

UNIVERSIDADE FEDERAL  
DE UBERLÂNDIA

# MINICURSO DE PYTHON3



# HISTÓRIA

- Criada por Guido Van Rossum;
- É uma linguagem de programação interpretada, de script, imperativa, orientada a objetos, funcional;
- Foi pensada para ser uma linguagem fácil e intuitiva, tão inteligível quanto inglês.



# VERSÕES



PYTHON2

```
>>> print "Hello World!"  
>>> 3/2 = 1
```

PYTHON3

```
>>> print ("Hello World!")  
>>> 3/2 = 1.5
```

# INSTALAÇÃO DO PYTHON

Windows:

<https://python.org.br/instalacao-windows/>

Linux:

<https://python.org.br/instalacao-linux/>

# PARA O MINICURSO:

Para o mini curso usaremos a seguinte IDE online:

<https://www.programiz.com/python-programming/online-compiler/>

Para estudo:

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

Para exercícios:

<https://www.urionlinejudge.com.br/judge/pt/problems/index/1>

# VARIÁVEIS E TIPOS DE DADOS

- Possui tipagem dinâmica;

```
main.py
1  variavel = 1
2  print(type(variavel))
3  variavel = "texto"
4  print(type(variavel))
5  variavel = 1.56
6  print(type(variavel))
7  variavel = 4+5j
8  print(type(variavel))
9  variavel = False
10 print(type(variavel))
11
```

```
<class 'int'>
<class 'str'>
<class 'float'>
<class 'complex'>
<class 'bool'>
```

# FUNÇÕES BÁSICAS

main.py

```
1  variavel = input("Digite variavel: ")
2  print(variavel)
3  type(variavel)
4  variavel = int(2.6) #variavel vai ser 2
5  print(variavel)
6  variavel = float(2) #variavel vai ser 2.0
7  print(variavel)
8  variavel = str(2.6) #variavel vai ser 2.6
9  print(variavel)
```

Digite variavel: 34

34

2

2.0

2.6

✖

OPERADORES



# OPERADORES: BÁSICOS

```
main.py
1  num1 = 3
2  num2 = 2
3
4  #Basicos:
5  print(num1 + num2)
6  print(num1 - num2)
7  print(num1 * num2)
8  print(num1 / num2)
```

5  
1  
6  
1.5  
❏

# OPERADORES: ARITMÉTICOS

main.py

```
1  num1 = 3
2  num2 = 2
3
4  #Aritimeticos:
5  print(num1 % num2)  #Modulo
6  print(num1 ** num2) #exponencial
7  print(num1 // num2) #divisão inteira
```

1

9

1

0



# OPERADORES: ATRIBUIÇÃO

main.py

5

```
1 #Atribuição:
2 num1 = 3 #Atribuindo o numero 3 para
    a variavel num1
3 num2 = 2
4 num1 += num2
5 print(num1)
```

main.py

6

```
1 #Atribuição:
2 num1 = 3 #Atribuindo o numero 3 para
    a variavel num1
3 num2 = 2
4 num1 *= num2
5 print(num1)
```

main.py

1

```
1 #Atribuição:
2 num1 = 3 #Atribuindo o numero 3 para
    a variavel num1
3 num2 = 2
4 num1 -= num2
5 print(num1)
```

main.py

9

```
1 #Atribuição:
2 num1 = 3 #Atribuindo o numero 3 para
    a variavel num1
3 num2 = 2
4 num1 **= num2
5 print(num1)
```

# OPERADORES: COMPARAÇÃO

main.py

```
1  num1 = 3
2  num2 = 2
3
4  #Comparação:
5  print(num1==num2) #é igual?
6  print(num1!=num2) #é diferente?
7  print(num1>num2)  #é maior?
8  print(num1<num2)  #é menor?
9  print(num1<=num2) #é menor ou igual?
10 print(num1>=num2) #é maior ou igual?
```

False

True

True

False

False

True



# OPERADORES: LÓGICOS

```
main.py
1 print("Tabela do E")
2 print(True and True)
3 print(True and False)
4 print(False and True)
5 print(False and False)
```

Tabela do E

```
True
False
False
False
```

```
main.py
1 print("Tabela do OU")
2 print(True or True)
3 print(True or False)
4 print(False or True)
5 print(False or False)
6
```

