برنامه نویسی کامپیوتر ــ زمستان ۹۷
تدریس توسط: حسین جوهری
دانشگاه خواجه نصیرالدین طوسی

خلاصه هفته چهارم

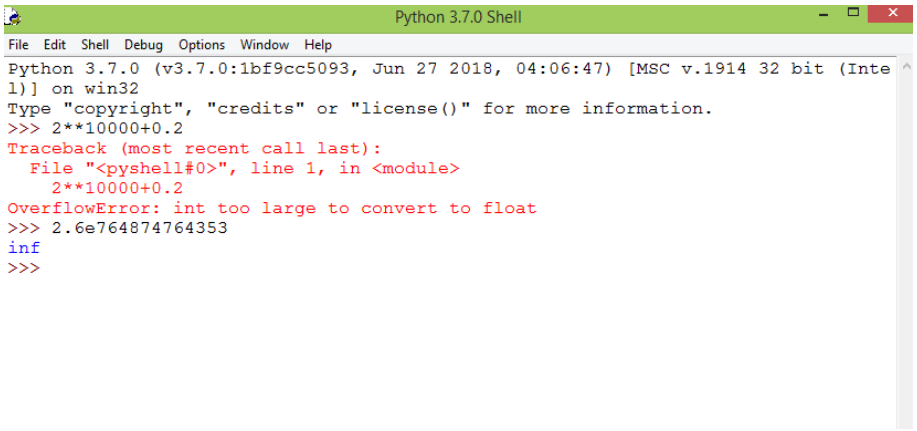
مفاهیم:
ادامه انواع داده اصلی در پایتون، ورودی و خروجی در پایتون

در این هفته بحث انواع داده اصلی در پایتون را ادامه می‌دهیم. سپس سراغ برنامه نویسی به زبان پایتون می‌رویم و چند نکته اساسی در مورد ساختار یک برنامه پایتون ذکر می‌کنیم. در نهایت با چند تابع ساده پایتون برای نمایش در خروجی (مانیتور) و گرفتن داده از ورودی (کیبورد) آشنا می‌شویم.

۱ انواع داده اصلی در پایتون

در هفته قبل به بررسی انواع داده عددی (عدد صحیح و عدد اعشاری) در پایتون پرداختیم. در مورد اعداد اعشاری (یا ممیز شناور floating point) این نکته را اضافه می‌کنیم که اعداد اعشاری در پایتون به دو شیوه نمایش داده می‌شوند. شیوه ممیز شناور که همان شیوه معمول است. برای مثال اعداد 23.41 و 0.00231 به شیوه ممیز شناور معمولی نمایش داده شده‌اند. شیوه دیگر، نمایش علمی و یا توانی است که معمولا برای نمایش اعداد خیلی بزرگ و یا خیلی کوچک استفاده می‌شود. در این حالت عدد اعشاری بصورت توانی ضرب یک عدد در توانی از ۱۰ نمایش داده می‌شود. برای مثال عدد 23.41 بصورت $2.341e1$ نمایش داده می‌شود که معادل با 2.341×10^1 است. عدد 0.00231 بصورت $2.31e-3$ نمایش داده می‌شود که معادل با 2.31×10^{-3} است. بطور معمول عدد قبل از حرف e یک عدد کمتر از 10 است و عدد بعد از حرف e توان 10 است که باید یک عدد صحیح (مثبت یا منفی) باشد.

این نکته را هم اضافه کنیم که محدودیتی برای بزرگی و کوچکی اعداد اعشاری در پایتون وجود دارد. همانطور که در هفته قبل ذکر شد، هر عدد اعشاری را نمی‌توان بصورت دقیق در کامپیوتر ذخیره کرد چون بعضی از اعداد اعشاری در مبنای دو نیاز به بینهایت رقم دارند. از این رو، برای ذخیره نوع اعشاری تعداد محدودی بیت حافظه در نظر گرفته شده است. اگر عدد خیلی بزرگ باشد (یا خیلی کوچک باشد) مفسر پایتون مقدار سمبلیک inf به معنای بینهایت را چاپ می‌کند یا اینکه پیام خطا چاپ می‌شود.



