: abs(x)

تابعی است که یک عدد میگیرد و قدر مطلق آن را برمیگرداند

```
print(abs(-123))
>>> 123

print(abs(7800))
>>> 7800
```

: all(iterable)

این تابع مانند عملگر and کار میکند

```
number = (3>2),(1<0)
print(all(number))
>>> False

number = (3>2),(1<100)
print(all(number))
>>> True
```

: any(iterable)

این تابع مانند عملگر or کار میکند

```
number = (3>2),(1<0)
print(any(number))
>>> True

number = (3>20),(100<100)
print(all(number))
>>> False
```

---

bin(x):

تابعی است که یک عدد به عنوان ورودی دریافت میکند و آن را برحسب مبنای 2 برمیگرداند

```
number = 17
print(bin(number))
>>> 0b10001

number = 54
print(bin(number))
>>> 0b110110
```

---

bool(object):

تابعی که خروجی آن به صورت True یا False است

```
print(bool(''))
>>> False

print(bool(' '))
>>> True

print(bool(3>20))
>>> False
```

---

chr(i):

یک عدد از ورودی میگرد و کارکتر یونیکد آن را برمیگرداند

```
print(chr(97))
>>> a

print(chr(8364))
>>> €
```

#در صورتی که کارکتر یونیکد برای عدد وجود نداشته باشد کارکتر خالی چاپ میشود

---

complex() :

این تابع میتواند یا number یا string یا real,imag بگیرد

این تابع اعداد مختلط را برمیگرداند

```
print(complex('+1.23'))
>>> (1.23+0j)

print(complex(+17))
>>> (17+0j)

print(complex(real = 5 , imag=4.5))
>>> (5+4.5j)
```

---

dict(a) :

دیتا تایپی که قابلیت تبدیل دارد را به دیکشنری تبدیل میکند

```
a = [1,2,3,4,5,6]
print(dict(a))
>>> Error

b = [('ali' , 15) , ('amir' , 20) , ('reza',25)]
print(dict(b))
>>> {'ali' : 15 , 'amir' : 20 , 'reza' : 25}
```

: divmod(a,b)

تابعی است که دو ورودی میگیرد ورودی اول را تقسیم بر ورودی دوم میکند و خارج قسمت و باقی مانده را برمیگرداند

```
(a,b)
>>> (a//b , a%b)

print(divmod(23,2))
>>> (11,1)
```

: enumerate(iterable , start=0)

برای دیتا تایپ مورد نظر به اعضای هر شی یک عدد برای شماره گذاری آن میگذارد

```
a = [‘ali’ , ‘amir’ , ‘reza’ , 25]
print(list(enumerate(a))
>>> [(0 , ‘ali’) , (1 , ‘amir’) , (2 , ‘reza’ ) , (3 , 25)]

b = [‘ali’ , ‘amir’ , ‘reza’ , 25]
print(list(enumerate(b , start=10))
>>> [(10 , ‘ali’) , (11 , ‘amir’) , (12 , ‘reza’ ) , (13 , 25)]
```

: eval()

دستورات داخل استرینگ را انجام میدهد

```
a = ‘17+20-2’
print(eval(a))
>>> 35

b = ‘15*2’
print(eval(b))
>>> 30
```

```
c = ‘print(‘alireza’)’
print(eval(c))
>>> alireza
```

---

exec() :

دستورات داخل استرینگ را انجام میدهد

این تابع میتواند کد های پیشتری از تابع eval را اجرا کند

```
c = ‘‘‘
def ali(a):
    return (f‘welcome {a}’)
b = ‘amir’
print(ali(b))
’’’
print(exec(c))
>>> welcome amir
```

#این کد با دستور eval ارور میدهد

:filter(function , iterable)

تابعی است که دو ورودی میگیرد

ورودی اول روی اعضای ورودی دوم اجرا میشوند

```
b = list(range(0,10+1))
print(b)
>>> [0,1,2,3,4,5,6,7,8,9,10]
c = (filter(lambda x:True if x>5 else False , b))
print(list(c))
>>> [6,7,8,9,10]


#d = lambda x:True if x>5 else False
#def d(x):
    if x>5:
        return True
    else:
        return False
```

دو کامنت بالا یکسان هستند

به عبارتی کارکتر های ورودی دوم از فیلتر ورودی اول رد میشوند

مقادیر True میماند و مقادیر False دور ریخته میشوند

---

frozenset():

تابعی است با تمام قابلیت های مجموعه اما با این تفاوت که نمیتوان به آن عضوی اضافه کرد

```
b = set(range(0,5+1))
b = b.add(10)
print(b)
>>> {0,1,2,3,4,5,10}

b = frozenset(range(0,5+1))
b = b.add(10)
print(b)
>>> Error
```

