# PYTHON SEM COMPLICAÇÕES

Por Julia Marcolan
Contato: julia@marcolan.com




## SUMÁRIO

# O que é Python ?

## O que ele faz ?

Python é uma linguagem de programação inicialmente desenvolvida por Guido van Rossum, um grande fã do grupo de comédia Monty Python, nos anos 80. O objetivo do desenvolvimento do Python era criar uma linguagem de fácil compreensão para humanos e máquinas de forma simultânea. Isso quer dizer que muitas vezes programar em Python é intuitivo e fácil para aprender.

Python é uma linguagem com "tipagem dinâmica" isso quer dizer que não é preciso atribuir um tipo para uma variável, o interpretador de Python é responsável por descobrir esse tipo. A linguagem é orientada a objetos e muito poderosa, tem aplicação em diversas áreas como criação de sites e jogos, softwares científicos, gráficos e etc.

# Instalação



A versão do Python que deve ser instalada é o Python 3. Para fazer isso siga as instruções do tutorial do Django Girls para o seu sistema operacional.

https://tutorial.djangogirls.org/pt/python_installation/?q=


Imagem retirada do site Django Girls em 28 de janeiro de 2020;

O Django Girls é uma organização internacional sem fins lucrativos criada por duas mulheres polonesas, Ola Sitarska e Ola Sendecka, para inspirar mulheres de todas as origens a se interessarem por tecnologia e a se tornarem programadores, oferecendo um ambiente seguro e amigável. Eu, além de adorar a iniciativa, adoro os tutoriais, pois são muito bem explicados, por isso resolvi passar o link do tutorial ao invés de apenas utilizar a referência.

OBS: Caso você não tenha experiência usando o terminal do seu computador, nesse mesmo tutorial você encontra algumas dicas e até mesmo instruções para abrir o terminal em cada sistema operacional.

# Primeiros passos

## Python direto do terminal

Umas das forma de utilizar Python é direto do terminal. Para isso basta abrir o terminal e inserir o comando $python3. Nesse momento passará a aparecer o símbolo “>>>” que indica que o Python está ativo e todos os comando devem ser em Python. Para sair desse modo use Ctrl+d.

```
julia@julha:~$ python3
Python 3.7.5 (default, Nov 20 2019, 09:21:52)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 4+2
6
>>>
```

## Arquivos.py

Uma desvantagem de utilizar Python diretamente do terminal é que só é possível digitar um comando por vez. Dessa forma é mais eficiente, na grande maioria dos casos, escrever seu código num arquivo.py e depois rodar no terminal utilizando o comando $python3 exemplo.py

```
julia@julha:~$ python3
Python 3.7.5 (default, Nov 20 2019, 09:21:52)
[GCC 9.2.1 20191008] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 4
6
>>>
julia@julha:~$ python3 exemplo.py
6
julia@julha:~$
```

Como a ideia é que esse documento seja acessível para todo mundo, até quem está começando do ZERO, acho que vale a pena mencionar que é preciso de um editor de códigos para escrever os arquivos. Eles facilitam para codar pois deixam partes do código, como as funções, coloridas e ajudam na hora de fazer a indentação do código. Eu, particularmente, gosto de usar o Visual Studio Code que é um editor desenvolvido pela Microsoft para Windows, Linux e macOS. Mas você pode usar o de sua preferência.

Obs: Não é #publi, mas poderia ser #MicrosoftMePatrocina

Você com certeza percebeu que quando o terminal estava no modo de comandos de Python eu apenas digitei 2 + 4 e a conta foi feita. Essa é exatamente uma das vantagens que mencionei sobre ser uma linguagem muito intuitiva. As operações matemáticas são tão simples de se fazer direto pelo terminal que existe quem diga que Python é a melhor calculadora já inventada.

# Funções print e input

## Função print

A função print é utilizada para escrever coisas na tela. Você pode praticar criando seu primeiro programa em Python para escrever o texto “Fora Bolsonaro!” na tela do seu computador.

