



30 de setembro de 2019

# Introdução ao Python

Ativando os poderes Pythônico  
SIDIA - Setembro/2019

**Jailson P. Januário**

jpj.ads@uea.edu.br

Github:@jailsonpj



# Apresentação

- **Jailson Pereira Januário**
- Sistemas de Informação (EST/UEA)
- Laboratório de Sistemas Inteligentes (LSI/UEA)
- Buritech
- *Machine Learning*
- *Deep Learning*
- Entusiasta Python
- Coordenador do PyData Manaus



# O que é Python?

- **Linguagem de altíssimo nível**
  - Suporte nativo a estruturas de dados de alto nível
- **Multiplataforma**
  - Unix, Windows, Symbian, Solaris, etc...
- **Multiparadigma**
  - Procedural, OO, Funcional
- **Opensource**
- **Dinâmica e forte**
- **Joga com outras linguagens**
  - (.NET) IronPython, (JAVA) Jython, C e C++



## Por que Python?

- Aprendizado fácil
- Sintaxe limpa e de fácil leitura
- Forte suporte da comunidade
- Bem documentada
- Biblioteca
- Divertida
- Mais com menos *[código]* - Pythônico
- Liberdade



## Quem usa Python?

- Google
- NASA
- Méliuz
- Accenture
- Mercado livre
- Serasa Consumidor
- Globo.com
- Entre outras<sup>1</sup>

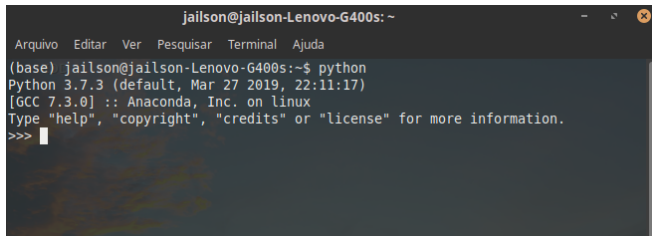
---

<sup>1</sup><https://python.org.br/empresas/>



## Instalação do Python - Linux

- Já vem por padrão instalado nas distribuições;
- Opção de Instalação:
  - *\$ sudo apt-get install python*
- Execução via terminal:
  - *\$ python*



```
jailson@jailson-Lenovo-G400s: ~  
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda  
(base) jailson@jailson-Lenovo-G400s:~$ python  
Python 3.7.3 (default, Mar 27 2019, 22:11:17)  
[GCC 7.3.0] :: Anaconda, Inc. on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 
```

Figura 1: Execução pelo terminal.



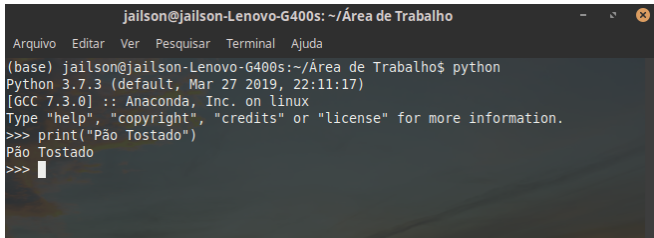
## Instalação - Windows

- Dowload: <https://www.python.org/downloads/windows/>
- Execute o programa baixado. Ex: `python-3.7.4.msi`
- Pacotes instalados



# Interpretador Interativo

```
print(" Pão Tostado")
```



```
jailson@jailson-Lenovo-G400s: ~/Área de Trabalho
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
(base) jailson@jailson-Lenovo-G400s:~/Área de Trabalho$ python
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Pão Tostado")
Pão Tostado
>>> 
```

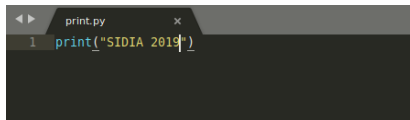
Figura 2: Utilizando o interpretador.





