Python 3

- print("texto") - imprime para o ecrã

Variáveis

```
character_name = "Tom"
character_age = "50"
is_male = True
is-female = False
print("There once was a man named " + character_name + ", ")
```

→ só é possível concatenar strings a strings

} booleanos

↗ concatenação de strings

```
#comment
```

↗ comentário (só existe single line)

```
character_name = "Mike"
print("He really liked the name " + character_name + ",  ")
print("but he didn't like being " + character_age + ".")
```

Strings

Escape Sequences

- \n - nova linha

```
print("Giraffe\nAcademy")
```

- \" - aspas dentro de strings (evitando "fecho" da string)

```
print("Giraffe \" Academy\"")
```

- \\ - caracter \

```
print("Giraffe\\Academy")
```

- string variable

```
phrase = "New Giraffe Academy"
print(phrase)
```

- concatenação de strings

```
print(phrase + " is cool")
```

Funções Aplicadas a Strings

- sintaxe: aplicação de funções e métodos em objetos

```
print(phrase.lower())
```

↙ converte em lower case

↳ todas as funções têm ( ), mesmo não tendo argumentos

```
print(phrase.upper())
```

↳ converte em upper case

-
  ```
  print(phrase.isupper())
  False
  ```

  ↳ verifica se a string é upper case

  ↳ devolve um bool

- funções podem ser aplicadas em sequência

```
print(phrase.upper().isupper())
True
```

↳ aplica upper à string phrase ①

→ verifica se a string phrase é upper case ②

- função comprimento

```
print(len(phrase))
19
```

recebe um argumento e devolve o seu comprimento

- índice numa string

```
print(phrase[5])
i
```

↳ começa no índice zero

- função índice - recebe um argumento e devolve o seu índice na string onde foi aplicada. Em caso de repetição devolve a primeira ocorrência

```
print(phrase.index('A'))
12
```

→ recebe o caracter A como argumento e devolve a sua posição na string phrase

↳ string onde a função index é aplicada

↳ posição onde o caracter A se encontra na string

- - no caso de ter recebido uma string, devolve a posição de onde ela começa

```
print(phrase.index("Acad"))
12
```

- - - se o parâmetro não estiver contido na string é devolvido um erro

- função replace

```
print(phrase.replace("New", "Old"))
Old Giraffe Academy
```

→ não muda o estado de memória da string onde foi aplicada

Valores Numéricos

- resto da divisão inteira

```
print(10 % 3)
1
```

- tal como noutras linguagens, operações entre inteiros e fracionários levam à conversão do inteiro em fracionário

- conversão de valores numéricos em caracteres: útil para os concatenar a strings e os poder imprimir

```
print(str(my-num)
5
print("O valor é " + str(my-num) + ".")
O valor é 5.
```

→ caso contrário causaria um erro porque a função print não pode receber valores numéricos como argumento, apenas as suas representações, no formato de caracter

Funções Matemáticas mais Comuns

- abs - valor absoluto (módulo)

```
my-num = -5
print(abs(my_num))
5
```

- pow - potência

```
print(pow(my-num, 2)
25
print(pow(3, 3)
27
```

→ $(-5)^2$

→ 3^3

- max - máximo: devolve o maior valor (aplicável a caracteres, em função dos valores ASCII)

```
print(max(4, 6))
6
print( max(max(4,6), 9)
9
```

- min - mínimo

```
print(min(4, -3))
-3
```

- round - arredondamento

```
print(round(3.2))
3
print(round(3.7))
4
```

- floor - arredonda para baixo (math module)

```
from math import *
```

→ importar o module math (topo do ficheiro)

```
print(floor(3.7))
3
```

- ceil - arredonda para cima (math module)

```
print(ceil(4.7))
5
```

- sqrt - raiz quadrada (math module)

```
print(sqrt(16))
4
```

Input

- input("prompt")

```
<variavel> = input("prompt")
```

- por defeito, o input vai ser convertido numa string e a soma de valores numéricos resulta na concatenação dessas strings. São necessárias as funções str e (int) para contornar essa situação

```
num1 = int(input("Enter a number :")
num2 = int(input("Enter another number : ")
result = num1 + num2
print(result)
print(str(num1) + " + " + str(num2) + " = " + str(result))
```

} recebe os inputs e converte-os para int (também existe float)

↳ só faz sentido fazer isto se quiser manter os valores guardados em formato numérico

↙ volta a converter em string

↙ concatena

- é possível pedir inputs sequenciais

```
print("Insert a color, a plural noun and a celebrity")
(color, plural_noun, celebrity) = input(), input(), input()
print("Roses are " + color)
print(plural_noun + " are blue")
print("I love " + celebrity)
```