



Curso de Python

<https://github.com/tsbressan/CursoPython>

Ementa:

- O curso de Python envolve o desenvolvimento de códigos na linguagem de programação mais utilizada para análise e processamento de dados. O curso é ideal para profissionais iniciais ou mesmo aqueles que desejam rever conceitos e utilizações da linguagem Python. Será abordado conceitos iniciais da linguagem Python, histórico, características, instalação, execução, estrutura inicial da linguagem, tipo de dados, estrutura de controle e repetição, bem como exemplos práticos de utilização de dados geofísicos nas bibliotecas Numpy, Pandas, Scipy e Matplotlib.

Conteúdo Programático do Curso:

- Introdução a linguagem Python (Histórico, Características, Versões, Instalação, Editores, execução)
- Estrutura da Linguagem – Sintaxe Inicial
- Módulo ou bibliotecas
- Tipo de Dados
- Estrutura de Controle
- Estrutura de Repetição
- Importação de Arquivos (.csv)
- Processamento numérico (bibliotecas)
- Pandas: Análise e Estrutura de Dados
- NumPy: Array, Matrix e outras funções principais
- SciPy: processamento científico (álgebra linear entre outras)
- Matplotlib: gráficos
- Exercícios Práticos (IODP)

Horário do curso:

- 19:00 às 21:00 (terça, quarta e sexta)
- Dias: 06, 07, 09, 13, 14, 16, 20, 21, 23, 27 /Julho

Livros/Material Suplementar:

- BORGES, L. E. Python para Desenvolvedores. 2ª Edição. Rio de Janeiro, 2010.
- GALVÃO, F. Aprenda Python Básico Rápido e Fácil de entender.
- LABAKI, J. Introdução a Python – Módulo A. Universidade Estadual Paulista – Grupo Python.

Livros/Material Suplementar 2:

- Beginning Python – From Novice to Professional
- Beginning Programming with Python For Dummies
- Geospatial Development By Example with Python
- Learning Python – Fourth Edition
- Practical Programming – Na Introduction to Computer Science Using Python 3 - Second Edition
- Programming in Python 3 – A complete Introduction to the Python Language
- Python in Practice
- Scientific Computing with Python 3
- The Quick Python Book

Introdução a Linguagem Python

- Criada no final dos anos 80 (1989-1990) por Guido Van Rossum, no Centro de Matemática e Tecnológica da Informática, na Holanda. (baseada em linguagem ABC, Algol, C, Java, Perl, entre outras)
- Nome da linguagem relaciona-se com o gosto do criador por humoristas britânicos (Monty Python) ou a relação com o réptil com o mesmo nome (em português-píton)
- Linguagem de programação muito utilizada para análise de dados (científico), programação web, mobile, games,...

Introdução a Linguagem Python

- **CARACTERÍSTICAS:**
- Base no Sistema Operacional escrito em C (principal CPython)
- Linguagem interpretada e de alto nível; Interoperabilidade;
- Orientada a objetos; Multiplataforma;
- Tipagem dinâmica e forte;
- Filosofia de poucas linhas de código em relação a outras linguagens 'tradicionais' – programação rápida e legível
- 5ª linguagem mais popular (considerada uma linguagem simples)