Editor de texto:

```
# Faça um programa que imprima na tela a seguinte mensagem "Fora Bolsonaro!"
print("Fora Bolsonaro!")
```

Terminal:

```
julia@julha:~/python-exercicios$ python3 1.py
Fora Bolsonaro!
julia@julha:~/python-exercicios$
```

Eu optei por fazer o exercício gravando um arquivo.py e depois rodá-lo no terminal. Esse exemplo é bastante simples e poderia ter sido feito sem problemas direto por linha de comando, no entanto eu sugiro que você faça arquivos enquanto estiver estudando e guarde eles numa pasta “Exercícios-python” para consultar quando for preciso. Ou treine dos dois jeitos!!

## Função input

A função input é utilizada para ler uma entrada do usuário via terminal. Inicialmente para praticar faça o exemplo abaixo de ler o seu nome e depois escrevê-lo no terminal utilizando a função print.



Editor de texto:

```
nome = input("Informe seu nome:").strip()
print(nome)
```

Terminal:

```
julia@julha:~/exercicios-python$ python3 2.py
Informe seu nome:julia
julia
julia@julha:~/exercicios-python$
```

A função strip() é utilizada para remover possíveis espaços no início ou final da string.

A função input lê a entrada como uma string (estudaremos o que é uma string daqui a pouco, mas resumindo é um conjunto de caracteres). Em determinados momentos é preciso converter os tipos. Veremos isso conforme for necessário, mas para matar a curiosidade caso você queira ler um inteiro deve informar o comando

$numero = int(input(“Informe o numero”)) .

# Operações matemáticas

## Usando Python como calculadora

As operações matemáticas básicas são muito simples em Python.

**Soma**

```
>>> a + b
>>> 5 + 2
>>> 7
```

**Subtração**

```
>>> a - b
>>> 5 - 2
>>> 3
```

**Multiplicação**

```
>>> a * b
>>> 5 * 2
>>> 10
```

**Divisão**

```
>>> a / b
>>> 5 / 2
>>> 2.5
```

Em números decimeis, utiliza-se ponto no lugar de vírgula.

**Parte inteira da divisão**

```
>>> a // b
>>> 5 // 2
>>> 2
```

**Resto da divisão**

```
>>> a % b
>>> 5 / 2
>>> 1
```



**Potenciação**

```
>>> a ** b
>>> 5 ** 2
>>> 25
```

**Raíz**

```
>>> a ** 0.5
>>> 25**0.5
>>> 5.0
```

Lembre-se da propriedade:

$$\sqrt[n]{a} = a^{\frac{1}{n}}$$

Obs: Pratique essas operações com Python direto no terminal.

# Módulos matemáticos

Os módulos são coleções de funções pré definidas que podem ser utilizadas pelo usuário. Os módulos matemáticos possuem funções que são utilizados para fazer operações mais complicadas do que as operações básicas. Os módulos são math para números reais e o cmath que inclui os números complexos. Para utilizar o módulo é preciso "importá-lo", ou seja informar para o interpretador que você deseja usar uma função daquele módulo. A relação completa das operações disponíveis encontra-se na documentação oficial do Python mas entre estas funções estão raiz quadrada, funções trigonométricas como seno e cosseno e constantes como o pi.

Documentação oficial: https://docs.python.org/3/library/math.html

- Funções Aritméticas:

Funções para cálculos aritméticos com números reais, exemplos:

- gcd -> maior divisor comum: math.gcd(a,b)
- factorial -> calcula o fatorial: math.factorial(a)

- Funções Trigonométricas:

Funções envolvendo ângulos e relações trigonométricas.

- sin -> calcula o seno: math.sin(a)
- cos -> calcula o cosseno: math.cos(a)

- Funções de Potência:

Uma outra forma de ser fazer potenciação é utilizando as funções de potência.

