

**Guia de estudos para aprender Python**

Comunidade Python em Moçambique

2018

**ÍNDICE**

[Controle e avaliações 3](#_Toc716723627)

[Literais 3](#_Toc767051906)

[Boolean 3](#_Toc1281379930)

[Strings 4](#_Toc1318254693)

[Operadores numéricos: \*\*,\*,/,%,//,+,- 5](#_Toc42802086)

[Expoente de um numero 5](#_Toc809059819)

[Multiplicação 5](#_Toc901399184)

[Divisão 5](#_Toc656342984)

[Adição 5](#_Toc1172241086)

[Subtração 5](#_Toc1778073350)

[Modulo da divisão 5](#_Toc287384016)

[Divisão inteira 5](#_Toc1049341695)

[Operadores lógicos 6](#_Toc1041809182)

[Instrução pass 6](#_Toc508635590)

[Argumentos end= e sep= para formatação na função print() 6](#_Toc713714235)

[Uso do sep 6](#_Toc369231842)

[Uso do end 6](#_Toc1406325011)

[Estrutura condicional: if, if-else, if-elif, if-elif-else 6](#_Toc1778637423)

[Listas simples: Construção de vetores, Indexação e cortes (Slicing), e a função len() 7](#_Toc1761911494)

# Controle e avaliações

Python é uma linguagem de programação poderosa fácil de aprender. Possui uma estrutura de dados de alto nível eficiente e simples, porém flexível e tem uma abordagem a metodologia de Programação Orientada a Objectos.

A síntaxe elegante e tidos dinâmicos, juntamente com a natureza do seu interpretador, fazem do Python uma linguagem ideal para *scripting* e para o desenvolvimento rápido em diversas áreas de estudos e para a maiorias de plataformas.

## Literais

Literal é um dado bruto dado em uma variável ou constante. Em Python, existem vários tipos de literais que são os seguintes: Boolean, Integer, Float, Numerical e Complex.

### Boolean

Um literal booleano pode ter dois valores, True ou False, o exemplo explica como.

x = (1 == True)

y = (1 == False)

a = True + 4

b = False + 10

**print**("x é", x)

**print**("y é", y)

**print**("a:", a)

**print**("b:", b)

Os literais podem ser de 3 tipos em Python, eles podem ser do tipo Float, Integer e complexos.

* Integer: Usado para representação dos números inteiros
* Numerical: Usado para representação de números reais
* Floating-point numbers: Usado para a representação de Números reais de ponto flutuante.
* Scientific notation: Usado para representação cientifica dos números, são também chamados de números complexos.

a = 0***b1010*** #Binary Literals

b = 100 #Decimal Literal

c = 0***o310*** #Octal Literal

d = 0x12c #Hexadecimal Literal

#Literal flutuante

float\_1 = 90.5

float\_2 = 2.5e2

#Literal complexo

x = 34.94j

**print**(a, b, c, d)

**print**(float\_1, float\_2)

**print**(x, x.imag, x.real)

### Strings

Literais do tipo String são uma sequência de caracteres delimitados por aspas. Podemos utilizar as aspas simples, duplas ou mesmo triplas para uma string e um literal do tipo carácter é aquele que apenas possui um e somente um carácter delimitado por aspas simples ou duplas.

Exemplo: Como usar um literal do tipo string em Python?

strings = "String em Python"

char = "C"

str\_multilinha = """Esta string pode conter multiplas linhas e de enorme conteudo"""

unicode = u"\u00dcnic\u00f6de"

raw\_str = r"string \n no seu formato puro"

**print**(strings)

**print**(char)

**print**(str\_multilinha)

**print**(unicode)

**print**(raw\_str)

## Operadores numéricos: \*\*,\*,/,%,//,+,-

### Expoente de um número

a = 20

b = 3

**print**(a\*\*b) #Vai calcular o valor de a elevado a b

### Multiplicação

a = 20

b = 3

**print**(a \* b) #Vai calcular o valor de a vezes b

### Divisão

a = 20

b = 3

**print**(a / b) #Vai calcular o valor da divisão de a por b

### Adição

a = 20

b = 3

**print**(a + b) #Vai calcular o valor da adição de a e b

### Subtração

a = 20

b = 3

**print**(a - b) #Vai calcular o valor da subtração de a e b

### Modulo da divisão

a = 20

b = 3

**print**(a % b) #Vai calcular o valor do resto da divisão entre a e b

### Divisão inteira

a = 20

b = 3

**print**(a // b) #Vai calcular o valor inteiro da divisão de a e b

## Operadores lógicos

Os operadores lógicos unem expressões lógicas formando assim, uma nova expressão que é composta por 2 ou mais sub-expressões. O resultado lógico de expressões compostas será a relação entre as sub-expressões.

