملگرهای منطقی و اثر اجرای آن‌ها را نشان می‌دهد. در این جدول فرض می‌کنیم `x` و `y` دو عبارت بولی هستند. سه عملگر منطقی در پایتون وجود دارد که عبارت‌اند از `not` (نقیض یا «نه» شرطی)، `and` («و» شرطی) و `or` («یا» شرطی).

توجه کنید که عملگرهای `and` و `or` به صورت **مدار کوتاه**[5] عمل می‌کنند؛ یعنی در صورتی عملوند دوم را ارزیابی می‌کنند که نتوان از ارزیابی عملوند اول، نتیجهٔ عبارت را تعیین کرد. این موضوع در مواردی به کار می‌رود که می‌خواهیم عبارت بولی بسازیم و نخست بررسی کنیم شرط خاصی برقرار باشد (مثلاً مرجع شیء، `None` نباشد) و سپس شرط دیگری را بررسی کنیم. عبارت زیر را ببینید:

```
5 < 7 or 3 > 8
```

عبارت اول `5 < 7` درست است و ارزش درستی دارد. در `or` وقتی یک عملوند درست باشد، کل عبارت ارزش `True` دارد و نیازی به بررسی عبارت `3 > 8` نیست که این عمل را ارزیابی مدار کوتاه می‌نامند.

5. Short circuit

۵-۸-۱. عملگرهای بیتی

عملگرهای بیتی برای کار کردن روی بیت‌های نوع صحیح ارائه شده‌اند. این عملگرها و مثال‌هایی از کاربرد آن‌ها را در جدول ۱-۸ مشاهده می‌کنید. فرض کنید x و y دو عملوند بیتی با مقادیر `x = 0b00000111` و `y = 0b00001111` باشند. عملگرها روی تک‌تک بیت‌ها عمل می‌کنند. نتیجهٔ عملگر & روی دو بیت وقتی یک می‌شود که هر دو بیت یک باشند، وگرنه صفر می‌شود.

عملگر	نام	مثال	نتیجه (باینری)	شرح
&	و (بیتی)	`x & y`	`00000111`	بیت به بیت AND می‌شود.
|	یا (بیتی)	`x | y`	`00001111`	بیت به بیت OR می‌شود.
^	یای انحصاری	`x ^ y`	`00001000`	بیت به بیت XOR می‌شود.
<<	شیفت به چپ	`x << 1`	`00001110`	یک بیت به چپ شیفت می‌دهد.
>>	شیفت به راست	`y >> 1`	`00000111`	یک بیت به راست شیفت می‌دهد.
~	نقیض بیتی	`~x`	`11111000`	صفرها و یک‌ها را برعکس می‌کند (در نمایش هشت‌بیتی).

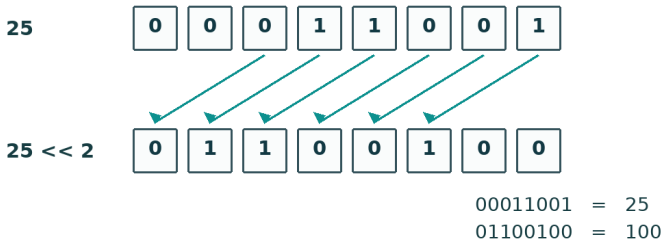
جدول ۱-۸. عملگرهای بیتی

در عملگر | نتیجه وقتی یک می‌شود که دست‌کم یکی از دو بیت یک باشد؛ وگرنه صفر می‌شود. در عملگر ^ نتیجه وقتی یک است که یکی از دو بیت صفر و دیگری یک باشد. عملگر ~ روی یک بیت عمل می‌کند و صفر را به یک و یک را به صفر تبدیل می‌کند.

به عنوان مثال، مقدار ۲۵ در نمایش بیتی به صورت زیر است:

```
25(10) = 00011001(2)
```

اگر عملگر شیفت به چپ را روی ۲۵ اجرا کنیم و به اندازهٔ دو بیت به سمت چپ شیفت دهیم، نتیجه به صورت زیر خواهد بود:



شیفت به چپ دو بیتی مقدار ۲۵ و به‌دست‌آمدن مقدار ۱۰۰

```
25         00011001
25 << 2    01100100   # 100
```

هر عمل شیفت به چپ به معنی ضرب در ۲ و دو شیفت به چپ به معنای ضرب در ۴ است. بنابراین نتیجهٔ `25 << 2` باید `100` باشد.

۶-۸-۱. عملگرهایی روی دنباله‌ها

همان‌طور که دیدید، در پایتون انواع داده‌ای وجود دارند که دنباله‌ای از اشیا هستند. عملگرهایی وجود دارند که روی این دنباله‌ها عمل می‌کنند. این عملگرها را به طور مختصر شرح می‌دهیم:

`in` بررسی می‌کند که آیا یک مقدار خاص در دنباله وجود دارد یا خیر؛ به صورت زیر به کار می‌رود (آیا مقدار `val` در دنبالهٔ `s` وجود دارد یا خیر):

```
val in s
```

`not in` بررسی می‌کند که یک مقدار خاص در یک دنباله وجود ندارد و به صورت زیر به کار می‌رود (عدم وجود `val` را در `s` بررسی می‌کند):

```
val not in s
```

۱-۸-۷. عملگرهای تساوی

عملگر `is` روی دو عملوند عمل می‌کند و وقتی ارزش `True` را برمی‌گرداند که آن دو عملوند به شیء یکسانی اشاره کنند:

```
a is b
```

اگر `a` و `b` به یک شیء اشاره کنند مقدار `True` وگرنه مقدار `False` برگردانده می‌شود. عملگر `is not` نیز روی دو عملوند عمل می‌کند و در صورتی که دو عملوند یکسان نباشند مقدار `True` وگرنه مقدار `False` را برمی‌گرداند.