۱.۱ نوع داده منطقی

حاصل یک عبارات عددی مثل $4 \times 3 - 12$ و یا $1 + 2^4 - \frac{5}{2.3}$ یک عدد است. در این عبارات از عملگرهای اصلی مانند جمع + و ضرب $\times$ و تفریق ـ و توان استفاده شده است. اما عملگرهایی دیگری نیز مانند بزرگتری $>$، کوچکتری $<$ و تساوی $=$ نیز وجود دارند که عباراتی از نوع دیگر را می‌سازند. برای مثال $3 < 4$ هم یک عبارت معمول در ریاضیات است که مانند یک گزاره خبری است. اینگونه عبارات (یا گزاره‌ها) یکی از دو ارزش درست true یا نادرست false را دارند. عبارت $3 < 4$ درست است و ارزش true دارد اما عبارت $4 \geq 7$ نادرست است و ارزش false دارد.

تعریف: در پایتون، یک عبارت منطقی (یا گزاره منطقی) عبارتی است که با اعداد، متغیرها و عملگرهای منطقی مانند تساوی $==$، بزرگتر یا مساوی $\geq$، عطف منطقی and، یای منطقی or و یا نقیض منطقی not ساخته شده است. یک عبارت منطقی می‌تواند ترکیبی باشد و حاصل ترکیب چند عبارت منطقی کوچکتر باشد. معمولا از پرانتز برای نظم و ترتیب دادن به بخشهای عبارت استفاده می‌شود. برای مثال گزاره‌های زیر، عبارات منطقی هستند.

```
3 == 5

x+y =>  10

(x==0) or (y != z)

2**x => 4

(10+y <= 5.4) and (x*z > 4)
```

- هر عبارت منطقی بستگی به مقداری که متغیرهایش دارند، ارزش درست True و یا نادرست False را دارد. برای مثال عبارت زیر اگر مقدار x برابر با 4 باشد و y برابر با 6 باشد ارزش True خواهد داشد. اما اگر x برابر با 3 و y برابر با 5 باشد ارزش False را خواهد داشت.

```
x+y =>  10
```

- کلمات True و False از کلمات کلیدی پایتون هستند. واضح است که خود True ارزش True دارد و False ارزش False دارد.
- یک عدد تنها نیز یک عبارت منطقی است. طبق قرارداد همه اعداد غیر صفر ارزش True و عدد صفر ارزش False دارد. در واقع در میان اعداد تنها صفر است که ارزش False دارد.

```
1       (True)
0       (False)
2.312   (True)
0.003   (True)
0.0     (False)
-343    (True)
```

- تساوی در پایتون با نماد == و عدم تساوی با نماد =! نشان داده می‌شود.

```
4 == 5       (False)
4 != 5       (True)
```

- دو عبارت منطقی را می‌توان با استفاده از عملگرهای منطقی از قبیل and و or با هم ترکیب کرد و یک عبارت منطقی جدید بدست آورد.

```
True and False
True and 1
3  or -5
(10+y <= 5.4) and (x*z > 4)
x  or   (z !=3)
```

- همانطور که گفته شد، عملگر منطقی and دو عبارت منطقی را با هم ترکیب (عطف) می‌کند. به هر یک از این عبارتهای منطقی دو طرف and یک عملوند گفته می‌شود. حاصل عبارت and ارزش True دارد اگر و فقط اگر هر دو عملوند ارزش True داشته باشند. (اگر یکی False باشد، حاصل False خواهد بود)

```
True and False              (False)
True and 1                  (True)
3  and -5                   (True)
(5 <= 5.4) and (10 > 4)     (False)
1.2  and  (0 !=3)           (True)
```

- حاصل عبارت or ارزش True دارد اگر یکی از عملوندها (یا هر دو) ارزش True داشته باشند. (اگر هر دو False باشند، حاصل False خواهد بود)

```
True or False                (True)
True or 1                    (True)
0  or  0                     (False)
(5 <= 5.4) or (10 < 4)       (False)
1.2  or  (0 != 0)            (True)
```

- عملگر not تنها یک عملوند دارد و نقیض ارزش عملوند را برمی‌گرداند. اگر عملوند ارزش False داشته باشد، حاصل True می‌شود و اگر True باشد حاصل False می‌شود.

```
not(False)       (True)
not(True)        (False)
not(1)           (False)
not(0)           (True)
not(3 > 2)       (False)
```

- می‌توان با ترکیب عبارات ساده‌تر، عبارات منطقی پیچیده‌تر ساخت. برای نظم بخشیدن به عبارت و مشخص کردن اولویتها بهتر است از پرانتز استفاده شود.