Tabela do OU

```
True
True
True
False
```

```
main.py
1 print("Tabela do NÃO")
2 print(not True)
3 print(not False)
4
```

Tabela do NÃO

```
False
True
```

# OPERADORES: PERTENCIMENTO

```
main.py
1  x = ["Maça", "banana", "jabuticaba"]
2  y = ["Maça", "banana", "jabuticaba"]
3
4  z = y
5
6  print("x é y? " + str(x is y))
7  print("z é y? " + str(z is y))
```

x é y? False  
z é y? True

Existe também o **'is not'**, que funciona como: `not(x is y) == x is not y`

HORA DE EXERCITAR

FIM DA PRIMEIRA AULA



STRINGS

# STRING: O QUE É...

```
main.py
1  #String é uma cadeia de caracteres
2  #cercado por aspas duplas ou simples
3
4  print("Herou")
5  print('Herou')
```

Herou  
Herou  
❏

```
main.py
1  #uma variavel pode receber uma string
2  a = 'Herou'
3  print(a)
```

Herou  
❏

# STRING: MÉTODOS

main.py

```
1  #Método strip
2  a = " Cabeça, ombro, joelho e pé "
3  print(a + '.')
4  print(a.strip() + '.') # Observe q os espaços no
   começo e no fim sumiram
```

Cabeça, ombro, joelho e pé .  
Cabeça, ombro, joelho e pé.



main.py

```
1  #Método lower/uper
2  a = " Cabeça, Ombro, Joelho e Pé "
3  print(a)
4  print(a.lower()) # tudo minusculo
5  print(a.upper())
```

Cabeça, Ombro, Joelho e Pé  
cabeça, ombro, joelho e pé  
CABEÇA, OMBRO, JOELHO E PÉ



# STRING: MÉTODOS...

main.py

```
1  #Método replace
2  a = " Cabeça, Ombro, Joelho e Pé "
3  print(a)
4  print(a.replace("o", 'u'))#troca dos letra 'o' por 'u'
5  #Observe q a letra 'O' nao
   foi trocada
```

```
Cabeça, Ombro, Joelho e Pé
Cabeça, Ombro, Juelhu e Pé
>
```

main.py

```
1  #Método split
2  a = " Cabeça; Ombro; Joelho e Pé "
3  print(a)
4  print(a.split(';'))#retorna um vetor
```

```
Cabeça; Ombro; Joelho e Pé
[' Cabeça', ' Ombro', ' Joelho e Pé ']
>
```

# STRING: OPERADOR IN/NOT IN

main.py

```
1  #Operador 'in'
2  a = "Cabeça, Ombro, Joelho e Pé"
3  print(a)
4  print('Pé' in a)#tem 'Pé' na frase a cima?
5  print('elho' in a)
6  print(['omb' in a])
```

```
Cabeça, Ombro, Joelho e Pé
True
True
False
➤ []
```

main.py

```
1  #Operador 'not in'
2  a = "Cabeça, Ombro, Joelho e Pé"
3  print(a)
4  print('Pé' not in a)# não tem 'Pé' na frase a cima?
5  print('elho' not in a)
6  print('omb' not in a)
```

```
Cabeça, Ombro, Joelho e Pé
False
False
True
➤ []
```

# OPERADORES CONDICIONAIS

# OPERADORES CONDICIONAIS - IF E ELSE

main.py

```
1 if True:
2     print('é verdade')
3
4 if False:
5     print("é falso")
6
7 variavel = 5
8 if variavel > 10:
9     print('é maior que 10')
10
11 if variavel < 10:
12     print('é menor que 10')
13
14 if variavel == 10:
15     print('é igual que 10')
16
17 print("fim")
```

```
é verdade
é menor que 10
fim
```

main.py

```
1 if True:
2     print('é verdade')
3 else:
4     print('é falso')
5
6 print("\nSegunda parte:\n")
7 if False:
8     print('é verdade')
9 else:
10    print('é falso')
11
12 print("fim")
```

