
Desenvolvimento para a Internet e Aplicações Móveis

2022/23

Python 3

Introdução



ISCTE  Instituto Universitário de Lisboa
Gabinete de Comunicação e Imagem

Gabinete de Comunicação e Imagem
ISCTE  Instituto Universitário de Lisboa

Python 3



<https://www.python.org>

Linguagem de programação:

- orientada para objectos
- interpretada
- ideal para scripting e desenvolvimento rápido para web

Instalação

Faça o download da última versão de Python 3 a partir de <https://www.python.org> . Escolha um ficheiro de instalação executável. Instale Python 3. Durante a instalação escolha a opção para instalar Python no path do sistema operativo.

Java vs Python

| Característica | Java | Python |
|----------------------|---|--|
| Compilação | Compilada | Interpretada |
| Estática ou Dinâmica | Definição específica dos tipos de dados | Definição dinâmica dos tipos de dados |
| Herança | Herança múltipla é parcialmente feita através de interfaces | Admite herança múltipla |
| Blocos de código | Definidos por chavetas | Definidos por indentação |
| Rapidez | - | + |
| Portabilidade | + exige JVM | - exige interpretador |
| Arquitetura | JVM converte bytecode em linguagem máquina | Interpretador converte código em linguagem máquina |

Exemplos de diferenças em código

Java

```
public class Print5 {  
    public static void main(String[] args) {  
        System.out.println("3+2=" +  
                            (Integer.toString(3+2)));  
    }  
}
```

Python

```
def main():  
    print('3+2=', 3+2)  
  
if __name__ == "__main__":  
    main()
```

Java

```
class PrintNumber {
    int left;
    int right;
    PrintNumber(int left, int right) {
        this.left = left;
        this.right = right;
    }
    public int getleft() {
        return left;
    }
    public int getRight() {
        return right;
    }
}

public class Print5 {
    public static void main(String[] args) {
        PrintNumber printNumber = new PrintNumber (3,2);
        String sum =
            Integer.toString(printNumber.getleft()
                + printNumber.getRight() );
        System.out.println("3+2=" + sum);
    }
}
```

Python

```
class Number:
    def __init__(self, left, right):
        self.left = left
        self.right = right

number = Number(3, 2)
print("3+2=", number.left + number.right)
```

Instruções básicas

Abra um projeto no PyCharm, crie um novo ficheiro Python e teste as seguintes instruções. Verifique a indentação.

```
o_mundo_e_plano = True
if o_mundo_e_plano:
    print("Cuidado para nao cair !")
```

```
print(5+8)
```

```
print(40 - 30*2)
```

```
print((40 - 30*2) / 2)
```

```
print(8 / 5)
```

```
print(int(8 / 5))
```

```
print (float(2 + 2))
```

```
print (17 / 3)
```

```
print (17 // 3) # divisão inteira
```

```
print (17 % 3)  # resto da divisão
```

```
print (5 ** 2)  # potência
```

Comentários:

```
# ...
```

```
"""
```

```
...
```

```
...
```

```
"""
```

```
'''
```

```
...
```

```
...
```

```
'''
```

Variáveis:

- são case-sensitive
- devem começar com uma letra ou underscore
- não podem começar com um número
- podem apenas conter caracteres alfanuméricos (A-z, 0-9, e _)

```
y = "Quantos alunos há no ISCTE?"  
x = 10000  
print (y)  
print (x)
```

```
s = "esta é uma string"  
s = 10000  
print (s)  
print (str(s) + " é um int ")
```

```
x = str(3)  
print(x)  
print(type(x))  
print()
```

```
y = int(3)
print(y)
print(type(y))
print()
z = float(3)
print(z)
print(type(z))
```

```
x = "ISCTE-IUL"
x = 'ISCTE-IUL'
print(x)
```

```
x, y, z = "ISCTE", "IST", "ISEG"
print(x)
print(y)
print(z)
```

```
x = y = z = "ISCTE"
print(x)
print(y)
print(z)
```

```
institutos = ["ISCTE", "IST", "ISEG"]
x, y, z = institutos
```

```
print(x)
print(y)
print(z)
```

```
x = "Estar no ISCTE "
y = "e genial"
z = x + y
print(z)
```

```
x = 5
y = 10
print(x + y)
```

```
x = "O ISCTE tem "
y = 10
z = " mil alunos"
print(x + str(y) + z)
```

```
x = "Lisboa"

def myfunc():
    x = "ISCTE"
    print("Dentro de myfunc() estou no "
          + x)

myfunc()

print("Fora de myfunc() estou em " + x)
```

```
x = "Lisboa"

def myfunc():
    global x
    x = "ISCTE"
    print("Dentro de myfunc() estou no "
          + x)

myfunc()

print("Fora de myfunc() estou em " + x)
```