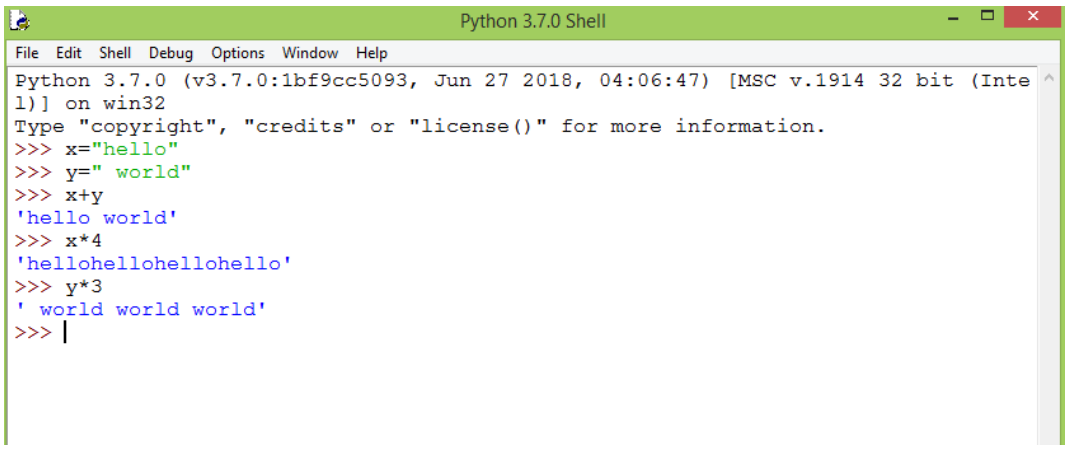
```
not((x > y) and (z ==0))
(x*x +1 < 0) or not(y)
```

۲.۱ نوع داده رشته string

متغیرها علاوه بر اعداد و یا مقادیر منطقی (مثل True و False) می‌توانند مقادیر رشته‌ای نیز دریافت کنند. یک رشته (string) دنباله‌ای از حروف است. یک رشته در پایتون با استفاده از علامت گیومه بالا و یا دو گیومه بالا (نقل قول یا quotation) نشان داده می‌شود.