---

id(object):

زمانی که در پایتون یک متغیر تعریف میشود یک شناسه در موری مختص آن تعریف میشود میتوان با تابع id آن شناسه را رویت کرد

```
a = ‘amir’
print(id(a))
>>> 1487797245232
```

#به دلایل امنیتی بعد هر بار اجرا برنامه شناسه متغیر تغیر میکند

:isinstance(object , classinfo)

این تابع دو ورودی میگرد و چک میکند که ورودی اول از نوع و تایپ ورودی دوم باشد

```
a = ‘amir’
print(isinstance(a , str))
>>> True

b = ‘amir’
print(isinstance(b , int))
>>> False

c = ‘amir’
print(isinstance(c , (int , str , tuple , bool)))
>>> True
```

#میتوان برای ورودی دوم این تابع یک تاپل را ارسال کرد اگر که ورودی اول از نوع یکی از اشیا ورودی دوم بود True را برمیگرداند

در مثال بالا متغیر c از جنس bool,int,tuple نیست اما از جنس str هست

:map(function,iterable)

این تابع دوورودی میگرد ورودی اول روی اعضای ورودی دوم اعمال میشود

```
def power(x):
    return x**2
a = [0,1,2,3,4,5]
print(list(map(power,a)))
>>> [0,1,4,9,16,25]
```

:max(iterable)

این تابع بزرگترین آیتم ورودی را برمیگرداند برای حروف این تابع بر اساس ترتیب حروف الفبا کار میکند

```
a = [110 , 17 , 22 , 35 , 499 , 51]
```

```
print(max(a))
>>> 499

print(max('a' , 'bb' , 'aa'))
>>> bb
```

#در این تابع حرف z بزرگترین کارکتر و a کوچکترین کارکتر است

:min(iterable)

این تابع کوچکترین آیتم ورودی را برمیگرداند برای حروف این تابع بر اساس ترتیب حروف الفبا کار میکند

```
a = [110 , 17 , 22 , 35 , 499 , 51]
print(min(a))
>>> 17

print(min('a', 'bb' , 'aa'))
>>> a
```

#در این تابع حرف a بزرگترین کارکتر و z کوچکترین کارکتر است

:oct(int)

این تابع مبنای هشت عدد ورودی را برمیگرداند

```
print(oct(22))
>>> 0o26
```

:ord()

کارکتر یونیکد را میگید و عدد آن را برمیگرداند

این تابع بر خلاف تابع chr میباشد

```
print(ord('5'))
>>> 53

print(ord('A'))
>>> 65

print(ord('a'))
>>> 97
```

:pow(x,y,z)

این تابع پارامتر اول را به توان پارامتر دوم میرساند وجود دو ورودی اول اجباری و وجود ورودی سوم اختیاری است

جواب عملیات توان دو ورودی اول با ورودی سوم تقسیم شده و باقی مانده را برمیگرداند

```
pow(x,y,z)
>>> ((x**y)%z)

print(pow(4,3))
>>> 64

print(pow(4,3,5))
>>> 4 #(4**3)%5
```

---

round():

این تابع یک عدد میگیرد و به تعدادی که در ورودی دوم آمده آن را روند میکند

ورودی دوم به طور پیشفرض 0 است و کلا عدد را گرد میکند

```
print(round(4.147,2))
```

```
>>> 4.15
```

```
print(round(4.147))
>>> 4
```

:sum()

این تابع اعضای یک متغیر را جمع میکند دو ورودی میگرد ورودی اول اجباری است و ورودی دوم اختیاری

در صورتی که ورودی دوم را وارد کنید حاصل جمع پارامتر اول با پارامتر دوم جمع میشود

```
a = [2 , 17 , 20 , 3]
print(sum(a))
>>> 42
```

```
a = [2 , 17 , 20 , 3]
print(sum(a , start= -2))
>>> 40
```

:zip()

این تابع چند ورودی قابل پیمایش مثل رشته یا لیست یا تاپل و ...

را به هم وصل میکند به طوری که اشیا صفرم باهم قرار میگرند اشیا اول با هم قرار میگرند و به همین ترتیب

```
a = [2 , 17 , 20 , 3]
b = [5 , 6 , 'ali' , 'm']
print(list(zip(a,b)))
>>> [(2,5) , (17,6) , (20,'ali') , (3,'m')]

a = [2 , 17 , 20 ]
b = [5 , 6 , 'ali' , 'm']
print(list(zip(a,b)))
>>> [(2,5) , (17,6) , (20,'ali')]
```

#اگر ورودی ها تعداد عضو های برابر نداشته باشند کمترین عضو در نظر گرفته میشود

پایان