Python Easy

Prof. Iury Adones

Python

O que é Python?

- Python é uma linguagem de programação de alto nível

O que é Python?

- Python é uma linguagem de programação de alto nível
- Python é interpretada

O que é Python?

- Python é uma linguagem de programação de alto nível
- Python é interpretada
  - de script

O que é Python?

- Python é uma linguagem de programação de alto nível
- Python é interpretada
  - de script
  - imperativa

O que é Python?

- Python é uma linguagem de programação de alto nível
- Python é interpretada
  - de script
  - imperativa
  - orientada a objetos

O que é Python?

- Python é uma linguagem de programação de alto nível
- Python é interpretada
  - de script
  - imperativa
  - orientada a objetos
  - funcional

O que é Python?

- Python é uma linguagem de programação de alto nível
- Python é interpretada
  - de script
  - imperativa
  - orientada a objetos
  - funcional
  - de tipagem dinâmica e forte

História

Linguagem **Python**

- Foi lançada por *Guido van Rossum* em 1991

História

Linguagem **Python**

- Foi lançada por *Guido van Rossum* em 1991
- Atualmente o desenvolvimento é aberto (*open-source*)

História

Linguagem **Python**

- Foi lançada por *Guido van Rossum* em 1991
- Atualmente o desenvolvimento é aberto (*open-source*)
- Gerenciado pela organização sem fins lucrativos **Python Software Foundation**

Lançada pela comunidade python

Versão estável

Python 3.7.x

Estilo de tipagem

- Dinâmica

Estilo de tipagem

- Dinâmica
- Forte

Python é uma linguagem multiparadigma

- Programação Orientação a Objetos

Python é uma linguagem multiparadigma

- Programação Orientação a Objetos
- Programação Imperativa

Python é uma linguagem multiparadigma

- Programação Orientação a Objetos
- Programação Imperativa
- Programação Funcional

Referência

- wikipedia.org/Python

Download and Install

Install Python 3.7 no Windows

Download

site: python.org

Install python3 with Ubuntu

Open terminal

shortcut: `ctrl + alt + t`

write command line:

```
sudo apt-get install python3
```

Names

sudo: SuperUser Do

apt: Advanced Package Tool

Install python with Archlinux

Open terminal

write command line:

```
sudo pacman -S python
```

Names

pacman: Package Manager

-S or –sync: Synchronize packages

Executar Python

Windows

Pesquise por **`prompt de commandos`**, mas também conhecido como **`cmd`**

Linux e Mac

Pesquise por **`terminal`**

Sobre `terminal` e `prompt`

O **`terminal`** e **`prompt`** são usados para gerenciar o sistema operacional e ambiente de desenvolvimento.

Podemos enviar linhas de commandos que serão interpretadas e executadas.

Interpretadores de comandos são chamados de **`shell`**.

Checar a versão do python

Digite a linha de comando, depois pressione a teclar `Enter`

```
python --version
```

```
Python 3.7.x
```

Vamos iniciar o shell do python

Terminal ou prompt

```
python
```

```
Python 3.7.x (default, May ## ####, ##:##:##)
[GCC #.#.# ########] on linux
Type "help", "copyright", "credits" or "license" ...
>>> 2 + 2
4
>>>
```

Como identificar se estamos no shell do python?

Na linha de comando de python tem >>>

Imprime o “hello world”

Shell do Python

```
>>> print('Olá mundo!')
Olá mundo!
>>> exit()
```

Por que no comando `exit` colocar ( )?

- Linguagem de programação é puramente linguagem matemática

Por que no comando `exit` colocar ( )?

- Linguagem de programação é puramente linguagem matemática
  - $f(x) = x^2$

Por que no comando `exit` colocar ( )?

- Linguagem de programação é puramente linguagem matemática
  - $f(x) = x^2$
    - $f(x = 0) = 0$

Por que no comando `exit` colocar ( )?

- Linguagem de programação é puramente linguagem matemática
  - $f(x) = x^2$
    - $f(x = 0) = 0$
    - $f(1) = 1$

Por que no comando `exit` colocar ( )?

- Linguagem de programação é puramente linguagem matemática
  - $f(x) = x^2$
    - $f(x = 0) = 0$
    - $f(1) = 1$
    - $f(2) = 4$

Por que no comando `exit` colocar ( )?