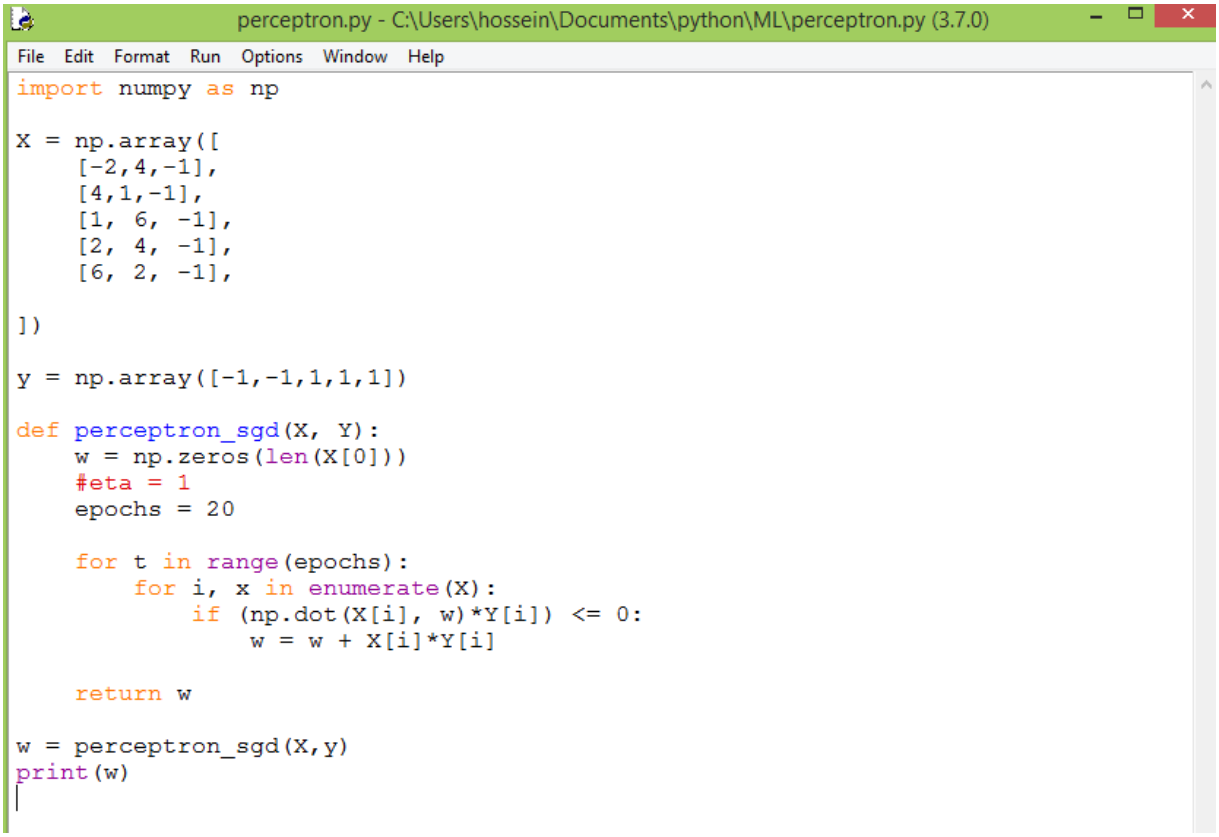
```
x = "hello"
x = 'hello'
y = "hello world"
z = '23ahdag5'
```

دو متغیر x و y که مقادیر رشته‌ای داشته باشند، حاصل $x + y$ یک رشته است که از به چسپاندن دو رشته x و y بدست می‌آید. $x * 4$ به معنی تکرار x به تعداد ۴ بار است.



۲ چند نکته در مورد ساختار یک برنامه پایتون

یک برنامه پایتون در واقع لیستی از دستورات پایتون است که پشت سر هم اجرا می‌شوند. برنامه‌های پایتون در فایلهای متنی با پسوند py. ذخیره می‌شوند. در شکل زیر یک نمونه برنامه پایتون نشان داده شده است.



نگارش برنامه‌های پایتون تابع قوانین و اصولی است که در زیر به ذکر چند نمونه از آنها می‌پردازیم.

- در هر خط تنها یک دستور باید نوشته شود. بعضی از دستورات پایتون در چند خط نوشته می‌شوند ولی در خط تنها یک دستور باید نوشته شود.

```
x = "hello"    y = 23            #   (error)

x = "hello"
y = 23                           #   (correct)
```

- قبل از شروع دستور در خط نباید فاصله خالی باشد. وجود فاصله خالی در شروع دستور باعث تولید پیام خطا یا ابهام در برنامه می‌شود. (البته در دستورات چند خطی از فاصله خالی در شروع استفاده می‌شود که در دروس بعدی به آن پرداخته می‌شود)

```
x = "hello"                 #   (error)
   y = 23


x = "hello"
y = 23                      #   (correct)
```

- خط خالی در برنامه مجاز است.

```
x = "hello"

y = 23
```

- اگر بخواهیم خطی اجرا نشود می‌توان به شروع دستور کاراکتر # را اضافه کنیم. این باعث می‌شود که مفسر پایتون از اجرای خط مربوطه صرف نظر کند. در کل مفسر پایتون با دیدن کاراکتر # از اجرای بقیه خط صرف نظر می‌کند. از این ابزار برای اضافه کردن توضیحات comment به برنامه نیز استفاده می‌شود. در واقع به این کار، اضافه کردن کامنت به برنامه و یا کامنت کردن یک خط نیز گفته می‌شود.

```
#  x = "hello"

# y is an integer
y = 23
```

