Politécnico de Leiria

Escola Superior de Tecnologia e Gestão

Matemática Discreta - Componente PL

EI (D)

Ano letivo 2023/2024 - 2.º Sem. Ficha prática 3

Conjuntos e operações entre conjuntos

Conjuntos em Python

Em Python, um **conjunto** é uma coleção não ordenada de elementos que não se repetem, ou seja, que não podem surgir em duplicado. Sendo uma coleção não ordenada, não é armazenada posição de elementos ou ordem de inserção dos mesmos. Assim, num conjunto não é possível aceder a um elemento pela sua posição.

- É possível criar um conjunto de duas formas distintas: ou usando chavetas ({}) ou usando a função `set`.

Exemplo: Definição do conjunto $A = \{1, 2, 3\}$:

```
In [1]: A={1,2,3}
        print(A)
```

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

```
In [2]: A=set([1,2,3])
        print(A)
```

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

Exemplo: Definição do conjunto $Nomes = \{\text{João, Maria, Manuel}\}$:

```
In [3]: Nomes={"João","Maria","Manuel"}
        print(Nomes)
```

```
{'Manuel', 'João', 'Maria'}
```

```
In [4]: Nomes=set(["João","Maria","Manuel"])
        print(Nomes)
```

```
{'Manuel', 'João', 'Maria'}
```

Exemplo: Definição do conjunto $B = \{x \in \mathbb{N} : x \text{ é par e } x \leq 20\}$:

```
In [5]: B=set(range(2,21,2))
        print(B)
```

```
{2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
```

Exemplo: Definição do conjunto C, composto pelos carateres que a compõem a string "Matemática":

```
In [6]: C=set("Matemática")
        print(C)
```

```
{'m', 'e', 'M', 'á', 't', 'c', 'i', 'a'}
```

- Para criar um **conjunto vazio**, deve ser usada a função `set()`

- Na seguinte tabela encontram-se comandos úteis em Teoria de conjuntos:

Teoria de conjuntos	

`A&B`
ou
`A.intersection(B)`
	Efetua a interseção entre os conjuntos `A` e `B`
	Devolve o conjunto composto pelos elementos que pertencem a `A` **e** a `B`

`A|B`
ou
`A.union(B)`
	Efetua a união entre os conjuntos `A` e `B`
	Devolve o conjunto composto pelos elementos que pertencem a `A`, a `B` **ou** a ambos

`A-B`
ou
`A.difference(B)`
	Efetua a diferença entre o conjunto `A` e o `B`
	
	Devolve o conjunto composto pelos elementos que pertencem a `A` **mas não a** `B`

`A^B`	Efetua a diferença simétrica entre os conjuntos `A` e `B` Devolve o conjunto com os elementos que pertencem a `A` **ou a** `B`, mas **não a ambos**
`len(A)`	Devolve a cardinalidade de `A` Devolve o número de elementos de `A`
`A==B`	Testa se os conjuntos `A` e `B` são iguais
`A<B`	Testa se o conjunto `A` está contido em `B`
`x in A`	Testa se `x` pertence a `A`
`x not in A`	Testa se `x` não pertence a `A`
`A.add(elem)`	Adiciona `elem` ao conjunto `A`
`A.remove(elem)`	Remove `elem` do conjunto `A` (se `A` não contiver o `elem`, devolve KeyError)
`A.discard(elem)`	Remove `elem` do conjunto `A`, se este conjunto contiver o `elem`
`A.clear()`	Remove todos os elementos do conjunto

Exemplos:

```
In [7]: A=set([1,2,3,4,5])
        B=set([3,5,7,8,9])
        C=set([0,3,6,9])

In [8]: B|C
Out[8]: {0, 3, 5, 6, 7, 8, 9}

In [9]: B&C
Out[9]: {3, 9}

In [10]: B-C
Out[10]: {5, 7, 8}

In [11]: A^B
Out[11]: {1, 2, 4, 7, 8, 9}

In [12]: len(A)
Out[12]: 5
```

Exemplos:

```
In [13]: 0 in C
Out[13]: True

In [14]: 5 not in A
Out[14]: False

In [15]: A.add(10)
         print(A)
         {1, 2, 3, 4, 5, 10}

In [16]: A.remove(3)
         print(A)
         {1, 2, 4, 5, 10}

In [17]: A.discard(12)
         print(A)
         {1, 2, 4, 5, 10}
```

Exemplos:

```
In [18]: F={c for c in 'Discreta' if c not in 'Matemática'}
         print(F)
         {'r', 's', 'D'}

In [19]: F.discard('D')
         print(F)
         {'r', 's'}
```

Listas em Python

Em Python, uma **lista** é uma coleção ordenada de objetos. A posição de cada elemento da lista é clara dado que todos os seus itens são indexados.