Tipos de dados

| | |
|-----------------|---|
| Text Type: | <code>str</code> |
| Numeric Types: | <code>int</code> , <code>float</code> , <code>complex</code> |
| Sequence Types: | <code>list</code> , <code>tuple</code> , <code>range</code> |
| Mapping Type: | <code>dict</code> |
| Set Types: | <code>set</code> , <code>frozenset</code> |
| Boolean Type: | <code>bool</code> |
| Binary Types: | <code>bytes</code> , <code>bytearray</code> , <code>memoryview</code> |

| Example | Data Type |
|---|-------------------------|
| <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> |

Construtores

(também podem ser utilizados para casting, ou conversão de tipos)

| Example | Data Type |
|---|-------------------------|
| <code>x = str("Hello World")</code> | <code>str</code> |
| <code>x = int(20)</code> | <code>int</code> |
| <code>x = float(20.5)</code> | <code>float</code> |
| <code>x = complex(1j)</code> | <code>complex</code> |
| <code>x = list(("apple", "banana", "cherry"))</code> | <code>list</code> |
| <code>x = tuple(("apple", "banana", "cherry"))</code> | <code>tuple</code> |
| <code>x = range(6)</code> | <code>range</code> |
| <code>x = dict(name="John", age=36)</code> | <code>dict</code> |
| <code>x = set(("apple", "banana", "cherry"))</code> | <code>set</code> |
| <code>x = frozenset(("apple", "banana", "cherry"))</code> | <code>frozenset</code> |
| <code>x = bool(5)</code> | <code>bool</code> |
| <code>x = bytes(5)</code> | <code>bytes</code> |
| <code>x = bytearray(5)</code> | <code>bytearray</code> |
| <code>x = memoryview(bytes(5))</code> | <code>memoryview</code> |

Números aleatórios

(mais em https://www.w3schools.com/python/module_random.asp)

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

Strings

```
x = "ISCTE-IUL"
x = 'ISCTE-IUL'
print(x)
```

```
a = '''Tudo vale a pena quando a alma não
é pequena.
Fernando Pessoa'''
print(a)
```

```
a = "Tudo vale a pena"  
print(a[1])
```

```
for x in "Tudo vale a pena":  
    print(x)
```

```
a = "Tudo vale a pena"  
print(len(a))
```

```
txt = "Tudo vale a pena"
```

```
print("pena" in txt)
```

```
print("alma" in txt)
```

```
if "pena" in txt:
```

```
    print("pena existe na frase")
```

```
if "alma" not in txt:
```

```
    print("alma não existe na frase")
```

```
word = 'Python'

print(word[0])
# 'P'

print (word[5])
# 'n'

print (word[-1])
# 'n'

print (word[-2])
# 'o'

print (word[-6])
# 'P'

print (word[0:2]) # de 0 (incluído) a 2 (excluído)
# 'Py'

print (word[2:5])
# 'tho'

print (word[:2])
# 'Py'

print (word[4:])
# 'on'

print (word[-2:])
# 'on'
```

```
a = "Tudo vale a pena"

print(a.upper())

print(a.lower())

print(a.replace("Tudo", "Nada"))

print(a.split(", "))

b = "quando a alma não é pequena."
c = a + " " + b
print(c)
```

Impressão de alguns caracteres especiais

| Code | Result |
|------|-----------------|
| \' | Single Quote |
| \\ | Backslash |
| \n | New Line |
| \r | Carriage Return |
| \t | Tab |