é verdade

\Segunda parte:

```
é falso
fim
```

# OPERADORES CONDICIONAIS - ELIF

main.py

```
1 a = 100
2 b = 20
3
4 if a > b:
5     print('a é maior q b')
6
7 if b < a:
8     print('b é menor q a')
9
10 else:
11     print('a é menor ou igual a b')
```

a é maior q b  
b é menor q a



main.py

```
1 a = 100
2 b = 20
3
4 if a > b:
5     print('a é maior q b')
6
7 elif b < a:
8     print('b é menor q a')
9
10 else:
11     print('a é menor ou igual a b')
```

a é maior q b





# OPERADORES CONDICIONAIS - E/OU/NOT

main.py

```
1 #and
2 if True and True:
3     print('Verdade e Verdade')
4 if False and True:
5     print('Falso e Verdade')
6 if True and False:
7     print('Verdade e Falso')
8 if False and False:
9     print('Falso e Falso')
```

Verdade e Verdade



main.py

```
1 #or
2 if True or True:
3     print('Verdade ou Verdade')
4 if False or True:
5     print('Falso ou Verdade')
6 if True or False:
7     print('Verdade ou Falso')
8 if False or False:
9     print('Falso ou Falso')
```

Verdade ou Verdade  
Falso ou Verdade  
Verdade ou Falso



main.py

```
1 #not
2 if not True:
3     print('Não é verdade')
4 if not False:
5     print('não é falso')
```

não é falso



LOOPS

# LOOP - WHILE

main.py

```
1  #while loop
2  i = 1
3  while i < 6:
4      print(i)
5      i += 1
```

1

2

3

4

5

❌

# LOOP - FOR RANGE

```
main.py
1  #for range 2 loop
2  for x in range(2,10,2):
3      if x == 3:
4          break
5      print(x)
```

2  
4  
6  
8

```
main.py
1  #for rage loop
2  for x in range(6):
3      print(x)
```

0  
1  
2  
3  
4  
5

# LOOP - FOR IN

main.py

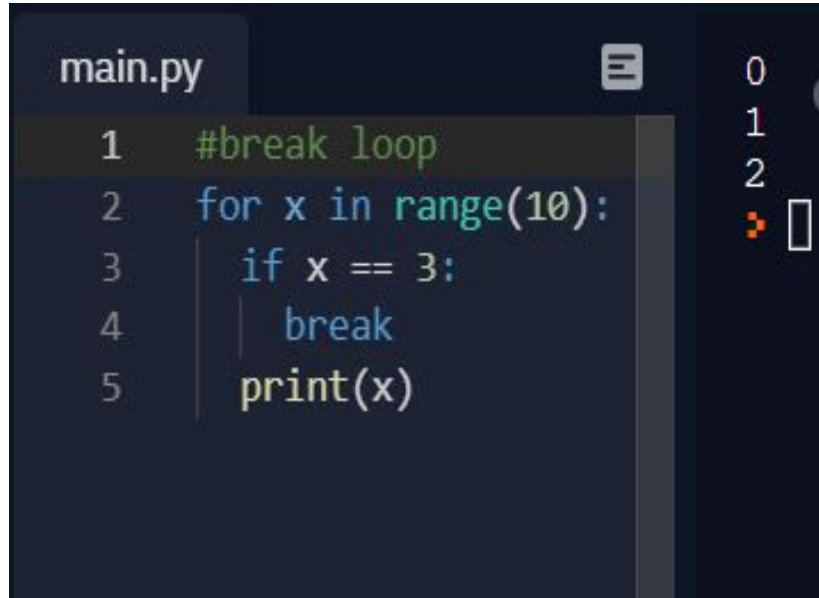
```
1  #for each loop
2  fruits = ["apple", "banana", "cherry"]
3  for x in fruits:
4      print(x)
```

