

# Introdução à Linguagem Python

Paulo Henrique Junqueira Amorim

Novembro de 2018

# Agenda

Introdução

Entrada

Condicional

Laço de Repetição

Lista

Exceção

Tupla

Dicionário

Função

Classe

Arquivo

## Formação:



Mestre em Ciência da Computação (2012-2015)



Bacharel em Ciência da Computação  
(2006-2009)



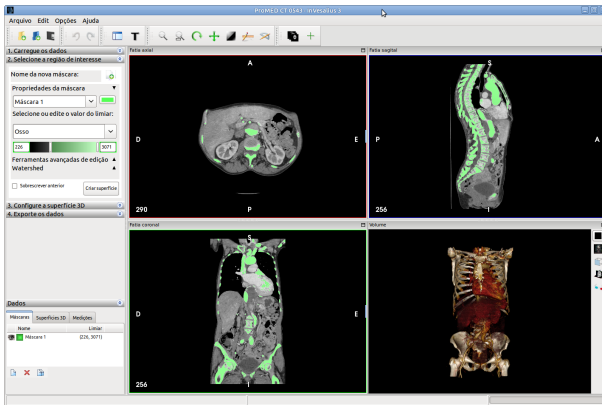
Técnico em Informática (2003-2005)

## Profissionalmente:



Estagiário / Pesquisador (2007-Atualmente)

## CTI Renato Archer - Projeto InVesalius:



<http://www.cti.gov.br/invesalius>

# Introdução

## Python - Características

Linguagem fácil e intuitiva;

# Introdução

## Python - Características

Linguagem fácil e intuitiva;

Código aberto;

# Introdução

## Python - Características

Linguagem fácil e intuitiva;

Código aberto;

Sintaxe tão interlegível quanto inglês;

# Introdução

## Python - Características

Linguagem fácil e intuitiva;

Código aberto;

Sintaxe tão interlegível quanto inglês;

Desenvolvimento rápido;



# Introdução

## Python - Características

Linguagem fácil e intuitiva;

Código aberto;

Sintaxe tão interlegível quanto inglês;

Desenvolvimento rápido;

Interpretada;

# Introdução

## Python - Características

Linguagem fácil e intuitiva;

Código aberto;

Sintaxe tão interlegível quanto inglês;

Desenvolvimento rápido;

Interpretada;

Procedural;

# Introdução

## Python - Características

Linguagem fácil e intuitiva;

Código aberto;

Sintaxe tão interlegível quanto inglês;

Desenvolvimento rápido;

Interpretada;

Procedural;

Orientada Objetos;

# Introdução

## Python - Características

Linguagem fácil e intuitiva;

Código aberto;

Sintaxe tão interlegível quanto inglês;

Desenvolvimento rápido;

Interpretada;

Procedural;

Orientada Objetos;

Tudo é um objeto;

# Introdução

## Python - Características

Linguagem fácil e intuitiva;

Código aberto;

Sintaxe tão interlegível quanto inglês;

Desenvolvimento rápido;

Interpretada;

Procedural;

Orientada Objetos;

Tudo é um objeto;

Comunica com C/C++ através de Wrappers.

# Introdução

Python - O criador...



Figura: Guido van Rossum - Holanda

# Introdução

Python - O nome...



Figura: Monty Python

# Introdução

## Algumas Bibliotecas

- Numpy - Numeric Python.
- Scipy - Scientific python.
- SymPy (Matemática Simbólica)
- PIL - Python Imaging Library
- Matplotlib
- Scikit (learn, image ...)
- Astropy
- PyWavelets
- Python - OpenCV
- VTK - Visualization Toolkit
- ITK - Insight Segmentation and Registration Toolkit
- Mahotas (Visão Computacional)
- GDCM - Grassroots DICOM
- PyDICOM

- Simplejson
- SQL Alchemy (ORM)
- Peewee (ORM)
- Simplejson
- TensorFlow (Deep Learning)
- Keras (Deep Learning)
- Pygame (Jogos)
- Pyglet (Jogos e Multimedia)
- wxPython (Interface Gráfica)
- PyQt (Interface Gráfica)
- pyGTK (Interface Gráfica)
- BeautifulSoup (parser XML e HTML)
- Requests
- Twisted (Manipular Protocolos Redes)
- Peewee (Sniff)