ردیف تقدم	عملگرها	شرح	شرکت‌پذیری
۱	`()`	عبارت داخل پرانتز تقدم بالاتر دارد.	L → R
۲	`**`	توان	R → L
۳	`+x -x`	مثبت و منفی یگانی	R → L
۴	`* / % //`	ضرب، تقسیم، باقی‌مانده و تقسیم صحیح	L → R
۵	`+ -`	جمع و تفریق	L → R
٦	`<< >>`	شیفت به چپ و شیفت به راست	L → R
۷	`&`	و (بیتی)	L → R
۸	`^`	یای انحصاری بیتی	L → R
۹	`|`	یای بیتی	L → R
۱۰	`<= >= < >`	عملگرهای مقایسه‌ای	L → R
۱۱	`== !=`	تساوی و نامساوی	R → L
۱۲	`+= -= *= /=`	عملگرهای ترکیبی	R → L
۱۳	`is is not`	عملگرهای هویت	L → R
۱۴	`in not in`	عملگرهای عضویت	L → R
۱۵	`and or not`	عملگرهای منطقی	L → R

جدول ۱-۹. تقدم عملگرها در حالت کلی

ارائهٔ مثال‌هایی دربارهٔ این عملگرها را به فصول دیگر موکول می‌کنیم که اطلاعات کافی دربارهٔ دستورات پایتون کسب کرده باشیم.

۱-۹. تقدم عملگرها

بدیهی است که در عبارات محاسباتی، چندین عملگر با عملوندهای مختلف ترکیب شوند. در ارزیابی عبارت باید ترتیب اجرای عملگرها را بدانیم که به این عمل، تعیین تقدم عملگرها گفته می‌شود. برای پی بردن به مفهوم تقدم عملگرها، عبارت زیر را در نظر بگیرید:

```
4 + 3 * 7
```

حاصل این عبارت برابر با 25 می‌شود؛ زیرا ابتدا عملگر * روی مقادیر 3 و 7 اجرا می‌شود که نتیجه‌اش 21 است و سپس 4 با 21 جمع می‌شود که حاصل آن 25 است. می‌گوییم تقدم عملگر * نسبت به عملگر + بیشتر است و در نتیجه زودتر انجام می‌شود. به طور کلی وقتی چندین عملگر در یک عبارت محاسباتی با هم ترکیب شوند، ترتیب اجرای آن‌ها مهم است. تقدم عملگرها در پایتون را در جدول ۱-۹ مشاهده می‌کنید. همان‌طور که در این جدول مشاهده می‌کنید، هر عبارتی که در داخل پرانتز باشد زودتر از بقیه ارزیابی می‌شود.

در عباراتی که دو عملگر با تقدم یکسان پشت سر هم بیایند، کدام‌یک نسبت به دیگری تقدم دارد؟ به عنوان مثال، در عبارت زیر، اول عملگر * انجام می‌شود یا عملگر /؟

```
16 / 2 * 5
```

در این گونه موارد، قاعدهٔ شرکت‌پذیری زبان مشخص می‌کند که کدام زودتر انجام شود. قواعد شرکت‌پذیری در زبان‌ها معمولاً به دو صورت است:

- **از چپ به راست** (L → R): در این حالت از سمت چپ شروع می‌کنیم و هر عملگری که زودتر ظاهر شود، زودتر انجام می‌گیرد. بر این اساس، در عبارت مذکور، ابتدا عملگر / و سپس عملگر * انجام می‌شود.
- **از راست به چپ** (R → L): در این حالت از سمت راست به چپ حرکت می‌کنیم و هر عملگری که زودتر ظاهر شود، زودتر انجام خواهد شد. بر این اساس، در عبارت مذکور، ابتدا عملگر * و سپس عملگر / انجام خواهد شد. در پایتون غالباً از قاعدهٔ شرکت‌پذیری چپ به راست استفاده می‌شود و در نتیجه، در عبارت مذکور، ابتدا عملگر تقسیم انجام می‌گیرد و حاصل آن ۸ است که در ۵ ضرب می‌شود و نتیجهٔ نهایی ۴۰ به دست می‌آید. اما عملگر توان از شرکت‌پذیری راست به چپ پیروی می‌کند.

```
a ** b ** c
```

در این عبارت، ابتدا b به توان c می‌رسد و سپس a به توان نتیجهٔ آن‌ها می‌رسد، مثل عبارت زیر:

```
a ** (b ** c)
```

۱-۱۰. ساختار برنامه در پایتون

در بخش‌های قبلی، موضوعات ساده و در عین حال مهمی را دربارهٔ پایتون یاد گرفتیم، ولی تاکنون ساختار یک برنامه را در پایتون بررسی نکردیم. در این بخش دربارهٔ ساختار برنامه در پایتون و رهنمودهایی برای کدنویسی بهینه بحث خواهیم کرد.