- sqrt -> rais quadrada: math.sqrt(a)
- pow -> potência: math.pow(a,b) (a elevado a b)

- Funções Logarítimicas:

- log2(x) -> log na base 2 de x
- log10(x) -> log na base 10 de x
- log(x, base) -> log de x para qualquer base direfente de 2 e 10.

Obs: A unidade usada em funções trigonométricas é rad.



Constante pi: math.pi

Exemplo de utilização:

Editor de texto:

```
import math

numero = int(input("Insira um número inteiro:"))

raiz = math.sqrt(numero)
print("Raíz quadrada do número inserido:", raiz)

print("O valor de pi é:", math.pi)

print("Cosseno de pi:", math.cos(math.pi))
```

Repare que eu usei a conversão do tipo de entrada da função input para int.

Terminal:

```
julia@julha:~/exercicios-python$ python3 3.py
Insira um número inteiro:25
Raíz quadrada do número inserido: 5.0
O valor de pi é: 3.141592653589793
Cosseno de pi: -1.0
julia@julha:~/exercicios-python$
```

## Exercícios:

1. Faça um programa que peça para que o usuário informe dois números e print as seguintes informações sobre eles: 1) soma 2) subtração 3) produto.
2. Faça um programa que peça dois números para o usuários e print 1) A divisão dos dois números 2) A divisão inteira 3) o resto da divisão.



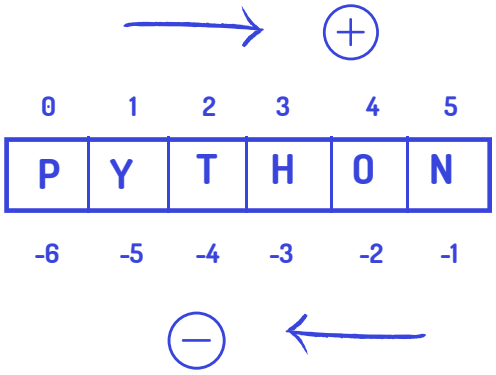
Obs: As soluções dos exercícios se encontram-se disponíveis porém seria interessante tentar resolvê-los antes de espiar.

# Strings

Uma string é uma sequência de caracteres utilizada para representar palavras ou frases. Em Python uma string pode ser colocada 'entre aspas simples' ou " entre aspas duplas", o importante é que comece e termine sempre da mesma forma.

## índices

Cada carácter que compõe a string pode ser representado por um índice. Contando da esquerda para a direita seguimos índices crescentes positivos iniciados do zero, e da direita para a esquerda índices negativos iniciados de -1.




As strings podem ser acessadas por meio dos seus índices. Por exemplo, podemos pegar apenas um carácter da string, ou ainda fazer um "fatiamento" da string mostrando apenas parte dela. No exemplo de programa abaixo é possível compreender como isso funciona.



Editor de texto:

```
# Atribuindo uma string para a variável palavra
palavra = 'Python'

# Os índices 0 e -6 da string correspondem a mesma posição, por isso o resultado será o mesmo
#Obs: quando escrevo coisas "entre aspas" para serem printadas na tela, são strings
print("índice 0:", palavra[0])
print("índice -6:", palavra[-6])
print()


# Fatiamento: Par fazer o fatiamento devemos informar o índice inicial que queremos e o final. Porém cuidado
# pois o último elemento de índice não entra. Por exemplo, quando pefirmos palavra[1:5] -> indica que queremos
# os elementos da string de 1 até 5 porém o elemente 5 fica de fora.
print("índices de 1 até 5 [1:5]", palavra[1:5])
print()

#Também é possível pegar os índices de -1 em -1
print("índices de -1 até -4 [-1:-4:-1] de -1 em -1 :", palavra[-1:-4:-1])
print()

# Por último, vimos que nossa string vai até o índice 5 (no sentido positivo)
# porém, se estipularmos o fatiamento palavra[1:6] (para conseguir pegar o 5) nenhum erro é acusado
# O python vai até onde dá
print("De 1 até 6 [1:6]:", palavra[1:6])
print()
print("De 1 até 7 [1:7]:", palavra[1:7])
```