- Linguagem de programação é puramente linguagem matemática
  - $f(x) = x^2$
    - $f(x = 0) = 0$
    - $f(1) = 1$
    - $f(2) = 4$
    - $f(3) = 9$

Por que no comando `exit` colocar ( )?

- Linguagem de programação é puramente linguagem matemática
  - $f(x) = x^2$
    - $f(x = 0) = 0$
    - $f(1) = 1$
    - $f(2) = 4$
    - $f(3) = 9$
  - `exit()` -> A função envia uma mensagem ao sistema operacional que deseja sair do shell do python

Python tipos de variáveis

Interger

int()

```
0
1
-53
22
100
```

Float

float()

```
0.0
1.2
-34.44
234.23
3e3
5E4
6e-10
2E-10
```

Complex

complex()

```
(1+2j)
(0j)
(5+3j)
(1-2j)
```

String

str()

```
"Olá mundo"
"1º lugar"
'Preço 3.56'
'A festa foi "divertida"'
```

List

list()

```
[1, 2, 3, 4, 5]
['maçã', 'banana']
[1, 'maçã', 2, 'banana']
```

Dicionary

dict()

```
{'cpf': '094.940.490-04'}
{'001': {'nome': 'Usuário de Python', 'idade': 22}}
{'x': 10, 'y': 3}
```

Set

set()

```
{'maçã', 'uva', 'queijo'}
{1, 2, 3, 4}
{[1,2,3],'maçã'}
```

Boolean

bool()

```
True
False
```

Empty

NoneType

```
None
```

Python Básico

Python Básico - Tipagem

Tipagem dinâmica

```
>>> a = 1
>>> type(a)
<class 'int'>
>>> a = 'abacaxi'
>>> type(a)
<class 'str'>
>>> a = 1.0
>>> type(a)
<class 'float'>
```

Python Básico - Tipagem

Tipagem dinâmica

```
>>> a = [1, 2.2, '23', 'ola', None]
>>> type(a)
<class 'list'>
>>> a = {1,2}
>>> type(a)
<class 'set'>
>>> a = {1:2}
>>> type(a)
<class 'dict'>
```

Python Básico

Entrada de variáveis

```
>>> valor_in = input("Digite um valor: ")
Digite um valor: 10
>>> type(valor_in)
<class 'str'>
>>> valor_in
'10'
>>> valor_in = int(valor_in)
>>> type(valor_in)
<class 'int'>
>>> valor_in
10
>>>
```

Python Básico - Extensão

Extensão do arquivo em python é **`.py`**

Python Básico

Script com python

```
touch programa-init.py
echo "print('Ola mundo')" >> programa-init.py
python programa-init.py
```

```
Ola mundo
```

Python Básico

Script com python

```
echo "print('Ola novamente')" > programa-other.py
python programa-other.py
```

```
Ola novamente
```

Python Básico

Operações básicas da matemática

Operação	**Operador**
Adição	+
Subtração	-
Multiplicação	*
Divisão	\

Python Básico

Operações básicas da matemática

Operação	**Operador**
Exponenciação	**
Parte inteira	//
Módulo	%

Python Básico - exemplo de script

Crie um arquivo chamado `mult.py`

```
# coding: utf-8
"""
calcule: f(x,y) = x*y
Sabemos que x e y pertencem aos números Naturais.
"""

x = input("x: ")
y = input("y: ")

x, y = int(x), int(y)

result = x * y

print('x: %d, y: %f' %(x,y))
print('Resulta: %i' %result)
```

Python Básico - Modos de imprimir

1 - Forma de imprimir na tela

```
print('x: {}, y: {}'.format(x, y))
print('Resulta: {}'.format(result))
```

Python Básico - Modos de imprimir

2 - Forma de imprimir na tela

```
print('x: {1}, y: {0}'.format(y, x))
print('Resulta: {0}'.format(result))
```

Python Básico - Modos de imprimir

3 - Forma de imprimir na tela

```
print('x: {_x}, y: {_y}'.format(_y=y, _x=x))
print('Resulta: {r}'.format(r=result))
```

