Conjuntos

Alessandro Botelho Bovo

O que é um `set`

- Coleção **sem ordem** e **sem elementos repetidos**.
- Ideal para **testar pertinência** (usar `in`) e fazer operações de **teoria dos conjuntos**.

```
frutas = {"maçã", "banana", "maçã", "pera"}
print(frutas)
```

```
{'banana', 'pera', 'maçã'}
```

```
print("banana" in frutas)
```

```
True
```

Criação de conjuntos

- Literal `{...}`

```
numeros = {1, 2, 3}
print(numeros)
```

```
{1, 2, 3}
```

Criação de conjuntos

- Construtor `set(...)`

```
vazia = set()
print(vazia)
```

```
set()
```

Criação de conjuntos

- Construtor `set(...)`

```
letras = set("alessandro")
print(letras)
```

```
{'o', 'n', 'a', 'l', 'r', 'e', 'd', 's'}
```

Criação de conjuntos

- Construtor `set(...)`

```
lista_sem_duplicatas = set([1, 2, 2, 3, 3, 3])
print(lista_sem_duplicatas)
```

```
{1, 2, 3}
```

Operações fundamentais

- União
- Interseção
- Diferença

- Versão operador
- Versão método
- Versão *in-place* (altera o próprio conjunto)

União (| ou union)

```
a = {1, 2, 3}
b = {3, 4, 5}

uniao1 = a | b
uniao2 = a.union(b)
print(uniao1)
print(uniao2)
```

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

Interseção (`&` ou `intersection`)

```
a = {1, 2, 3}
b = {2, 3, 4}

inter1 = a & b
inter2 = a.intersection(b)
print(inter1)
print(inter2)
```

```
{2, 3}
{2, 3}
```

Diferença (`-` ou `difference`)

```
a = {1, 2, 3, 4}
b = {3, 4, 5}

dif_ab = a - b              # Elementos de a que NÃO estão em b
dif_ba = b.difference(a) # Elementos de b que NÃO estão em a
print(dif_ab)   # {1, 2}
print(dif_ba)   # {5}
```

```
{1, 2}
{5}
```

Diferença simétrica (`^` ou `symmetric_difference`)

- Elementos que estão em **um OU no outro**, mas não nos dois.

```
a = {1, 2, 3}
b = {3, 4, 5}

dif_sim1 = a ^ b
dif_sim2 = a.symmetric_difference(b)
print(dif_sim1)
print(dif_sim2)
```

```
{1, 2, 4, 5}
{1, 2, 4, 5}
```

Versões *in-place* (atualizam o próprio conjunto)

```
a = {1, 2, 3}
b = {3, 4}

a.update(b)                          # união in-place: a = {1, 2, 3, 4}
a.intersection_update(b)             # interseção in-place: a = {3, 4}
a.difference_update({4})             # diferença in-place: a = {3}
a.symmetric_difference_update({2, 3})  # a = {2}
print(a)  # {2}
```

Adição de elementos

- Adiciona um elemento:

```
cidades = {"Londrina", "Cambé"}

cidades.add("Rolândia")     # adiciona um elemento
print(cidades)
```

```
{'Cambé', 'Londrina', 'Rolândia'}
```

Remoção de elementos

- Remove se existir (sem erro se não existir):

```
cidades = {"Londrina", "Cambé", "Rolândia"}
cidades.discard("Cambé")
print(cidades)

cidades.discard("Ibiporã")
print(cidades)
```

```
{'Rolândia', 'Londrina'}
{'Rolândia', 'Londrina'}
```

Remoção de elementos

- Remove se existir (gera erro se não existir):

```
cidades = {"Londrina", "Cambé", "Rolândia"}
cidades.remove("Londrina")
print(cidades)
```

```
{'Cambé', 'Rolândia'}
```

```
cidades.remove("Ibiporã") # Vai gerar KeyError
print(cidades)
```

```
KeyError: 'Ibiporã'
```

Remoção de elementos

- Remove e retorna um elemento "qualquer":

```
cidades = {"Londrina", "Cambé", "Rolândia"}
print(cidades)
item = cidades.pop()
print(item)
print(cidades)
```

```
{'Rolândia', 'Londrina', 'Cambé'}
Rolândia
{'Londrina', 'Cambé'}
```

Quando usar o `.pop()`

- Para "consumir" rapidamente todos os elementos sem se importar com a ordem:

```
conjunto = {1, 2, 3, 4}
while conjunto:
    x = conjunto.pop()
    print("Processando:", x)
```

```
Processando: 1
Processando: 2
Processando: 3
Processando: 4
```

Remoção de elementos

- Esvazia o conjunto:

```
cidades = {"Londrina", "Cambé", "Rolândia"}
if not cidades:
    print("Vazio")
else:
    print("Tem elementos")

cidades.clear()  # Esvazia o conjunto
if not cidades:
    print("Vazio")
else:
    print("Tem elementos")
```

```
Tem elementos
Vazio
```

Contato

Alessandro Botelho Bovo

alessandrobovo@utfpr.edu.br