apple  
banana  
cherry

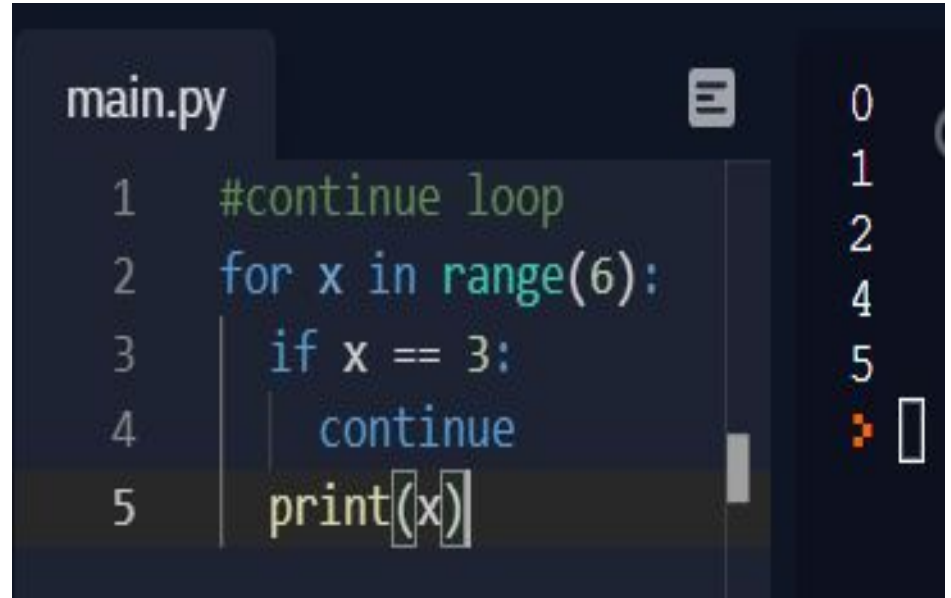


# LOOP - BREAK E CONTINUE

```
main.py
1  #break loop
2  for x in range(10):
3      if x == 3:
4          break
5      print(x)
```



```
main.py
1  #continue loop
2  for x in range(6):
3      if x == 3:
4          continue
5      print(x)
```



FUNÇÕES

# FUNÇÕES - TEORIA

- Indentação
- Exemplo:

```
def nome_minha_funcao() :  
    print("oi")
```

- Parâmetros
- Parâmetros Padrão (DEFAULT)
- return ( 1 ou múltiplos parâmetros)



# FUNÇÕES - PRÁTICA

```
main.py 4
1  #função
2  def soma(num1,num2):
3      num3 = num1+num2
4      return num3
5
6  print(soma(1,3))
```

```
main.py 4
1  #função ex2
2  def ola():
3      print("ola")
4
5  ola()
```

# FUNÇÕES - PRÁTICA

```
main.py 1
2
3
4
5
6
7
8
9
10
1 #função ex3
2 def recurção(i):
3     if i > 10:
4         return
5     print(i)
6     i = i+1
7     recurção(i)
8
9
10 recurção(1)
```

HORA DE EXERCITAR

FIM DA SEGUNDA AULA

LISTAS []

# LISTAS [] - TEORIA

- Lista é identificada por colchete []
- É um tipo de dado **ordenado** e **mutável**.
- Acessar itens (os itens são sequenciais, por isso ordenado, verificar se item está na lista)
- Mudar valores (por isso é mutável)

# LISTA - MÉTODOS

main.py

```
1  #listas
2  minhaLista = ["Abacate", "mamão"]
3
4  print("MinhaLista = " + str(minhaLista))
5  print("primeira posição de MinhaLista => " +minhaLista[0])
6  print("tamanho da MinhaLista = " +str(len(minhaLista)))
```

MinhaLista = ['Abacate', 'mamão']  
primeira posição de MinhaLista => Abacate  
tamanho da MinhaLista = 2

➤

main.py

```
1  #listas
2  minhaLista = ["Abacate", "mamão"]
3
4  #append adiciona um novo elemento no final da lista
5  minhaLista.append("Jabuticaba")
6  minhaLista.append("Mangericão")
7  minhaLista.append("Abacate")
8
9  print("MinhaLista = " + str(minhaLista))
10
11 #remove e pop removem elementos da lista
12 minhaLista.remove("Abacate") #o remove a partir do elemento
13 #observe q o remove tirou apenas a primeira ocorrência de abacate
14
15 minhaLista.pop(0) #o pop a partir da posição
16
17 print("MinhaLista = " + str(minhaLista))
18 print("primeira posição de MinhaLista => " +minhaLista[0])
```

MinhaLista = ['Abacate', 'mamão', 'Jabuticaba', 'Mangericão', 'Abacate']  
MinhaLista = ['Jabuticaba', 'Mangericão', 'Abacate']  
primeira posição de MinhaLista => Jabuticaba

➤

TUPLES()



# TUPLAS() - TEORIA

- É um tipo de dado **ordenado** e **imutável**.
  - Não tem como mudar os itens da tupla;
  - Não tem como remover os itens de uma tupla;
- Acessar itens, verificar se pertence.

