

Introdução ao Python para computação científica

André Pacheco

`pacheco.comp@gmail.com`

Universidade Federal do Espírito Santo



Março de 2018

AGENDA

INTRODUÇÃO

CONCEITOS BÁSICOS - PYTHON

NUMPY E SCIPY

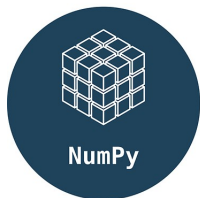
MATPLOTLIB

REDES NEURAIIS COM SCIKIT-LEARN

INTRODUÇÃO

- ▶ O que é python?
 - ▶ Linguagem de programação interpretada
 - ▶ Orientada a objetos
 - ▶ Tipagem dinâmica
 - ▶ **Muito utilizada no meio científico**
- ▶ Por que python?
 - ▶ Diversas biblioteca e frameworks
 - ▶ Permite construção de scripts de maneira fácil
 - ▶ Plotagem de gráficos de maneira fácil
 - ▶ Comunidade de desenvolvimento ativa
 - ▶ **Software livre**
- ▶ Python 2.x × 3.x?
 - ▶ Apesar de lançado em 2008, a migração para o python 3.x vem sendo lenta
 - ▶ Muitas bibliotecas precisam ser atualizadas
 - ▶ Esta introdução será realizada no python 2.7.x

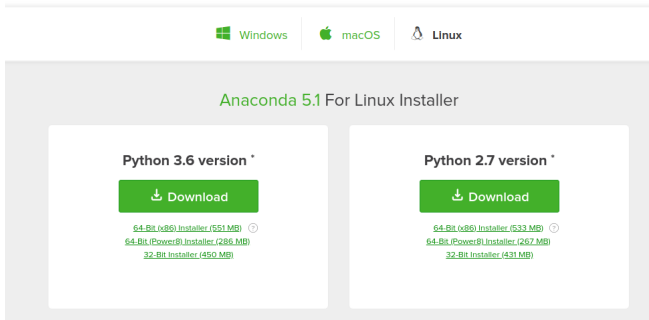
BIBLIOTECAS NECESSÁRIAS



- ▶ **Kit básico:**
 - ▶ Numpy
 - ▶ Scipy
 - ▶ Matplotlib
 - ▶ Scikit-Learn
- ▶ **Extras:**
 - ▶ Pandas
 - ▶ Seaborn
 - ▶ Tensorflow
- ▶ **Plataformas de desenvolvimento:**
 - ▶ Spyder
 - ▶ Jupyter notebook

INSTALAÇÃO

- ▶ A instalação pode ser realizada para qualquer sistema operacional
- ▶ Pode ser realizada manualmente, baixando cada pacote
- ▶ Pode ser realizado por meio do **pacote Anaconda**
 - ▶ `https://www.anaconda.com/download`
 - ▶ Compatível com todos os SOs



The screenshot shows the Anaconda 5.1 For Linux Installer download page. At the top, there are three tabs: Windows, macOS, and Linux. The Linux tab is selected. Below the tabs, the text "Anaconda 5.1 For Linux Installer" is displayed. There are two main sections: "Python 3.6 version" and "Python 2.7 version". Each section has a green "Download" button. Below the buttons, there are links for different installer types and bitnesses, such as "64-Bit (x86) Installer (551 MB)", "64-Bit (PowerBI) Installer (286 MB)", and "32-Bit Installer (450 MB)".

Windows macOS Linux

Anaconda 5.1 For Linux Installer

Python 3.6 version *

Download

[64-Bit \(x86\) Installer \(551 MB\)](#) ⓘ
[64-Bit \(PowerBI\) Installer \(286 MB\)](#)
[32-Bit Installer \(450 MB\)](#)

Python 2.7 version *

Download

[64-Bit \(x86\) Installer \(533 MB\)](#) ⓘ
[64-Bit \(PowerBI\) Installer \(267 MB\)](#)
[32-Bit Installer \(431 MB\)](#)

INSTALAÇÃO

- ▶ Após instalado o pacote anaconda, para instalar o Spyder:
 - ▶ `conda install -c anaconda spyder`
- ▶ O Jupyter Notebook já vem instalado!