هر برنامه در پایتون شامل تعدادی دستور است که هر دستور کاری را انجام می‌دهد. بهتر است هر دستور از برنامه در یک سطر جداگانه نوشته شود. در ادامهٔ این فصل با برخی از دستورات پایتون برای برنامه‌نویسی مقدماتی آشنا می‌شوید و سپس در فصل‌های بعدی به تدریج با دستورات و امکانات بیشتری آشنا خواهید شد.

۱-۱۰-۱. دستور انتساب

دستور انتساب یکی از ساده‌ترین دستورات در پایتون است که شیئی را به یک مرجع شیء نسبت می‌دهد. دستور انتساب با عملگر = مشخص می‌شود. دستور زیر یک دستور انتساب است:

```
x = 15
```

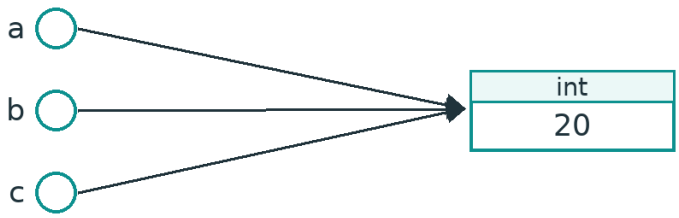
با این دستور انتساب، متغیر یا مرجع x به شیئی با مقدار ۱۵ اشاره خواهد کرد. در واقع، x به مکانی از حافظه اشاره خواهد کرد که در آن محل، شیئی با مقدار ۱۵ ذخیره شده است.

همان‌طور که مشاهده می‌کنید، همراه با مقدار شیء، نوع داده‌ای آن نیز ذخیره می‌شود تا مفسر پایتون بداند در هر لحظه، متغیر x به چه نوع شیئی اشاره دارد. این عمل را انتساب پویای نوع در زبان پایتون می‌گویند.

اکنون دستور زیر را ببینید:

```
a = b = c = 20
```

این دستور را انتساب چندگانه می‌گویند که در آن سه متغیر a و b و c ایجاد می‌شوند و به شیء ۲۰ اشاره خواهند کرد:

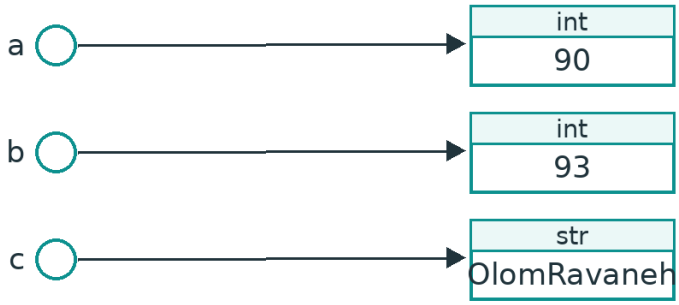


سه متغیر با ارجاع مشترک به شیء صحیح ۲۰

اکنون دستور زیر را ببینید:

```
a, b, c = 90, 93, "OlomRavaneh"
```

این دستور هم یک دستور انتساب چندگانه است، ولی باعث می‌شود که سه متغیر a و b و c هر کدام به یک شیء مختلف اشاره کنند:



ارجاع سه متغیر به سه شیء متفاوت

این دستورات مرا به یاد سال‌های ۹۰ و ۹۳ می‌اندازد که انتشارات علوم رایانه به عنوان ناشر برتر کشور انتخاب شده است.

دستورات زیر و اثر اجرای آن‌ها را ببینید:

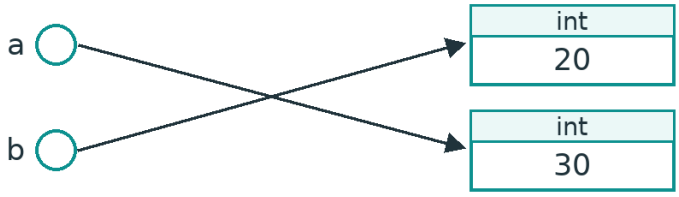
```
a = 20
b = 30
```



مقدارهای اولیهٔ دو متغیر

اکنون دستورات زیر و اثر اجرای آن‌ها را ببینید:

```
a, b = b, a
```



جابه‌جایی مراجع دو متغیر بدون متغیر کمکی

همان‌طور که مشاهده می‌کنید، این دستور موجب می‌شود که متغیر a به جایی اشاره کند که b قبلاً اشاره می‌کرد و متغیر b به جایی اشاره کند که قبلاً a به آن اشاره می‌کرد.

۲-۱۰-۱. دستور `import`

از این دستور در پایتون به وفور استفاده خواهیم کرد. تعدادی از امکانات پایتون، جزء هستهٔ پایتون محسوب می‌شوند و به محض نصب پایتون، مستقیماً می‌توانید از آن‌ها استفاده کنید؛ به عنوان مثال،

کلاس‌های `int`، `float`، `dict`، `complex`، `set` و `frozenset` از این دسته امکانات محسوب می‌شوند. اما برخی دیگر از امکانات پایتون، مستقیماً قابل استفاده نیستند. این امکانات در ماژول‌هایی قرار می‌گیرند که قبل از استفاده از آن‌ها باید آن را وارد برنامهٔ خود کنید. برای این منظور، از دستور `import` استفاده می‌شود که به روش‌های مختلفی قابل استفاده است. یک روش سادهٔ آن به صورت زیر است:

```
import moduleName
```

`moduleName` نام ماژولی است که باید وارد برنامه شود تا از آن استفاده کنید. شکل دیگر کاربرد آن به صورت زیر است:

```
from moduleName import className
```

این دستور، کلاس `className` را از ماژول `moduleName` وارد برنامه می‌کند.

