

class2

December 11, 2021

1 Introdução a Python

1.1 Aula 2

2 Sumário

- Operações lógicas
- Operações de pertença
- Fluxos de controlo
- Scripts e comentários
- Funções
- Exercícios

3 Operações lógicas

and: Devolve **True** se ambas as expressões forem verdadeiras

```
[1]: x = 3  
  
x < 10 and x < 5
```

```
[1]: True
```

or: Devolve **True** se alguma das expressões for verdadeira

```
[2]: x = 5  
  
x == 8 or x == 4
```

```
[2]: False
```

not: Reverte o resultado

```
[3]: x = 12  
  
not(x > 5 and x > 10)
```

[3]: False

4 Operações de pertença

`in`: Retorna **True** se a sequência com o valor especificado está presente no objeto

```
[4]: x = "ab"
     y = "abcd"

     x in y
```

[4]: True

`not in`: Retorna **True** se a sequência com o valor especificado não está presente no objeto

```
[5]: x = "ab"
     y = "cd"

     x not in y
```

[5]: True

5 Fluxos de controlo

5.1 If... else

`if`: Permite executar um conjunto de instruções caso a condição em questão se verifique

```
[6]: x = 2
     y = 4

     if y > x:
         print("y é maior do que x") #atenção à indentação
```

y é maior do que x

`elif`: Permite executar um conjunto de instruções caso a condição em questão se verifique e todas as anteriores não

```
[7]: x = 4
     y = 2

     if y > x:
         print("y é maior do que x")
     elif y < x:
```

```
print("y é menor do que x")
```

y é menor do que x

else: Permite capturar tudo o que não tenha sido capturado pelas condições anteriores

```
[8]: x = 5
     y = 5

     if y > x:
         print("y é maior do que x")
     elif y < x:
         print("y é menor do que x")
     else:
         print("y é igual a x")
```

y é igual a x

5.2 Ciclos while

while: Permite executar um conjunto de instruções enquanto a condição for verdadeira

```
[9]: i = 0

     while i<6:
         i+=1

     i
```

[9]: 6

break: Permite terminar o ciclo mesmo que a condição seja verdadeira

```
[10]: i = 0

      while i < 5:
          i += 1
          if i == 3:
              break
          print(i)
```

1

2

continue: Utiliza-se quando queremos passar uma certa iteração à frente

```
[11]: i = 0

      while i < 5:
```

```
i += 1
if i == 3:
    continue
print(i)
```

```
1
2
4
5
```

5.3 Ciclos for

for: Permite executar um conjunto de instruções para cada item de uma lista, de um tuplo, etc.

```
[12]: for x in "banana":
      print(x)
```

```
b
a
n
a
n
a
```

6 Scripts e comentários

Para correr um script:

```
python nome_do_script.py
```

Para correr um script em modo interativo:

```
python -i nome_do_script.py
```

7 Funções

def: Utiliza-se para definir uma função

```
[13]: def my_function(): #sem argumento
      print("Hello world!")

      my_function() #chamada da função
```

```
Hello world!
```

```
[2]: def check_parity(i): #com argumento
      if i%2==0:
          print(f"Number {i} is even.")
      else:
          print(f"Number {i} is odd.")

      check_parity(8)
```

Number 8 is even.

```
[4]: def favourite_color(color = "pink"): #argumento por omissão
      print(f"My favourite color is {color}.")

      favourite_color()
```

My favourite color is pink.

```
[7]: def children_names(*args): #número indefinido de argumentos
      if len(args) == 0:
          print("There are no children!")
      else:
          print("The children names are:")
          for name in args:
              print(f"- {name}")

      children_names("Maria", "John", "Frank")
```

The children names are:

- Maria
- John
- Frank

```
[1]: def get_age(**kwargs): #número indefinido de argumentos com nome
      for key, value in kwargs.items():
          print(f"{key}: {value}")

      get_age(name = "Peter", age = 30)
```

name: Peter

age: 30

return: Utiliza-se para a função devolver um resultado

```
[18]: def double(i):
        return i*2

      double(2)
```

[18]: 4

```
[10]: def factorial(i): #função recursiva
      if i == 0:
          return 1
      else:
          return i * factorial(i-1)

      factorial(5)
```

[10]: 120

pass: Utiliza-se para definir funções sem conteúdo

```
[20]: def empty_function():
      pass
```

lambda: função anónima que pode receber qualquer número de argumentos, mas só pode ter uma expressão

```
[21]: x = lambda a: a+10

      x(5)
```

[21]: 15

Algumas funções pré-definidas relevantes: - abs() - len() - range() - etc.

```
[22]: abs(-7)
```

[22]: 7

```
[23]: len("abcde")
```

[23]: 5

```
[13]: for i in range(10):
      print(i)
```

0
1
2
3
4
5
6
7
8
9

8 Exercícios

- 1) Escreva uma função que verifica se uma string contém o caractere “a”.
- 2) Dados dois números inteiros, escreva uma função que retorna o maior.
- 3) Dado o raio de uma circunferência, escreva uma função que retorna o seu perímetro.
- 4) Escreva uma função que retorna um quadrado de 5 por 5 cardinais (“#”).
- 5) Dados dois números inteiros, escreva uma função que retorna o seu produto caso este seja maior do que 1000, caso contrário retorna a sua soma.
- 6) Dados os termos a , b e c de uma equação de segundo grau, escreva uma função que retorna o resultado da aplicação da fórmula resolvente.
- 7) Dado um número natural, escreva uma função que imprime os seus divisores. Tenha em conta que os divisores de um número natural n são os inteiros $k \leq n$ tais que o resto da divisão de n por k é 0.
- 8) Dado um número natural, escreva uma função que calcule o termo de Fibonnaci para esse número.
- 9) Escreva uma função que percorre os números inteiros de 1 até 50. Para os múltiplos de 3 imprime “Fizz”, para os múltiplos de 5 imprime “Buzz” e para os múltiplos de 3 e 5 simultaneamente imprime “FizzBuzz”.
- 10) Escreva uma função que verifica a velocidade dos condutores: se a velocidade for menor que 70 deverá imprimir “ok”, por cada 5km acima do limite de velocidade (70) deverá implicar um furo na carta e imprimir o número de furos, se o condutor tiver mais de 12 furos na carta deverá imprimir “carta de condução suspensa”.