

# ASIMOV

---

## Set e Booleanos

Existem dois outros tipos de objeto em Python que devemos cobrir rapidamente. Conjuntos(Sets) e Booleanos.

### Sets

Os conjuntos são uma coleção não ordenada de elementos *únicos*. Podemos construí-los usando a função `set()`. Avançemos e façamos um conjunto para ver como funciona

```
In [3]: x = set()
```

```
In [4]: # Adicionamos elementos com o método add().  
x.add(1)
```

```
In [5]: # Mostra  
x
```

```
Out[5]: {1}
```

Observe os colchetes. Isso não indica um dicionário! Embora você possa montar analogias como um set sendo um dicionário com apenas chaves. Sabemos que um conjunto tem apenas entradas únicas. Então, o que acontece quando tentamos adicionar algo que já está em um conjunto?

```
In [6]: # Adiciona um elemento novo  
x.add(2)
```

```
In [7]: # Mostra  
x
```

```
Out[7]: {1, 2}
```

```
In [8]: # Adiciona o mesmo elemento  
x.add(1)
```

```
In [10]: # Mostra  
x
```

```
Out[10]: {1, 2}
```

Observe como ele não colocará mais 1 lá. Isso porque um conjunto apenas se ocupa de elementos exclusivos! Podemos transformar uma lista com múltiplos elementos repetidos para um conjunto para obter os elementos exclusivos. Por exemplo:

```
In [11]: # Cria uma lista com elementos repetidos  
l = [1,1,2,2,3,4,5,6,1,1]
```

```
In [12]: # Transforma em um set com elementos únicos  
set(l)
```

```
Out[12]: {1, 2, 3, 4, 5, 6}
```

## Booleanos

O Python possui também Booleanos (com True e False predefinidas que são basicamente apenas os números inteiros 1 e 0). Ele também possui um objeto reservado chamado None. Passemos por alguns exemplos rápidos de booleanos (vamos mergulhar mais profundamente neles mais tarde neste curso).

```
In [13]: # Define um objeto como True  
a = True
```

```
In [14]: # Mostra  
a
```

```
Out[14]: True
```

Também podemos usar operadores de comparação para criar booleanos. Examinaremos todos os operadores de comparação mais tarde no curso.

```
In [15]: # O output é booleano  
1 > 2
```

```
Out[15]: False
```

Nós podemos usar None como um espaço reservado para um objeto que não queremos reatribuir ainda:

```
In [16]: b = None
```

É isso aí! Agora você deve ter uma compreensão básica de objetos Python e tipos de estrutura de dados. Em seguida, vá em frente e faça o teste de avaliação!