Python Básico - Modos de imprimir

4 - Forma de imprimir na tela

```
print(f'x: {x}, y: {y}')
print(f'Resulta: {result}')
```

Python Básico - Modos de imprimir

5 - Forma de imprimir na tela

```
print(f'x: {x}, y: {y}')
print(f'Resulta: {x * y}')
```

Python Básico - Prática

Exercício

Faça um Programa que peça um número e então mostre a mensagem.
O número informado foi [número]

Python Básico - Solução

1 - programa_1.py

```
number = input("Digite um número: ")

content = "O número informado foi"

print(content, number)
```

Python Básico - Solução

2 - programa_2.py

```
number = input("Informe o número: ")


print(f"O número informado foi {number}")
```

Python Básico - Solução

3 - programa_3.py

```
print(f"O número informado foi {input('Qual o número: ')}")
```

Python Básico - Prática

Exercício

Faça um Programa que peça as 4 notas bimestrais e mostre a média.

Python Básico - Solução

1 - programa_1.py

```
nota_1 = input("1º nota do bimestre: ")
nota_2 = input("2º nota do bimestre: ")
nota_3 = input("3º nota do bimestre: ")
nota_4 = input("4º nota do bimestre: ")

semestre_1 = float(nota_1) + float(nota_2)
semestre_2 = float(nota_3) + float(nota_4)

soma = semestre_1 + semestre_2
média = soma / 4

print(média)
```

Python Básico - Solução

2 - programa_2.py

```
nota_1 = float(input("1º nota do bimestre: "))
nota_2 = float(input("2º nota do bimestre: "))
nota_3 = float(input("3º nota do bimestre: "))
nota_4 = float(input("4º nota do bimestre: "))

média = (nota_1 + nota_2 + nota_3 + nota_4) / 4

print(média)
```

Python Básico - Solução

3 - programa_3.py

```
soma = 0
i = 1; nota = input(f"{i}º nota do bimestre: ")
soma = soma + float(nota)
i = i + 1; nota = input(f"{i}º nota do bimestre: ")
soma = soma + float(nota)

i += 1
nota = input(f"{i}º nota do bimestre: ")
soma += float(nota)
i += 1
nota = input(f"{i}º nota do bimestre: ")
soma += float(nota)

média = soma / i
print(média)
```

Python Básico - Prática

Exercício

Faça um Programa que peça a temperatura em graus Farenheit, transforme e mostre a temperatura em graus Celsius.

$$C = \frac{5(F - 32)}{9}$$

Python Básico - Solução

1 - programa_1.py

```
farenheit = float(input("Temperatura Farenheit: "))

c = (5 * (farenheit - 32)) / 9

print("Temperatura em ºC é", c)
```

Python Básico - list()

Listas []

```
r_index = [0, 1, 2, 3]
l_index = [-4, -3, -2, -1]

str_list = ['U', 'F', 'P', 'E']

print(str_list[0], str_list[-4])
print(str_list[1], str_list[-3])
print(str_list[2], str_list[-2])
print(str_list[3], str_list[-1])
```

Python Básico - `list()`

Listas []

```
str_list = ['U', 'F', 'P', 'E']
print(str_list[0::])
print(str_list[1::])
print(str_list[2::])
print(str_list[3::])
```

[start : end : step]

Python Básico - list()

Listas []

```
>>> a = ["A", "C", 1, 2, 5.0]
>>> print(a[0])
"A"
>>> print(len(a))
5
>>> type(a[4])
<class 'float'>
>>> type(a[0])
<class 'str'>
```

Python Básico - `list()`

É legal mexer com listas

```
liguagem = "python"
caracter = list(liguagem)
print(caracter)

palavra = "".join(caracter)
print(palavra)
```

Python Básico - `list()`

Lista é mutável

```
matrix = [0]
matrix = matrix * 3
print(matrix)
matrix[1] = 2
print(matrix)
```