## Executando arquivos \*.py

- Use um editor de texto;
- Escreva um código para imprimir uma mensagem



```
print.py
1 print("SIDIA 2019")
```

Figura 3: Exemplo de arquivo .py

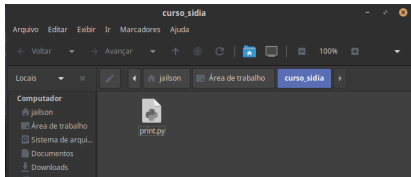


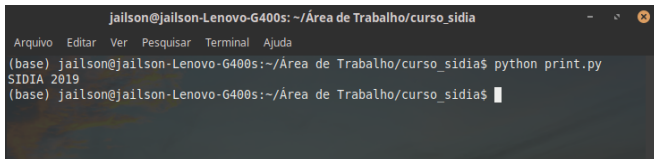
Figura 4: Arquivo .py

- Salve o arquivo
- print.py



## Executando arquivos \*.py

- No terminal, entre no diretório onde salvou o arquivo print.py;
- Digite o comando para execução de arquivos python
  - *\$ python print.py*



```
jailson@jailson-Lenovo-G400s: ~/Área de Trabalho/curso_sidia
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
(base) jailson@jailson-Lenovo-G400s:~/Área de Trabalho/curso_sidia$ python print.py
SIDIA 2019
(base) jailson@jailson-Lenovo-G400s:~/Área de Trabalho/curso_sidia$
```

Figura 5: Execução do arquivo print.py



## Conceitos

- Case sensitive
  - `This`  $\neq$  `this`
- Blocos por endentação
- Tipagem dinâmica
  - `a = 2` `Integer`
  - `a = 'alguma coisa'` `String`
  - `a = 2.3` `Float`
- Tudo é objeto
- Não tem ponto e vírgula no final (";")
- Comentários começam com `#`



# Variáveis e Tipos Básicos

- [illegible]

```
>>> a = True
>>> b = False
>>> type(a)
<type 'bool'>
>>> type(b)
<type 'bool'>
```

Tente usar **dir(a)** e **help(a)**



# Conversão dos Tipos Básicos

- `int()`, `float()`, `str()`, `bool()`, `long()`

```
>>> int(3.1415)
3
```

```
>>> float(3)
3.0
```

```
>>> str(25)
'25'
```

```
>>> str(True)
'True'
```

```
>>> bool(1)
True
```

```
>>> bool(0)
False
```

```
>>> bool(43)
True
```

```
>>> bool( 'lala' )
True
```

```
>>> bool( " ")
True
```

```
>>> float(2333333333333377777777778888888889999999999933334421L)
2.333333333333377e+56
```

```
>>> long(2.56)
2L
```

Teste no interpretador!



# Operadores Aritméticos

• +, -, \*, /, //, \*\*, %

Divisão com números inteiros resulta em um número inteiro

// - divisão inteira  
\*\* - exponenciação  
% - resto da divisão

Teste no interpretador

```
>>> a = 2
>>> b = 3
>>> a + b
5
>>> a - b
-1
>>> a / b
0
>>> a // b
0
>>> a ** b
8
```

```
>>> a = 2
>>> b = 3.5
>>> a + b
5.5
>>> a - b
-1.5
>>> a * b
7.0
>>> a / b
0.5714285714285714
>>> a // b
0.0
>>> a ** b
11.313708498984761
```

```
>>> 4 // 1.3
3.0
>>> 10 // 1.3
7.0
>>> 10 // 3.3
3.0
```

```
>>> 10 % 3
1
>>> 10 % 2
0
>>> 5 % 3
2
```



# Operadores Relacionais

- $>$ ,  $<$ ,  $>=$ ,  $<=$ ,  $==$ ,  $!=$

$!=$  significa "diferente"

É possível usar vários operadores na relação:  
 $1 < 2 <= 3 < 4 < 5 > 4 > 3 >= 2 != 1$

$\#$  - indica um comentário

```
>>> 2 > 3
False
>>> 2 < 3
True
>>> 3 >= 3
True
>>> 4 <= 3
False
```

```
>>> 2 == 2
True
>>> 2 == 1
False
>>> 3 != 2
True
>>> 3 != 3
False
```

```
>>> x = 3
>>> 2 < x < 4
True
>>> 7 > x > 1
True
>>> 3 <= x < 4
True
```