Métodos para strings

Explorar

https://www.w3schools.com/python/python_strings_methods.asp

Booleans

```
a = 198
```

```
b = 33
```

```
if b > a:
```

```
    print("b é maior que a")
```

```
else:
```

```
    print("b não é maior que a")
```

```
print(bool("alma"))
```

```
print(bool(123))
```

```
print(bool(["Tudo", "vale", "a", "pena"]))
```

```
print(bool(True))
```

```
print(bool(""))
```

```
print(bool(0))
```

```
print(bool([]))
```

```
print(bool(False))
```

```
print(bool(None))
```

```
print(bool(()))
```

```
x = 200
print(isinstance(x, int))
print(isinstance(x, str))
```

Operações aritméticas

| Operator | Name | Example |
|----------|----------------|---------|
| + | Addition | x + y |
| - | Subtraction | x - y |
| * | Multiplication | x * y |
| / | Division | x / y |
| % | Modulus | x % y |
| ** | Exponentiation | x ** y |
| // | Floor division | x // y |

Operações de atribuição

| Operator | Example | Same As |
|----------|---------|-----------|
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| *= | x *= 3 | x = x * 3 |
| /= | x /= 3 | x = x / 3 |

Operações de comparação

| Operator | Name | Example |
|----------|--------------------------|---------|
| == | Equal | x == y |
| != | Not equal | x != y |
| > | Greater than | x > y |
| < | Less than | x < y |
| >= | Greater than or equal to | x >= y |
| <= | Less than or equal to | x <= y |

Operações lógicas

| Operator | Description | Example |
|----------|---|--|
| and | Returns True if both statements are true | <code>x < 5 and x < 10</code> |
| or | Returns True if one of the statements is true | <code>x < 5 or x < 4</code> |
| not | Reverse the result, returns False if the result is true | <code>not(x < 5 and x < 10)</code> |

Operações de identidade

| Operator | Description | Example |
|----------|--|-------------------------|
| is | Returns True if both variables are the same object | <code>x is y</code> |
| is not | Returns True if both variables are not the same object | <code>x is not y</code> |

Operações de pertença

| Operator | Description | Example |
|----------|--|------------|
| in | Returns True if a sequence with the specified value is present in the object | x in y |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y |

Lista completa de operadores em

https://www.w3schools.com/python/python_operators.asp

Colecções

- Lista
- Tuplo
- Conjunto
- Dicionário

Listas

| Method | Description |
|------------------|--|
| <u>append()</u> | Adds an element at the end of the list |
| <u>clear()</u> | Removes all the elements from the list |
| <u>copy()</u> | Returns a copy of the list |
| <u>count()</u> | Returns the number of elements with the specified value |
| <u>extend()</u> | Add the elements of a list (or any iterable), to the end of the current list |
| <u>index()</u> | Returns the index of the first element with the specified value |
| <u>insert()</u> | Adds an element at the specified position |
| <u>pop()</u> | Removes the element at the specified position |
| <u>remove()</u> | Removes the item with the specified value |
| <u>reverse()</u> | Reverses the order of the list |
| <u>sort()</u> | Sorts the list |

```
quadrados = [1, 4, 9, 16, 25]
print (quadrados)
# [1, 4, 9, 16, 25]
```

```
print (quadrados[0])
# 1
print (quadrados[-1])
# 25
print (quadrados[-3:])
# [9, 16, 25]
print (quadrados[:])
# [1, 4, 9, 16, 25]
print (quadrados + [36, 49, 64, 81, 100])
# [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
cubos = [1, 8, 27, 65, 125]
print (4 ** 3)
# 64
cubos[3] = 64
print (cubos)
# [1, 8, 27, 64, 125]
```

```
cubos.append(216)
cubos.append(7 ** 3)
print (cubos)
# [1, 8, 27, 64, 125, 216, 343]
```

```
letras = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print (letras)
# ['a', 'b', 'c', 'd', 'e', 'f', 'g']
letras[2:5] = ['C', 'D', 'E']
print (letras)
# ['a', 'b', 'C', 'D', 'E', 'f', 'g']
```

```
letras[2:5] = []
print (letras)
# ['a', 'b', 'f', 'g']
letras[:] = []
print (letras)
# []
```

```
letras = ['a', 'b', 'c', 'd']
```

```
print (len(letras))
```

```
# 4
```

```
a = ['a', 'b', 'c']
```

```
n = [1, 2, 3]
```

```
x = [a, n]
```

```
print (x)
```

```
# [['a', 'b', 'c'], [1, 2, 3]]
```

```
print (x[0])
```

```
# ['a', 'b', 'c']
```

```
print (x[0][1])
```

```
# 'b'
```

```
umaLista = ["Ana", 22, True, "feminino"]
```

```
print(type(umaLista))
```

```
print(umaLista[1])
```

```
print(umaLista[-1])
```

```
print(umaLista[1:3])
```

```
print(umaLista[-2:-1])
```

```
umaLista[0] = "Rodrigo"
```

```
umaLista[3] = "masculino"
```

```
print(umaLista)
umaLista.append(16)
print(umaLista)
umaLista.insert(1, "ISCTE")
print(umaLista)
UCs = ["DIAM", "IP", "POO"]
umaLista.extend(UCs)
print(umaLista)
umaLista.remove("POO")
print(umaLista)
umaLista.pop(1)
print(umaLista)
del umaLista[0]
print(umaLista)
del umaLista
print(umaLista)
umaLista = ["Ana", 22, True, "feminino"]
umaLista.clear()
print(umaLista)
for x in umaLista:
    print(x)
for i in range(len(umaLista)):
    print(umaLista[i])
i = 0
```

```
while i < len(umaLista):  
    print(umaLista[i])  
    i = i + 1
```

```
umaLista = ["José", "Ana", "Raquel",  
            "António", "Filipa"]  
umaLista.sort()  
print(umaLista)  
umaLista.sort(key = str.lower)  
print(umaLista)
```

```
umaLista = [33, 22, 2, 11, 8787]  
umaLista.sort()  
umaLista.sort(reverse = True)  
print(umaLista)
```

```
umaLista = [33, 22, 2, 11, 8787]  
umaLista.reverse()  
print(umaLista)
```

```
umaLista = [33, 22, 2, 11, 8787]  
outraLista = umaLista.copy()
```

```
print(outraLista)
aindaOutraLista = list(umaLista)
print(aindaOutraLista)
```

```
umaLista = ["José", "Ana", "Raquel",
"Antônio", "Filipa"]
outraLista = [33, 22, 2, 11, 8787]
duasListas = umaLista + outraLista
```