Python Básico - `list()`

Builtins `list()`

- `len(list)`

Python Básico - `list()`

Builtins `list()`

- `len(list)`
- `max(list)`

Python Básico - `list()`

Builtins `list()`

- `len(list)`
- `max(list)`
- `min(list)`

Python Básico - `list()`

Builtins `list()`

- `len(list)`
- `max(list)`
- `min(list)`
- `list(sequence)`

Python Básico - `list()`

Methods `list()`

- `list.append(object)`

Python Básico - `list()`

Methods `list()`

- `list.append(object)`
- `list.count(object)`

Python Básico - `list()`

Methods `list()`

- `list.append(object)`
- `list.count(object)`
- `list.extend(sequence)`

Python Básico - `list()`

Methods `list()`

- `list.append(object)`
- `list.count(object)`
- `list.extend(sequence)`
- `list.index(object)`

Python Básico - `list()`

Methods `list()`

- `list.append(object)`
- `list.count(object)`
- `list.extend(sequence)`
- `list.index(object)`
- `list.insert(index, object)`

Python Básico - `list()`

Methods `list()`

- `list.append(object)`
- `list.count(object)`
- `list.extend(sequence)`
- `list.index(object)`
- `list.insert(index, object)`
- `list.pop(object=list[-1])`

Python Básico - `list()`

Methods `list()`

- `list.append(object)`
- `list.count(object)`
- `list.extend(sequence)`
- `list.index(object)`
- `list.insert(index, object)`
- `list.pop(object=list[-1])`
- `list.remove(object)`

Python Básico - `list()`

Methods `list()`

- `list.append(object)`
- `list.count(object)`
- `list.extend(sequence)`
- `list.index(object)`
- `list.insert(index, object)`
- `list.pop(object=list[-1])`
- `list.remove(object)`
- `list.reverse(object)`

Python Básico - `list()`

Methods `list()`

- `list.append(object)`
- `list.count(object)`
- `list.extend(sequence)`
- `list.index(object)`
- `list.insert(index, object)`
- `list.pop(object=list[-1])`
- `list.remove(object)`
- `list.reverse(object)`
- `list.sort(object)`

Python Básico - list()

About list()

```
>>> help(list) or help([])
```

```
>>> dir(list) or dir([])
```

Python Básico - Prática

Exercício

Faça um Programa que peça as 4 notas bimestrais e adicione em uma lista, depois calcule e mostre as informações a média, mediana, desvio padrão, nota máxima e nota mínima.

Média:

$$\bar{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i$$

Desvio padrão:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^2}$$

Desvio padrão amostral:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n-1}(x_i - \bar{x})^2}$$

Python Básico - Prática

Iterations - for

```
>>> notas = [10, 4.5, 2.8, 6.0]
>>> for nt in notas:
>>>     print(nt)
10
4.5
2.8
6.0
```

Obs: Na linha do **for** tem que ser finalizado com **:** e nas linhas abaixo coloque sempre *4 espaços*, logo podes escrever o código que será executado pelo **for**.

Refaça o exercício anterior, mas agora use **for**.

Python Básico - dict()

Dicionários dict()

```
dic = {'lang': "python"}
print(dic['lang'])

dic["lib"] = 'django'
dic["lib"] = 'numpy'
dic["lib"] = 'pandas'
print(dic)

print("{}\n".format(dic.keys()))
print("%s\n" %dic.values())
```

About dict()

```
>>> help(dict)
```

```
>>> dir(dict)
```

Python Básico

Tuplas tuple()

```
>>> 1,2,3
(1, 2, 3)
>>> tuple([1, 2, 3])
(1, 2, 3)
>>> t = tuple([0, 0, 0])
>>> t[1] = 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
>>> l = list(t)
>>> print(l)
[0, 0, 0]
>>> l[1] = 2
>>> print(l)
[0, 2, 0]
```

Python Básico

About tuple()

```
>>> help(tuple)
```

```
>>> dir(tuple)
```

Python Básico

Conjuntos set()

```
>>> set([1,2,3,3,2,1])
{1, 2, 3}
>>> s = {1, 2, 3}
>>> s.union({2,3,5})
{1, 2, 3, 5}
```

Python Básico

About set()

```
>>> help(set)
```

```
>>> dir(set)
```

Estrutura de controle

Delimitado por indentação

```
a = 0
print("O valor de a é ")
if a == 0:
    print "zero"
else:
    print a
```

- 4 espaços representa uma indentação.

