

## مقدمه

هرگز به آنقدر که می‌پری قانع نباش .

هرگز نگو بیشتر از این ممکن نیست.

دائم از خودت عبور کن!

شاگردی که کمتر از معلمش بداند،

دنیا را به عقب می‌راند،

شاگردی که به قدر معلمش بداند دنیا را متوقف می‌کند.

این تویی که پیش می‌رانی...

عصر حاضر از نظر بسیاری از روشنفکران انقلاب ارتباطات و اطلاعات نام گرفته است. در واقع کلیه کارهای روزمره‌ی انسان با سیستم های ارتباطی نوین و فناوری اطلاعات گره خورده است. لذا اهمیت علم کامپیوتر به عنوان نمود اصلی فناوری اطلاعات و ارتباطات بر کسی پوشیده نیست. در دنیای مدرن امروز تسلط و آشنایی با نحوه کار با سیستم های کامپیوتری به عنوان یک سطح سواد مطلوب در جامعه شناخته می شود که اگر کسی آن را نداشته باشد باید بهای عدم آشنایی با دنیای کامپیوتر در زندگی روزمره خویش با مراجعه به کافی نت ها، درخواست های الکترونیکی و موارد دیگر پردازد.

بدون شک، برنامه نویسی یکی از مهم ترین مهارت هایی است که امروزه نه تنها برای فارغ التحصیلان و دانشجویان رشته مهندسی کامپیوتر، بلکه برای سایر رشته ها و زمینه ها نیز به شکل ضروری، مورد نیاز است. هر ساله، بر تعداد موقعیت های شغلی که مرتبط با برنامه نویسی هستند و یا پیش نیاز اصلی آن ها برنامه نویسی است، افزوده می شود. از طرفی، بخشی جدا نشدنی از فرآیند تحقیق و پژوهش در اکثر رشته های دانشگاهی امروزی، کار با

کامپیوتر و توانایی پیاده‌سازی ایده‌ها و الگوریتم‌ها در قالب برنامه‌های کامپیوتری است. قطعا در این مسیر، دانشجویان نیازمند یادگیری و کسب مهارت‌های برنامه‌نویسی هستند.

پایتون یک زبان برنامه‌نویسی همه منظوره، سطح بالا، شیء‌گرا و مفسر است که توسط خودو فان روسوم در سال ۱۹۹۱ در کشور هلند طراحی شد. فلسفه ایجاد آن تأکید بر دو هدف اصلی خوانایی بالای برنامه‌های نوشته شده و کوتاهی و بازدهی نسبی بالای آن است. کلمات کلیدی و اصلی این زبان به صورت حداقلی تهیه شده‌اند و در مقابل کتابخانه‌هایی که در اختیار کاربر است بسیار وسیع هستند.

علی رغم استقبال گسترده دانش پژوهان عرصه کامپیوتر از زبان پایتون، در زمینه آموزش این زبان قدرتمند کتاب‌های زیادی به زبان فارسی منتشر نشده است که کتاب حاضر برای پر کردن خلأ موجود در زمینه نوشته شده و توسط انتشارات فن آوری‌های نوین به زیور چاپ آراسته شده است. امید است اثر حاضر همانند سایر آثار مولفین مورد استقبال قرار گیرد.

از تمامی اساتید و دانشجویان عزیز تقاضا داریم، هر گونه اشکال، ابهام در متن کتاب، پیشنهاد و انتقادات را به آدرس پست الکترونیک **fanavarienovin@gmail.com** ارسال نمایند.

### مولفین

fanavarienovin@gmail.com

# فصل آشنایی با زبان پایتون ۱

## ۱-۱ مقدمه‌ای بر زبان برنامه نویسی پایتون

پایتون یکی از معدود زبان‌های برنامه‌نویسی است که می‌توان ادعا کرد ساختاری ساده و قدرتمند دارد، از این رو، یادگیری این زبان همواره به افراد مبتدی که شاید هیچ تجربه‌ای در برنامه‌نویسی نداشته باشند، توصیه می‌شود و از طرف دیگر، استفاده از این زبان برای حل مسائل مختلف و پیچیده انتخاب اول بسیاری از برنامه نویسان حرفه‌ای بوده است.

بر اساس رتبه‌بندی سایت Tiobe، زبان برنامه‌نویسی Python در سپتامبر سال ۲۰۱۵ با سه پله صعود نسبت به زمان مشابه در سال قبل در جایگاه پنجم قرار گرفته است که نشان‌دهنده‌ی رشد محبوبیت این زبان در میان برنامه نویسان سراسر دنیا است.

همان‌طور که می‌دانید هر زبان برنامه‌نویسی ویژگی‌ها و قابلیت‌های خاص خود را دارد که آن را از سایر زبان‌ها متمایز می‌سازد و علت شکل‌گیری زبان‌های مختلف نیز پاسخگویی به نیازهای متفاوت و متنوع کاربران با استفاده از همین قابلیت‌های متمایز است. به همین دلیل، پیش از شروع به یادگیری هر زبان ابتدا باید نیازها و هدف خود را از یادگیری آن زبان در کنار قابلیت‌هایش قرار دهیم و در صورت تطبیق آن‌ها باهم، قدم درراه یادگیری بگذاریم. بنابراین، برای آشنایی بیش‌تر با زبان پایتون، در ادامه به معرفی برخی از ویژگی‌ها و قابلیت‌های آن می‌پردازیم:

۱. **سادگی و صراحت**<sup>۱</sup>: پایتون یک زبان ساده و کمینه‌گرا است. وقتی نگاهی به سورس کد یک برنامه‌ی نوشته‌شده به زبان پایتون بی‌اندازیم، احساس می‌کنیم که با یک متن انگلیسی صریح مواجه هستیم. شاید بتوان گفت این بزرگ‌ترین نقطه‌ی قوت پایتون است که به‌جای درگیر کردن برنامه‌نویس

---

<sup>۱</sup>. Simplicity

<sup>۲</sup>. Low Learning Curve

به جزئیات زبان به او اجازه می‌دهد تا روی حل مسئله تمرکز داشته باشد. همین موضوع سرعت کد نویسی و خوانایی این زبان را هم افزایش داده است.

۲. **منحنی یادگیری کم شیب**<sup>۲</sup>: قطعاً عامل اصلی این موضوع که یادگیری پایتون به عنوان قدم اول به مشتاقان برنامه‌نویسی و حتی کودکان توصیه می‌شود سینتکس فوق‌العاده ساده‌ی آن است. همان‌طور که گفتیم صراحت زبان پایتون نه تنها خوانایی آن را افزایش داده است، بلکه با حذف پیچیدگی‌ها سهولت یادگیری آن را نیز بیش‌تر کرده است.

۳. **رایگان و متن‌باز بودن**<sup>۱</sup>: توزیع‌های مختلف زبان برنامه‌نویسی پایتون کاملاً رایگان بوده و هر برنامه‌نویس می‌تواند سورس کد آن را بخواند، آن را تغییر دهد، و در برنامه‌های خود از اسکریپت‌های آن استفاده کند.

۴. **سطح بالا بودن**<sup>۳</sup>: پایتون از جمله زبان‌های قدرتمند سطح بالا است که برنامه‌نویس را درگیر جزئیات سطح پایین مثل مدیریت حافظه یا کار با ثبات‌ها (Registers) و غیره نمی‌کند.

۵. **قابل حمل بودن**<sup>۴</sup>: ماهیت متن‌باز پایتون موجب شده است که این زبان با پلتفرم‌های مختلف سازگار باشد. بنابر اعلام رسمی سایت پایتون، در حال حاضر این زبان روی ۲۱ پلتفرم از جمله Windows، GNU/Linux، Macintosh، Solaris، Android، iOS، و ... کار می‌کند و برنامه‌های نوشته‌شده به این زبان بدون نیاز به تغییر یا با تغییرات بسیار جزئی روی تمام پلتفرم‌ها اجرا می‌شوند.

۶. **زبانی مفسری**<sup>۵</sup>: برخلاف زبان‌های کامپایلری مانند C یا جاوا، زبان برنامه‌نویسی پایتون یک زبان مفسری است و سورس کد برنامه‌های نوشته‌شده به این زبان با استفاده از یک مفسر اجرا می‌شود که همین موضوع قابل حمل بودن آن را افزایش می‌دهد.

۷. **شیء‌گرایی**<sup>۶</sup>: پایتون در مقایسه با زبان‌هایی مانند جاوا یا ++C، روش قدرتمندتر و ساده‌تری را برای اجرا برنامه‌های شیء‌گرا به کار می‌گیرد.

۸. **توسعه‌پذیری**<sup>۱</sup>: یکی از مشکلات زبان مفسری پایتون سرعت پایین اجرا در مقایسه با زبان‌های کامپایلری مانند C یا جاوا است. حال اگر بخواهید قطعه‌ای از کدها سریع‌تر اجرا شود یا اگر بخواهید

<sup>۱</sup>. Free & Open Source  
<sup>۵</sup>. Object Oriented

<sup>۲</sup>. High-level  
<sup>۶</sup>. Extensible

<sup>۳</sup>. Portable  
<sup>۷</sup>. Embeddable

<sup>۴</sup>. Interpreted

### آشنایی با زبان پایتون ۱۳

بخشی از الگوریتم برنامه‌ی خود را پنهان کنید می‌توانید آن بخش را به زبان C، C++ یا جاوا بنویسید و آن را در میان کدهای پایتون برنامه‌ی خود قرار دهید.

۹. تعبیه‌پذیری<sup>۱</sup>: علاوه بر این که می‌توان کدهای زبان‌های دیگر را در برنامه‌های نوشته‌شده به زبان پایتون قرار داد، می‌توان قطعه کدهایی را به زبان پایتون نوشت و در سورس کد برنامه‌های C، C++ یا جاوا نشان داد و به این ترتیب قابلیت‌های اسکریپتی به سورس کد مدنظر اضافه نمود.

۱۰. کتابخانه‌ی گسترده: پایتون از یک کتابخانه‌ی استاندارد غنی بهره می‌برد و در کنار این کتابخانه‌ی وسیع، کتابخانه‌های سایر توسعه‌دهندگان نیز به سرعت در حال توسعه می‌باشند که در مجموع ابزارهای مناسبی را برای ایجاد اسناد، رابط‌های گرافیکی کاربر (GUI)، مرورگرهای وب، رمزنگاری، هوش مصنوعی، پست الکترونیکی، بازی‌سازی، داده‌کاوی، ایجاد و مدیریت وب‌سایت، و بسیاری کاربردهای دیگر در اختیار برنامه نویسان قرار می‌دهد.