- É possível criar uma lista usando os parêntesis retos ([]) ou usando a função `list`.
  Exemplo: Criação da lista $L = [1, 2, 3, 4]$:

```
In [20]: L=[1,2,3,4]
         print(L)
         [1, 2, 3, 4]
```

- Uma lista é uma sequência de elementos, que podem ou não ser do mesmo tipo.
  Exemplo: Criação da lista $L =$ ["Maria", $1, [2, 3]$] :

```
In [21]: L=["Maria", 1,[2,3]]
         print(L)
         ['Maria', 1, [2, 3]]
```

- **Exemplo:** Criação da lista $L = [10, 20, 30, 40]$ de duas formas distintas:

```
In [22]: L=[10,20,30,40]
         print(L)

         [10, 20, 30, 40]

In [23]: L=list(range(10,50,10))
         print(L)

         [10, 20, 30, 40]
```

Nota: de recordar que o 50 não será incluído na lista.

Esta estrutura vai ser útil na definição de conjuntos de conjuntos; para tal, será criada uma lista de conjuntos.

- **Exemplo:** Definição do conjunto de conjuntos $P = \{\{1, 2\}, \{3\}, \{7, 8, 9\}\}$:

```
In [24]: P=[{1,2},{3},{7,8,9}]
         print(P)

         [{1, 2}, {3}, {8, 9, 7}]

In [25]: P=[set([1,2]),set([3]),set([7,8,9])]
         print(P)

         [{1, 2}, {3}, {8, 9, 7}]
```

- Na tabela seguinte são apresentados os principais comandos associados a listas:

Comandos a usar em listas	
`L=list()` ou `L=[]`	Cria a lista vazia `L`
`L[i]`	Devolve o `(i+1)`-ésimo item da lista `L`
`L[i]=x`	Substitui o `(i+1)`-ésimo item da lista `L` por `x`
`L[i:j]`	Devolve a sublista de `L`, do `(i+1)`-ésimo item até ao `(j)`-ésimo
`del L[i]`	Remove o item que se encontra na `(i+1)` posição da lista `L`
`L.clear()`	Remove todos os itens da lista `L`
`L.pop(i)`	Remove o item da `(i+1)`-ésima posição da lista `L` e retorna-o. Se não for
	especificado índice, `L.pop()` remove e devolve o último item da lista `L`
`len(L)`	Número de itens da lista `L`
`L.count(x)`	Devolve o número de vezes que o item `x` aparece na lista `L`
`L.append(L2)`	Adiciona a lista `L2` como item da lista `L`
`L.insert(i,x)`	Insere o item `x` na `(i+1)`-ésima posição da lista `L`
`L.remove(x)`	Remove o primeiro item encontrado na lista cujo valor é igual a `x`
`L.index(x)`	Devolve o índice do primeiro item encontrado na lista cujo valor é igual a `x`
`L.sort()`	Ordena por ordem crescente os itens da lista `L`
`L.sort(reverse=True)`	Ordena por ordem decrescente os itens da lista `L`
`L2=sorted(L)`	Cria a lista `L2` definindo-a como uma cópia ordenada da lista `L`
`list(string)`	Devolve a lista constituída pelos carateres que compõem a `string`
`L=list([string1,string2])` `' '.join(L)`	Devolve a string que resulta da concatenação dos itens (em formato string) de `L`
`L1+L2`	Concatena as listas `L1` e `L2`