به عنوان مثال، ماژولی به نام `fractions` در پایتون نوشته شده که حاوی کلاسی به نام `Fraction` است که نوع داده‌ای کسری را در پایتون تعریف و مدیریت می‌کند. دستورات زیر را ببینید:

```
from fractions import Fraction
```

این دستور کلاس `Fraction` را از ماژول `fractions` وارد برنامه می‌کند. اکنون دستور زیر را ببینید:

```
Fraction(2, 3)
```

با این دستور کسر 2/3 ایجاد می‌شود.

۳-۱۰-۱. توضیحات در پایتون

وجود توضیحات در برنامه می‌تواند به افزایش خوانایی برنامه کمک کند. توضیحات را می‌توان در داخل برنامه، برای برنامه توضیح داد تا کسی که این برنامه را می‌خواند، از جمله خود نویسندهٔ برنامه، به راحتی آن را درک کند و تغییرات احتمالی را در آن به آسانی ایجاد کند.

یک روش ایجاد توضیحات در برنامه، استفاده از علامت # است. سطر زیر را ببینید:

```
#Program for sum of numbers
```

اگر با این دستور توضیحات را در ابتدای برنامه قرار دهید، می‌توانید هدف برنامه را بیان کنید.

روش دیگر ایجاد توضیحات در برنامه، استفاده از سه علامت کوتیشن در ابتدا و سپس سه علامت کوتیشن در انتهای توضیحات است. استفاده از کوتیشن یگانی و دوتایی فرقی نمی‌کند. دستورات زیر را ببینید:

```
"""
In this book we want to
learn python programming
that is a modern object oriented
language
"""
```

این دستور شامل چهار سطر توضیحات است که ابتدا و انتهای آن با سه علامت کوتیشن مشخص شده است.

۴-۱۰-۱. فاصله‌گذاری و بلاک‌بندی

در زبان پایتون، فاصله‌گذاری و تورفتگی برای ایجاد بلاک‌ها ضروری است. منظور از بلاک، مجموعه‌ای از دستورات مرتبط به هم است. دستورات برنامه باید از ابتدای سطر شروع شوند و تمام دستورات یک برنامه باید از همان ابتدای سطر آغاز شوند تا مشخص شود که یک بلاک را تشکیل می‌دهند. شروع و پایان هر بلاک با تورفتگی‌ها مشخص می‌شوند. تورفتگی‌ها با فضای خالی (Tab یا Space) ایجاد می‌شوند. سعی کنید یا از Space یا از Tab استفاده کنید و ترکیبی از آن‌ها را به کار نبرید. اگر مفهوم بلاک‌بندی را متوجه نشدید جای نگرانی نیست؛ در فصل‌های بعدی با آن کاملاً آشنا خواهید شد. شکل زیر مفهوم بلاک‌بندی را در پایتون نشان می‌دهد (فعلاً به مفهوم دستورات توجه نکنید):

```
#define a function for factorial
def fact(x):
    if x != 0:
        return x * fact(x - 1)
    else:
        return 1

#Driver program to test
m = int(input("Enter a positive number"))
print("Number = ", m, ", factorial = ", fact(m))
```

```
#define a function for factorial
def fact(x):
    if x != 0:
        return x * fact(x - 1)
    else:
        return 1

#Driver program to test
m = int(input("Enter a positive number"))
print("Number = ", m, ", factorial = ", fact(m))
```

نمایش ساختار بلوک‌ها با تورفتگی در برنامهٔ نمونه

در این بلاک‌بندی، به عنوان مثال، اگر دستور `if` و دستور `else` از یک ستون شروع نشوند، یعنی تورفتگی یکسانی نداشته باشند، با خطا مواجه خواهید شد.

اکنون که با برخی مفاهیم آشنا شدید، ساختار سادهٔ برنامه‌ای در پایتون را به صورت زیر می‌نویسیم:

```
import module
statement 1
statement 2
...
...
```

این ساختار نشان می‌دهد که یک برنامه در صورت نیاز به ماژول‌ها، باید آن‌ها را وارد کند و سپس دستوراتی برای انجام وظایف خود داشته باشد.

۱-۱۱. دریافت، پردازش و نمایش داده‌ها

همان‌طور که گفته شد، به طور کلی هر برنامهٔ کامپیوتری سه وظیفه را بر عهده دارد:

- دریافت داده‌ها یا تولید آن‌ها
- پردازش داده‌ها (هر نوع تغییر شکل در داده‌ها)
- چاپ نتایج

۱-۱۱-۱. تابع `input()` برای دریافت داده‌ها

برای دریافت داده‌ها از خارج از کامپیوتر و انتساب آن‌ها به متغیرهایی در برنامه، از تابع `input()` به صورت زیر استفاده می‌شود:

```
var = input("message")
```

در این روش کاربرد، `var` متغیری است که دادهٔ خوانده‌شده به آن نسبت داده می‌شود. `message` پیامی است که به کاربر نمایش داده می‌شود تا بپرسد چه اطلاعاتی را باید وارد برنامه کند. دستور زیر را ببینید:

```
test = input("Enter your course:")
```