۱۱. همه‌منظوره بودن<sup>۱</sup>: پایتون یک زبان برنامه‌نویسی با طیف گسترده‌ای از کاربردها است که در حوزه‌های مختلف و متنوع کاربرد داشته است که از جمله مهم‌ترین کاربردهای آن در طی سالیان گذشته می‌توان به موارد زیر اشاره کرد:

🚀 موتور جستجوگر گوگل و موتور گرافیکی یوتیوب

🚀 ساخت برنامه‌های کاربردی علمی در سازمان فضایی ناسا، Fermilab

🚀 بخشی از سرویس ایمیل یاهو

🚀 تست سخت‌افزار در Cisco، Intel، IBM

🚀 ابزارهای نصب لینوکس در نسخه‌ی Redhat

🚀 سرویس ابری Dropbox

🚀 و بسیاری کاربردهای دیگر نظیر طراحی سایت‌های دینامیک، تولید نرم‌افزارهای دسکتاپ، انیمیشن‌سازی، بازی‌سازی، شبکه، امنیت، پایگاه داده، داده‌کاوی، ساخت برنامه‌های محاسباتی و کاربردی در رشته‌های مختلف نظیر ریاضی، فیزیکی، آمار، زیست و غیره.

---

<sup>1</sup> . General-Purpose

در نهایت می‌توان گفت که پایتون ابزاری مهیج و قدرتمند در اختیار برنامه‌نویسان است که کار با آن ساده و سرگرم‌کننده می‌باشد و تسلط بر آن کاربران را وارد دنیایی شگفت‌انگیز و بی‌نهایت می‌کند که هر کس می‌تواند متناسب با توانایی‌هایش از امکانات آن برای حل مسائل خود بهره‌مند شود.

### ۱-۴-۱. سطرها

مفسر پایتون و همچنین کاربران، کدهای درون هر ماژول را به صورت تعدادی سطر مشاهده می‌کنند. در پایتون دو نوع سطر وجود دارند. ۱. **سطرهای فیزیکی**<sup>۱</sup>، سطرهایی هستند که توسط ویرایشگرهای متن شماره‌گذاری می‌شوند و به سادگی توسط کاربر قابل تشخیص می‌باشند. ۲. **سطرهای منطقی**<sup>۲</sup>، برداشت مفسر از اجرای برنامه است. هر سطر بیان‌گر یک دستور پایتون است. به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
>>> name = "Fanavarienovin.net"
>>> print(name)
```

دستور اول رشته fanavarienovin.net را به متغیر name نسبت می‌دهد و دستور دوم، عبارت fanavarienovin.net را نمایش می‌دهد. در این دستورات، هر سطر منطقی یک سطر فیزیکی در نظر گرفته شده است. با اجرای این دستورات خروجی زیر نمایش داده می‌شود:

```
Fanavarienovin.net
```

گاهی اوقات هر سطر فیزیکی می‌تواند شامل چند سطر منطقی باشد. در این حالت، باید بین سطرها، کاراکتر ”؛“ قرار داد. به عنوان مثال، دستورات زیر را ببینید:

```
>>> name = "Fanavarienovin.net"; print(name)
```

با اجرای این دستورات نیز خروجی زیر نمایش داده می‌شود:

```
Fanavarienovin.net
```

گاهی اوقات برای خوانایی بیش‌تر بهتر است دستورات یک سطر منطقی در چند سطر فیزیکی تایپ شود؛ به عنوان مثال، دستورات زیر را مشاهده کنید:

```
>>> message = "Python is a \
good programing language"
>>> print(message)
```

<sup>۱</sup>. Physical Lines

<sup>۲</sup>. Logical Lines

## آشنایی با زبان پایتون ۱۵

در این مثال، خطوط اول و دوم یک دستور منطقی هستند که در دو سطر آمده‌اند. برای توسعه یک دستور در چند سطر فیزیکی از کاراکتر “\” استفاده می‌شود. با اجرای این دستورات خروجی زیر نمایش داده می‌شود:

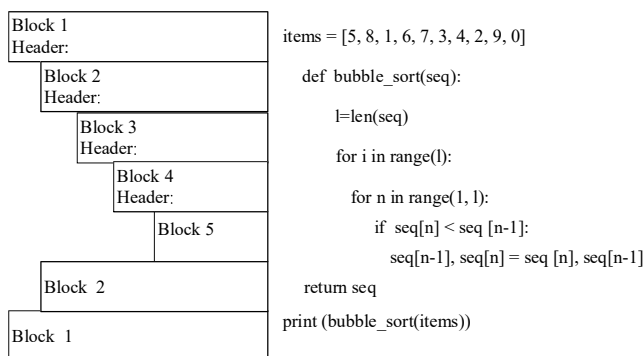
```
Python is a good programing language
```

تعداد کاراکترهای هر سطر فیزیکی نباید از ۷۹ کاراکتر بیش‌تر شود.

سطرهای خالی<sup>۱</sup>، برای افزایش خوانایی برنامه به کار می‌روند که شامل فضای خالی (Space یا Tab) هستند و توسط مفسر نادیده گرفته می‌شوند و به بایت کد ترجمه نمی‌گردند.

### ۲-۴-۱. بلاک بندی

بلاک بندی، یکی از امکاناتی است که برای افزایش خوانایی کد پایتون به کار می‌رود. در زبان پایتون برای ایجاد بلاک از تورفتگی<sup>۲</sup> سطرها استفاده می‌شود. درواقع، تورفتگی میزان فضای خالی (Space یا Tab) است که در ابتدای هر سطر فیزیکی قرار می‌گیرد. تمام دستورات موجود در یک بلاک باید به یک میزان نسبت به سرآیند خود تورفتگی داشته باشند. یعنی، تعداد فضای خالی تمام دستورات آن بلاک نسبت به سرآیند یکی باشد. شکل ۱-۱ نمونه‌ای از این بلاک بندی را نشان می‌دهد.



شکل ۱-۱ بلاک‌بندی در پایتون.

برای ایجاد هر تورفتگی از چهار جای خالی (کلید Space) استفاده کنید.

هرگز برای تورفتگی از کلیدهای Space و Tab باهم استفاده نکنید

<sup>۱</sup>. Blank Lines

<sup>۲</sup>. Indentation

#### ۴-۵-۱. عملگرهای منطقی

عملگرهای منطقی، بر روی عبارات منطقی درست یا نادرست عمل می‌کنند. نتیجه عملگرهای منطقی در جدول ۱-۶ آمده است. همان‌طور که در جدول ۱-۶ می‌بینید، زمانی نتیجه عملگر and (و منطقی) درست است که هر دو عملوند نتیجه درست داشته باشند. اما نتیجه عملگر or (یا منطقی) هنگامی نادرست است که هر دو عملوند نادرست باشند. عملگر not، نتیجه درست را نادرست و نتیجه نادرست را به درست تبدیل می‌کند.

اکنون دستورات زیر را ببینید.

```
>>> x = true
>>> y = false
>>> z1 = x and y
>>> z2 = x or y
>>> z3 = not y
```

دستور اول شیء true و x را تعریف کرده، True را به x پیوند می‌دهد، دستور دوم، اشیاء y و False را ایجاد کرده، بین y و false پیوند برقرار می‌کند، دستور سوم، نتیجه x and y (یعنی false) را در شیء ایجادشده z1 قرار می‌دهد، دستور چهارم، نتیجه x or y (یعنی true) را در z2 قرار می‌دهد و دستور پنجم، not y (یعنی true) را در z3 قرار می‌دهد.

| جدول ۴-۱ عملگرهای رابطه‌ای (مقایسه‌ای). |                  |          |       |                                                                                               |
|-----------------------------------------|------------------|----------|-------|-----------------------------------------------------------------------------------------------|
| عملگر                                   | نام              | مثال     | نتیجه | توضیحات                                                                                       |
| >                                       | بزرگ‌تر          | $2 > 3$  | False | اگر عملوند اول بزرگ‌تر از عملوند دوم باشد، نتیجه درست است، وگرنه نتیجه نادرست می‌باشد.        |
| >=                                      | بزرگ‌تر یا مساوی | $5 >= 3$ | True  | اگر عملوند اول بزرگ‌تر یا مساوی عملوند دوم باشد، نتیجه درست است، وگرنه، نتیجه نادرست می‌باشد. |
| <                                       | کوچک‌تر          | $5 < 7$  | True  | اگر عملوند اول کوچک‌تر از عملوند دوم باشد، نتیجه درست است، وگرنه نتیجه نادرست است.            |
| <=                                      | کوچک‌تر یا مساوی | $5 <= 3$ | Talse | اگر عملوند اول کوچک‌تر یا مساوی عملوند دوم باشد، نتیجه درست است، وگرنه نتیجه نادرست خواهد شد. |
| <>                                      | نامساوی          | $2 != 5$ | True  | اگر عملوند اول مخالف عملوند دوم باشد، نتیجه درست است،                                         |



## آشنایی با زبان پایتون ۱۷

|          |       |      |       |                                                                                    |
|----------|-------|------|-------|------------------------------------------------------------------------------------|
| یا<br>!= |       |      |       | وگرنه، نتیجه نادرست خواهد بود.                                                     |
| ==       | تساوی | ۲==۳ | Talse | اگر عملوند اول مساوی عملوند دوم باشد، نتیجه درست است، وگرنه نتیجه نادرست خواهد شد. |

| جدول ۵-۱ عملگرهای ترکیبی. |             |                |       |           |
|---------------------------|-------------|----------------|-------|-----------|
| عملگر                     | روش استفاده | مثال           | نتیجه | عملکرد    |
| +=                        | x += y      | x = 3; x += 5  | ۸     | x = x + y |
| -=                        | x -= y      | x = 7; x -= 3  | ۴     | x = x - y |
| *=                        | x *= y      | x = 3; x *= 5  | ۱۵    | x = x * y |
| /=                        | x /= y      | x = 17; x /= 5 | ۳٫۴   | x = x / y |
| %=                        | x %= y      | x = 17; x %= 5 | ۲     | x = x % y |
| **=                       | x **= y     | x=3; x **=2    | ۹     | x = x**y  |
| //=                       | x //= y     | x=17; x//=3    | ۵٫۰   | x = x//y  |

| جدول ۶-۱ عملکرد عملگرهای منطقی. |       |         |        |       |       |
|---------------------------------|-------|---------|--------|-------|-------|
| Y                               | x     | x and y | x or y | not x | not y |
| False                           | False | False   | False  | True  | True  |
| True                            | False | False   | True   | True  | False |
| False                           | True  | True    | True   | False | True  |
| True                            | True  | True    | True   | False | False |

## ۵-۵-۱. عملگرهای بیتی

عملگرهای بیتی، عملگرهایی که بر روی بیت‌های داده کار می‌کنند و می‌توانند آن‌ها را دست‌کاری کنند، برخی از این عملگرها عبارت‌اند از:

۱. عملگر &، "و" بیتی را انجام می‌دهد. این عملگر، دو عملوند را بیت به بیت باهم "و" بیتی می‌نماید (نتیجه و بیتی زمانی یک است که هر دو بیت ۱ باشند). به‌عنوان مثال، دستورات زیر را ببینید:

