عملگرهای مفید

چند تابع و "عملگر" در پایتون وجود دارد که به خوبی در هیچ یک از دسته‌بندی‌ها جا نمی‌گیرند، بنابراین ما در این سخنرانی به آن‌ها می‌پردازیم، بیایید شروع کنیم!

range

تابع range به شما اجازه می‌دهد تا به سرعت یک لیست از اعداد صحیح را *تولید* کنید، این خیلی کاربردی است، بنابراین توجه کنید که چگونه از آن استفاده کنید! سه پارامتر وجود دارد که می‌توانید منتقل کنید، یک شروع، یک توقف و یک اندازه گام. بیایید چند مثال را ببینیم:

In [1]:

```
range(0,11)
```

Out[1]:

```
range(0, 11)
```

توجه داشته باشید که این یک تابع **generator** است، بنابراین برای دریافت یک لیست از آن، ما نیاز داریم که آن را با **list()** به یک لیست تبدیل کنیم. چه چیزی یک تولید کننده است؟ این یک نوع خاص از تابع است که اطلاعات را تولید می‌کند و نیازی به ذخیره آن در حافظه ندارد. ما هنوز درباره توابع یا تولید کننده‌ها صحبت نکرده‌ایم، بنابراین فعلاً این را در یادداشت‌های خود نگه دارید، ما در جزئیات بسیار بیشتری در آینده در آموزش شما به این موضوع خواهیم پرداخت!

In [3]:

```
# Notice how 11 is not included, up to but not including 11, just like slice notatio
list(range(0,11))
```

Out[3]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [4]:

```
list(range(0,12))
```

Out[4]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [6]:

```
# Third parameter is step size!
# step size just means how big of a jump/leap/step you
# take from the starting number to get to the next number.

list(range(0,11,2))
```

Out[6]:

```
[0, 2, 4, 6, 8, 10]
```

In [7]:

```
list(range(0,101,10))
```

Out[7]:

```
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

enumerate

enumerate یک تابع بسیار مفید برای استفاده با حلقه‌های for است. بیایید این وضعیت را تصور کنیم:

In [8]:

```
index_count = 0

for letter in 'abcde':
    print("At index {} the letter is {}".format(index_count,letter))
    index_count += 1
```

```
At index 0 the letter is a
At index 1 the letter is b
At index 2 the letter is c
At index 3 the letter is d
At index 4 the letter is e
```

ردیابی اینکه چند حلقه از طریق رفته‌اید به قدری متداول است که enumerate ایجاد شده است تا نیازی نباشد نگران ایجاد و به‌روزرسانی این متغیر index_count یا loop_count باشید.

In [10]:

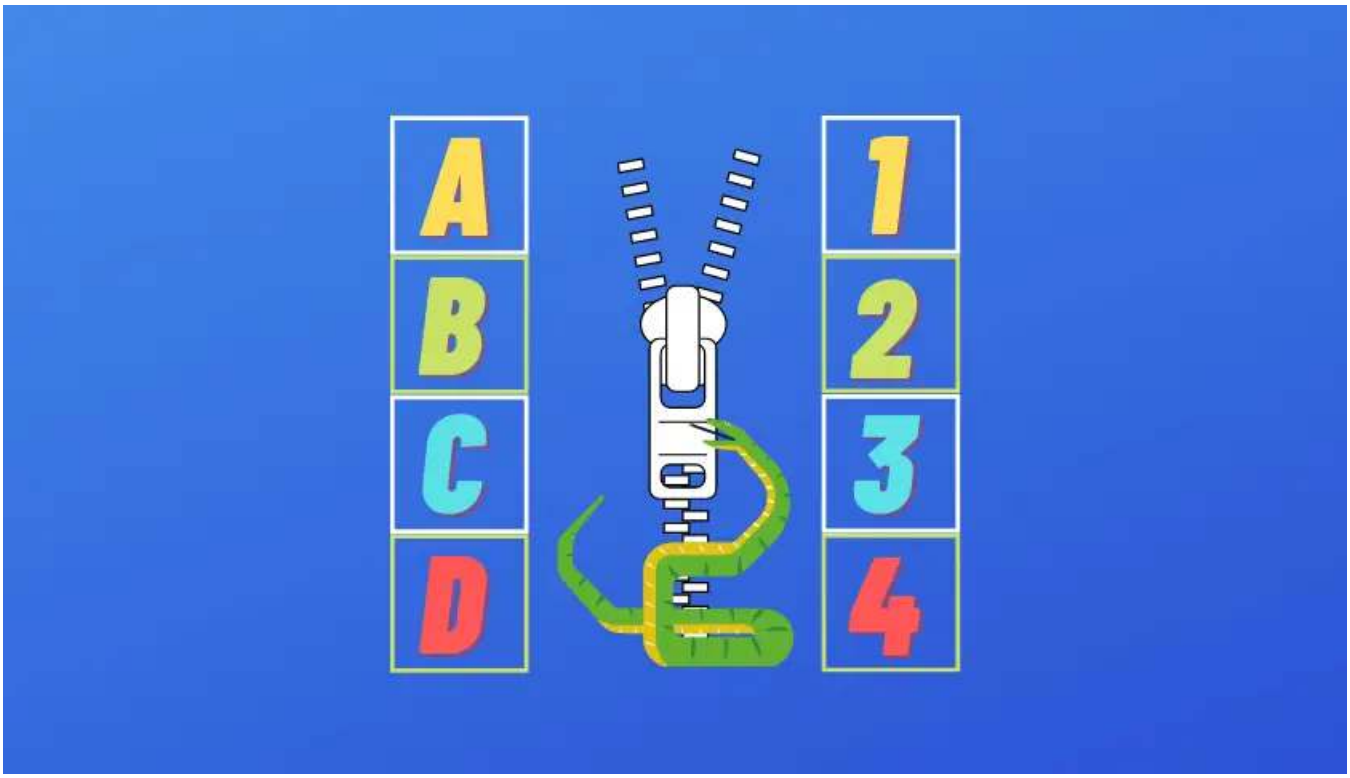
```
# Notice the tuple unpacking!

