
SET (Conjunto)

Set:

- Sets (conjuntos) são usados para armazenar uma coleção de dados.
  - Um set é definido entre chaves `{ }`.
  - Um set é uma coleção **não ordenada** e **não indexada**

In [1]:
```
conjunto = {"apple", "banana", "cherry"}
print(conjunto)
```

```
{'banana', 'cherry', 'apple'}
```

Conjuntos não são ordenados

- Os itens de um conjunto não possuem uma ordem definida
- Os itens do conjunto podem aparecer em uma ordem diferente sempre que você os utiliza

Conjuntos não são indexados

- O itens não podem ser referenciados por índice ou chave

Itens não repetidos

- Os conjuntos não podem ter itens com o mesmo valor.
- Valores duplicados serão ignorados.

In [3]:
```
conjunto = {"apple", "banana", "cherry", "apple"}
print(conjunto)
```

```
{'banana', 'cherry', 'apple'}
```

Conjuntos são heterogêneos:

- Podem armazenar itens de tipos diferente
  - True e 1 são considerados o mesmo valor

In [21]:
```
conjunto = {"abc", 34, 40, "nome" , 1, 1, True}
print(conjunto)
```

```
{1, 34, 40, 'nome', 'abc'}
```

Tamanho do conjunto

- A função **`len()`** é utilizada para obter o tamanho de um conjunto.

In [7]:
```
conjunto = {"apple", "banana", "cherry", "apple"}
print(len(conjunto))
```

```
3
```

Acessando itens

- Não é possível acessar itens consultando um índice ou uma chave.
- Mas você pode percorrer os itens usando um **`for`** ou verificar se um valor está presente em um conjunto usando a instrução **`in`**.

In [1]:
```
conjunto = {"apple", "banana", "cherry"}

for x in conjunto:
  print(x)

if "apple" in conjunto:
    print("OK")
```

```
cherry
banana
apple
OK
```

Inserindo itens

- Itens podem ser adicionados ao conjunto usando a função **`add()`**.
- Ao adicionar um item repetido, ele será ignorado.

In [12]:
```
conjunto = {"apple", "banana", "cherry"}
conjunto.add("orange")
conjunto.add("apple")
print(conjunto)
```

```
{'banana', 'cherry', 'orange', 'apple'}
```

Removendo itens

- Itens podem ser removidos utilizando a função **`remove()`** ou **`discard()`**.
  - Se o item a ser removido não existir, `remove()` gerará um erro.
  - Se o item a ser removido não existir, `discard()` NÃO gerará um erro.

In [22]:
```
nomes = {'Paulo', 'Ana', 'Pedro', 'Maria'}
nomes.remove('Ana')
nomes.discard('Maria')
print(nomes)
```

```
{'Paulo', 'Pedro'}
```

Preenchendo conjuntos com entrada do usuário

- Podemos preencher um conjunto com dados inseridos pelo usuário utilizando uma estrutura **`for`**.
- Para criar um conjunto vazio utilize a função **`set()`**

In [17]:
```
numeros = set()         # conjunto vazio
for i in range(5):
    n = int(input('Informe um Número: '))
    numeros.add(n)
print(numeros)
```

```
Informe um Número: 1
Informe um Número: 2
Informe um Número: 3
Informe um Número: 4
Informe um Número: 5
{1, 2, 3, 4, 5}
```

Unindo conjuntos:

- Existem várias maneiras de unir dois ou mais conjuntos em Python.
- O método **`union()`** retorna um novo conjunto contendo todos os itens de ambos os conjuntos.

In [23]:
```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set3 = set1.union(set2)

print(set3)
```

```
{1, 2, 3, 'c', 'a', 'b'}
```

Unindo conjuntos:

- O método **`intersection`** retorna um conjunto que contém os itens que existem em ambos os conjuntos.

In [19]:
```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.intersection(y)

print(z)
```

```
{'apple'}
```

Unindo conjuntos:

- O método **difference** retorna um conjunto que contém os itens que existem apenas em um conjunto e não existem no outro

In [20]:
```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.difference(y)

print(z)
```

```
{'banana', 'cherry'}
```

Função set

- A função `set` pode ser utilizada para copiar os itens de uma lista, tupla ou string para um conjunto.
  - Valores duplicados serão ignorados.

In [2]:
```
texto = 'bananeira'
conjunto = set(texto)
print(conjunto)

lista = ["apple", "banana", "cherry", "apple"]
conjunto = set(lista)
print(conjunto)
```

```
{'e', 'r', 'i', 'n', 'b', 'a'}
{'cherry', 'banana', 'apple'}
```