پیامی که در دستور `input` آمده است، اعلان می‌کند که کاربر باید رشتهٔ تحصیلی خودش را وارد کند که به متغیر `test` نسبت داده خواهد شد. وقتی کنترل اجرای برنامه به دستور `input` رسید، پیام آن به صورت زیر چاپ می‌شود و منتظر دریافت رشتهٔ تحصیلی کاربر می‌ماند. کاربر باید رشته‌ای را وارد کند و سپس کلید `Enter` را فشار دهد:

```
Enter your course: Computer
```

کاربر رشتهٔ `Computer` را وارد می‌کند و کلید `Enter` را فشار می‌دهد که این رشته به عنوان یک شیء، در حافظهٔ کامپیوتر ذخیره می‌شود و متغیر `test` به آن اشاره می‌کند.



متغیر `test` به شیء رشته‌ای «Computer» اشاره می‌کند

تابع `input()` تمام اطلاعات واردشده را به صورت رشته‌ای دریافت می‌کند. دستور زیر را ببینید:

```
stno = input("Enter student number: ")
ave = input("Enter your grade: ")
```

دستور اول شمارهٔ دانشجویی را که یک مقدار صحیح است از ورودی می‌خواند و به متغیر `stno` نسبت می‌دهد. در حالی که دستور دوم، نمرهٔ دانشجو را می‌خواند و به متغیر `ave` نسبت می‌دهد. وقتی کنترل اجرا به دستور اول رسید، پیامی ظاهر می‌شود، کاربر شمارهٔ دانشجویی را وارد می‌کند و کلید `Enter` را فشار می‌دهد:

```
Enter student number: 150
```

پس از آن کنترل به دستور `input` بعدی می‌رسد و پیامی صادر می‌شود که کاربر یک نمره را وارد کند که اعشاری است و کلید `Enter` را فشار می‌دهد:

```
Enter your grade: 16.5
```

پس از اجرای این دو دستور، حافظه به شکل زیر است:

تعجب نکنید؛ بله، هر اطلاعاتی که تابع `input()` از ورودی می‌خواند، آن‌ها را به صورت رشته‌ای ذخیره می‌کند. اگر می‌خواهید آن‌ها را به عددی تبدیل کنید، باید با توجه به نوع موردنظرتان از متد سازندهٔ متناظر استفاده کنید. به عنوان مثال، اگر بخواهید `stno` را به مقدار صحیح تبدیل کنید، باید متد سازندهٔ کلاس `int` یعنی `int()` را روی آن به کار ببرید و برای تبدیل `ave` به نوع ممیز شناور، باید متد سازندهٔ کلاس `float` یعنی `float()` را روی آن اعمال کنید. یعنی دو دستور مذکور باید به صورت زیر نوشته شوند:

```
stno = int(input("Enter student number: "))
ave = float(input("Enter your grade: "))
```

اکنون وقتی این دستورات اجرا شوند و همان اطلاعات قبلی وارد شوند، حافظه به شکل زیر خواهد بود:



ارجاع `stno` و `ave` به اشیای عددی پس از تبدیل نوع

پس از بررسی روش نمایش داده‌ها در بخش بعدی، مثال‌هایی را خواهید دید.

۲-۱۱-۱. نمایش داده‌ها به کاربران

یکی از وظایف برنامه‌ها این است که بتوانند اطلاعات موجود در کامپیوتر را به کاربر ارائه دهند. پایتون برای این منظور از تابع `print()` استفاده می‌کند. این تابع به شکل‌های مختلف به کار می‌رود که انواع گوناگونی از اطلاعات را به روش‌های مختلفی در خروجی (صفحه‌نمایش) ارائه می‌دهد. شکل سادهٔ کاربرد `print()` به صورت زیر است:

```
print(parameters)
```

در این روش کاربرد، parameters پارامترهایی هستند که تابع print() باید آن‌ها را در خروجی نمایش دهد. دقت داشته باشید که تابع print() بدون پارامترها نیز قابل استفاده است که در این صورت فقط یک سطر خالی در خروجی تولید می‌کند. به نکات زیر در مورد این تابع توجه کنید:

1. اگر پارامتر تابع یک مقدار رشته‌ای در داخل کوتیشن (یگانی یا دوتایی) باشد، عیناً در خروجی چاپ می‌شود.
2. با هر بار اجرای تابع print() یک سطر از خروجی تولید می‌شود و دستور چاپ بعدی، در سطر جدیدی عمل می‌کند.
3. اگر پارامتر تابع print() یک عبارت محاسباتی باشد، پس از ارزیابی، نتیجهٔ آن به خروجی می‌رود و چنانچه یک متغیر باشد محتوای آن متغیر چاپ می‌شود.
4. اگر print() چند پارامتر داشته باشد باید آن‌ها را با کاما از هم جدا کنید.

مثال ۱-۱. برنامه‌ای که دو مقدار صحیح را از ورودی خوانده، آن دو مقدار را جمع و حاصل جمع آن‌ها را نمایش می‌دهد.