- از کلمات کلیدی پایتون نباید برای نامگذاری متغیرها استفاده کرد. کلمات کلیدی کلماتی هستند که توسط زبان پایتون قبلا رزرو شده‌اند. برای مثال کلمات for و while و def جزو زبان پایتون هستند و نباید برای نامگذاری متغیرها از آنها استفاده کرد. انجام این کار تولید خطا خواهد کرد.

```
for = 23          (error)
while =  "hello"  (error)
```

- پایتون (مانند همه زبانهای برنامه‌نویسی) نسبت به حرف کوچک و بزرگ حساس است. اصطلاحا case-sensitive است. این بدین معنی است که برای مثال نامهای len و Len دو شناسه متفاوت در پایتون هستند.

۱.۲ دستور print

یکی از مهمترین و ابتدایی ترین دستورات در زبان پایتون دستور print است. از این دستور برای چاپ در خروجی (صفحه نمایش کامپیتور) استفاده می‌شود. برای مثال دستور زیر کلمه hello را در خروجی چاپ می‌کند.

```
print("hello")
```

دستور زیر عدد 345 را در خروجی چاپ می‌کند.

```
x = 345
print(x)
```

دستور print در واقع یک تابع است (در مورد توابع در پایتون در دروس بعدی بحث خواهد شد). هر تابع یک سری ورودی دارد که به آنها پارامترهای تابع گفته می‌شود. به دستور print می‌توان یک سری مقدار ثابت و متغیر به عنوان پارامتر ورودی برای چاپ داد. این دستور مقادیر داده شده را با ترتیبی که مشخص شده چاپ می‌کند. دستور print بعد از مقدار هر پارامتر یک فاصله می‌اندازد. به مثال زیر توجه کنید.

```
x = "Hello"
y = "world!"
print(x,y," This is a good day.")
```

دستور بالا خط زیر را چاپ می‌کند.

```
Hello world! This is a good day.
```

اگر بخواهیم به جای کاراکتر فاصله چیز دیگری بین مقادیر داده شده چاپ شود می‌توان از پارامتر sep استفاده کرد.

```
x = "Hello"
y = "world!"
print(x,y," This is a good day.",sep= "---")
```

دستور بالا خط زیر را چاپ می‌کند.

```
Hello---world!---This is a good day.
```

همیشه بطور پیش فرض اجرای دستور print از خط جدید شروع می‌شود.

```
x = "Hello"
y = "world!"
print(x)
print(y)
print( "This is a good day.")
```

دستورات بالا خروجی زیر را چاپ می‌کند.

```
Hello
world!
This is a good day.
```

۲.۲ دستور input

برای گرفتن ورودی از کاربر (برای مثال از طریق کیبورد) بطور معمول از دستور input استفاده می‌شود. حاصل دستور input را می‌توان در یک متغیر ذخیره کرد.

وقتی که دستور بالا اجرا می‌شود، در منوی خروجی، برنامه منتظر می‌ماند تا اینکه کاربر مقدار را وارد کند و کلید اینتر را فشار دهد. هر چیزی که قبلا از اینتر وارد شده در متغیر x ریخته می‌شود. در واقع خروجی input یک رشته است. در این حالت متغیر x حاوی یک رشته است.

```
x = input()
```

برای اینکه قبل از گرفتن ورودی پیامی برای کاربر چاپ شود، می‌توان یک رشته را برای چاپ به دستور input داد. به مثال زیر توجه کنید.

```
x = input("Please enter a name:")
```

دستور بالا موقع اجرا، عبارت Please enter a name: را در خروجی چاپ می‌کند و سپس منتظر می‌ماند تا اینکه کاربر مقدار را وارد کند. کاربر هرچه وارد کند (آنچه قبل از اینتر وارد شده) در متغیر x ذخیره می‌شود.

برنامه زیر یک اسم را از کاربر دریافت کرده و در خروجی به اسم داده شده سلام می‌کند.

```
x = input("Please enter a name:")
print("Hello",x,"!")
```

در درس بعدی مثالهای بیشتری را با print و input انجام می‌دهیم.