VAMOS COMEÇAR!



INTRODUÇÃO

- ▶ Antes de usar as bibliotecas temos que dominar o próprio Python
- ▶ Esta aula não é um curso de programação. É esperado de vocês:
 - ▶ Conhecimento da sintaxe de alguma linguagem de programação
 - ▶ Lógica de programação
- ▶ Vamos trabalhar basicamente a sintaxe do Python

COMO USAR O PYTHON?

- ▶ Podemos trabalhar com Python:
 - ▶ Via console (terminal ou CMD)
 - ▶ Via IDE e console
 - ▶ Via IPython
 - ▶ Via IDE e IPython

- ▶ Vamos trabalhar ao longo dessa introdução:
 - ▶ **Spyder**
 - ▶ **Jupyter Notebook**

PRIMEIRO PROGRAMA EM PYTHON

- ▶ Primeiro programa em Python:

```
>> print("Hello World!")  
Hello World!  
>>
```

- ▶ **Não é usado ; no final da instrução**

OPERAÇÕES ARITIMÉTICAS

- ▶ Sintaxe similar as demais linguagens de programação

Símbolo	Operação
+	Soma
-	Subtração
*	Multiplificação
/	Divisão
**	Exponenciação
//	Divisão inteira
%	Resto da divisão

OPERAÇÕES ARITIMÉTICAS

A ordem de precedência dos operadores é a mesma utilizada na matemática

```
>> 3 + 4 - 1
```

```
6
```

```
>> 8 - 3 * 3
```

```
-1
```

```
>> (8 - 3) * 3
```

```
15
```

```
>> 2**2
```

```
4
```

LÓGICA BOOLEANA

- Os operadores de comparação

Símbolo	Comparação
==	Igualdade
>	Maior
<	Menor
>=	Maior ou igual
<=	Menor ou igual
!=	Diferente

LÓGICA BOOLEANA

- Os operadores de comparação

Símbolo	Comparação
==	Igualdade
>	Maior
<	Menor
>=	Maior ou igual
<=	Menor ou igual
!=	Diferente

LÓGICA BOOLEANA

- ▶ Exemplo do uso dos operadores:

```
>> 9 == 9
True
>> 10.2 >= 12
False
>> x = 7 != 7
False
>> print(x)
False
```

LÓGICA BOOLEANA

- ▶ Os operadores de lógica booleana

Símbolo	Operação lógica
or	ou
and	e
not	Negação

- ▶ Exemplo:

```
>> 1 >= 10 or 7.7 < 11
True
>> 1 >= 10 and 7.7 < 11
False
>> not True
>> False
```

STRING

- ▶ Aspas duplas ou simples
- ▶ Operações sobrecarregadas

```
>> print('programar ' + 'em ' + 'Python')  
programar em python
```

```
>> 'teste' * 2  
'testeteste'
```

```
>> '10' * '10'  
TypeError: can't multiply sequence by non-int of  
type 'str'
```

VARIÁVEIS

- ▶ Tipagem dinâmica
- ▶ Todavia, não significa que ela não tenha um tipo

```
>> x = 10
>> print x
10
>> type(x)
<type, 'int'>
>> print (x+2)
12
>> x = 10.1
<type, 'float'>
```