|  |  |
| --- | --- |
| **Operador Python** | **Operação** |
| not | não |
| and | e |
| or | ou |

## Instrução pass

Na linguagem de programação Python, a instrução **pass** é nula. A diferença que ela possui em relação a um comentário é simples, o interpretador ignora por completo os comentários mas o **pass** não é ignorado, é reconhecido como uma instrução para não fazer nada.

## Argumentos end= e sep= para formatação na função print()

### Uso do sep

comunidade = 'PyMoz '

membros = 'Todos Moz Devz'

juncao = comunidade + membros

**print**(comunidade, membros, sep = '-')

### Uso do end

comunidade = 'PyMoz '

membros = 'Todos Moz Devz'

juncao = comunidade + membros

**print**(juncao, end = '+')

## Estrutura condicional: if, if-else, if-elif, if-elif-else

As condições servem para selecionar quando uma parte do programa deve ser ativada e quando deve ser simplesmente ignorada.

*if <condição>:*

*bloco verdadeiro*

# Uso do if

## 

comunidade = 'PyMoz'

if comunidade == 'PyMoz':

membros = 'Todos Moz Devz'

print(comunidade, membros, sep='-')

else:

pass

comunidade = 'PyMoz'

if comunidade == 'PyMoz':

membros = 'Todos Moz Devz'

print(comunidade, membros, sep='-')

## Uso do if-else

# if-elif

comunidade = 'PyMoz'

if comunidade == 'PyMoz':

membros = 'Todos Moz Devz'

print(comunidade, membros, sep='-')

elif comunidade == MozDevz:

membros = 'De todas comunidades'

print(comunidade, membros, sep='-')

## if-elif-else

comunidade = 'PyMoz'

if comunidade == 'PyMoz':

membros = 'Todos Moz Devz'

print(comunidade, membros, sep='-')

elif comunidade == MozDevz:

membros = 'De todas comunidades'

print(comunidade, membros, sep='-')

else:

print(‘Quero ir pra casa’)

## Listas simples: Construção de vetores, Indexação e cortes (Slicing), e a função len()

Lista é qualquer sequencia de objectos/elementos em qualquer ordem. Todas as estruturas de dados são também uma lista, só que cada uma com sua particularidade.

A lista pode ser considerada a principal estrutura de dados ou a mãe de todas as estruturas.

Todas as estruturas de dados tem como objectivo, armazenar um conjunto de informação geralmente do mesmo tipo.

Lista é uma estrutura de dados mutável onde a ordenação natural é estabelecida pela ordem de entrada. O ultimo item de uma lista não ordenada sempre será último elemento adicionado.

# Norma gerais das listas:

1-Novos itens sempre serão adicionados após o último item,

2-O primeiro item adicionado a lista sempre será o primeiro elemento da estrutura.

a = [1,2,3,4,5,6,7,8,9]

type(a)

len(a)

a[0]

a[-1]

a[4]

a.reverse()

lista = list("PyCMoz")

a.append(10)

# Slicing (fatiar) uma lista

Podemos numa lista fazer cortes a fim de obter uma nova lista.

Para fatiar uma lista iremos usar a seguinte estrutura:

**Lista[start:stop:step]**

**Start** - inicio do recorte da lista (default=0)

**Stop** -fim do recorte da lista (default=[-1])

**Step** - intervalo que queremos saltar (default=1)

lista = "Python Community - Moz"

lista[15] #retornal elemento na posição 15

lista[:15] #retornar elementos até a posição 15

lista[::2] #retornando elemtentos com salto de dois elementos

lista[::-1] #invertendo lista

lista[3::2] #retornandos elemento aprtir da posição 3 co salto de 2