# Introdução

## Alguns Frameworks Web



Pyramid™



Flask



WEB2PY



# Introdução

## Algumas IDE

- PyCharm
- Spyder
- Atom
- Eclipse com pyDev
- Visual Studio Code

# Introdução

## Python - Tipos

Números: int, long, float, complex

Booleano: True, False

String: str, unicode

Listas e Tuplas: list, tuple

Dicionários: dict

Arquivos: file

Nulo: None

# Introdução

## Python - Operadores

### Comparaçãõ

==, !=, <, <=, >, >=

### Lógicos

and or not

### Matemáticos

+, -, \*, /, %, \*\*

### Bit

<<, >>, ^, |, ~, &

# Introdução

## Python - Um pequeno exemplo

```
1  # -*- coding: UTF-8 -*-  
2  
3  print("Olá Mundo!!")
```

# Introdução

## Python - Comentário

```
1  # -*- coding: UTF-8 -*-
2
3  """
4  Aqui está um bloco de
5
6  comentário
7
8  """
9
10 #Um comentário de linha
11 print("Olá Mundo!!")
```

# Entrada

```
1  # -*- coding: UTF-8 -*-
2
3  nome = input("Qual o seu nome?
              ")
4
5  print("Olá " + nome)
```

## Condicional

```
1  # -*- coding: UTF-8 -*-
2
3  linguagem = input("Digite a
                     linguagem: ")
4
5  if linguagem == "python":
6      print( "= ")
7  else:
8      print( " ( = " )
```

Blocos são delimitados por 4 espaços ou 1 tab



# Condicional

Uma forma de substituir Switch/Case

```
1  # -*- coding: UTF-8 -*-
2
3  linguagem = input("Digita a
                     linguagem: ")
4
5  if linguagem == "python":
6      print( "=) ")
7  elif linguagem == "c":
8      print( "=| ")
9  elif linguagem == "c++":
10     print( ":| ")
11  else:
12     print( "=(" )
```

## Condicional

```
1  # -*- coding: UTF-8 -*-
2
3  valor = int(input("Digite um n
                     úmero: "))
4
5  if valor >= 0 and valor <= 30:
6      print("Está entre 0 e 30")
7  else:
8      print("Não está entre 0 e
              30")
```

## Expressão Condicional

```
1  # -*- coding: UTF-8 -*-
2
3  valor = int(input("Digite um n
                     úmero: "))
4
5  resultado = "par" if valor % 2
               == 0 else "impar"
6
7  print(resultado)
```

## Laço de repetição

```
1  # -*- coding: UTF-8 -*-  
2  
3  for i in range(0,10):  
4      print(i)
```

de 0 (inclusive) até 10 (exclusive)

## Laço de repetição

```
1  # -*- coding: UTF-8 -*-  
2  
3  for i in range(0,10,2):  
4      print(i)
```

de 0 (inclusive) até 10 (exclusive), pulando 1 número a cada iteração.

## Laço de repetição

```
1  # -*- coding: UTF-8 -*-
2  n = 0
3  while (n <= 10):
4      print(n)
5      n += 1
```

Agora sim.. de 0 até 10.

# Lista

```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ["Abacate", "Manga", "
            Maçã", "Abacaxi", "Caju"]
4
5  for f in frutas:
6      print(f)
```

# Lista






## Ordenar

```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ["Abacate", "Manga", "
           Maçã", "Abacaxi", "Caju"]
4  frutas.sort()
5
6  for f in frutas:
7      print(f)
```



# Lista

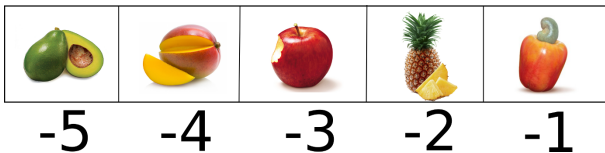
## Índice

				
0	1	2	3	4

```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ["Abacate", "Manga", "
           Maçã", "Abacaxi", "Caju"]
4
5  print(frutas[3])
```