# Manipulação de Strings

- Podemos representar strings com (') ou (' ')

```
(base) jailsonpereira@lsidesktop3:~$ python
Python 3.7.2 (default, Dec 29 2018, 06:19:36)
[GCC 7.3.0] :: Anaconda custom (64-bit) on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 'spam eggs'
'spam eggs'
>>> 'doesn\'t'
"doesn't"
>>> '"Yes," he said.'
'"Yes," he said.'
>>> "\"Yes,\" he said."
'"Yes," he said.'
>>> '"Isn\'t," she said.'
'"Isn\'t," she said.'
>>> 
```





# Textos em várias linhas

## Definição

```
1 hello = "Redes Neurais é um modelo de ML que se baseia no funcionamento dos neurônios humano\n\
2 Mas as redes neurais não são boas para todos para todos os problemas\n\
3 É isto."
4
5 print(hello)
```

## Resultado

```
Redes Neurais é um modelo de ML que se baseia no funcionamento dos neurônios humano
Mas as redes neurais não são boas para todos para todos os problemas
É isto.
```



## Operações com Strings

- `+` : Concatenação
- `*` : Replicação
- `str[i]` : retorna o caracter de índice `i` da string `str`
- `str[inicio:fim]`, retorna uma substring de `str`

```
>>> st = 'curso'
>>> st + 'python'
'curso python'
```

```
>>> st = 'Casa'
>>> st*3
'CasaCasaCasa'
```

```
>>> st = 'arquivo.mp3'
>>> st[0]
'a'
>>> st[-1]
'3'
>>> st[-4]
'.'
```

```
>>> st = 'arquivo.mp3'
>>> st[2:]
'quivo.mp3'
>>> st[0:-4]
'arquivo'
>>> st[-3:]
'mp3'
```



## Métodos de Strings

- `split(char)`

- Retorna uma **lista** com os elementos separados por **char**

```
>>> a = '1+2+3+4+5+6'  
>>> a.split('+')  
['1','2','3','4','5','6']
```

- `(chars)`

- Retorna uma string onde os **chars** da direita e da esquerda foram removidos

- `len(str)`

- Retorna o tamanho da string **str**

```
>>> a = 'curso de python'  
>>> len(a)  
15
```

```
>>> a = ' !!! STRING DE RETORNO ! !!!'  
>>> a.strip(' !')  
'STRING DE RETORNO'
```



# Métodos de Strings

- `find(substring)`
  - Retorna a posição da primeira ocorrência da `substring`.
  - Caso não seja encontrada, retorna -1
- `lower()`, `upper()`
  - Retornam uma string em minúsculo/maiúsculo



## Exercícios

- A partir da string "!! ! a;b;c;d;e;f;gh!" gere o resultado:
  - ['a','b','c','e','f','g']
- A partir da string 'ring ring! - hello!' gere o resultado:
  - 'hello!'
- Transforme a string 'isso deve ser bom' para 'Isso Deve Ser Bom'
- Transforme a string 'abacate azul' em '4b4c4te 4zul'

**Dica: para os dois últimos, pesquise os métodos de string usando dir()**



## Exercícios

- A partir da string '!! ! a;b;c;d;e;f;gh!#####' gere o resultado:
  - ['a','b','c','e','f','g']

```
string = '!! ! a;b;c;d;e;f;gh!#####'
```

```
print( string.strip('!#h').split(';'))
```



## Exercícios

- A partir da string 'ring ring! - hello!' gere o resultado:
  - 'hello!'

```
string = 'ring ring! - hello!'
print(string[string.find('hello'):])
```

**OU**

```
print(string[13:])
```



## Exercícios

- Transformar a string 'isso deve ser bom' para 'Isso Deve Ser Bom'

```
string = 'isso deve ser bom'  
print(string.title())
```

- Transformar a string 'abacate azul' em '4b4c4te 4zul'

```
string = 'abacate azul'  
print(string.replace('a', '4'))
```





# Difícil?