- **Exemplos:**

```
In [26]: L=[3,6,8,9,3,6]
         L[0]
```

```
Out[26]: 3
```

```
In [27]: L[2:4]
```

```
Out[27]: [8, 9]
```

```
In [28]: L[3]=10
         print(L)
```

```
[3, 6, 8, 10, 3, 6]
```

```
In [29]: del L[1]
         print(L)
```

```
[3, 8, 10, 3, 6]
```

```
In [30]: L.count(3)
```

```
Out[30]: 2
```

```
In [32]: L.append([7,9])
         print(L)
```

```
[3, 8, 10, 3, 6, 1, [7, 9]]
```

```
In [33]: L.insert(2,0)
         print(L)
```

```
[3, 8, 0, 10, 3, 6, 1, [7, 9]]
```

```
In [34]: [7,9] in L
```

```
Out[34]: True
```

```
In [35]: L1=[{1,2},{3,5,6},{11}]
         L2=[{4},{8,9}]
         L1+L2
```

```
Out[35]: [{1, 2}, {3, 5, 6}, {11}, {4}, {8, 9}]
```

```
In [36]: L1.pop(1)
```

```
Out[36]: {3, 5, 6}
```

```
In [37]: print(L1)
```

```
[{1, 2}, {11}]
```

Exercícios propostos

1. Considere os conjuntos $A = \{1, 2, 3, 4\}$, $B = \{3, 4, 5, 6, 7\}$ e $C = \{1, 5, 9, 10\}$.
   (a) Determine o universo $\mathcal{U}$ sabendo que $\mathcal{U} = A \cup B \cup C$.
   (b) Determine a cardinalidade de $\mathcal{U}$, $n(\mathcal{U})$.
   (c) Determine o complementar de A, $\overline{A}$.
   (d) Determine a diferença de A por B, $A - B$.
   (e) Determine a diferença simétrica de A e B, $A \oplus B$.

2. Considere os conjuntos $A = \{0, 1, 2, 4, 8\}$, $B = \{3, 5\}$ e $C = \{4, 6, 8\}$. Crie um script em Python que determine:
   (a) a cardinalidade do conjunto $A \cap C$.
   (b) o conjunto $A \oplus (B \cup C)$.

3. Construa uma função, designada por `simdiff`, que receba dois conjuntos A e B e devolva a sua diferença simétrica, sem fazer uso de `A^B`.

4. Considere os conjuntos $A = \{0, 2, 5, 8, 9\}$ e $B = \{2, 3, 5\}$ definidos no universo $\mathcal{U} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Crie um script em Python que:
   (a) determine a cardinalidade do conjunto $\overline{A}$.
   (b) determine o conjunto $(A \oplus B) \cap \mathcal{U}$.

5. Considere os conjuntos $A = \{x : x \text{ é múltiplo de 3, com } 0 < x \leq 100\}$, $B = \{x : x \text{ é divisível por 7, com } 66 \leq x \leq 140\}$ e $C = A \oplus B$. Crie um script em Python que determine:
   (a) o conjunto B por extensão.
   (b) a cardinalidade do conjunto $(A \cup C) \cap B$.

6. A Catarina tem 410 cartões numerados de 1 a 410. Crie um script em Python que:
   (a) determine quantos desses cartões têm um número par que não é múltiplo de 7.
   (b) determine quais desses cartões têm um número que é um quadrado perfeito. Nota: recorde que q é um quadrado perfeito se existir um $n \in \mathbb{N}$ tal que $n^2 = q$.
   (c) determine quais desses cartões têm um número que é múltiplo de 3 mas não é um quadrado perfeito.

7. Considere a lista de preferências indicada por um grupo de pessoas em determinado questionário:

   Lista_Atividades=["Futebol", "Ioga", "Cinema", "Futebol", "Concertos", "Cinema", "Concertos"].

   Crie um script em Python que construa o conjunto das atividades elencadas na lista anterior.

8. Considere o conjunto $B = \{1, 2, 3, 4\}$. Crie um script em Python que permita verificar se o conjunto $P = \{\{1\}, \{2\}, \{3, 4\}\}$ é uma partição de B.

9. Crie um script em Python que:
   - verifique se um determinado conjunto P, introduzido inicialmente, pode ou não ser uma partição de um dado conjunto A;
   - em caso afirmativo, indique qual será o conjunto A.