# مجموعه و بولین‌ها

دو نوع دیگر از اشیاء در پایتون وجود دارند که باید به طور سریع به آنها اشاره کنیم: مجموعه‌ها و بولین‌ها.

## مجموعه‌ها

مجموعه‌ها مجموعه‌ای نامرتب از عناصر *منحصربفرد* هستند. ما می‌توانیم آنها را با استفاده از تابع set() بسازیم. بیایید از این تابع برای ساختن یک مجموعه استفاده کنیم تا نحوه عملکرد آن را ببینیم.

### دو نکته مهم در مجموعه های ریاضی که در مجموعه های پایتون نیز صادق می باشند:

1. تکرار در مجموعه های بی معنا می باشد.
2. ترتیب در مجموعه ها مهم نیست

### نوع داده مجموعه در پایتون بیشتر برای ذخیره داده ها و بررسی عضویت مقادیر در یک مجموعه کاربرد دارد. ما در مجموعه امکان بازیابی مقادیر از طریق ایندکس گذاری یا کلید و ... نداریم.

In [1]:

```
x = set()
```

In [2]:

```
type(x)
```

Out[2]:

```
set
```

In [3]:

```
# We add to sets with the add() method
x.add(1)
```

In [4]:

```
#Show
x
```

Out[4]:

```
{1}
```

توجه کنید که از پرانتزهای آکولادی استفاده شده است. این نشان دهنده یک دیکشنری نیست! با این حال، می‌توانید بگویید مجموعه یک دیکشنری است که فقط دارای کلیدهاست.

ما می‌دانیم که یک مجموعه تنها ورودی‌های منحصربفرد دارد. پس چه اتفاقی می‌افتد وقتی سعی می‌کنیم چیزی را به مجموعه اضافه کنیم که از قبل در آن وجود دارد؟

In [5]:

```
# Add a different element
x.add(2)
```

In [6]:

```
#Show
x
```

Out[6]:

```
{1, 2}
```

In [7]:

```
# Try to add the same element
x.add(1)
```

In [8]:

```
#Show
x
```

Out[8]:

```
{1, 2}
```

In [9]:

```
# membership check with in operator
1 in x
```

Out[9]:

```
True
```

In [10]:

```
3 in x
```

Out[10]:

```
False
```

توجه کنید که مجموعه مجدداً یک عدد 1 را در آن قرار نمی‌دهد. این به این دلیل است که مجموعه تنها در ارتباط با عناصر منحصربفرد اهمیت دارد! ما می‌توانیم یک لیستی که شامل عناصر تکراری است را به یک مجموعه تبدیل کنیم تا عناصر منحصربفرد را بدست آوریم. به عنوان مثال:

In [13]:

```
# Create a list with repeats
list1 = [1,1,1,2,2,3,4,4,4,5,6,6,7,7,7,7,7]
```

In [14]:

```
# Cast as set to get unique values
unique = set(list1)


unique
```

Out[14]:

```
{1, 2, 3, 4, 5, 6, 7}
```

In [15]:

```
list(unique)
```

Out[15]:

```
[1, 2, 3, 4, 5, 6, 7]
```

In [16]:

```
tuple(unique)
```

Out[16]:

```
(1, 2, 3, 4, 5, 6, 7)
```

In [17]:

```
set('Hello wollll')
```

Out[17]:

```
{' ', 'H', 'e', 'l', 'o', 'w'}
```

## اعمال مجموعه ها

1. اشتراک

In [18]:

```
x = {1, 2, 3, 4, 5, 6, 7}
y = {4, 6, 10, 11, 12}
```

In [19]:

```
x.intersection(y)
```

Out[19]:

```
{4, 6}
```

2. اجتماع

In [20]:

```
x.union(y)
```

Out[20]:

```
{1, 2, 3, 4, 5, 6, 7, 10, 11, 12}
```

3. تفاضل مجموعه ها

In [21]:

```
x.difference(y)
```

Out[21]:

```
{1, 2, 3, 5, 7}
```

4. تفاضل متقارن مجموعه ها

In [22]:

```
x.symmetric_difference(y)
```

Out[22]:

```
{1, 2, 3, 5, 7, 10, 11, 12}
```

## بولین‌ها

پایتون دارای بولین‌ها است (با نمایش‌های از پیش تعیین شده True و False که در واقع فقط عددهای صحیح 1 و 0 هستند). همچنین یک شیء پوشاننده به نام None نیز دارد. بیایید از طریق چند مثال سریع از بولین‌ها بگذریم (بعدها در این دوره به طور عمیق‌تر به آنها خواهیم پرداخت).

In [23]:

```
# Set object to be a boolean
a = True
```

In [24]:

```
#Show
a
```

Out[24]:

```
True
```

In [25]:

```
type(a)
```

Out[25]:

```
bool
```

ما همچنین می‌توانیم از عملگرهای مقایسه برای ایجاد بولین‌ها استفاده کنیم. در طول دوره، به تمامی عملگرهای مقایسه خواهیم پرداخت.

In [26]:

```
# Output is boolean
1 > 2
```

Out[26]:

```
False
```

In [27]:

```
2 < 5
```

Out[27]:

```
True
```

ما می‌توانیم از None به عنوان یک نمادگذار برای یک شیء که هنوز نمی‌خواهیم آن را مجدداً اختصاص دهیم، استفاده کنیم.

In [29]:

```
# None placeholder
b = None
```

In [30]:

```
# Show
b
```

In [31]:

```
type(b)
```

Out[31]:

```
NoneType
```

درسته! اکنون باید درک اولیه‌ای از اشیا و نوع ساختار داده‌های پایتون داشته باشید. حالا بروید و آزمون ارزیابی را انجام دهید!