Terminal:

```
julia@julha:~/python-exercicios$ python3 exemplo.py
indice 0: P
indice -6: P

indices de 1 até 5 [1:5] ytho

indices de -1 até -4 [-1:-4:-1] de -1 em -1 : noh

De 1 até 6 [1:6]: ython

De 1 até 7 [1:7]: ython
julia@julha:~/python-exercicios$
```

Uma observação importante é que as strings em Python são imutáveis. Dessa forma, não é possível redefinir elementos da string.

# Operações com strings

Apesar das strings serem imutáveis, é possível criar novas strings manipulando as strings já existentes.

## Soma e multiplicação de strings:

Editor de texto:

```
1  #Soma  e Multiplocação de Strings
2
3  string1 = "Fora"
4  string2 = "Bolsonaro"
5
6  print("Somando as strings:", string1 + string2)
7  print()
8
9  #Também posso atribuir uma string nova para a soma
10 string3 = string1 + string2
11 print("Soma:", string3)
12 print()
13
14 #Multiplicação das strings
15 print("Fora Bolsonaro 13 vezes:", string3*13)
```



Terminal:

```
julia@julha:~/python-exercicios$ python3 exemplo5.py
Somando as strings: ForaBolsonaro

Soma: ForaBolsonaro

Fora Bolsonaro 13 vezes: ForaBolsonaroForaBolsonaroForaBolsonaroForaBolsonaroForaBolsonaroForaBolsonaroForaBolsonaroForaBolsonaroForaBolsonaro
ForaBolsonaroForaBolsonaroForaBolsonaroForaBolsonaro
julia@julha:~/python-exercicios$
```

# Métodos string

Métodos são procedimentos associados a um determinado objeto fazendo com que este objeto passe a ter determinado comportamento. Python possui vários metodos para strings, todos presentes na documentação ofiacial https://docs.python.org/pt-br/3/library/stdtypes.html#string-methods, alguns deles estão listados abaixo:

### str.upper()

Retorna uma cópia da String com todos os caracteres maiúsculos.

### str.capitalize()

Retorna uma cópia da String com o seu primeiro caractere em maiúsculo e o restantes em minúsculo.

### str.format(pode receber mais de um argumento)

Insere uma string no local da frase indicado por {}.

Editor de texto:

```
1   #Métodos com string
2   nome = "julia"
3   #format -> a string nome é o argumento que sera colocado
4   # no espaço indicado por {}
5   print("Bem vinda {}." .format(nome))
6   print()
7
8   print("NOME MAIÚSCULO:", str.upper(nome) )
9   print()
10  print("Nome novamente:", nome)
11  print()
12
13  print("Nome com a primeira letra Maiúscula:", str.capitalize(nome))
```

Terminal:

```
julia@julha:~/python-exercicios$ python3 exemplo4.py
Bem vinda julia.

NOME MAIÚSCULO: JULIA

Nome novamente: julia

Nome com a primeira letra Maiúscula: Julia
julia@julha:~/python-exercicios$
```

### Função len()

A função len retorna o tamanho da string.

### Função split()

Separa as palavras da string, criando cópias, para que possam ser agrupadas em outras strings.

Editor de texto:

```
#Usando funções para manipular strings
frase = " Itachi é o melhor personagem de Naruto"

#Calculando o tamanho da string
print( " A string informada contém {} caracteres." .format(len(frase)))
print()

#Separando as palavras da prase
p1,p2,p3,p4,p5,p6,p7 = frase.split()
print("Palavra 1:", p1, "\n Palavra 2:", p2, "\n Palavra 3:", p3, "\n Palavra 4:", p4)
```

Terminal:

```
julia@julha:~/python-exercicios$ python3 exemplo6.py
 A string informada contém 39 caracteres.

Palavra 1: Itachi
 Palavra 2: é
 Palavra 3: o
 Palavra 4: melhor
julia@julha:~/python-exercicios$
```

Depois dos exemplos, você pode estar se perguntando qual a diferença entre uma função e um método. Bom, a diferença é que um método só é válido para um tipo de objeto, os métodos de string só podem ser usados para strings. Uma função no entanto pode ser usada por vários tipos de objetos, a função len pode ser usada em listas por exemplo (veremos essa estrutura mais adiante).