Tuplos

Conjunto de valores separados por virgulas, ordenados, que não podem ser alterados e que permitem duplicados. Pode conter vários tipos de dados.

```
t = tuple(("ISCTE", "ISEG", "IST"))  
print(t)
```

```
t = 12345, 54321, 'Olá!'  
print (t)  
print (t[0])
```

```
u = "t", (1, 2, 3, 4, 5)  
print (u)  
u[0] = 88888
```

TypeError: 'tuple' object does not support item assignment

Como uma string, um duplo não pode ser alterado.
Como já vimos, as listas podem ser alteradas.

Tuplo com um só elemento

```
# este é um tuplo:  
umTuplo = ("ISCTE",)  
print(type(umTuplo))
```

```
# este não é um tuplo:  
outroTuplo = ("ISEG")  
print(type(outroTuplo))
```

```
ensinoSup = ("ISCTE", "ISEG", "IST", "UL",  
             "UNL", "UC", "UC", "UP")  
print(ensinoSup[1])  
print(ensinoSup[-1])  
print(ensinoSup[2:5])  
print(ensinoSup[:4])
```

```
print(ensinoSup[2:])
print(ensinoSup[-4:-1])
if "ISCTE" in ensinoSup:
    print("O ISCTE está no ensinoSup")
```

Um tuplo não pode ser alterado, mas uma lista sim...

```
x = ("ISCTE", "ISEG", "IST")
print(x)
y = list(x)
y[0] = "ISCTE-IUL"
x = tuple(y)
print(x)
```

```
y = list(x)
y.append("UL")
x = tuple(y)
print(x)
```

```
y = list(x)
y.remove("ISEG")
x = tuple(y)
```

```
print(x)
```

```
del x
```

```
print(x)
```

```
# NameError: name 'x' is not defined
```

```
ensinoSup = ("ISCTE", "ISEG", "IST")
```

```
for x in ensinoSup:
```

```
    print(x)
```

```
i = 0
```

```
while i < len(ensinoSup):
```

```
    print(ensinoSup[i])
```

```
    i = i + 1
```

Métodos específicos para tuplos

| Method | Description |
|----------------|---|
| <u>count()</u> | Returns the number of times a specified value occurs in a tuple |
| <u>index()</u> | Searches the tuple for a specified value and returns the position of where it was found |