for i,letter in enumerate('abcde'):
    print("At index {} the letter is {}".format(i,letter))
```

```
At index 0 the letter is a
At index 1 the letter is b
At index 2 the letter is c
At index 3 the letter is d
At index 4 the letter is e
```

zip

توجه داشته باشید که فرمت enumerate در واقع چه چیزی را برمی گرداند ، بیایید با تبدیل آن به list() نگاهی به آن بیندازیم



In [12]:

```
1 list(enumerate('abcde'))
```

Out[12]:

```
[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd'), (4, 'e')]
```

این لیست از tuple ها بود ، به این معنی که می توانستیم از tuple unpacking در حلقه for ما استفاده کنیم. این ساختار داده در پایتون واقعاً بسیار رایج است ، به خصوص هنگام کار با کتابخانه های خارجی. شما می توانید از تابع **zip()** برای ایجاد سریع لیست tuple ها با "زیپ کردن" دو لیست با هم استفاده کنید.

In [13]:

```
1 mylist1 = [1,2,3,4,5]
2 mylist2 = ['a','b','c','d','e']
```

In [15]:

```
1 # This one is also a generator! We will explain this later, but for now let's transf
2 zip(mylist1,mylist2)
```

Out[15]:

```
<zip at 0x1d205086f08>
```

In [17]:

```
1 list(zip(mylist1,mylist2))
```

Out[17]:

```
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd'), (5, 'e')]
```

برای استفاده از generator ، ما می توانستیم فقط از یک حلقه for استفاده کنیم

In [20]:

```
for item1, item2 in zip(mylist1,mylist2):
    print('For this tuple, first item was {} and second item was {}'.format(item1,it
```

```
For this tuple, first item was 1 and second item was a
For this tuple, first item was 2 and second item was b
For this tuple, first item was 3 and second item was c
For this tuple, first item was 4 and second item was d
For this tuple, first item was 5 and second item was e
```

min and max

برای بدست اوردن، حداقل مقدار یا حداکثر مقدار داخل یک لیست یا تاپل می توانیم از توابع min() و max() استفاده کنیم

In [1]:

```
mylist = [10,20,30,40,100]
```

In [27]:

```
min(mylist)
```

Out[27]:

```
10
```

In [44]:

```
max(mylist)
```

Out[44]:

```
100
```

sum

برای بدست اوردن مجموع اعضای یک لیست یا تاپل می توانیم از تابع sum() استفاده نماییم.

In [2]:

```
mylist = [10,20,30,40,50]
```

In [3]:

```
sum(mylist)
```

Out[3]:

```
150
```

random

پایتون با یک کتابخانه تصادفی داخلی همراه است. تعداد زیادی از توابع در این کتابخانه تصادفی وجود دارد ، بنابراین ما فقط دو تابع مفید را برای حال نشان می دهیم.

In [29]:

```
from random import shuffle
```

In [35]:

```
# This shuffles the list "in-place" meaning it won't return
# anything, instead it will effect the list passed
shuffle(mylist)
```

In [36]:

```
mylist
```

Out[36]:

```
[40, 10, 100, 30, 20]
```

In [39]:

```
from random import randint
```

In [41]:

```
# Return random integer in range [a, b], including both end points.
randint(0,100)
```

Out[41]:

```
25
```

In [42]:

```
# Return random integer in range [a, b], including both end points.
randint(0,100)
```

Out[42]:

```
91
```

input

همانگونه که در بخش قبل مشاهده نمودیم، این تابع برای دریافت ورودی از کاربر استفاده می شود

In [43]:

```
1 input('Enter Something into this box: ')
```

```
Enter Something into this box: great job!
```

Out[43]:

```
'great job!'
```

توجه داشته باشید که تمام مقادیری که توسط این تابع دریافت میشوند به رشته تبدیل میشوند:

In [4]:

```
1 x = input('Enter Something into this box: ')
```

```
Enter Something into this box: 6
```

In [5]:

```
1 type(x)
```

Out[5]:

```
str
```

In [6]:

```
1 x
```

Out[6]:

```
'6'
```