اهداف آموزشی:

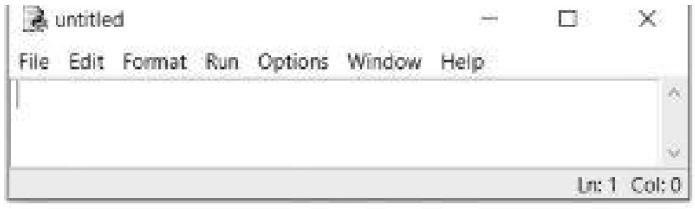
- یاد می‌گیریم داده‌ها را چگونه از ورودی دریافت و در خروجی چاپ کنیم.
- با روش نوشتن برنامه در پایتون آشنا می‌شویم.

متغیر	شرح
num1	عدد اول
num2	عدد دوم
sum	مجموع دو عدد

متغیرهای برنامه

مراحل طراحی و اجرا:

1. پایتون را راه‌اندازی کنید.
2. در منوی File گزینهٔ New File را انتخاب کنید تا محیط ویرایشگر برای تایپ و ذخیرهٔ برنامه باز شود:



پنجرهٔ ویرایشگر برنامه در پایتون

1. دستورات زیر را (بدون شماره‌ها) تایپ و تحت عنوان ch1-1.py ذخیره کنید:

```
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
sum = num1 + num2
print("First number is: ", num1)
print("Second number is: ", num2)
print("Sum is: ", sum)
```

سطر اول، عدد نخست را از ورودی می‌خواند و با تبدیل به نوع `int` به `num1` نسبت می‌دهد. سطر دوم نیز عدد دوم را به نوع `int` تبدیل و به `num2` نسبت می‌دهد. سطر سوم مجموع دو عدد را در `sum` قرار می‌دهد و سطرهای چهارم تا ششم، عددهای خوانده‌شده و مجموع آن‌ها را چاپ می‌کنند.

1. با کلید `F5` برنامه را اجرا کنید. عدد اول را وارد کنید و کلید `Enter` را فشار دهید؛ عدد دوم را نیز وارد و تأیید کنید تا نتیجه را ببینید:

```
Enter first number: 10
Enter second number: 50
First number is:  10
Second number is:  50
Sum is:  60
```

همان‌طور که مشاهده می‌کنید، خروجی در صفحه‌ای غیر از صفحهٔ تایپ برنامه ظاهر می‌شود و کافی است برای برگشتن به صفحهٔ برنامه روی آن کلیک کنید یا پنجرهٔ خروجی را ببندید.

آموخته‌های برنامه:

- هر دستور `print()` در یک سطر جداگانه عمل می‌کند.
- همراه هر خروجی بهتر است پیامی ظاهر شود تا مفهوم خروجی مشخص باشد.
- برای اجرای برنامه باید کلید `F5` را فشار دهیم یا در منوی `Run` گزینهٔ `Run Module` را انتخاب کنیم.

مثال ۱-۲. محاسبهٔ مساحت و محیط مستطیل

برنامه‌ای که طول و عرض مستطیلی را بخواند، مساحت و محیط آن را محاسبه و نمایش دهد.

مساحت مستطیل = طول × عرض

محیط مستطیل = ۲ × (طول + عرض)

اهداف آموزشی: یاد می‌گیریم چگونه اطلاعات را بخوانیم، محاسبات را روی آن‌ها انجام دهیم و نتایج را به خروجی ببریم.

متغیر	شرح
`length`	طول مستطیل
`width`	عرض مستطیل

متغیرهای برنامه

مراحل طراحی و اجرا:

1. پایتون را راه‌اندازی و در منوی `File` گزینهٔ `New File` را انتخاب کنید تا صفحه‌ای برای تایپ و ذخیرهٔ برنامه باز شود.
2. دستورات زیر را در فایل `ch1-2.py` تایپ و ذخیره کنید (بدون شماره‌ها):

```
length = int(input("Enter the length : "))
width = int(input("Enter width: "))
print("Area is: ", length * width, end = ", ")
print("Perimeter is: ", (length + width) * 2)
```

سطر اول، طول مستطیل را به length و سطر دوم، عرض را به width نسبت می‌دهد. سطر سوم همراه با پیامی مساحت را چاپ می‌کند؛ پارامتر end باعث می‌شود در انتهای خروجی، به‌جای شروع سطر جدید[1] ، متن تعیین‌شده چاپ شود. بنابراین نتیجهٔ دستور print() بعدی نیز در همان سطر قرار می‌گیرد.

1. برنامه را با کلید F5 یا گزینهٔ Run Module اجرا کنید. طول و عرض مستطیل را وارد و نتیجه را مشاهده کنید:

```
Enter the length : 10
Enter width: 8
Area is:  80, Perimeter is:  36
```

آموخته‌های برنامه: نتایج اجرای دو یا چند دستور print را می‌توان در یک سطر خروجی چاپ کرد. برای این منظور پارامتر end تعیین می‌کند در انتهای سطر چه چیزی قرار گیرد؛ در حالت معمول پایان سطر با \n مشخص می‌شود.

مثال ۱-۳. محاسبهٔ مساحت و محیط دایره

برنامه‌ای که شعاع دایره‌ای را از ورودی بخواند، مساحت و محیط آن را محاسبه و به خروجی ببرد (در آن r شعاع دایره و π یک مقدار ثابت است).

$$\text{مساحت دایره} = \pi r^2$$

اهداف آموزشی: یاد می‌گیریم از ماژول math مقدار π را به دست آوریم و با قابلیت جدیدی از تابع print() آشنا شویم.