Conjunto

Colecção desordenada e sem elementos duplicados. Elementos não podem ser alterados. Admite vários tipos no mesmo conjunto. Admite adicionar elementos.

```
cesto = {'maçã', 'laranja', 'maçã',  
'pêra', 'laranja', 'banana'}  
# ou  
cesto = set(('maçã', 'laranja', 'maçã',  
'pêra', 'laranja', 'banana'))  
  
print(cesto)    # não há duplicados  
  
print ('laranja' in cesto)  
# True  
  
print ('cenoura' in cesto)  
# False  
  
print(len(cesto))  
print(type(cesto))
```

```
for x in cesto:
    print(x)

print("maçã" in cesto)

cesto.add("uvas")
print(cesto)

outroCesto = {"ananás", "manga"}
cesto.update(outroCesto)
print(cesto)

cesto.remove("maçã")
print(cesto)
cesto.discard("ananás")
print(cesto)
# se elemento não existe, remove dá erro
# mas discard não dá erro

x = cesto.pop()
print(x)

cesto.clear()
```

```
print(cesto)
```

```
del cesto
```

```
print(cesto)
```

| Method | Description |
|--------------------------------------|--|
| <u>add()</u> | Adds an element to the set |
| <u>clear()</u> | Removes all the elements from the set |
| <u>copy()</u> | Returns a copy of the set |
| <u>difference()</u> | Returns a set containing the difference between two or more sets |
| <u>difference_update()</u> | Removes the items in this set that are also included in another, specified set |
| <u>discard()</u> | Remove the specified item |
| <u>intersection()</u> | Returns a set, that is the intersection of two other sets |
| <u>intersection_update()</u> | Removes the items in this set that are not present in other, specified set(s) |
| <u>isdisjoint()</u> | Returns whether two sets have a intersection or not |
| <u>issubset()</u> | Returns whether another set contains this set or not |
| <u>issuperset()</u> | Returns whether this set contains another set or not |
| <u>pop()</u> | Removes an element from the set |
| <u>remove()</u> | Removes the specified element |
| <u>symmetric_difference()</u> | Returns a set with the symmetric differences of two sets |
| <u>symmetric_difference_update()</u> | inserts the symmetric differences from this set and another |
| <u>union()</u> | Return a set containing the union of sets |
| <u>update()</u> | Update the set with the union of this set and others |

Dicionário

Dicionários são compostos por pares chave-valor e são indexados pelas chaves. São ordenados, elementos podem ser alterados e não admitem repetidos. Admitem vários tipos de dados em simultâneo.

```
tel = {  
    'João': 213404098,  
    'Sofia': 213404139  
}  
  
print (tel)  
# {'Rui': 213404127, 'João': 213404098,  
  'Sofia': 213404139}  
  
print(len(tel))  
print(type(tel))  
  
print (tel['João'])  
# 213404098  
  
tel['João'] = 213404127 # muda um valor  
print (tel['João'])
```

```
print (tel.get('João'))

del tel['Sofia']
tel['Inês'] = 213404127
print (tel)
# {'Rui': 213404127, 'João': 213404098,
# 'Inês': 213404127}

print (list(tel.keys()))
# ['Rui', 'João', 'Inês']

print (list(tel.values()))
# ['213404127', '213404098', '213404127']

x = tel.items()

print (sorted(tel.keys()))
# ['Inês', 'João', 'Rui']

print ('Rui' in tel)
# True

print ('João' not in tel)
```

False

```
if "João" in tel:  
    print("Sim tenho o número do João")
```

```
tel["Raquel"] = "913456789"  
tel.update({"Filipe": "8767665"})  
print(tel)
```

```
tel.pop("Raquel")  
print(tel)
```

```
tel.popitem()  
print(tel)
```

```
del tel["Inês"]  
print(tel)
```

```
tel.clear()  
print(tel)
```

```
tel.update({"Filipe": "8767665",  
"Adriana": "8764543"})
```

```
for x in tel:
    print(x + " : " + tel[x])

for x in tel.values():
    print(x)

for x in tel.keys():
    print(x)

for x, y in tel.items():
    print(x, y)

# dicionários embutidos
tel.update({"Leopoldina":      {"Portátil":
"912345678", "Casa": "213456789"}})
print(tel)
```