## Exercícios:

3. Atribua a frase “Pacific rim melhor filme de 2013” a uma determinada variável, depois calcule quantos caracteres tem essa frase e a escreva toda em letras maiúsculas.

4. Faça um programa que leia 2 strings e informe o conteúdo delas seguido do seu comprimento.

5. Faça um programa que permita ao usuário digitar o seu nome e em seguida mostre o nome do usuário de trás para frente utilizando somente letras maiúsculas. Dica: lembre-se que ao informar o nome o usuário pode digitar letras maiúsculas ou minúsculas.

6. Dado uma string com uma frase informada pelo usuário (incluindo espaços em branco), conte os espaços em branco presentes nela.



Atenção: Nem todas as funções/metódos que você precisará para fazer esse estes exercícios estão nessa apostila. Chegou a hora de explorar a documentação oficial e as outras referências;

Obs: As soluções dos exercícios se encontram-se disponíveis porém seria interessante tentar resolvê-los antes de espiar.

## Comandos de comparação

- x maior que y $x > y$
- x menor que y $x < y$
- x maior ou igual a y $x >= y$
- x menor ou igual a y $x <= y$
- x igual a y $x == y$
- x diferente de y $x! = y$

## Comparação dupla

**and** — Usado quando duas condições precisam ser satisfeitas de forma simultânea.

**or** — Usado quando pelo menos uma, das duas condições estipuladas, precisa ser satisfeita.

## Booleanos

- True -> verdadeiro
- False -> falso

OBS: A grafia deve ser exatamente assim, com o primeiro maiúsculo e o restante minúsculo.

## Estruturas de comando de controle de fluxo



Ferramentas de controle de fluxo são ferramentas que permitem que um determinado conjunto de comandos seja repetido determinado número de vezes satisfazendo certas condições.

### While

O while é um laço de repetição que executa um determinado comando enquanto a condição do laço for verdadeira.

OBS: A indentação é muito importante em Python!!!!

**Estrutura**

```
>>> while condição:
>>>         print("oi")
```

As condições geralmente são feitas com comandos de comparação. Ex

```
>>> while x<10:
>>>         faça alguma coisa
```

→ É preciso ter alguma ferramenta que atualize a condição para que não caia num loop infinito.

## if

O comando if executa o conjunto de comandos seguinte caso a condição seja verdadeira naquele momento.

**Estrutura**

```
>>> if  condição:
>>>        print("oi")
```

## else

A condição else é executada caso a condição do if não seja satisfeita.

**Estrutura**

```
>>> if  condição:
>>>        print("oi")
>>>else
>>>        print("tchau")
```

## elif

O elif é uma abreviação de else if, usado para quando preciso impor mais de uma possível condição/ação

**Estrutura**

```
>>> if  condição:
>>>        print("oi")
>>>elif condição:
>>>        print("tchau")
>>else
>>>        print("bjs")
```



## For

O comando for itera sobre itens de diversas estruturas, como strings, listas e dicionários (que ainda serão estudados).

**Estrutura**

```
>>> for i in a range(10):
>>>        print("i")
```

## Função range()

A função range() é usada para iterar numa sequência numérica.

- range(10) cria uma sequência de índices de 0 até 9.
- range(5:10) cria uma sequencia de índices de 5 até 9.
- range(0,10,2) cria uma sequência de 0 até 9 indo de 2 em 2.

