



## **Tipos de dados**

**Números, Strings, Booleanos e Listas**

**Variáveis**



# Tipos de dados



# Tipos de dados

Exemplo	Tipo de dados
<code>x = "Hello World"</code>	<code>str</code>
<code>x = 20</code>	<code>int</code>
<code>x = 20.5</code>	<code>float</code>
<code>x = 1j</code>	<code>complex</code>
<code>x = ["apple", "banana", "cherry"]</code>	<code>list</code>
<code>x = ("apple", "banana", "cherry")</code>	<code>tuple</code>
<code>x = range(6)</code>	<code>range</code>
<code>x = {"name" : "John", "age" : 36}</code>	<code>dict</code>
<code>x = {"apple", "banana", "cherry"}</code>	<code>set</code>
<code>x = frozenset({"apple", "banana", "cherry"})</code>	<code>frozenset</code>
<code>x = True</code>	<code>bool</code>
<code>x = b"Hello"</code>	<code>bytes</code>
<code>x = bytearray(5)</code>	<code>bytearray</code>
<code>x = memoryview(bytes(5))</code>	<code>memoryview</code>
<code>x = None</code>	<code>NoneType</code>



# Números



# Números

- Existem três tipos de números em Python:

Designação	Sintaxe	Exemplo
Inteiros	int	x=1
Fracionários	float	y=2.3
Complexos	complex	z=4j

- O Python não tem uma função random() para gerar um número aleatório, mas contém um módulo built-in chamado random para esse fim:

```
import random  
print(random.randrange(1, 10))
```



# Strings



# Strings

- Strings são linhas de texto que podem conter quaisquer caracteres.
- As strings em Python são colocadas entre aspas simples ou duplas. 'Olá' é o mesmo que "Olá".
- Uma string literal pode ser exibida com a função `print()`:

```
print("Olá Python!")  
print('Olá Python!')
```



# Booleanos



# Booleanos

- Os booleanos representam um valor True (verdadeiro) ou False (falso).
- É importante observar que a primeira letra é sempre maiúscula.
- Estes representam dados que só podem ser uma coisa ou outra. Por exemplo:

```
estaRegistado = True # Utilizado numa base de dados de utilizadores  
estaVivo = False # Utilizado num jogo, quando o jogador morre
```



# Listas



# Listas

- As listas são usadas para agrupar dados.
- São conhecidas por arrays em quase todas as outras linguagens.

```
vaziaLista = []
numerosLista = [1, 2, 3]
stringsLista = ["arroz", "feijão"]
mistList = ["Olá", [1, 2, 3], False]
```

- As listas podem conter quaisquer tipos de dados, incluindo outras listas ou nada.
- Os elementos das listas podem ser acedidos recorrendo aos índices de lista:

```
numerosLista[1] #retorna 2
stringLista[0] #retorna arroz
mixedList[1][2] #retorna 3
```



# Variáveis



# Variáveis

- **Sintaxe:**

```
nome = valor
```

- As variáveis em Python podem ter qualquer nome (exceto algumas palavras-chave) e os seus valores podem ser de qualquer tipo de dado.
- Podem mesmo mudar de tipo de dados após a atribuição.



# Variáveis - Exemplos

```
x = 5
y = "Maria"
print(x)
print(y)

x = 4      # x é do tipo int
x = "Sally" # x é agora do tipo str
print(x)
```



# Variáveis

- Quando é necessário especificar o tipo de dado utiliza-se o casting: nome = tipo(valor)
- O tipo de dado de qualquer variável pode ser obtido através da função type: print(type(nome))

```
x = str(3)      # x será '3'  
y = int(3)      # y será 3  
z = float(3)    # z será 3.0  
  
x = 5  
y = "Maria"  
print(type(x))  # retorna <class 'int'>  
print(type(y))  # retorna <class 'str'>
```



# Variáveis - Exemplos

- Outras propriedades:

```
x = "Maria"  
# é o mesmo que  
x = 'Maria'  
  
a = 4  
A = "Sally"  
# a e A são duas variáveis diferentes
```



# Variáveis

- Podem ainda ser atribuídos múltiplos valores, ou um único valor, a várias variáveis numa única linha.

```
x, y, z = "Maria", "José", "Marta"
```

```
print(x)
```

```
print(y)
```

```
print(z)
```

```
x = y = z = "Maria"
```

```
print(x)
```

```
print(y)
```

```
print(z)
```



# Variáveis

- Se houver uma coleção de valores numa lista, tuplo, etc., o Python permite a sua extração para variáveis. A esta operação chama-se *Unpacking* (descompactação).

```
frutas = ["maçã", "banana", "cereja"]
x, y, z = frutas
print(x)
print(y)
print(z)
```



# Variáveis - Output

- Na função print(), o output de múltiplas variáveis pode ser visualizado utilizando a vírgula ou o operador soma.

```
x = "O Python"  
y = "é"  
z = "fantástico"  
  
print(x, y, z)
```

```
x = "O Python "  
y = "é "  
z = "fantástico"  
  
print(x+y+z)
```

- No entanto repare no espaço depois de “Python “ e de ”é “. Sem isso o resultado seria ”O Pythonfantástico”



# Variáveis - Output

- Na função print(), a combinação de strings com números não é permitida com o operador soma:

```
x = 5  
y = "Maria"  
print(x + y)
```



- A melhor forma de o fazer é utilizar a vírgula como operador:

```
x = 5  
y = "Maria"  
print(x, y)
```





# Variáveis globais

- As variáveis que são criadas fora de uma função chamam-se variáveis globais.
- Estas variáveis podem ser utilizadas em qualquer zona do programa, tanto dentro como fora das funções:

```
x = "fantástico"

def mfuncao():
    print("O Python é " + x)

mfuncao()
```



# Variáveis globais

- Se for criada uma variável com o mesmo nome dentro de uma função, esta variável será local, e só poderá ser utilizada dentro da função.
- A variável global com o mesmo nome permanecerá como estava, global e com o valor original:

```
x = "fantástico"

def mfuncao():
    x = "incrível"
    print("O Python é" , x)

mfuncao()

print("O Python é", x)
```



# Variáveis globais

- Para alterar o valor de uma variável global dentro de uma função, declare a variável usando a palavra-chave global.
- Ou seja, use a palavra-chave global se quiser alterar uma variável global dentro de uma função.

```
x = "fantástico"

def mfuncao():
    global x
    x = "incrível"
    print("O Python é", x)

mfuncao()

print("O Python é", x)
```



# Exercícios

1) Crie um programa que peça ao utilizador para introduzir o seu nome e a sua idade. O programa deverá calcular em que ano o utilizador fará 100 anos.  
Nota: neste exercício, a indicação do ano corrente é explícita.

Output:

```
Indique o seu nome: Maria  
Que idade tem: 25  
Maria, terá 100 anos em 2098
```