| Method | Description |
|---------------------|---|
| <u>clear()</u> | Removes all the elements from the dictionary |
| <u>copy()</u> | Returns a copy of the dictionary |
| <u>fromkeys()</u> | Returns a dictionary with the specified keys and value |
| <u>get()</u> | Returns the value of the specified key |
| <u>items()</u> | Returns a list containing a tuple for each key value pair |
| <u>keys()</u> | Returns a list containing the dictionary's keys |
| <u>pop()</u> | Removes the element with the specified key |
| <u>popitem()</u> | Removes the last inserted key-value pair |
| <u>setdefault()</u> | Returns the value of the specified key. If the key does not exist: insert the key, with the specified value |
| <u>update()</u> | Updates the dictionary with the specified key-value pairs |
| <u>values()</u> | Returns a list of all the values in the dictionary |

Estruturas de controlo

If

```
x = int(input("Introduza um inteiro: "))
if x < 0:
    x = 0
    print('Inteiro negativo alterado para
zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Um dígito')
else:
    print('Outro')
```

Pode ser usada apenas uma linha de código:

```
a = 2000
b = 200
if a > b: print("a é maior que b")

a = 330
```

```
b = 330
print("A") if a > b else print("=") if a
== b else print("B")
```

```
a = 200
b = 33
c = 500
if a > b and c > a:
    print("Ambas as condições são True")
if a > b or a > c:
    print("Pelo menos uma das condições é
True")
```

```
x = 41
if x > 10:
    print("Above ten,")
    if x > 20:
        print("and also above 20!")
    else:
        print("but not above 20.")
```

```
a = 33
b = 200
```

```
if b > a:
    pass    # quando por alguma razão temos que ter um if
            # mas não queremos executar instruções no seu
            # corpo
```

While

```
a = 1
while a < 10:
    if a == 6:
        break
    print(a)
    a += 1
else:
    print("a já chegou a 10")
```

For

For itera ao longo dos itens de uma sequência, que pode ser uma lista ou uma string.

```
frase = ['Quem', 'ri', 'por', 'último',
         'ri', 'melhor']
```

```
for w in frase:
    print(w, len(w))

for x in "ISCTE":
    print(x)

for w in frase:
    print(w)
    if w == "último":
        break

for i in range(5):
    print(i)

for x in range(2, 30, 3):
    print(x)

for x in range(10):
    if x == 4:
        continue
    if x == 7: break
    print(x)
else:
    print("Cheguei ao fim!")
```

```
universidades = ["UL", "UP", "UNL"]
institutos = ["ISCTE", "ISEG", "ICC"]
for x in universidades:
    for y in institutos:
        print(x, y)
```

For pode iterar ao longo de um número:

```
a = ['Ter', 'a', 'consciência', 'limpa',
     'é', 'ter', 'a', 'memória', 'fraca']

for i in range(len(a)):
    print(i, a[i])
```

```
personagens = {'Zé': 'do Telhado',
               'Maria': 'da Fonte'}
```

```
for k, v in personagens.items():  
    print(k, v)
```

```
for i, v in enumerate(['tic', 'tac',  
    'toc']):  
    print(i, v)
```

Funções

Função que calcula a serie de Fibonacci até um dado numero:

```
def fib(n) :  
    """Imprime a Série de Fibonacci até  
    ao número n."""  
    a, b = 0, 1  
    while a < n:  
        print(a, ' ' )  
        a, b = b, a+b  
        print()
```

n é um argumento. Não se declara o tipo do argumento.

Execução da função:

```
fib(2000)
```

ou:

```
f = fib
```

```
f(100)
```

Retorno de uma função:

```
def fib(n):  
    result = []  
    a, b = 0, 1  
    while a < n:  
        result.append(a)  
        a, b = b, a+b  
    return result
```

```
f100 = fib(100)  
print (f100)
```

Valores por defeito para argumentos de funções

```
def tudo_fixe(prompt, retries=4,  
              complaint='ou Sim ou Não !'):  
    while True:  
        print ('Está tudo fixe?')  
        ok = input(prompt)  
        if ok in ('sim', 'Sim', 'SIM',  
                  'OK', 'ok', 'ya'):  
            return True  
        if ok in ('não', 'Não', 'NÃO',  
                  'nem por isso'):  
            return False  
        retries = retries - 1  
        if retries < 0:  
            raise OSError('O utilizador não  
                           nos liga !!')  
        print (complaint + '\n')  
  
tudo_fixe('')
```

Os argumentos com valores por defeito podem ser ignorados ou invocados; são chamados *Keyword Arguments*, ou *kwargs*

Argumentos arbitrários

Uma função pode ser declarada com um valor arbitrário de argumentos.