# Lista

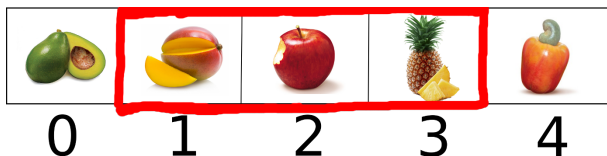
## Índice



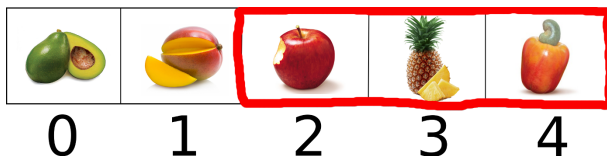
```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ["Abacate", "Manga", "
           Maçã", "Abacaxi", "Caju"]
4
5  print(frutas[-1])
```

# Lista

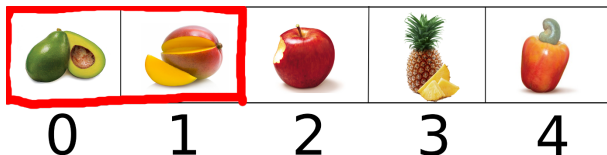
## Índice



```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ["Abacate", "Manga", "
           Maçã", "Abacaxi", "Caju"]
4
5  print(frutas[1:4])
```



```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ["Abacate", "Manga", "
           Maçã", "Abacaxi", "Caju"]
4
5  print(frutas[2:])
```



```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ["Abacate", "Manga", "
           Maçã", "Abacaxi", "Caju"]
4
5  print(frutas[:2])
```

# Lista

## Métodos

- **append(object)**
- **count(value)**
- **extend(iterable)**
- **index(value, [start, [stop]])**
- **insert(index, object)**

- **pop([index])**
- **remove(value)**
- **reverse()**
- **sort(cmp=None, key=None, re-verse=False)**

# Lista

## Exercício

- Além da lista de frutas, criar uma outra lista chamada legumes e preenchê-la;

# Lista

## Exercício

- Além da lista de frutas, criar uma outra lista chamada legumes e preenchê-la;
- Armazenar cada lista em uma terceira lista (cesto);



# Lista

## Exercício

- Além da lista de frutas, criar uma outra lista chamada legumes e preenchê-la;
- Armazenar cada lista em uma terceira lista (cesto);
- Exibir o conteúdo das duas posições da terceira lista;

# Lista

## Exercício

- Além da lista de frutas, criar uma outra lista chamada legumes e preenchê-la;
- Armazenar cada lista em uma terceira lista (cesto);
- Exibir o conteúdo das duas posições da terceira lista;
- Unir a lista frutas com legumes e exibi-las.

# Lista

## Exercício (Uma possível solução)

```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ["Abacate", "Manga", "Maçã", "Abacaxi", "Caju"]
4
5  legumes = ["Batata", "Cenoura", "Abóbora", "Mandioca"]
6
7  cesto = [frutas, legumes]
8
9  for c in cesto:
10     print(c)
11
12  items = frutas + legumes
13  print(items)
14
15  #ou
16
17  frutas.extend(legumes)
18  print(frutas)
```

## Exceção

```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ["Abacate", "Manga", "
            Maçã", "Abacaxi", "Caju"]
4
5  try:
6      print(frutas[10])
7  except (IndexError):
8      print("Erro")
```

# Tupla

Tuplas são imutáveis!

```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ("Abacate", "Manga",
            "Maçã", "Abacaxi", "Caju")
```

```
1  # -*- coding: UTF-8 -*-
2  frutas = ("Abacate", "Manga",
            "Maçã", "Abacaxi", "Caju")
3  frutas[0] = "Melão"
```

TypeError: 'tuple' object does not support item assignment

## Dicionário

```
1  # -*- coding: UTF-8 -*-
2
3  frutas = {"Abacate":1, "Manga":3, "Maçã":10, "Abacaxi":3,
            "Caju":1}
4
5  legumes = {"Batata":2, "Cenoura":3}
6
7  print("Caju: " + str(frutas["Caju"]))
8  print("Batata: " + str(legumes["Batata"]))
```

# Dicionário

## Alguns Métodos

- add
- clear
- copy
- difference
- discard

- intersection
- pop
- remove
- union
- update

## Função

```
1  # -*- coding: UTF-8 -*-
2
3  def soma(x, y):
4      total = x + y
5      return total
6
7  a = int(input("Digite o
               primeiro número: "))
8  b = int(input("Digite o
               segundo número: "))
9
10 resultado = soma(a, b)
11 print(resultado)
```