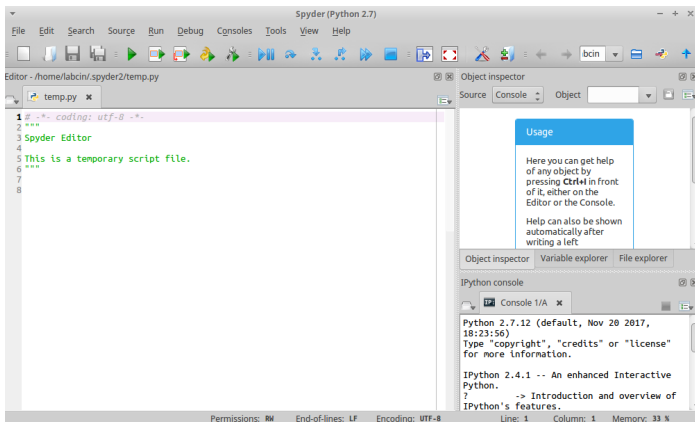
- ▶ **Atenção!** Toda atribuição em python é feita por REFERÊNCIA!

DÚVIDAS?



USANDO UMA IDE

- ▶ Uma IDE facilita o desenvolvimento de código
- ▶ Como nossos códigos vão usar mais linhas, é melhor migrarmos para IDE desde já



ESTRUTURA DE CONTROLE DE FLUXO

- ▶ Similar a maioria das linguagens:

```
if (expressao) :  
    codigo
```

- ▶ Python utiliza a indentação para delimitar blocos!

ESTRUTURA DE CONTROLE DE FLUXO

- ▶ Respeitando a indentação, podemos aninhar if's

```
x = 10
if x % 2 == 0: #verificando se o numero e par
    print 'E par'
    if x >= 10:
        print 'E maior do que 10'
print x + 20
```

ESTRUTURA DE CONTROLE DE FLUXO

- ▶ E por trás de um "grande" *if*, sempre existe um *else*

```
x = float (input('Digite um numero: '))
if x % 2 == 0: #verificando se o numero e par
    print 'E par'
else:
    print 'E impar'
```

ESTRUTURAS DE REPETIÇÃO

- ▶ Estruturas para avaliar uma condição repetidas vezes

```
while (condicao):  
    codigo...
```

- ▶ Exemplo:

```
x = 1  
while (x < 4):  
    print (x)  
    x = x + 1  
print 'Acabou'
```

ESTRUTURAS DE REPETIÇÃO

- ▶ Podemos utilizar também a estrutura de repetição *for*:

```
for i in range(4):  
    print i  
print 'Acabou'
```

- ▶ *range* é utilizado para gerar um sequencia de valores de acordo com o parâmetro

DÚVIDAS?



CONTAINERS

- ▶ São estruturas que armazenam dados em Python
- ▶ 3 tipos básicos:
 - ▶ **Listas**
 - ▶ Tuplas
 - ▶ Dicionário
- ▶ Nesta introdução vamos trabalhar apenas com listas
 - ▶ Mas as demais são semelhantes
 - ▶ Vocês são encorajados a estudá-las por conta própria

LISTAS

- conjunto de valores em que cada valor é representado por um índice

```
lista1 = [1,2,3,4]
lista2 = [0.0, 0.1, 0.2]
lista3 = [True, False]
palavras = ['Goku', 'Gohan', 'Vegeta']
```

```
print lista1
print lista2[0]
print lista3[1]
print palavras[2]
```

LISTAS

- ▶ Podemos iniciar uma lista vazia

```
lista1 = []  
lista2 = list()  
print (lista1, lista2)
```

- ▶ Ou com elementos de tipos

```
lista = [1, 'a', '1.0', 'Ola']  
print (lista)
```

LISTAS

- ▶ Se tentarmos acessar um índice que não existe na lista, é retornado um erro:

```
lista1 = []  
lista2 = list()  
print (lista1, lista2)
```

```
l1 = [0,1,2,3]  
print l1[4]  
# IndexError: list index out of range
```