```
>>> a, b = 3, 2
>>> z = a & b
```

```
a= 00000011
b=00000010
```

```
=====
```

$z = 00000010$

پس  $z$  برابر با 2 می شود.

۲. عملگر، "یا" **بیتی** را انجام می دهد. این عملگر، دو عملوند را بیت به بیت باهم "یا بیتی" نموده (نتیجه یا بیتی زمانی صفر است که هر دو بیت ۰ باشند). به عنوان مثال، دستورات زیر را ببینید:

```
>>> a, b = 3, 2
>>> z = a | b
```

$a = 00000011$

$b = 00000010$

=====

$z = 00000011$

پس  $z$  برابر 3 خواهد شد.

۳. عملگر **^**، **xor** (یا انحصاری) بیتی را انجام می دهد. این عملگر دو عملوند را بیت به بیت (بیت های متناظر) را باهم یا انحصاری می کند (نتیجه یا انحصاری زمانی یک است که دو بیت مخالف یکدیگر باشند). به عنوان مثال، دستورات زیر را ببینید:

```
>>> a, b = 7, 2
>>> z = a ^ b
```

$a = 00000111$

$b = 00000010$

=====

$z = 00000101$

۴. عملگر **~**، **نقیض بیتی** است. این عملگر قبل از یک عملوند قرار گرفته، تمام بیت های 1 آن را به 0 و تمام بیت های 0 را به 1 تبدیل می کند. به عنوان مثال، دستورات زیر را مشاهده کنید:

```
>>> a = 10
>>> b = ~ a
```

$a = 00001010$

$b = 11110101$

۵. عملگر **<<**، **شیفت به چپ** را انجام می دهد. این عملگر بین دو عملوند قرار گرفته و مقدار عملوند سمت چپ را به تعداد عملوند سمت راست به سمت چپ شیفت می دهد. به عنوان مثال، دستورات زیر را ببینید:

```
>>> a, b = 2, 3
>>> z = a << b
```

$a = 00000010$

b=3

-----  
z = 00010000

همان‌طور که مشاهده می‌شود، z برابر با ۱۶ است. یعنی، با هر شیفت به چپ، مقدار a در ۲ ضرب می‌شود و در z قرار می‌گیرد. پس مقدار a در ۸ ضرب شده ( $2^3 * 2$ ) تا ۱۶ به دست آمده، در z قرار می‌گیرد.

۶. عملگر >> شیفت به سمت راست را انجام می‌دهد. این عملگر بین دو عملوند قرار گرفته و مقدار عملوند اول را به تعداد عملوند دوم به سمت راست شیفت می‌دهد. به‌عنوان مثال، دستورات را مشاهده کنید:

```
>>> a=12
>>> z = a >> 2
```

a= 00001100

b=00000011

همان‌طور که در این دستورت مشاهده کردید، با هر شیفت به چپ عدد تقسیم‌بر ۲ می‌شود، مقدار ۱۲ تقسیم‌بر ۴ شده و مقدار ۳ (یعنی، 00000011) به‌دست آمده است.

## ۶-۱. انواع داده‌ها(اشیای آماده)

پایتون هر نوع داده را توسط یک کلاس ارائه می‌کند. بنابراین، هر داده نمونه‌ای<sup>۱</sup> یا یک شیء<sup>۲</sup> از کلاس مشخص است. علاوه بر کلاس‌های آماده، برنامه‌نویس می‌تواند کلاس‌های جدیدی تعریف کند که در فصل‌های بعدی خواهیم دید. در پایتون انواع داده‌های مختلفی وجود دارند که عبارت‌اند از:

- |                  |                     |               |
|------------------|---------------------|---------------|
| ۱. داده‌های عددی | ۲. داده‌های رشته‌ای | ۳. لیست‌ها    |
| ۴. مجموعه‌ها     | ۵. فایل‌ها          | ۶. دیکشنری‌ها |

در این فصل به داده‌های عددی می‌پردازیم و در فصول بعدی رشته‌ها، لیست‌ها، مجموعه‌ها، فایل‌ها و دیکشنری را خواهیم آموخت.

## ۱-۶-۱. انواع داده‌های عددی

<sup>۱</sup>. Instance

<sup>۲</sup>. Object

در پایتون گروهی از انواع اشیاء وجود دارند که برای کار با اعداد به کار می‌روند. این انواع اشیاء عبارت‌اند از:

- |                             |                                |
|-----------------------------|--------------------------------|
| ۱. داده‌های صحیح (Integer)  | ۲. داده‌های ممیز شناور (Float) |
| ۳. داده‌های مختلط (Complex) | ۴. داده‌های ده‌دهی (Decimal)   |
| ۵. داده‌های کسری (Fraction) | ۶. داده‌های منطقی (Boolean)    |

### داده‌های صحیح

این نوع داده‌ها برای معرفی اعداد صحیح مثبت و منفی (بدون ممیز اعشار) نظیر 1785، 0، -900 و غیره به کار می‌روند. در پایتون نسخه ۲ دو نوع داده صحیح وجود دارد که عبارت‌اند از:

🌈 داده‌های صحیح با محدودیت اندازه که `int` نامیده می‌شوند.

🌈 داده‌های صحیح بدون محدودیت اندازه که `long` نامیده می‌شوند. در پایتون نسخه ۲ برای تعیین

داده‌های صحیح با نوع `long`، انتهای داده کاراکتر `L` یا `l` قرار می‌گیرد.

چنانچه در نسخه ۲ پایتون داده‌ای را با نوع `int` در نظر بگیرید، سرریز<sup>۱</sup> اتفاق افتد (یعنی، داده‌ای را در آن متغیر قرار دهید که در متغیر جا نشود)، خطایی رخ نخواهد داد و پایتون به صورت خودکار نوع `int` را به شیء با نوع `long` تبدیل خواهد کرد.

دقت کنید که بیش‌ترین مقدار و کم‌ترین مقدار یک شیء نوع `int` را می‌توانید با `sys.maxint - 1` و `sys.maxint` ببینید. برای این منظور می‌توانید دستورات زیر را اجرا کنید:

```
>> import sys
>>> print sys.maxint, sys.maxint-1
```

اما در نسخه ۳ پایتون اعداد صحیح با یک نوع `int` ارائه می‌گردند که از لحاظ اندازه محدودیتی ندارند. لذا، استفاده از کاراکترهای `L` و `l` در پایان این اعداد مجاز نمی‌باشد. چون در این نسخه محدودیت نوع `int` حذف شده است، لذا، `sys.maxint` حذف شده است. اما، می‌توان به جای آن از دستور `sys.maxsize` استفاده کرد.

<sup>۱</sup>. Overflow    <sup>۲</sup>. Binary    <sup>۳</sup>. Octal    <sup>۴</sup>. Hexadecimal

## آشنایی با زبان پایتون ۲۱

اعداد صحیح را می‌توان در مبنای دو<sup>۲</sup>، مبنای هشت<sup>۳</sup> و مبنای شانزده<sup>۴</sup> بیان کرد. اعداد مبنای ۲ را باید با 0b یا 0B شروع نمود. به عنوان مثال، عدد زیر در مبنای ۲ است:

```
>>> a = 0b1101
```

اما، اعداد مبنای ۸ را می‌توان با 0O یا 0o شروع کرد. به عنوان مثال، عدد زیر در مبنای ۸ است:

```
>>> a = 0o743
```

ولی، اعداد مبنای ۱۶ را باید با 0x یا 0X آغاز نمود. به عنوان مثال، عدد زیر در مبنای ۱۶ است:

```
>>> a = 0xb7D
```

در پایتون توابعی برای تبدیل یک عدد از مبنای ۱۰ به مبنای ۲، ۸ و ۱۶ وجود دارند که عبارت‌اند

از:

🔗 **تابع bin()**، یک عدد مبنای ۱۰ را به عدد مبنای ۲ تبدیل می‌کند. به عنوان مثال، دستورات زیر

خروجی 0b101 را نمایش می‌دهند:

```
>>> a = 5
>>> print(bin(a))
```

🔗 **تابع oct()**، برای تبدیل عدد مبنای ۱۰ به مبنای ۸ به کار می‌رود. به عنوان مثال، دستورات زیر

خروجی 0o22 را نمایش می‌دهند:

```
>>> a = 18
>>> print(oct(a))
```

🔗 **تابع hex()**، برای تبدیل عدد مبنای ۱۰ به مبنای ۱۶ به کار می‌رود. به عنوان مثال، دستورات زیر

خروجی 0x14 را نمایش می‌دهند:

```
>>> a = 20
>>> print(hex(a))
```

🔗 **تابع int()**، برای تبدیل یک عدد از یک مبنا به مبنای ۱۰ به کار می‌رود. به عنوان مثال، دستورات

زیر 20 را نمایش می‌دهند:

```
>>> a = 0x14
>>> print(int(a))
```

## اعداد اعشاری

اعداد می‌توانند اعشاری باشند. پایتون برای نگه‌داری اعداد اعشاری (نظیر 3.1415، 0.5 و...) از اشیایی با نوع float استفاده می‌کند. علاوه بر نمایش اعداد اعشاری به صورت ممیز شناور می‌توان اعداد اعشاری را با نماد علمی<sup>۱</sup> نمایش داد که در پایتون برای نمایش اعداد اعشاری با نماد علمی از حرف E یا e استفاده می‌شود. به عنوان مثال، در پایتون اعداد  $5 \times 10^7$  و  $6 \times 10^{-10}$  به صورت‌های 5E7 (یا 5e7) یا 6E-10 (یا 6e-10) نمایش داده می‌شوند.

### اعداد مختلط

همان‌طور که در ریاضی دیدیم، هر عدد مختلط<sup>۲</sup> از دو بخش حقیقی<sup>۳</sup> و موهومی<sup>۴</sup> تشکیل شده است. اعداد مختلط در پایتون با نوع شیء complex تعریف می‌شوند. عدد مختلط در پایتون به صورت  $x + yj$  نمایش داده می‌شود که x نشان‌دهنده، بخش حقیقی و y نشان‌دهنده بخش موهومی است. به عنوان مثال، عدد  $3 + 5j$  یک عدد مختلط است که بخش حقیقی آن ۳ و بخش موهومی آن 5 می‌باشد.

از کلاس complex می‌توان برای تعریف اعداد مختلط استفاده نمود که این کلاس به صورت زیر به کار می‌رود:

```
complex(real, imag)
```

که real بخش حقیقی و imag بخش موهومی عدد مختلط را مشخص می‌کند. چنانچه هریک از این بخش‌ها به عنوان آرگومان ارسال نشوند، به صورت پیش فرض صفر در نظر گرفته می‌شوند، به عنوان مثال، دستورات زیر را ببینید:

```
>>> a = 5; b = -3
>>> complex(a, b)
```

دستور اول، مقادیر ۳ و ۴- را به ترتیب به اشیاء a و b تخصیص می‌دهد و دستور دوم، یک شیء complex با مقدار حقیقی ۵ و مقدار موهومی ۳- ایجاد می‌نماید (خروجی (5-3j) نمایش داده می‌شود). با دو صفت real و imag می‌توان بخش‌های حقیقی و موهومی یک عدد مختلط را به دست آورد. به عنوان مثال، دستورات زیر را ببینید:

```
>>> a = 4.5
```

<sup>۱</sup>. Scientific Notation

<sup>۲</sup>. Complex Number

<sup>۳</sup>. Real

<sup>۴</sup>. Imaginary

```
>>> b = complex(a)
>>> b.real
>>> b.imag
```

دستور اول، متغیر  $a$  با مقدار ۴,۵ را ایجاد کرده، دستور دوم، شیء  $z$  با مقدار  $4.5+0.0j$  را ایجاد می‌نماید و دستور سوم، بخش حقیقی شیء مختلط  $b$  (یعنی ۴,۵) را نمایش می‌دهد و دستور چهارم، بخش موهومی شیء  $b$  (یعنی ۰.۰) را نمایش می‌دهد.

**اعداد دسیمال (دہدہی)**

همان‌طور که بیان گردید، در پایتون اعداد اعشاری به‌صورت شیء با نوع float معرفی می‌گردند. مفسر پایتون برای ارائه نوع ممیز شناور از کدگذاری Binary-Floating Point استفاده می‌نماید که برای محاسبات حسابداری مناسب نمی‌باشد. چون، پایتون اعدادی از قبیل 0.1، 0.2 و 0.3 را به‌صورت 0.10000000000000001، 0.20000000000000001 و 0.30000000000000001 نشان می‌دهد که دقیقاً برابر 0.1، 0.2 و 0.3 نمی‌باشند. این موضوع در برخی از اوقات موجب خطای منطقی خواهد شد. به‌عنوان مثال، دستورات زیر را ببینید:

```
>>> a = 0.2 + 0.2 + 0.2
>>> a == 0.6
```

با اجرای این دستورات False نشان داده می شود. یعنی،  $0.2 + 0.2 + 0.2$  برابر  $0.6$  نخواهد شد و این موضوع یک خطای منطقی برنامه است. اکنون اگر  $a$  را نشان دهیم، یعنی، دستور زیر را تایپ کنیم، خروجی  $0.6000000000000001$  نشان داده می شود:

>>> a

و دستورات زیر مقدار 0.30000000000000004 را نشان می دهند:

```
>>> a = 0.1 + 0.1 + 0.1
>>> a
```

دستور اول، سه بار 0.1 را جمع کرده و در a قرار می‌دهد که انتظار می‌رود، نتیجه 0.3 شود، اما، با نمایش a می‌بینیم که a برابر با 0.30000000000000004 می‌باشد که این انحراف ناشی از نحوه کدگذاری اعداد اعشاری می‌باشد. به همین دلیل، در پایتون یک نوع شیء جدید به نام Decimal

طراحی شده است. این نوع در ماژول decimal قرار دارد. برای استفاده از نوع Decimal ابتدا باید با دستور زیر این ماژول را به برنامه اضافه کنید:

```
>>> import decimal
```

دستورات زیر True را نشان می دهند:

```
>>> import decimal
>>> a = decimal.Decimal("0.6")
>>> b = decimal.Decimal("0.2")
>>> a == b + b + b
```

دستور اول، ماژول decimal را اضافه می کند، دستور دوم، مقدار ده دهی 0.6 را در شیء a قرار می دهد، دستور سوم، مقدار ده دهی 0.2 را در شیء b قرار می دهد، دستور چهارم، a (0.6) را با حاصل جمع  $b + b + b$  (0.2+0.2+0.2) مقایسه کرده و نتیجه True را برمی گرداند.

## ۲-۶-۱. رشته

در پایتون رشته<sup>۱</sup>، مجموعه ای از کاراکترهای پشت سر هم است که در بین جفت کتیشن ("") یا تک کتیشن (') قرار می گیرند. به عنوان مثال، دستورات زیر را ببینید:

```
>>> a = "Python language"
>>> a
>>> print(a)
```

دستور اول، شیء ای به نام a با نوع رشته ای تعریف می کند و شیء رشته ای 'Python Language' را به آن تخصیص می دهد، دستور دوم، مقدار a (یعنی، 'Python language') را نمایش می دهد و دستور سوم نیز مقدار a (یعنی، Python language) را نمایش می دهد.

در پایتون برخلاف برخی از زبان های برنامه نویسی دیگر نوع کاراکتری<sup>۲</sup> وجود ندارد. یعنی، در زبان پایتون کاراکتر، رشته ای با طول یک است.

در پایتون می توان از کاراکترهای کتیشن در داخل یکدیگر استفاده کرد. در این حالت فقط باید نوع کتیشن داخلی با بیرونی متفاوت باشد. اما، اگر بخواهید از کاراکتر کتیشن یکسان استفاده کنید، باید از کاراکتر<sup>۳</sup> قبل از کتیشن استفاده کنید. به عنوان مثال، دستورات زیر را مشاهده کنید:

```
>>> "Python 'language'"
```

<sup>۱</sup>. String

<sup>۲</sup>. Char

<sup>۳</sup>. Escape



## آشنایی با زبان پایتون ۲۵

```
>>> 'I'm a student'
```

دستور اول، کاراکتر تک کتیشن را در داخل جفت کتیشن استفاده می کند (خروجی را به صورت "Python 'language'" نمایش می دهد) و دستور دوم، کاراکتر تک کتیشن را در داخل کاراکتر تک کتیشن دیگر استفاده می کند. برای این منظور، از کاراکتر \ قبل از کاراکتر تک کتیشن داخلی استفاده می نماید (عبارت "I'm a student" را نمایش خواهد داد).

### عملگرهای رشته

➦ **عملگر +**، این عملگر برای اتصال (الحاق) دو رشته به کار می رود. به طوری که رشته سمت راست را به انتهای رشته سمت چپ اضافه می کند. به عنوان مثال، دستورات زیر را ببینید:

```
>>> s1 = "Fanavarienovin"
>>> s2 = ".net"
>>> s1 + s2
```

دستور اول، رشته s1 را ایجاد کرده، شیء Fanavarienovin را به آن تخصیص می دهد، دستور دوم شیء s2 را ایجاد نموده، رشته .net را در آن قرار می دهد و دستور سوم، رشته s2 را به انتهای رشته s1 می چسباند. یعنی 'Fanavarienovin.net' را نمایش می دهد.

➦ **عملگر \***، برای تکرار یک رشته به کار می رود. این عملگر دو عملوند یکی از نوع رشته ای و دیگری از نوع عدد صحیح را دریافت کرده رشته را به تعداد عدد دریافت شده تکرار می کند و برمی گرداند. به عنوان مثال، دستور زیر عبارت 'Fanavarienovin Fanavarienovin Fanavarienovin' را نمایش می دهند:

```
>>> "Fanavarienovin " * 3
```

## ۸-۱. تابع print()

همان طور که قبلاً بیان گردید، زمانی که یک عبارت را در مفسر تایپ کرده باشید و کلید Enter را بزنید، عبارت فوراً ارزیابی شده، نتیجه ارزیابی عبارت نمایش داده می شود. به عنوان مثال، دستور زیر را تایپ کرده تا نتیجه ارزیابی عبارت (یعنی، 57.125) را ببینید:

```
>>> 8*(5+3)-110/6
```

این ویژگی برای زمانی به کار می‌رود که بخواهید نتیجه یک دستور محاسباتی را حساب کرده یا بخواهید املائی عبارت را ارزیابی کنید.

حال، اگر این دستورات را در یک ماژول تایپ کنید، با اجرای این دستورات خروجی آن‌ها نمایش داده نمی‌شود. برای نمایش اطلاعات در ماژول می‌توانید از تابع `print()` استفاده کنید. در تابع `print()`، می‌توانید هر دنباله‌ای از عباراتی را بی‌آورید. این عبارات با کاما (,) از هم جدا می‌شوند. در هنگام استفاده از تابع `print()` به نکات زیر دقت کنید:

۱. اگر تابع `print()` را بدون آرگومان استفاده کنید، یک سطر خالی چاپ خواهد شد.
۲. با هر بار اجرای تابع `print`، یک سطر چاپ خواهد شد.
۳. اگر آرگومان تابع `print()` رشته‌ای باشد، عین رشته را در خروجی نمایش می‌دهد.
۴. اگر در آرگومان تابع `print()` یک عبارت آورده شود، نتیجه عبارت در خروجی نمایش داده می‌شود.
۵. اگر در آرگومان تابع `print()` نام یک متغیر آورده شود، مقدار متغیر در خروجی نمایش داده می‌شود.

به عنوان مثال، دستورات زیر را ببینید:

```
>>> x, y = 3, 5
>>> print(x, " + ", y, " = ", x + y)
```

دستور اول، مقدار ۳ را در `x` و مقدار ۵ را در `y` قرار می‌دهد، دستور دوم، ابتدا مقدار `x` (یعنی ۳)، سپس علامت `" + "`، در ادامه مقدار `y` (یعنی ۵)، در پایان علامت `" = "` و نتیجه عبارت `x + y` (یعنی ۸) را نمایش می‌دهد؛ یعنی، خروجی زیر:

```
3 + 5 = 8
```

در واقع هر چیزی که در آرگومان تابع `print()` استفاده می‌شود، برای نمایش به نوع رشته تبدیل می‌گردد، به عنوان مثال، اگر متغیر `n` عددی صحیح باشد که به مقدار ۱۰ ارجاع می‌دهد، اما وقتی به عنوان آرگومان `print()` استفاده می‌گردد، در نهایت مقدار ۱۰ به یک رشته تبدیل می‌شود. با این وجود، باید دقت کنید که متغیر `n` همچنان به یک عدد صحیح ارجاع می‌دهد. به عنوان مثال، دستورات زیر را ببینید:

## آشنایی با زبان پایتون ۲۷

```
>>> n = 10
>>> print("n is" + n)
```

با اجرای این دستور انتظار داریم که عبارت زیر نمایش داده شود:

```
n is 10
```

در صورتی که با اجرای این دستور خطای زیر صادر می‌گردد:

Traceback (most recent call last):

```
File "<pyshell#11>", line 1, in <module>
    print("n is" + n(
```

TypeError: Can't convert 'int' object to str implicitly

چون n از نوع عددی است. پس باید به نوع رشته تبدیل شود یا دستور به صورت زیر به کار رود:

```
>>> print("n is", n)
```

اکنون خروجی زیر نمایش داده می‌شود:

```
n is 10
```

برای تبدیل n به نوع رشته‌ای می‌توانید از تابع str() استفاده کنید (مانند دستور زیر):

```
>>> print("n is " + str(n))
```

با اجرای این دستور، خروجی زیر نمایش داده می‌شود:

```
n is 10
```

## ۹-۱. تایپ، ذخیره و اجرای برنامه در پایتون

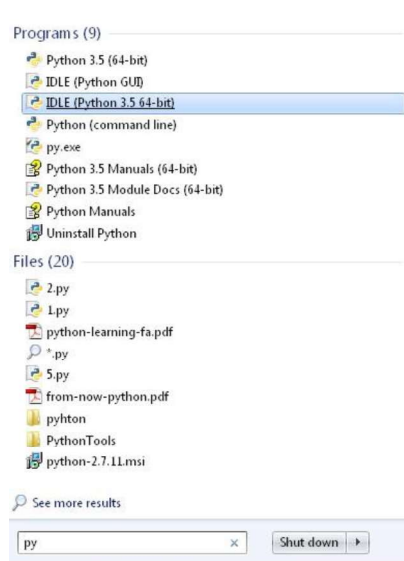
تاکنون، دستورات را به صورت تکی نوشته و اجرا کردیم و نتایج را مشاهده نمودیم. در پایتون امکانی وجود دارد تا بتوانید دستورات را به صورت یکجا تایپ کرده و اجرا نمایید. برای این منظور، به ویراستاری نیاز دارید تا برنامه را در آن تایپ کنید. سپس آن را اجرا کنید. در نسخه‌های مختلف پایتون، ویراستاری آماده شده است که می‌توانید در آن برنامه‌تان را تایپ و اجرا کنید.

به عنوان مثال، در پایتون نسخه ۳ به بعد فرآیند اجرا و ویرایش مانند مثال ۱-۱ است.

مثال ۱-۱. برنامه‌ای که مراحل تایپ، ذخیره و اجرای یک برنامه ساده را نشان می‌دهد.

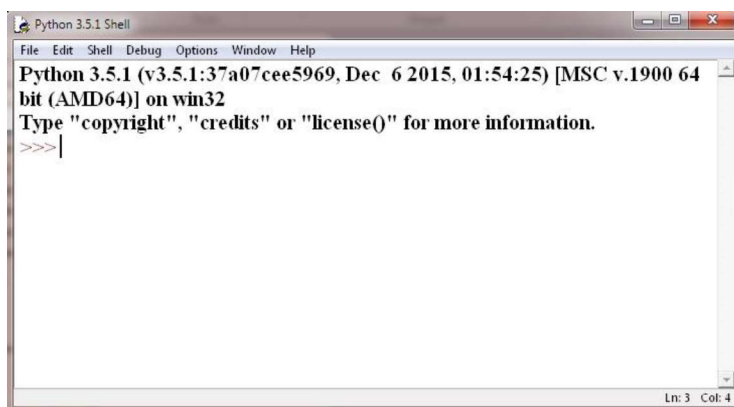
۱. در منوی Start، در باکس Search، عبارت py را تایپ کرده تا لیست برنامه‌هایی که با py شروع

می‌شوند، ظاهر شود (شکل ۲-۱).



شکل ۲-۱ لیست برنامه‌هایی که با py شروع می‌شوند.

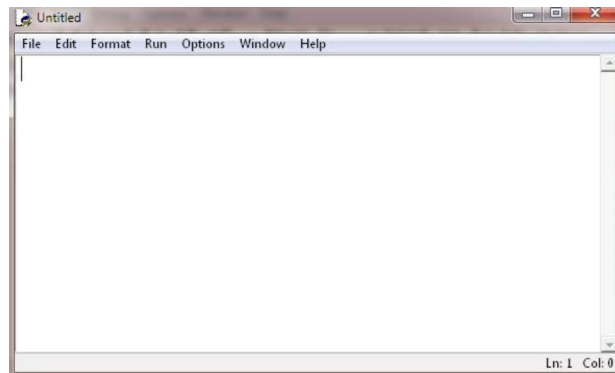
۲. برنامه (python 3.5 64-bit) IDLE را اجرا کنید تا شکل ۳-۱ ظاهر شود.



شکل ۳-۱ python 3.5.1 shell

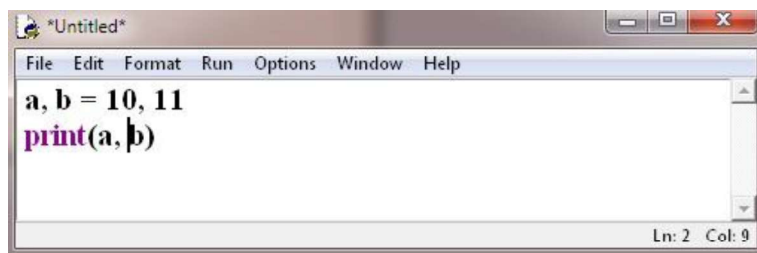
۳. گزینه File / New File (یا کلیدهای ترکیبی Ctrl+N) را فشار دهید تا فایل جدیدی ایجاد شود (شکل ۴-۱).

## آشنایی با زبان پایتون ۲۹



شکل ۴-۱ ایجاد فایل جدید python.

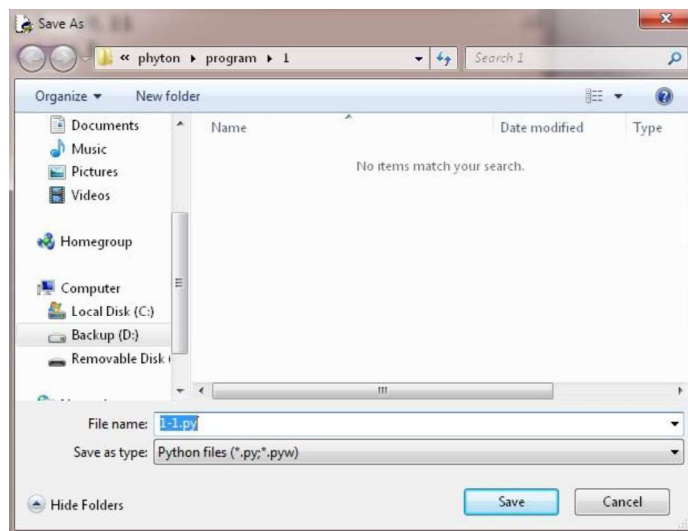
۴. اکنون دستورات برنامه‌تان را تایپ کنید (مانند شکل ۵-۱).



شکل ۵ - ۱ دستورات نمونه برای اجرا.

۵. گزینه File / Save As (یا کلیدهای ترکیبی Ctrl+ Shift+ S) را اجرا کنید تا پنجره Save As

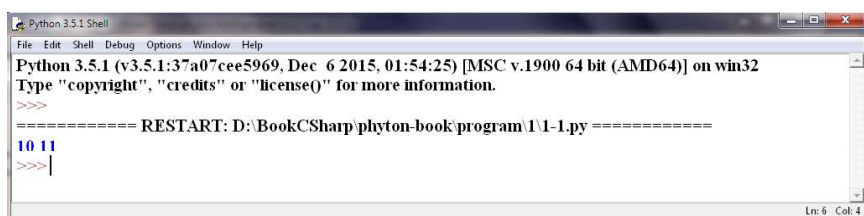
ظاهر شود (شکل ۶-۱).



شکل ۶-۱ پنجره Save As.

۶. نام فایل را 1-1.py انتخاب کرده، سپس Save را کلیک کنید (همان‌طوری که در این شکل می‌بینید، پسوند فایل py انتخاب شده است که فایل source برنامه‌های زبان پایتون می‌باشد).

۷. برای اجرا گزینه Run / Run Module (یا کلید F5) را انتخاب کنید تا خروجی زیر را ببینید (شکل ۷-۱).



شکل ۷-۱ نمونه خروجی برنامه.

## ۱۰-۱. خواندن داده

اکثر برنامه‌های واقعی باید اطلاعاتی را از کاربر بخوانند. برای این منظور، پایتون از تابع `input()` استفاده می‌کند. تابع `input()` به صورت زیر به کار می‌رود:

```
پیغام" = input()
```

### آشنایی با زبان پایتون ۳۱

وقتی کنترل اجرای برنامه به تابع `input()` برسد، ابتدا پیغام نمایش داده می‌شود، منتظر می‌ماند تا کاربر رشته‌ای را وارد کرده، کلید `Enter` را فشار دهد. به محض این که کاربر کلید `Enter` را فشار دهد، رشته وارد شده در متغیر قرار می‌گیرد. به عنوان مثال، دستور زیر را ببینید:

```
name = input("Enter your name:")
```

با اجرای این دستور عبارت زیر نمایش داده می‌شود:

```
Enter your name:
```

اکنون کاربر عبارت `fanavarienovin` را وارد کرده، کلید `Enter` را فشار دهد، عبارت `fanavarienovin` در متغیر `name` قرار می‌گیرد.

اکنون دستورات زیر را ببینید:

```
a = input("Enter a:")
b = input("Enter b:")
sum = a + b
print("Sum is ", sum)
```

با اجرای این دستورات خروجی زیر نمایش داده می‌شود:

```
Enter a:10
Enter b:20
Sum is 1020
```

همان‌طور که در این خروجی می‌بینید، خروجی حاصل جمع دو عدد وارد شده (یعنی جمع ۱۰ و ۲۰) نمی‌باشد. چون دستورات `input()` و `a` و `b` را به صورت رشته می‌خوانند. پس `a="10"` و `b="20"` می‌باشد. بنابراین `sum = a + b` (برابر ۱۰۲۰) اتصال دو رشته `a` و `b` می‌باشد.

پس اگر بخواهید عددی را بخوانید، ابتدا، با دستور `input()` می‌توانید آن را به صورت رشته‌ای بخوانید و با تابع `int()` آن را به عدد تبدیل کنید. اکنون، دستورات زیر را ببینید:

```
a = input("Enter a:")
b = input("Enter b:")
a = int(a)
b = int(b)
sum = a + b
print("Sum is ", sum)
```

با اجرای این دستورات و ورود اعداد ۱۰ و ۲۰ جلوی `a:` و `b:` خروجی زیر ظاهر می‌گردد:

```
Enter a:10
Enter b:20
Sum is 30
```

چون، دستور اول، رشته عددی ۱۰ را می‌خواند، در a قرار می‌دهد، دستور دوم، رشته عددی ۲۰ را خوانده، در b قرار می‌دهد، دستور سوم، مقدار صحیح a (int(a)) را در a قرار می‌دهد، دستور چهارم، مقدار صحیح b را در b قرار می‌دهد، دستور پنجم، جمع a و b (یعنی 30) را در sum قرار می‌دهد و دستور ششم، عبارت Sum is 30 را نمایش می‌دهد.

تابع input() را می‌توان در داخل تابع int() استفاده کرد. در این صورت رشته خوانده‌شده را به عدد صحیح تبدیل کرده، در متغیر قرار می‌دهد. به عنوان مثال، دستور زیر را مشاهده کنید:

```
n = int(input("Enter n:"))
```

با اجرای این دستور عبارت زیر ظاهر می‌شود:

```
Enter n:
```

اکنون اگر کاربر جلوی n، مقدار رشته‌ای ۱۵ را وارد کند، مقدار رشته‌ای "15" به عدد ۱۵ تبدیل و در n قرار می‌گیرد. حال اگر کاربر به جای یک عدد اشتبانه‌ا رشته‌ای را وارد کند که در آن کاراکترهای غیر عددی نظیر 'a' تا 'z' یا 'A' تا 'Z' و غیره وجود داشته باشند، مفسر پایتون پیغام خطای زیر را نمایش می‌دهد:

```
Enter n:12A12
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    n = int(input("Enter n('':
ValueError: invalid literal for int() with base 10: '12A12'
```

این پیغام خطا به این دلیل است که آرگومان تابع int() باید شامل رشته‌ای باشد که فقط از کاراکترهای عددی تشکیل می‌شود. یعنی، آرگومان تابع int()، رشته نمی‌تواند شامل کاراکترهای غیر عددی باشد.

مثال ۱-۲. برنامه‌ای که دو عدد را خوانده، حاصل جمع آن‌ها را نمایش می‌دهد (هدف این برنامه آشنایی با دستورات ورودی، خروجی و عملگر + است).

مراحل طراحی و اجرا:

۱. با گزینه File / New file (کلیدهای Ctrl + N) ماژول جدیدی ایجاد کنید.



۲. دستورات آن را به صورت زیر تایپ کنید:

```
a = input("Please enter number1: ")
a=int(a)
b = input("Please enter number2: ")
b = int(b)
print (a, '+ ', b, '= ', a + b)
```

| متغیر | هدف     |
|-------|---------|
| a     | عدد اول |
| b     | عدد دوم |

دستور اول، پیغام Please enter number1: را نمایش می‌دهد، سپس رشته‌ای را خوانده، در a قرار می‌دهد، دستور دوم، مقدار موجود در رشته a را به عدد تبدیل می‌کند و در a قرار می‌دهد، دستور سوم، پیغام Please enter number2: را نمایش داده، یک رشته عددی را خوانده، در b قرار می‌دهد، دستور چهارم، رشته b را به عدد تبدیل می‌کند و در b قرار می‌دهد و دستور پنجم، با تابع print، ابتدا مقدار a، سپس علامت +، در ادامه مقدار b و در پایان علامت = به همراه a + b را نمایش می‌دهد.

۳. با گزینه File / Save As (Ctrl + Shift + S) ماژول را به نام 1-2.Py ذخیره کنید.

۴. با گزینه Run / Run Module (کلید F5) ماژول را اجرا کنید و داده‌های ورودی را به صورت زیر وارد نمایید:

```
Please enter number1: 12
Please enter number2: 15
12 + 15 = 27
```

مثال ۱-۳. برنامه‌ای که شعاع دایره را خوانده، با استفاده از فرمول زیر مساحت دایره را نمایش می‌دهد:

$$Area = \pi r^2$$

مراحل طراحی و اجرا:

۱. با گزینه File / New file (Ctrl + N) ماژول جدیدی ایجاد کنید.

۲. دستورات زیر را در آن تایپ کنید:

```
response=input("What is your radius? ")
r = float(response)
area = 3.14159 * r**2
print("The area is ", area)
```

| متغیر | هدف         |
|-------|-------------|
| r     | شعاع دایره  |
| area  | مساحت دایره |

دستور اول پیغامی را نمایش داده، یک رشته را به عنوان شعاع دریافت می کند و در شیء response قرار می دهد، دستور دوم، مقدار response را به عدد اعشاری تبدیل می کند و در r قرار می دهد، دستور سوم، مساحت دایره را حساب کرده، به area تخصیص می دهد، دستور چهارم ابتدا، عبارت The area is و سپس در ادامه آن مقدار متغیر area (مساحت دایره) را نمایش می دهد.

۳. ماژول را به نام 1-3.Py ذخیره کنید.

۴. ماژول را اجرا کنید. جلوی What is your radius? مقدار 12.5 را وارد کرده تا خروجی زیر را مشاهده نمایید:

```
What is your radius? 12.5
The area is 490.87343749999997
```

مثال ۴-۱. هر سال برابر با  $3.156 \times 10^7$  است. برنامه ای که سن تان را به سال دریافت کرده، به ثانیه، دقیقه و ساعت تبدیل کند. هر دقیقه 60 ثانیه و هر ساعت 60 دقیقه است (هر ساعت ۳۶۰۰ ثانیه است).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
age=input("Enter your age: ")
Age=int(age)
second = Age * 3165e4
minute = second / 60
hour = second / 3600
print ('Second is ', second)
print ('Minute is ', minute)
print ('Hour is ', hour)
```

| متغیر  | هدف                          |
|--------|------------------------------|
| age    | سن ورودی (به صورت رشته)      |
| Age    | سن به سال و تبدیل شده به عدد |
| second | سن به ثانیه                  |
| minute | سن به دقیقه                  |
| hour   | سن به ساعت                   |

دستور اول، پیغام Enter your age: را نمایش داده، یک رشته را به عنوان سن تان دریافت می کند و در متغیر age قرار می دهد. دستور دوم، رشته عددی age را به عدد صحیح تبدیل می کند و در متغیر Age قرار می دهد (دقت داشته باشید که متغیر age و Age دو متغیر متفاوت هستند، چون پایتون نسبت به حروف بزرگ و کوچک حساس است)، دستور سوم، مقدار متغیر Age را در  $3165e4$  (یعنی  $3.156 \times 10^7$ ) ضرب کرده تا سن تان را به ثانیه تبدیل نماید و سپس آن را در متغیر second قرار می دهد، دستور چهارم، مقدار متغیر second را تقسیم بر ۶۰ می کند تا تعداد دقیق سن تان را به دست

### آشنایی با زبان پایتون ۳۵

آورده، سپس آن را در متغیر minute قرار می‌دهد، دستور پنجم، مقدار second را تقسیم بر ۳۶۰۰ می‌کند تا تعداد ساعات سن‌تان را حساب کرده، سپس آن را در متغیر hour قرار می‌دهد، دستورات ششم تا هشتم با پیغام‌های مناسب تعداد ثانیه‌ها، دقائق و ساعات سن‌تان را نمایش می‌دهند.

➔۲. ماژول را ذخیره و اجرا کنید. جلوی Enter your age: عدد ۴۷ را وارد کرده تا خروجی زیر را

مشاهده کنید:

```
Enter your age: 47
Second is 1487550000.0
Minute is 24792500.0
Hour is 413208.3333333333
```

مثال ۵-۱. برنامه‌ای که یک عدد دورقمی را خوانده، مقلوب آن را نمایش می‌دهد (هدف برنامه آشنایی با عملگرهای % و // است).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
a = input("Enter a number: ")
a = int(a)
r1 = a % 10
r2 = a // 10
print("Reverse is ", r1 * 10 + r2)
```

| متغیر          | هدف        |
|----------------|------------|
| a              | عدد دورقمی |
| r <sub>1</sub> | رقم یکان   |
| r <sub>2</sub> | رقم دهگان  |

دستور اول، با نمایش پیغام Enter a number: یک رشته عددی دورقمی را دریافت می‌کند و در a قرار می‌دهد، دستور دوم، مقدار a را به عدد صحیح تبدیل نموده، به a نسبت می‌دهد (همان‌طور که مشاهده کردید، در زمان اجرا می‌تواند نوع شیء تغییر کند، یعنی a از نوع رشته‌ای به نوع عددی صحیح تبدیل گردید)، دستور سوم، رقم یکان a را با عملگر % جدا کرده، در r<sub>1</sub> قرار می‌دهد، دستور چهارم، با عملگر // رقم دهگان (عملگر // برای انجام تقسیم صحیح به کار می‌رود) a را جدا نموده، در r<sub>2</sub> قرار می‌دهد و دستور پنجم، ابتدا عبارت Reverse is و سپس r<sub>1</sub>\*10+r<sub>2</sub> یعنی، همان مقلوب (a) را نمایش می‌دهد.

۲. ماژول را به نام 1\_5.Py ذخیره کرده و اجرا نمایید. جلوی Enter a number: عدد 47 را وارد

کرده تا خروجی زیر را مشاهده کنید:

```
Enter a number: 47
```

Reverse is 74

مثال ۶-۱. برنامه ای که دو عدد صحیح را خوانده، خارج قسمت و باقی مانده عدد اول بر عدد دوم را نمایش می دهد (هدف برنامه آشنایی با عملگرهای % (باقی مانده تقسیم صحیح) و // (تقسیم صحیح) است).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
a = int(input("Enter a: "))
b = int(input("Enter b: "))
print(a % b, " ", a // b)
```

| متغیر | هدف     |
|-------|---------|
| a     | عدد اول |
| b     | عدد دوم |

۲. ماژول را ذخیره کرده، اعداد ۱۴ و ۳ را وارد نمایید تا خروجی زیر را ببینید:

Enter a: 14

Enter b: 3

4 2

مثال ۷-۱. برنامه ای که x و y را خوانده و حاصل عبارت زیر را نمایش می دهد (هدف برنامه آشنایی با عملگر توان است):

$$z = x^3 + 2x^2y + 3y - 7$$

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
x = int(input("Enter x: "))
y = int(input("Enter y: "))
z = x ** 3 + 2 * x ** 2 * y + 3 * y - 7
print("Z = ", z)
```

۲. ماژول را ذخیره و اجرا کنید و اکنون دو عدد ۷ و ۶ وارد کرده تا خروجی زیر را ببینید:

Enter x: 7

Enter y: 6

Z = 942

مثال ۱۵-۱. برنامه ای که دو رشته را خوانده، این دو رشته را به هم الحاق کرده، نتیجه را نمایش می دهد (هدف این برنامه، آشنایی با عملگر + برای اتصال دو رشته است).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

| متغیر | هدف      |
|-------|----------|
| str1  | رشته اول |

### آشنایی با زبان پایتون ۳۷

```
str1 = input("Enter string1: ")
str2 = input("Enter string2: ")
str3 = str1 + str2
print (str3)
```

|                           |      |
|---------------------------|------|
| رشته دوم                  | str2 |
| حاصل الحاق رشته اول و دوم | str3 |

۲. ماژول را ذخیره و اجرا کنید و نمونه خروجی را به صورت زیر مشاهده نمایید:

```
Enter string1: Fanavarienovin
Enter string2: Publisher
Fanavarienovin Publisher
```

مثال ۱۶-۱. برنامه ای که یک رشته و تعداد تکرار آن را خوانده، رشته را به تعداد عدد وارد شده تکرار می نماید و نمایش می دهد (هدف برنامه استفاده از عملگر \* برای تکرار رشته است).

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
s = input("Enter a string: ")
rep = int(input("Enter repeat: "))
print (s * rep)
```

| متغیر | هدف              |
|-------|------------------|
| s     | رشته ورودی       |
| rep   | تعداد تکرار رشته |

۲. ماژول را ذخیره و اجرا کرده، سپس جلوی string، مقدار Python و جلوی repeat، مقدار ۵ را

وارد کنید تا خروجی زیر را مشاهده کنید:

```
Enter a string: Python
Enter repeat: 5
PythonPythonPythonPythonPython
```

مثال ۱۷-۱. برنامه ای که دو عدد را خوانده، معادل مختلط آن دو عدد را نمایش می دهد (عدد اول بخش real و عدد دوم بخش imag می باشد). هدف این برنامه ایجاد اعداد مختلط و نمایش آن ها است.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
a = int(input("Enter real part: "))
b = int(input("Enter image part: "))
complex1 = complex(a, b)
print (complex1)
```

| متغیر    | هدف                 |
|----------|---------------------|
| a        | بخش real            |
| b        | بخش imag            |
| complex1 | عدد مختلط تولید شده |

۲. ماژول را ذخیره و اجرا کرده، جلوی real part و image part مقادیر ۱۲ و -۴ را وارد کنید تا خروجی زیر را ببینید:

```
Enter real part: 12
Enter image part: -4
(j4 -12)
```

### ۱۱-۱. مسائل حل شده

مثال ۱. برنامه‌ای که سه ضلع مثلث را خوانده، با استفاده از فرمول‌های زیر محیط و مساحت مثلث را حساب می‌کند.

$$p = \frac{(a + b + c)}{2}$$

$$s = \sqrt{p(p - a)(p - b)(p - c)}$$

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
a = float(input("Enter a: "))
b = float(input("Enter b: "))
c = float(input("Enter c: "))
p = (a + b + c) / 2
print("Perime is ", p)
s = p * (p - a) * (p - b) * (p - c) ** 0.5
print("Area is ", s)
```

| متغیر | هدف                  |
|-------|----------------------|
| a     | ضلع اول مثلث         |
| b     | ضلع دوم مثلث         |
| c     | ضلع سوم مثلث         |
| p     | محیط و نصف محیط مثلث |
| s     | مساحت مثلث           |

۲. ماژول را ذخیره و اجرا کرده، جلوی a: ، b: و c: به ترتیب ۱۲، ۸ و ۹ را وارد کنید تا خروجی زیر را ببینید:

```
Enter a: 12
Enter b: 8
Enter c: 9
Perime is 29.0
Area is 35.99913193397863
```

مثال ۲. برنامه‌ای که دو عدد a و b را خوانده، حاصل عبارت زیر را محاسبه کرده و نمایش می‌دهد:

$$f(a, b) = \sqrt{a^b}$$

## مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
a = int(input("Enter a: "))
b = int(input("Enter b: "))
f = (a ** b) ** 0.5
print("f is ", f)
```

| متغیر | هدف           |
|-------|---------------|
| a     | عدد ورودی اول |
| b     | عدد ورودی دوم |
| f     | حاصل عبارت    |

۲. ماژول را ذخیره و اجرا کرده، جلوی a: و b: به ترتیب 7 و 2 را وارد کنید تا خروجی زیر را ببینید:

```
Enter a: 7
Enter b: 2
f is 7.0
```

مثال ۳. برنامه‌ای که یک عدد دورقمی را خوانده، حاصل رقم اول به توان رقم دوم و رقم دوم به توان رقم اول را نمایش می‌دهد.

## مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
a = int(input("Enter a: "))
r1 = a % 10
r2 = a // 10
print(r1, " ** ", r2, " = ", r1 ** r2)
print(r2, " ** ", r1, " = ", r2 ** r1)
```

| متغیر          | هدف        |
|----------------|------------|
| a              | عدد دورقمی |
| r <sub>1</sub> | رقم یکان   |
| r <sub>2</sub> | رقم دهگان  |

۲. ماژول را ذخیره و اجرا کرده، عدد ۳۴ را وارد کنید تا خروجی زیر را ببینید:

```
Enter a: 34
4 ** 3 = 64
3 ** 4 = 81
```

مثال ۴. برنامه‌ای که دو عدد را خوانده، بدون استفاده از متغیر کمکی محتوی آن‌ها را تعویض می‌کند.

## مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
a = int(input("Enter a: "))
b = int(input("Enter b: "))
a, b = b, a
print("a = ", a)
print("b = ", b)
```

| متغیر | هدف           |
|-------|---------------|
| a     | عدد ورودی اول |
| b     | عدد ورودی دوم |

#### ۴۰ فصل اول

۲. ماژول را ذخیره و اجرا کرده، جلوی a: و b: به ترتیب اعداد ۲۰ و ۱۲ را وارد کنید تا خروجی زیر را ببینید:

```
Enter a: 12
Enter b: 20
a = 20
b = 12
```

مثال ۵. برنامه‌ای که ارتفاع و قاعده مثلث را از ورودی خوانده، مساحت آن را محاسبه می‌کند و نمایش می‌دهد.

$$\text{مساحت مثلث} = \frac{\text{قاعده} \times \text{ارتفاع}}{2}$$

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده و در آن دستورات زیر را تایپ کنید:

```
a = int(input("Enter a: "))
h = int(input("Enter h: "))
s = a * h / 2
print("Area is ", s)
```

| متغیر | هدف        |
|-------|------------|
| a     | قاعده      |
| h     | ارتفاع     |
| s     | مساحت مثلث |

۲. ماژول را ذخیره و اجرا کنید. اکنون به ترتیب اعداد ۷ و ۹ را جلوی a: و h: وارد کرده تا خروجی زیر را ببینید:

```
Enter a: 7
Enter h: 9
Area is 31.5
```

مثال ۶. یک دوچرخه‌سوار با سرعت x کیلومتر بر ساعت شروع به حرکت می‌کند و پس از n دقیقه سرعت آن به k کیلومتر بر ساعت می‌رسد. برنامه‌ای که با استفاده از فرمول زیر شتاب دوچرخه‌سوار را محاسبه کرده، نمایش می‌دهد:

$$\text{شتاب} = \frac{20 \times (\text{سرعت اولیه}) - (\text{سرعت نهایی})}{n(\text{زمان به دقیقه})}$$

مراحل طراحی و اجرا:

۱. ماژول جدید ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:



متغیر هدف



#### آشنایی با زبان پایتون ۴۱

```
x = int(input("Enter x: "))
k = int(input("Enter k: "))
n = int(input("Enter n: "))
a = (k-x) * 60 / n
print("a is ", a)
```

|   |                 |
|---|-----------------|
| x | سرعت اولیه      |
| k | سرعت نهایی      |
| n | مدت به دقیقه    |
| a | شتاب محاسبه شده |

۲. ماژول را ذخیره و اجرا کرده، جلوی x:، k:، n: به ترتیب ۱۰، ۱۵، ۳ را وارد کنید تا خروجی زیر

را ببینید:

```
Enter x: 10
Enter k: 20
Enter n: 3
a is 200.0
```

مثال ۷. برنامه‌ای که توان ۲، توان ۳ و توان ۴ اعداد ۱ تا ۹ را چاپ کند.

مراحل طراحی و اجرا:

۱. ماژول جدید ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
print('---', '\t', "---", '\t', "---")
print(1 ** 2, '\t', 1 ** 3, '\t', 1 ** 4)
print(2 ** 2, '\t', 2 ** 3, '\t', 2 ** 4)
print(3 ** 2, '\t', 3 ** 3, '\t', 3 ** 4)
print(4 ** 2, '\t', 4 ** 3, '\t', 4 ** 4)
print(5 ** 2, '\t', 5 ** 3, '\t', 5 ** 4)
print(6 ** 2, '\t', 6 ** 3, '\t', 6 ** 4)
print(7 ** 2, '\t', 7 ** 3, '\t', 7 ** 4)
print(8 ** 2, '\t', 8 ** 3, '\t', 8 ** 4)
print(9 ** 2, '\t', 9 ** 3, '\t', 9 ** 4)
```

۲. ماژول را ذخیره و اجرا کرده تا خروجی زیر را ببینید:

```
--- --- ---
1 1 1
4 8 16
9 27 81
16 64 256
25 125 625
36 216 1296
49 343 2401
64 512 4096
81 729 6561
```

همان‌طور که در کد این برنامه مشاهده می‌شود، هریک از اعداد ۱ تا ۹ را در یک سطر تایپ کردیم که با بیان حلقه تکرار در فصل بعدی نیازی به تکرار ۹ سطر نمی‌باشد.