Estrutura de controle

Delimitado por indentação

```
a = 0
print("O valor de a é ")
if a == 0:
    print "zero"
else:
    print a
```

- 4 espaços representa uma indentação.
- O bloco da indetação é inicializado por :

Estrutura de controle

- `if`

Estrutura de controle

- `if`
- `elif`

Estrutura de controle

- `if`
- `elif`
- `else`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`
- `and`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`
- `and`
- `or`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`
- `and`
- `or`
- `is`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`
- `and`
- `or`
- `is`
- `not`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`
- `and`
- `or`
- `is`
- `not`
- `==`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`
- `and`
- `or`
- `is`
- `not`
- `==`
- `!=`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`
- `and`
- `or`
- `is`
- `not`
- `==`
- `!=`
- `<=`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`
- `and`
- `or`
- `is`
- `not`
- `==`
- `!=`
- `<=`
- `>=`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`
- `and`
- `or`
- `is`
- `not`
- `==`
- `!=`
- `<=`
- `>=`
- `<`

Estrutura de controle

- `if`
- `elif`
- `else`
- `for-else`
- `while-else`
- `and`
- `or`
- `is`
- `not`
- `==`
- `!=`
- `<=`
- `>=`
- `<`
- `>`

Repetições

- `for`

Repetições

- `for`
- `while`

Repetições

- `for`
- `while`
- `iter`

Repetições

- `for`
- `while`
- `iter`
- `compreension`

Arquivos

- `open()`

Arquivos

- `open()`
  - `read()`

Arquivos

- `open()`
  - `read()`
  - `write()`

Arquivos

- `open()`
  - `read()`
  - `write()`
  - `close()`

Arquivos

- `open()`
  - `read()`
  - `write()`
  - `close()`
- `with`

Arquivos

- `open()`
  - `read()`
  - `write()`
  - `close()`
- `with`
  - `open()`

Arquivos

- `open()`
  - `read()`
  - `write()`
  - `close()`
- `with`
  - `open()`
  - `read()`

Arquivos

- `open()`
  - `read()`
  - `write()`
  - `close()`
- `with`
  - `open()`
  - `read()`
  - `write()`

Módulos padrões `builtins`

- Zen python

Módulos padrões `builtins`

- Zen python
  - `this`

Módulos padrões `builtins`

- Zen python
  - `this`
- `builtins`

Módulos padrões `builtins`

- Zen python
  - `this`
- `builtins`
- `sys`

Módulos padrões `builtins`

- Zen python
  - `this`
- `builtins`
- `sys`
  - módulos

Módulos padrões `builtins`

- Zen python
  - `this`
- `builtins`
- `sys`
  - módulos
- `math`

Módulos padrões `builtins`

- Zen python
  - `this`
- `builtins`
- `sys`
  - módulos
- `math`
- `os`

Módulos padrões `builtins`

- Zen python
  - `this`
- `builtins`
- `sys`
  - módulos
- `math`
- `os`
- `glob`

Módulos padrões `builtins`

- Zen python
  - `this`
- `builtins`
- `sys`
  - módulos
- `math`
- `os`
- `glob`
- `pathlib`

Módulos padrões `builtins`

- Zen python
  - `this`
- `builtins`
- `sys`
  - módulos
- `math`
- `os`
- `glob`
- `pathlib`
- `pickle`

Módulos padrões `builtins`

- Zen python
  - `this`
- `builtins`
- `sys`
  - módulos
- `math`
- `os`
- `glob`
- `pathlib`
- `pickle`
- `json`

Funções

- `def`

Funções

- `def`
- `async def`

Módulos e Pacotes

- Criação de módulos

Módulos e Pacotes

- Criação de módulos
- Usar módulos de terceiros

Classes

- `class`

Classes

- `class`
- `__init__`

Classes

- `class`
- `__init__`
- `__magic__`

Python funcional

- `lambda`

Python funcional

- `lambda`
- `map`

Python funcional

- `lambda`
- `map`
- `reduce`

Python funcional

- `lambda`
- `map`
- `reduce`
- `filter`

Python funcional

- `lambda`
- `map`
- `reduce`
- `filter`
- `compreension`

Python funcional

- `lambda`
- `map`
- `reduce`
- `filter`
- `compreensions`
- `return values of if-else`

Python funcional

- `lambda`
- `map`
- `reduce`
- `filter`
- `compreension`
- `return values of if-else`
- `return values of or`