# TUPLAS() - MÉTODOS

main.py

```
1  #tuplas
2  umaTupla = ("Abacate", "Mamão", "Pera")
3  print(umaTupla)
4
5  print(umaTupla[0])
6
7  #ver o ultimo item da tupla
8  print(umaTupla[-1])
9
10 print(len(umaTupla))
```

```
('Abacate', 'Mamão', 'Pera')
Abacate
Pera
3
❏
```

DICIONÁRIO {}

# DICIONÁRIO{} - TEORIA

- É um conjunto de dados que possui uma chave e o valor.
- É um tipo de dado **desordenado**, **mutável**, **indexável**;
- Dicionário também tem os métodos:
  - `len()` -> para pegar o tamanho;
  - `pop()` -> para retirar item;

# DICIONÁRIO{} - MÉTODOS

main.py

```
1  #Dicionario
2  meuDicio = {
3      "Lua" : "Um satelite da Terra",
4      "Terra" : "Um planeta do sistema solar",
5      "Sol" : "A estrela do sistema solar"
6  }
7
8  print("Meu dicionario => " + str(meuDicio))
9
10 print("\n\nLua é " + meuDicio["Lua"])
11 print("Terra é " + meuDicio.get("Terra"))
12
13 meuDicio["Lua"] = "Que brilha la no ceu"
14 print("\n\nLua é " + meuDicio["Lua"])
15 |
16 meuDicio.popitem()
17 print("\n\nMeu dicionario => " + str
    (meuDicio))
```

Meu dicionario => {'Lua': 'Um satelite da Terra',  
'Terra': 'Um planeta do sistema solar', 'Sol': 'A  
estrela do sistema solar'}

Lua é Um satelite da Terra  
Terra é Um planeta do sistema solar

Lua é Que brilha la no ceu

Meu dicionario => {'Lua': 'Que brilha la no ceu',  
'Terra': 'Um planeta do sistema solar'}  
➤ []

# CONJUNTOS

# CONJUNTOS - STRINGS

- O tipo de dado String é um “conjunto” de caracteres, mas não uma lista.

main.py

```
1 #String como lista
2 frase = "Mais um dia se vai para outro nascer"
3
4 print(frase)
5 print(frase[1])
6 print(frase[3:10])#3 incluso 10 fora -> pega do 3-9
7 print(len(frase))
8 palavras = frase.split(" ")
9 print(palavras)
10
```

Mais um dia se vai para outro nascer

a

s um di

36

['Mais', 'um', 'dia', 'se', 'vai', 'para', 'outro', 'nascer']

➤

# CONJUNTO ++

Não é muito usado. Mas se tiver interesse:

[https://www.w3schools.com/python/python\\_sets.  
asp](https://www.w3schools.com/python/python_sets.asp)



# IMPORTAR MÓDULOS

- importar o módulo completo:
  - **import** nome\_modulo
- renomear o módulo:
  - **import** nome\_modulo **as** novo
- importar apenas algumas funções:
  - **from** nome\_modulo **import** nome\_da\_funcao (ou classe)
- Palavras reservadas: **from**, **import**, **as**

HORA DE EXERCITAR

FIM DO CURSO

OBRIGADO POR PARTICIPAR

:)

## + LINKS

[https://www.python-course.eu/python3\\_object\\_oriented\\_programming.php](https://www.python-course.eu/python3_object_oriented_programming.php)

<https://www.tutorialspoint.com/python3/>



# AUTOR:

Esse material foi feito por mim, Higor Emanuel Souza Silva, e autorizo a sua utilização, desde que a menção seja feita;

Todos os códigos usados em sala de aula, e apostilas para leitura e revisão de conteúdo, podem ser acessados no meu github:

<https://github.com/higoress/MinicursoPython>

Posso ser contactado em:

higoress@gmail.com