Editor de texto:

```python
#Exemplo de utilização das estruturas de controle de fluxo
#while
print("Exemplo while")
print()
n = 0
while n<10:
    #identação
    print("n = ", n)
    #incrementação do n
    n = n +1

#if
print()
print("Exemplo if")
print()

nome = input("Informe um nome")

if nome == "julia":
    print("oi julia você é linda")
elif nome == "Naruto":
    print("O  Naruto nunca desiste!")
else:
    print("Oi {}, seja bem vindo" .format(nome))

# for
print()
print("Exemplo for")
print()

for i in range(5,10):
    print("oi,", i)
```



Terminal:

```
julia@julha:~/python-exercicios$ python3 exemplo7.py
Exemplo while

n =  0
n =  1
n =  2
n =  3
n =  4
n =  5
n =  6
n =  7
n =  8
n =  9

Exemplo if

Informe um nomejulia
oi julia você é linda

Exemplo for

oi, 5
oi, 6
oi, 7
oi, 8
oi, 9
julia@julha:~/python-exercicios$
```

OBS: A indêntação é muito importante em Python!!!! Se o código não estiver bem indentado  ele não funciona!!!

```
julia@julha:~/python-exercicios$ python3 exemplo7.py
  File "exemplo7.py", line 20
    else:
       ^
IndentationError: unindent does not match any outer indentation level
julia@julha:~/python-exercicios$
```

# Comandos break, continue, clausula else e pass

- Break-> Sai imediatamente do laço condicional mais interno.
- Continue -> Continua com a próxima linha de instrução.
- Clausula else -> Num laço condicional NUNCA está ligado ao condicional que contém break
- Pass -> Não faz nada kkk

## Exercícios:

7. Peça para que o usuário informe um número inteiro n > 0 e as notas finais de n alunos, determinar quantos alunos ficaram de recuperação. Um aluno está de recuperação se sua nota fina está entre 3.0 e 5.0 A nota máxima é 10.

8. Escreva um programa que exiba uma sequência de 12 números com cada elemento igual ao triplo do elemento anterior.

9. Escreva um programa que peça para o usuário informar um número inteiro e verificar se o número informado é primo. Em caso afirmativo, o programa deve mostrar a mensagem " O número informado é primo", caso contrário deve informar "Não é primo".

10. Escreva um programa que peça para o usuário informar um número inteiro e verificar se ele é múltiplo de 2. Em caso afirmativo, o programa deve mostrar a mensagem " O número informado é múltiplo de 2", caso contrário deve informar "Não é múltiplo de 2".



11. Escreva um programa que peça para o usuário informar um número inteiro e verificar se o número informado é múltiplo de 2 ou de 5. Caso seja múltiplo apenas de 2 o programa deve informar " Múltiplo apenas de 2", caso seja múltiplo apenas de 5 deve informar " Múltiplo apenas de 5", caso seja dos dois deve informar " Múltiplo de 2 e 5" e caso não seja múltiplo de nenhum deve informar "Não é múltiplo de 2 nem de 5.".

Atenção: Nem todas as funções/metódos que você precisará para fazer esse estes exercícios estão nessa apostila Chegou a hora de explorar a documentação oficial e as outras referências;

Obs: As soluções dos exercícios se encontram-se disponíveis porém seria interessante tentar resolvê-los antes de espiar.

# Outras referências

Documentação oficial de Python < https://docs.python.org/3/ > Acesso em 01 de fevereiro de 2019.

Site Django girls < https://tutorial.djangogirls.org/pt/python_installation/?q= > Acesso em 01 de fevereiro de 2019.

Introdução a programação com Python por Igor Magalhães e Giovani Louzada
< https://github.com/igormagalhaesr/aprendendoaprogramarempython >

Comunidade Python Brasil < https://wiki.python.org.br/PythonBrasil >



Além da documentação que foi utilizada para fazer este conteúdo, aqui você também encontra outras fontes de pesquisa para aprender mais.