1. New line

متغیر	شرح
r	شعاع دایره
area	مساحت دایره

متغیرهای برنامه

مراحل طراحی و اجرا:

1. پایتون را راه‌اندازی کنید و در منوی File گزینهٔ New File را انتخاب کنید تا صفحه‌ای برای تایپ و ذخیرهٔ برنامه باز شود.
2. دستورات زیر را در فایل ch1-3.py تایپ و ذخیره کنید (بدون شماره‌ها):

```
import math
r = float(input("Enter the radius :"))
area = math.pi * r ** 2
print("radius", "area", sep = 5 * " ")
print(r, area, sep = 7 * " ")
print("Perimeter = ", 2 * math.pi * r)
```

در سطر اول با دستور import ماژول math وارد برنامه می‌شود. سطر دوم شعاع را می‌خواند و به r نسبت می‌دهد. سطر سوم مساحت دایره را با استفاده از مقدار math.pi محاسبه می‌کند. در سطرهای چهارم و پنجم از پارامتر sep برای مشخص کردن فاصلهٔ بین اطلاعات استفاده شده است و سطر ششم محیط دایره را محاسبه و چاپ می‌کند.

1. برنامه را اجرا کرده، مقداری را به‌عنوان شعاع دایره وارد کنید و نتیجه را ببینید. نمونه‌ای از اجرای برنامه:

```
Enter the radius :10
radius     area
10.0       314.1592653589793
Perimeter =  62.83185307179586
```

آموخته‌های برنامه:

- یاد گرفتیم چگونه ماژولی را وارد برنامهٔ خود کنیم.
- با استفاده از پارامتر `sep` در `print()` می‌توان فاصلهٔ بین اقلام خروجی را تعیین کرد.

۳-۱۱-۱. کاراکترهای کنترلی

در زبان پایتون تعدادی کاراکتر ویژه به نام «کاراکترهای کنترلی» وجود دارند که می‌توانند چگونگی نمایش اطلاعات را در خروجی کنترل کنند. این کاراکترها را در جدول ۱-۱۰ مشاهده می‌کنید.

مثال ۱-۴. برنامه‌ای که با استفاده از کاراکترهای کنترلی، شیوهٔ نمایش خروجی‌ها را مشخص می‌کند.

هدف آموزشی: یاد می‌گیریم چگونه از کاراکترهای کنترلی استفاده کنیم.

کاراکتر کنترلی	کاری که انجام می‌دهد
`\newline`	از خط جدید صرف‌نظر می‌کند.
`\\`	کاراکتر \ را چاپ می‌کند.
`\'`	کاراکتر کوتیشن یگانی (') را چاپ می‌کند.
`\"`	کاراکتر کوتیشن دوتایی (") را چاپ می‌کند.
`\a`	بوق سیستم را به صدا درمی‌آورد.
`\b`	کاراکتر قبل از خودش را حذف می‌کند (Backspace).
`\f`	صفحهٔ جدیدی را به وجود می‌آورد.
`\n`	سطر جدیدی را ایجاد می‌کند.
`\N{name}`	کاراکتر یونیکد با نام تعیین‌شده.
`\ooo`	کاراکتر با مقدار مبنای هشت.
`\r`	بازگشت به ابتدای سطر (Carriage return).
`\t`	کنترل خروجی را به محل بعدی (Tab) می‌برد.
`\uhhhh`	کاراکتر یونیکد با مقدار مبنای ۱۶ چهاررقمی.

\Uhhhhhhhh	کاراکتر یونیکد با مقدار مبنای ۱۶ هشت‌رقمی.
\v	کنترل خروجی را به ابتدای ۸ سطر بعدی می‌برد.
\xhh	کاراکتر با مقدار مبنای ۱۶ هشت‌بیتی.

جدول ۱-۱۰. کاراکترهای کنترلی

مراحل طراحی و اجرا:

1. پایتون را راه‌اندازی کنید و از منوی File گزینهٔ New File را برای تایپ و ذخیرهٔ برنامه انتخاب کنید.
2. دستورات زیر را در فایل ch1-4.py تایپ و ذخیره کنید:

```
#using control characters
print("\"Computer\" \t \"Books\" \t \"Notebook\"\n")
print("\"Hardware\" \t \"Software\" \t \"Firmware\" ")
```

سطر نخست توضیحات است و سطرهای دوم و سوم نوشته‌هایی را با استفاده از کاراکترهای کنترلی به خروجی می‌برند. توجه کنید که \n باعث رفتن به سطر جدید می‌شود.

1. برنامه را اجرا کرده، نتیجه را ببینید. نمونه‌ای از خروجی:

```
"Computer"      "Books"      "Notebook"

"Hardware"      "Software"   "Firmware"
```

آموخته‌های برنامه:

- با کاراکترهای کنترلی می‌توان کاراکترهای ویژه مانند " یا ' را در خروجی چاپ کرد.
- با کاراکترهای کنترلی می‌توان بین اقلام خروجی فاصله ایجاد کرد و سطرهای خروجی را کنترل کرد.