Teste a seguinte função:

```
def concat(*args, sep="/") :  
    return sep.join(args)  
  
print (concat("Terra", "Marte", "Vénus"))
```

Qual é o resultado?

Módulos

Módulo: ficheiro .py com definição de funções e com instruções executáveis Python.

Utilização de um módulo:

```
import nomeDoModulo
```

ou

```
from nomeDoModulo import funcao
```

Execução a partir da linha de comando
o modulo deve ter a função main:

```
if __name__ == "__main__":  
    import sys  
    funcao(int(sys.argv[1]))
```

o que permite executar na linha de comando:

```
$ python3 nomeDoModulo <argumentos>
```

Bibliografia e links úteis

“The Python Language Reference
<https://docs.python.org/3/reference/index.html>

Python » English 3.8.2rc1 Documentation » Quick search Go | previous | next | modules | index

Previous topic
5. Editors and IDEs

Next topic
1. Introduction

This Page
Report a Bug
Show Source


The Python Language Reference

This reference manual describes the syntax and “core semantics” of the language. It is terse, but attempts to be exact and complete. The semantics of non-essential built-in object types and of the built-in functions and modules are described in [The Python Standard Library](#). For an informal introduction to the language, see [The Python Tutorial](#). For C or C++ programmers, two additional manuals exist: [Extending and Embedding the Python Interpreter](#) describes the high-level picture of how to write a Python extension module, and the [Python/C API Reference Manual](#) describes the interfaces available to C/C++ programmers in detail.

- [1. Introduction](#)
 - [1.1. Alternate Implementations](#)
 - [1.2. Notation](#)
- [2. Lexical analysis](#)
 - [2.1. Line structure](#)
 - [2.2. Other tokens](#)
 - [2.3. Identifiers and keywords](#)
 - [2.4. Literals](#)
 - [2.5. Operators](#)
 - [2.6. Delimiters](#)
- [3. Data model](#)
 - [3.1. Objects, values and types](#)
 - [3.2. The standard type hierarchy](#)
 - [3.3. Special method names](#)
 - [3.4. Coroutines](#)

“W3Schools Python Tutorial”

<https://www.w3schools.com/python/default.asp>

THE WORLD'S LARGEST WEB DEVELOPERS

[Home](#) [HTML](#) [CSS](#) [JAVASCRIPT](#) [SQL](#) [MORE ▾](#) [REFERENCES ▾](#) [EXERCISES ▾](#)

Python Tutorial

[Python HOME](#)
[Python Intro](#)
[Python Get Started](#)
[Python Syntax](#)
[Python Comments](#)
[Python Variables](#)
[Python Data Types](#)
[Python Numbers](#)
[Python Casting](#)
[Python Strings](#)
[Python Booleans](#)
[Python Operators](#)
[Python Lists](#)
[Python Tuples](#)
[Python Sets](#)
[Python Dictionaries](#)
[Python If...Else](#)
[Python While Loops](#)
[Python For Loops](#)
[Python Functions](#)
[Python Lambda](#)
[Python Arrays](#)
[Python Classes/Objects](#)



Condições válidas para todas as ofertas:
• Exclusivo novas adesões online entre 13 e 20 de fevereiro 2020. • Perda do direito a oferta no caso de incumprimento ou denúncia do Acordo de Utilização do Cartão de Crédito Wizink.
• Válido para compras ou adiantamentos de numerário a crédito*, no montante total mínimo de 300€, nos 2 primeiros meses após aprovação.
** O cash advance está sujeito a comissões, conforme preço em vigor.

Python Tutorial

[◀ Home](#)[Next ▶](#)

Python is a programming language.

Python can be used on a server to create web applications.

[Start learning Python now »](#)

Learning by Examples

With our "Try it Yourself" editor, you can edit the code and view the result.

Example

Screenshot