مثال ۸. برنامه‌ای که مختصات دونقطه را خوانده، فاصله بین دونقطه را محاسبه و نمایش می‌دهد. اگر دونقطه  $(x_1, y_1)$  و  $(x_2, y_2)$  باشند، فاصله بین دونقطه به‌صورت زیر محاسبه می‌شود:

مراحل طراحی و اجرا:

۱. ماژول جدیدی را ایجاد کرده، دستورات آن را به‌صورت زیر تایپ کنید:

```
x1 = int(input("Enter x1: "))
y1 = int(input("Enter y1: "))
x2 = int(input("Enter x2: "))
y2 = int(input("Enter y2: "))
d = ((x2 ** 2 - x1 ** 2) + (y2 ** 2 - y1 ** 2))
** 0.5
print("distance is ", d)
```

| متغیر | هدف               |
|-------|-------------------|
| $x_1$ | مختصات x نقطه اول |
| $y_1$ | مختصات y نقطه اول |
| $x_1$ | مختصات x نقطه دوم |
| $y_2$ | مختصات y نقطه دوم |
| d     | فاصله دونقطه      |

۲. ماژول را ذخیره و اجرا کرده، اطلاعات زیر را وارد کنید تا فاصله بین دونقطه را مشاهده نمایید:

```
Enter x1: 12
Enter y1: 14
Enter x2: 19
Enter y2: 24
distance is 24.43358344574123
```

مثال ۹. برنامه‌ای که سه مقدار را خوانده، نوع آن‌ها را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات آن را به‌صورت زیر تایپ کنید:

```
a = int(input("Enter a: "))
b = float(input("Enter b: "))
c = input("Enter c: ")
print("Type a is ", type(a))
print("Type b is ", type(b))
print("Type c is ", type(c))
```

| متغیر | هدف                           |
|-------|-------------------------------|
| a     | متغیر ورودی از نوع عدد صحیح   |
| b     | متغیر ورودی از نوع عدد اعشاری |
| c     | متغیر ورودی از نوع رشته‌ای    |

۲. پروژه را ذخیره و اجرا کرده، و به ترتیب مقادیر ۱۰، ۵، ۱۲ و string را جلوی a، b و c وارد

کنید تا خروجی زیر را ببینید:

### آشنایی با زبان پایتون ۴۳

```
Enter a: 10
Enter b: 12.5
Enter c: string
<'Type a is <class 'int
<'Type b is <class 'float
<'Type c is <class 'str
```

مثال ۱۰. برنامه‌ای که دو مقدار را خوانده، شماره شناسایی این اشیاء را نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی را ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

```
a = int(input("Enter a: "))
b = input("Enter b: ")
print("ID a is ", id(a))
print("ID b is ", id(b))
```

| متغیر | هدف             |
|-------|-----------------|
| a     | مقدار ورودی اول |
| b     | مقدار ورودی دوم |

۲. ماژول را ذخیره و اجرا کرده، مقادیر ۱۰ و Fanavarienovin را وارد کنید تا خروجی زیر را

ببینید:

```
Enter a: 10
Enter b: Fanavarienovin
ID a is 1498607696
ID b is 58207472
```

پروژه برنامه‌نویسی ۱: برنامه‌ای که یک عدد ۵ رقمی را خوانده، ارقام عدد را با فاصله نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی را ایجاد کرده، دستورات آن را به صورت زیر تایپ کنید:

| متغیر | هدف                                                   |
|-------|-------------------------------------------------------|
| num   | عدد پنج رقمی خوانده شده                               |
| temp  | متغیر کمکی که پس از حذف هر رقم num را نگه‌داری می‌کند |
| a۱    | رقم یکان                                              |
| a۲    | رقم دهگان                                             |
| a۳    | رقم صدگان                                             |
| a۴    | رقم هزارگان                                           |

```

num = int(input("Enter a number:"))
a1 = num % 10
temp = num // 10
a2 = temp % 10
temp = temp // 10
a3 = temp % 10
temp = temp // 10
a4 = temp % 10
temp = temp // 10
a5 = temp % 10
temp = temp // 10
print(a5, " ", a4, " ", a3, " ", a2, " ", a1)

```

|    |                |
|----|----------------|
| a5 | رقم ده هزارگان |
|----|----------------|

۲. ماژول را ذخیره و اجرا کرده، عدد ۶۷۱۸۱ را وارد کنید تا خروجی زیر را ببینید:

Enter a number:67181

1 8 1 7 6

پروژه برنامه‌نویسی ۲. برنامه‌ای که دو عدد را خوانده، اعمال زیر را انجام می‌دهد:

۱. حاصل جمع، تفریق، حاصل ضرب، تقسیم، باقی‌مانده تقسیم صحیح و توان آن‌ها را نمایش می‌دهد.
۲. حاصل عملگرهای and (و منطقی)، or (یا منطقی)، & (و بیتی)، | (یا بیتی) و □ (یا انحصاری بیتی) آن‌ها را انجام می‌دهد.
۳. دو عدد را به باینری نمایش داده، حاصل جمع، تفریق، ضرب، تقسیم، باقی‌مانده تقسیم و توان آن‌ها را به باینری نمایش می‌دهد.
۴. دو عدد را به مبنای ۸ تبدیل کرده، حاصل جمع، تفریق، ضرب، تقسیم، باقی‌مانده تقسیم و توان آن‌ها را به مبنای ۸ تبدیل می‌کند.
۵. دو عدد را به مبنای ۱۶ تبدیل کرده، حاصل جمع، تفریق، ضرب، تقسیم، باقی‌مانده تقسیم و توان آن‌ها را به مبنای ۱۶ تبدیل می‌نماید و نمایش می‌دهد.

مراحل طراحی و اجرا:

۱. ماژول جدیدی ایجاد کرده، دستورات زیر را در آن تایپ کنید:

```

num1 = int(input("Enter a number1:"))
num2 = int(input("Enter a number2:"))
print(num1, " + ", num2, " = ", num1 + num2)
print(num1, " - ", num2, " = ", num1 - num2)
print(num1, " * ", num2, " = ", num1 * num2)
print(num1, " / ", num2, " = ", num1 / num2)
print(num1, " // ", num2, " = ", num1 // num2)
print(num1, " % ", num2, " = ", num1 % num2)
print(num1, " ** ", num2, " = ", num1 ** num2)

```

| متغیر | هدف                |
|-------|--------------------|
| num1  | عدد اول خوانده شده |
| num2  | عدد دوم خوانده شده |

```
print(num1, " and ", num2, "=", num1 and num2)
print(num1, " or ", num2, "=", num1 or num2)
print(num1, " & ", num2, "=", num1 & num2)
print(num1, " | ", num2, "=", num1 | num2)
print(num1, " ^ ", num2, "=", num1 ^ num2)
print(bin(num1), " + ", bin(num2), "=", bin(num1 + num2))
print(bin(num1), " - ", bin(num2), "=", bin(num1 - num2))
print(bin(num1), " * ", bin(num2), "=", bin(num1 * num2))
print(bin(num1), " // ", bin(num2), "=", bin(num1 // num2))
print(bin(num1), " % ", bin(num2), "=", bin(num1 % num2))
print(bin(num1), " ** ", bin(num2), "=", bin(num1 ** num2))
print(oct(num1), " + ", oct(num2), "=", oct(num1 + num2))
print(oct(num1), " - ", oct(num2), "=", oct(num1 - num2))
print(oct(num1), " * ", oct(num2), "=", oct(num1 * num2))
print(oct(num1), " // ", oct(num2), "=", oct(num1 // num2))
print(oct(num1), " % ", oct(num2), "=", oct(num1 % num2))
print(oct(num1), " ** ", oct(num2), "=", oct(num1 ** num2))
print(hex(num1), " + ", hex(num2), "=", hex(num1 + num2))
print(hex(num1), " - ", hex(num2), "=", hex(num1 - num2))
print(hex(num1), " * ", hex(num2), "=", hex(num1 * num2))
print(hex(num1), " // ", hex(num2), "=", hex(num1 // num2))
print(hex(num1), " % ", hex(num2), "=", hex(num1 % num2))
print(hex(num1), " ** ", hex(num2), "=", hex(num1 ** num2))
```

۲. ماژول را ذخیره و اجرا کنید. دو عدد را وارد کرده تا خروجی زیر را ببینید:

```
Enter a number1:12
Enter a number2:5
12 + 5 = 17
12 - 5 = 7
12 * 5 = 60
12 / 5 = 2.4
12 // 5 = 2
12 % 5 = 2
12 ** 5 = 248832
12 and 5 = 5
12 or 5 = 12
12 & 5 = 4
12 | 5 = 13
12 ^ 5 = 9
0b1100 + 0b101 = 0b10001
0b1100 - 0b101 = 0b111
0b1100 * 0b101 = 0b111100
```

```
0b1100 // 0b101 = 0b10
0b1100 % 0b101 = 0b10
0b1100 ** 0b101 = 0b111100110000000000
0o14 + 0o5 = 0o21
0o14 - 0o5 = 0o7
0o14 * 0o5 = 0o74
0o14 // 0o5 = 0o2
0o14 % 0o5 = 0o2
0o14 ** 0o5 = 0o746000
0xc + 0x5 = 0x11
0xc - 0x5 = 0x7
0xc * 0x5 = 0x3c
0xc // 0x5 = 0x2
0xc % 0x5 = 0x2
0xc ** 0x5 = 0x3cc00
```