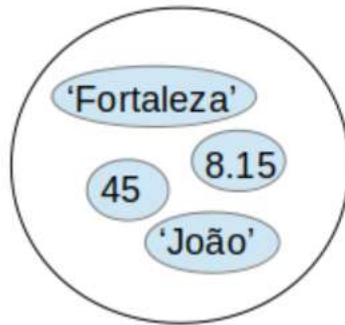


## Conjuntos em Python

Listas são coleções ordenadas (a ordem dos elementos na lista é significativa: o 1º elemento, o 2º elemento, etc), indexadas (cada elemento da lista é identificável por um índice) e que permitem a duplicação de elementos, a exemplo de:

'João'	45	8.15	'Fortaleza'	45
0	1	2	3	4

Conjuntos (sets), por outro lado, são coleções não ordenadas, não indexadas e sem duplicação de elementos, pode ser representada por algo como:



Sets (ou, como iremos chamar daqui para a frente, conjuntos) são estruturas disponíveis como *builtins* do Python, utilizadas para representar **coleções desordenadas de elementos únicos**.

É importante sempre lembrar dos conjuntos por suas duas principais características:

1. Os elementos não são armazenados em uma ordem específica e confiável;
2. Conjuntos não contém elementos repetidos.

A característica número 1 é importante, porque o desenvolvedor **jamais deve confiar na ordenação de um conjunto**, visto que a ordem em que os elementos são mantidos nos conjuntos varia de implementação para implementação do interpretador Python. Conjuntos não suportam indexação nem fatiamento (*slicing*).

## Criando um conjunto em Python:

```
Thonny - C:\Users\Luciana Rech\Documents\Luciana\Luciana UFSC\Ensino\2020 remoto\POO\CódigoFonte\Lista\Exemplo01_Set.py @ 12:9
File Edit View Run Device Tools Help
Exemplo01_Set.py
1 # Definindo um conjunto vazio por meio da função set()
2 c = set()
3
4 # Um elemento pode ser adicionado a um conjunto por meio do método add()
5 c.add(8.15)
6 c.add('João')
7 c.add(45)
8 c.add('Fortaleza')
9 c.add(45)
10
11 # Mostrar o conjunto
12 print(c)

Shell
>>> %cd 'C:\Users\Luciana Rech\Documents\Luciana\Luciana UFSC\Ensino\2020 remoto\POO\CódigoFonte\Lista'
>>> %Run Exemplo01_Set.py
>>> %Run Exemplo01_Set.py
{'Fortaleza', 'João', 8.15, 45}
>>> %Run Exemplo01_Set.py
{'João', 8.15, 45, 'Fortaleza'}
>>> %Run Exemplo01_Set.py
{8.15, 'Fortaleza', 45, 'João'}
>>>
```

## Também é possível criar um conjunto a partir de uma lista já existente.

```
Thonny - C:\Users\Luciana Rech\Documents\Luciana\Luciana UFSC\Ensino\2020 remoto\POO\CódigoFonte\Lista\Exemplo02_Set.py @ 9:1
File Edit View Run Device Tools Help
Exemplo02_Set.py
1 # Cria uma lista com valores repetidos
2 lista = [1001, 9834, 2, 1001, -43, 18, 2, 2, 9834]
3 print("Estrutura de Lista: ", lista)
4 # Cria um conjunto a partir da lista de valores, resultando na eliminação das repetições
5 x = set(lista)
6 print("Estrutura de conjunto: ", x)
7
8
9

Shell
Python 3.7.2 (bundled)
>>> %Run Exemplo02_Set.py
Estrutura de Lista: [1001, 9834, 2, 1001, -43, 18, 2, 2, 9834]
Estrutura de conjunto: {2, 1001, 9834, 18, -43}
>>> |
```

Algumas das funções e ações sobre conjuntos que são similares às vistas sobre listas, tuplas e strings:

- `len(c)` → número de elementos do conjunto (cardinalidade)
- `x in c` → verificar se um elemento **x** pertence ao conjunto **c**
- `x not in c` → verificar se um elemento **x** NÃO pertence ao conjunto **c**
- `for x in c:` → percorrer os elementos do conjunto **c**

### Pertinência

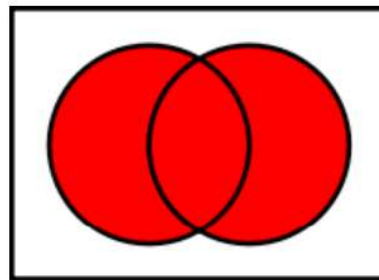
Além das operações tradicionais de união, interseção e diferença, também temos operações de verificação de pertinência. A seguir veremos algumas.