# Função

A variável `__name__`

```
1  # -*- coding: UTF-8 -*-
2
3  def soma(x, y):
4      total = x + y
5      return total
6
7  if __name__ == "__main__":
8      a = int(input("Digite o
9          primeiro número: "))
10
11     b = int(input("Digite o
12         segundo número: "))
13
14     resultado = soma(a, b)
15     print(resultado)
```

# Função

## O import

```
1  # -*- coding: UTF-8 -*-
2  import soma
3
4  if __name__ == "__main__":
5      print(soma.soma(1,2))
```

# Função

Parâmetros passados em **\*tupla**

```
1  # -*- coding: UTF-8 -*-
2
3  def soma(*val):
4      print(val)
5      total = 0
6      for v in val:
7          total += v
8      return total
9
10 if __name__ == "__main__":
11     resultado = soma(10,5,20)
12     print(resultado)
```

# Função

Parâmetros passados em **\*\*dicionário**

```
1  # -*- coding: UTF-8 -*-
2
3  def soma(**val):
4      print(val)
5      total = val["x"] + val["y"]
6      return total
7
8  if __name__ == "__main__":
9      resultado = soma(x=10, y
10         =12)
11      print(resultado)
```

# Função

## Argumento default

```
1  # -*- coding: UTF-8 -*-
2
3  def soma(x, y=10):
4      total = x + y
5      return total
6
7  if __name__ == "__main__":
8
9      resultado = soma(1)
10     print(resultado)
```

# Classe

```
1  # -*- coding: UTF-8 -*-
2  class Cesto:
3
4      def __init__(self):
5          self.items = []
6
7      def add(self, item):
8          self.items.append(item)
9
10     def get(self):
11         return self.items
12
13     if __name__ == "__main__":
14
15         c = Cesto()
16         c.add("Abacaxi")
17
18         f = c.get()
19         print(f)
```

# Classe - Herança

```
1  # -*- coding: UTF-8 -*-
2  class Item:
3      def __init__(self):
4          self.cor = None
5
6  class Fruta(Item):
7      def __init__(self):
8          self.nome = None
9          super().__init__()
10
11      def set_fruta(self, nome, cor):
12          self.nome = nome
13          self.cor = cor
14
15      def get_fruta(self):
16          return [self.nome, self.cor]
17
18  if __name__ == "__main__":
19      c = Fruta()
20      c.set_fruta("Maçã", "Vermelha")
21      print(c.get_fruta())
```

# Arquivo

## Escrita

```
1  # -*- coding: UTF-8 -*-
2
3  frutas = ["Abacate", "Manga", "
           Maçã", "Abacaxi", "Caju"]
4
5  f = open("frutas.txt", "w")
6
7  for item in frutas:
8      f.write(item + "\n")
9
10 f.close()
```



# Arquivo

## Leitura

```
1  # -*- coding: UTF-8 -*-
2
3  f = open("frutas.txt", "r")
4  lines = f.readlines()
5
6  for l in lines:
7      print(l)
8
9  f.close()
```

# Arquivo

## Exercício

- Baixar o arquivo: <https://goo.gl/sA5bMu>

# Arquivo

## Exercício

- Baixar o arquivo: <https://goo.gl/sA5bMu>
- Ler o arquivo texto e exibir cada linha;

# Arquivo

## Exercício

- Baixar o arquivo: <https://goo.gl/sA5bMu>
- Ler o arquivo texto e exibir cada linha;
- Separar o nome e as notas (dica usar **split**);

# Arquivo

## Exercício

- Baixar o arquivo: <https://goo.gl/sA5bMu>
- Ler o arquivo texto e exibir cada linha;
- Separar o nome e as notas (dica usar **split**);
- Exibir a média de cada aluno.

# Arquivo

## Exercício (Uma possível solução)

```
1  # -*- coding: UTF-8 -*-
2
3  f = open("notas.txt", "r")
4  lines = f.readlines()
5
6  for l in lines:
7      print(l)
8
9  for l in lines:
10     sep = l.split(" ")
11     nome = sep[0]
12     n1 = float(sep[1])
13     n2 = float(sep[2])
14     n3 = float(sep[3])
15     media = (n1 + n2 + n3) / 3
16     print(nome, media)
17
18  f.close()
```

# FIM

## Contato

paulojamorim @ gmail.com

<http://github.com/paulojamorim>