- ▶ Perceba que o índice começa sempre do número zero

LISTAS

- ▶ Lembra do *range*? Ele retorna uma lista!

```
# Sequencia de 0 a 4
```

```
a = range (5)
```

```
print a
```

```
# Sequencia de 5 a 9
```

```
b = range (5,10)
```

```
# Sequencia de 5 a 20, indo de 5 em 5
```

```
c = range (5,21,5)
```

```
print c
```

```
# Saida:
```

```
# [0,1,2,3,4]
```

```
# [5,6,7,8,9]
```

```
# [5,10,15,20]
```

LISTAS

- ▶ Para representar matrizes podemos inserir lista dentro de lista:

```
# Matriz = [1 2  
##### 3 4]  
M = [[1,2],[3,4]]  
print M  
print M[0][1]
```

OPERAÇÕES COM LISTAS

- ▶ Acessando posições na lista:

```
n = [2, 2, 2, 2]
# Acessa o segundo elemento da lista e o igual a 8
n[1] = 8
# Acessa o ultimo elemento da lista e o iguala a 10
n[-1] = 10
print n
```

OPERAÇÕES COM LISTAS

- ▶ O sinal + concatena duas listas:
 - ▶ Guarde essa informação

```
n = [0, 1, 2]
m = [3, 4, 5]
print m+n
```

```
# Saida:
# [0, 1, 2, 3, 4, 5]
```

OPERAÇÕES COM LISTAS

- ▶ Adicionando itens em uma lista vazia
- ▶ Pegando o tamanho da lista

```
# Criando uma lista vazia
n = list ()
# Adicionando elementos no fim da lista
n.append(3)
n.append(4)
n.append(5)
# Pegando o tamanho da lista
tam = len(n)
# Imprimindo a lista e o tamanho
print 'Lista: ', n, '\nTamanho: ', tam

# Saida:
# Lista: [3,4,5]
# Tamanho: 3
```

OPERAÇÕES COM LISTAS

- ▶ Funções de máximo, mínimo e soma:

```
n = [1, 4, 10, 2]
print max(n) # Valor maximo da lista
print min(n) # Valor minimo da lista
print sum(n) # Somatorio da lista

# Saída:
# 10
# 1
# 17
```

OPERAÇÕES COM LISTAS

- ▶ Utilizando um *for* para percorrer uma lista:

```
# Lista de frutas
starWars = ['Darth Vader', 'Luke', 'Chewbacca',
            'K2SO']

# Percorrendo toda lista e imprimindo cada fruta
for sw in starWars:
    print sw
```

OPERAÇÕES COM LISTAS

- ▶ Acessando posições da lista via *slicing*:

```
# Lista de frutas
starWars = ['Darth Vader', 'Luke', 'Chewbacca',
            'K2SO']

# Acessando os 2 primeiros elementos
print starWars[0:2]

# Acessando ate o terceiro
print starWars[:3]

# Acessando o ultimo elemento
print starWars[-1]

# Acessando os dois ultimos
print starWars[-2:]
```

FUNÇÕES

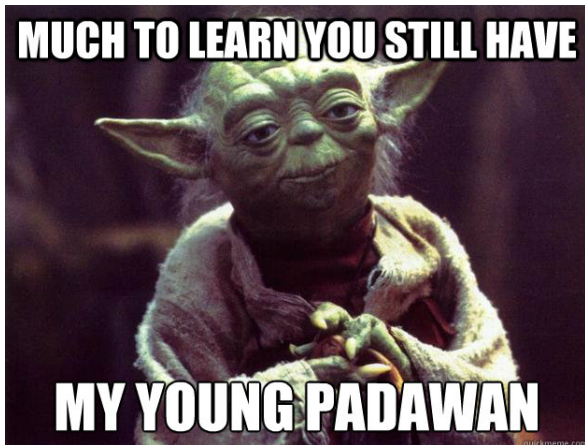
- ▶ Função é um artifício para modularizar e facilitar o desenvolvimento
- ▶ Python permite o retorno de várias variáveis e parâmetros pré definidos
- ▶ Vamos a um exemplo no Jupyter