۴-۱۱-۱. فرمت‌بندی خروجی

با استفاده از پارامترهای `end` و `sep` و نیز کاراکترهای کنترلی می‌توانیم خروجی‌ها را تحت کنترل خود درآوریم، اما مواردی وجود دارند که این امکانات به‌خوبی جواب‌گو نیست. برای این منظور از متد `format` مربوط به کلاس `str` استفاده می‌شود. نمونه‌ای از دستور `print()` را با استفاده از متد `format()` ببینید.

```
a = 156
b = 476.32
print("{0:5d}, {1:8.2f}".format(a, b))
```

در این مثال می‌خواهیم مقدار `a` را طوری چاپ کنیم که پنج مکان از خروجی را اشغال کند؛ یعنی پنج مکان برای آن در نظر گرفته شود، درحالی‌که سه مکان سمت راست آن برای رقم‌های ۱۵۶ و دو مکان سمت چپ خالی می‌ماند. عبارت `{0:5d}` به این معناست که عرض میدان پنج کاراکتر و نوع مقدار صحیح است؛ در آن `d` به معنی مقدار صحیح است.

مکان اول	مکان دوم	مکان سوم	مکان چهارم	مکان پنجم
خالی	خالی	۱	۵	۶

می‌خواهیم `b` را طوری چاپ کنیم که کل طول آن هشت مکان باشد: یک مکان برای نقطهٔ اعشار، دو مکان برای قسمت اعشار و بقیه برای قسمت صحیح در نظر گرفته شود. این موضوع با `{1:8.2f}` مشخص شده است. `1` شمارهٔ پارامتر متد `format()` را مشخص می‌کند؛ `8` کل طول میدان خروجی و `2` تعداد ارقام اعشار را بیان می‌کند.

شکل دیگری از کاربرد متد `format()` را در دستور زیر ببینید:

```
print("{a:5d}, {b:8.2f}".format(a = 156, b = 476.32))
```

این دستور دقیقاً معادل دستور قبلی است. همان‌طور که در این دو مثال مشاهده کردید، از حرف d برای چاپ نمایش مقادیر صحیح و از حرف f برای نمایش مقادیر اعشاری استفاده می‌شود. حرف s نیز برای چاپ مقادیر رشته‌ای به کار می‌رود.

مثال ۱-۵. محاسبه و نمایش میانگین سه عدد

برنامه‌ای که سه مقدار صحیح را از ورودی می‌خواند، میانگین آن‌ها را محاسبه می‌کند و با فرمت مناسبی به خروجی می‌برد.

متغیر	شرح
a, b, c	سه عدد ورودی
ave	میانگین اعداد

متغیرهای برنامه

هدف آموزشی: یاد می‌گیریم چگونه با متد format() از کلاس str خروجی را فرمت‌بندی کنیم.

مراحل طراحی و اجرا:

1. پایتون را راه‌اندازی کنید و گزینهٔ New File را از منوی File انتخاب کنید تا صفحه‌ای برای تایپ و ذخیرهٔ برنامه ظاهر شود.
2. دستورات زیر را در فایل ch1-5.py تایپ و ذخیره کنید:

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
c = int(input("Enter third number: "))
ave = (a + b + c) / 3
print(" a = {0:5d} \n b = {1:5d} \n c = {2:5d}".format(a, b,
c))
print("ave = {0:8.2f}".format(ave))
```

سطرهای اول تا سوم، مقادیر را خوانده و به متغیرهای a، b و c نسبت می‌دهند. سطر چهارم میانگین را محاسبه و در ave قرار می‌دهد. در دو سطر آخر نیز خروجی فرمت‌بندی می‌شود.

1. برنامه را اجرا کنید؛ نتیجهٔ نمونه‌ای از اجرا در زیر آمده است:

```
Enter first number: 10
Enter second number: 29
Enter third number: 47
 a =    10
 b =    29
 c =    47
ave =    28.67
```

آموخته‌های برنامه: متد format() ابزار مناسبی برای تعیین طول میدان خروجی برای انواع مختلفی از داده‌ها است.

فرمت‌بندی لیترال‌های رشته‌ای

برای استفاده از لیترال‌های رشته‌ای فرمت‌بندی‌شده، از پیشوند f برای لیترال استفاده می‌شود. فیلدهایی از لیترال که باید جایگزین شوند، در داخل {} قرار می‌گیرند. فیلدهای قابل جایگزینی عبارت‌هایی هستند که در زمان اجرا ارزیابی می‌شوند. دستور زیر را ببینید:

```
>>> price = 10
>>> print(f"price is {price}")
```

خروجی این دستورات به شرح زیر است:

```
price is 10
```

۱-۱۲. تبدیل مبنای اعداد

اعدادی که تاکنون با آن‌ها سروکار داشتیم، اعداد مبنای ده بودند. اگر در مسئله‌ای لازم بود با اعداد مبنای دیگری مثل دو، هشت یا شانزده کار کنید، در پایتون توابعی برای این کار فراهم آمده است.

تابع `bin()`: این تابع پارامترش را گرفته و آن را به مبنای دو تبدیل می‌کند.

تابع `oct()`: این تابع پارامتر خود را به مبنای هشت تبدیل می‌کند.

تابع `hex()`: این تابع پارامتر خود را به مبنای شانزده تبدیل می‌کند.

مثال ۱-۶. برنامه‌ای که یک عدد را در مبنای ده می‌خواند و به مبناهای دو، هشت و شانزده تبدیل می‌کند و به خروجی می‌برد.

هدف آموزشی: یاد می‌گیریم چگونه یک عدد را به مبناهای مختلف تبدیل کنیم.