Para verificar se um determinado elemento pertence a um conjunto, podemos usar o já conhecido operador de pertinência `in`:

```
1 >>> a = {1, 2, 3, 4}
2 >>> b = {3, 4, 5, 6}
3 >>> 1 in a
4 True
5 >>> 5 in a
6 False
```

Operações me conjuntos:

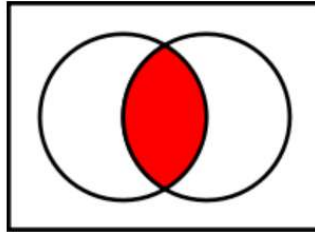
### União



$A \cup B$  (Crédito da imagem: [Wikipedia](#))

```
1 >>> a = {1, 2, 3, 4}
2 >>> b = {3, 4, 5, 6}
3 >>> print a.union(b)
4 set([1, 2, 3, 4, 5, 6])
```

## Interseção



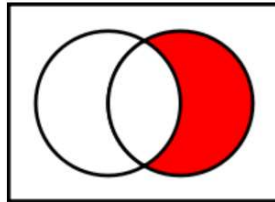
$A \cap B$  (Crédito da imagem: Wikipedia)

```
1 >>> print a.intersection(b)
2 set([3, 4])
```

Essa operação é muito útil quando precisamos descobrir elementos que duas listas possuem em comum:

```
1 >>> l1 = [1, 2, 3]
2 >>> l2 = [2, 4, 3]
3 >>> print set(l1).intersection(l2)
4 set([2, 3])
```

## Diferença



$B \setminus A$  (Crédito da imagem: Wikipedia)

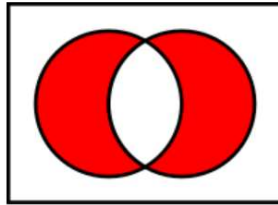
A diferença entre dois conjuntos A e B retorna somente os elementos de A que não estão em B, ou seja, retira de A todos os elementos comuns a ambos os conjuntos:

```
1 >>> a = {1, 2, 3, 4}
2 >>> b = {3, 4, 5, 6}
3 >>> print a.difference(b)
4 set([1, 2])
5 >>> print b.difference(a)
6 set([5, 6])
```

## Diferença simétrica

Diferença simétrica é uma operação sobre os dois conjuntos, que retorna todos os elementos (de ambos os conjuntos *a* e *b*) que pertencem a somente um dos conjuntos.

```
1 >>> a = {1, 2, 3, 4}
2 >>> b = {3, 4, 5, 6}
3 >>> print a.symmetric_difference(b)
4 set([1, 2, 5, 6])
```



$A \triangle B$  (Crédito da imagem: [Wikipedia](#))

Sintaxe	Descrição
<code>s.difference_update(t)</code>	Remove os itens que estão no conjunto <i>t</i> do conjunto <i>s</i> .
<code>s.intersection_update(t)</code>	Faz com que o conjunto <i>s</i> contenha a interseção dele com <i>t</i> .
<code>s.symmetric_difference_update(t)</code>	Faz com que o conjunto <i>s</i> contenha a diferença simétrica dele com <i>t</i> .
<code>s.isdisjoin(t)</code>	Retorna True se <i>s</i> e <i>t</i> não tem nenhum item em comum.
<code>s.issubset(t)</code> ( <i>s</i> <= <i>t</i> )	Retorna True se <i>s</i> é igual a ou um subconjunto de <i>t</i> .
<code>s.issuperset(t)</code> ( <i>s</i> >= <i>t</i> )	Retorna True se <i>s</i> é igual a ou <i>t</i> é um subconjunto de <i>s</i> .
<code>s.discard(x)</code>	Remove o item <i>x</i> do conjunto <i>s</i> .
<code>s.remove(x)</code>	Remove o item <i>x</i> se ele estiver em <i>s</i> se não retorna um <code>KeyError</code> .
<code>s.update(t)</code>	Adiciona cada item do conjunto <i>s</i> que não está no conjunto <i>t</i> para o conjunto <i>t</i> .

## Exercícios:

- 1) Considere um Centro de Treinamento Esportivo que oferece cursos de Futebol, Natação, Vôlei e Basquete. Em função do grande número de desistência de seus alunos, a administração resolveu oferecer um desconto de 50% na segunda modalidade.

Para isso eles precisarão verificar os alunos em comum (matriculados em mais de um esporte).

Você foi contratado para desenvolver um sistema que faça essa verificação e retorne as seguintes informações:

- 1- Mostre (imprimir na tela) a relação de alunos matriculados por modalidade: Futebol, Basquete, Natação e Vôlei.
- 2- Permita que novos alunos possam ser matriculados em qualquer uma das modalidades.
- 3- Faça a verificação se existem alunos matriculados em pelo menos 2 modalidades. E indique quais os alunos que possuem direito ao desconto.
- 4- Indique o número total de alunos por modalidade, e também o número total de alunos do Centro de Treinamento.

- 2) Considere que você foi contratado para desenvolver um sistema de integração entre 2 instituições financeiras (devido a uma fusão entre estas instituições). Em função do grande número de clientes, a administração resolveu oferecer algumas vantagens para clientes em comum (um desconto de 70% na tarifa de manutenção de contas correntes por exemplo), na expectativa que novos clientes sejam atraídos com a novidade.

Para isso eles precisarão verificar quais os clientes em comum (cadastrados tanto na empresa 1 como na empresa2).

Você foi contratado para desenvolver um sistema que faça essa verificação e retorne as seguintes informações:

- 5- Mostre (imprimir na tela) a relação de clientes cadastrados por empresa.
- 6- Permita que novos clientes possam ser cadastrados em qualquer uma das modalidades.
- 7- Faça a verificação se existem clientes já cadastrados nas 2 empresas. E indique quais são estes clientes.
- 8- Indique o número de clientes:
  - a. Da empresa 1
  - b. Da empresa 2
  - c. Clientes já pertencentes a ambas as empresas
  - d. Clientes pertencentes a apenas 1 das empresas
  - e. Número total de clientes