O BÁSICO DO BÁSICO DE ORIENTAÇÃO A OBJETOS

- ▶ É basicamente um conceito de programação que facilita o desenvolvimento de programas
 - ▶ Principalmente quando o projeto se torna mais complexo e grande
- ▶ Para instanciar um objeto, definimos uma classe
- ▶ Uma classe possui basicamente:
 - ▶ Atributos
 - ▶ Métodos
- ▶ Vamos a um exemplo no Jupyter

CONCEITOS BÁSICOS

- ▶ Os conceitos básicos se encerram aqui. Porém...



NUMPY E SCIPY

- ▶ Abreviação de *numerical python* e *scientific Python*
- ▶ São fundamentais para alta performance em computação científica em python
 - ▶ Disponibiliza uma maneira eficiente de trabalhar com vetores e matrizes
 - ▶ Implementa métodos extremamente eficientes
 - ▶ Permite *broadcast*
 - ▶ Funções matemáticas padrões
- ▶ Numpy e Scipy trabalham juntos. Importamos apenas o `numpy`, mas o `scipy` é utilizado "por baixo dos panos"
 - ▶ Embora podemos utilizar o `scipy` declaradamente

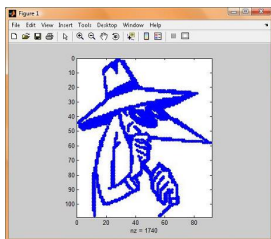
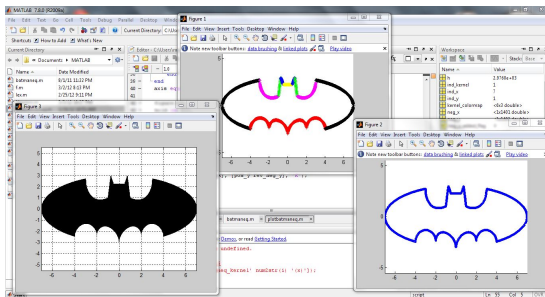
```
import numpy as np
```

MATPLOTLIB

- ▶ Biblioteca para plotar gráficos e figuras em python
- ▶ Funciona bem parecido com o MATLAB
- ▶ Para utilizar basta importar a biblioteca

```
import matplotlib.pyplot as plt
```

MATPLOTLIB

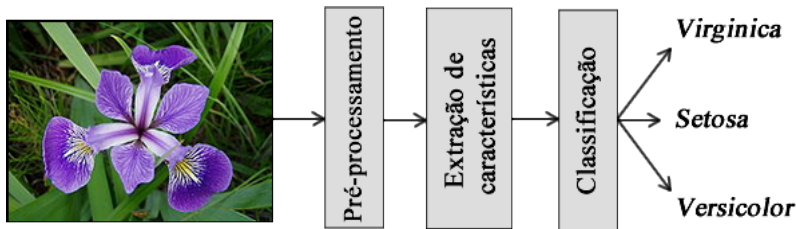


SCIKIT-LEARN

- ▶ Biblioteca que implementa diversos algoritmos de análise de dados e aprendizado de máquina
 - ▶ Classificação
 - ▶ Predição
 - ▶ Clusterização
- ▶ Nesta biblioteca importamos somente os métodos que vamos utilizar
- ▶ Vamos fazer um exemplo de classificação utilizando redes neurais e todas as ferramentas discutidas neste tutorial

CLASSIFICAÇÃO DA BASE DE IRIS

- ▶ A base de dados de IRIS possui as medidas de 150 flores da espécie IRIS
 - ▶ Largura e altura de sepálas e pétalas
 - ▶ Existem 3 grupos de flores: virginica, versicolor e setosa
 - ▶ Portanto, são 150 amostras e 4 atributos e 3 classes



Obrigado pela atenção

André Pacheco

`pacheco.comp@gmail.com`