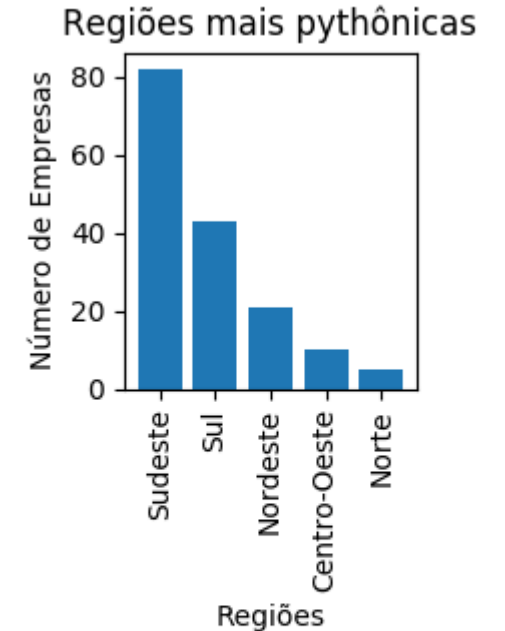
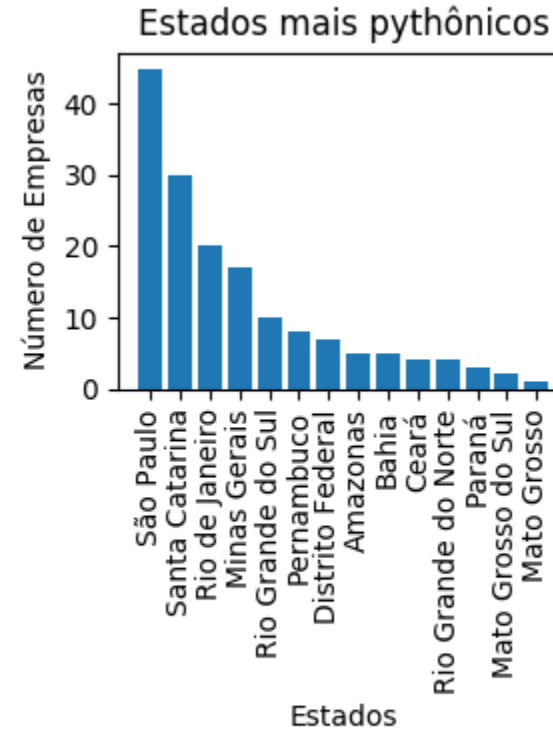
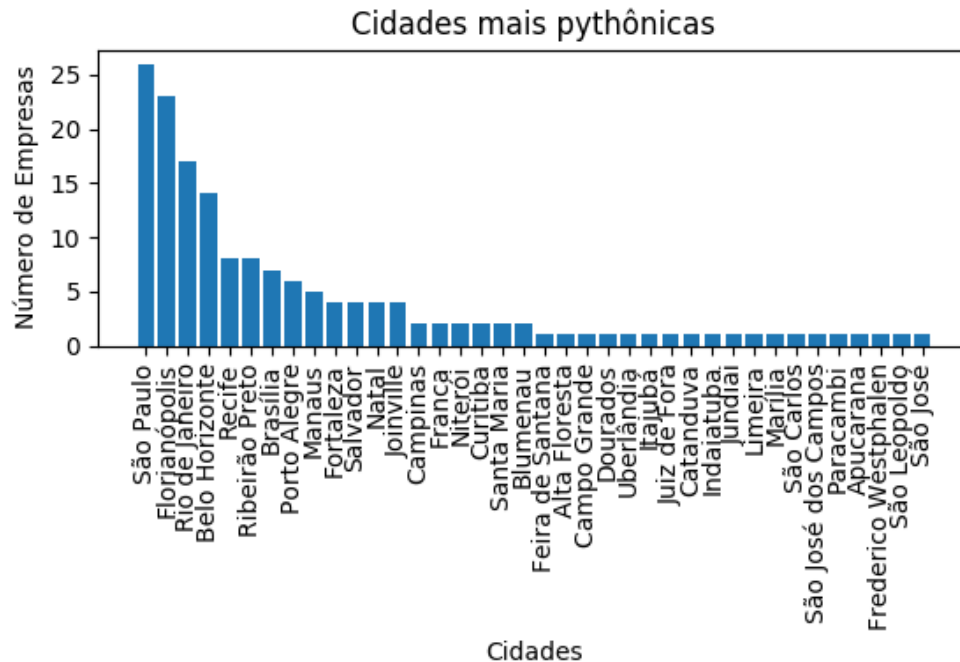
Introdução a Linguagem Python

- **CARACTERÍSTICAS:**

O que pode ser feito:

- Análise científica e estatística
- Construção de sistema web (Django, Flask) - frameworks
- Inteligência Artificial e Aprendizado de Máquina
- Aplicativos com interfaces gráficas (Tkinter, WxPython)

Introdução a Linguagem Python



Introdução a Linguagem Python

- **Versões e software:**

- Atualmente é gerenciada pela Python Software Foundation, com licença de código aberto e livre, compatível com a GNU (General Public License).

- Site: <https://www.python.org/> ou <https://www.python.org.br>

- Versão Atual: 3.9.6

- Anaconda: <https://anaconda.org/>

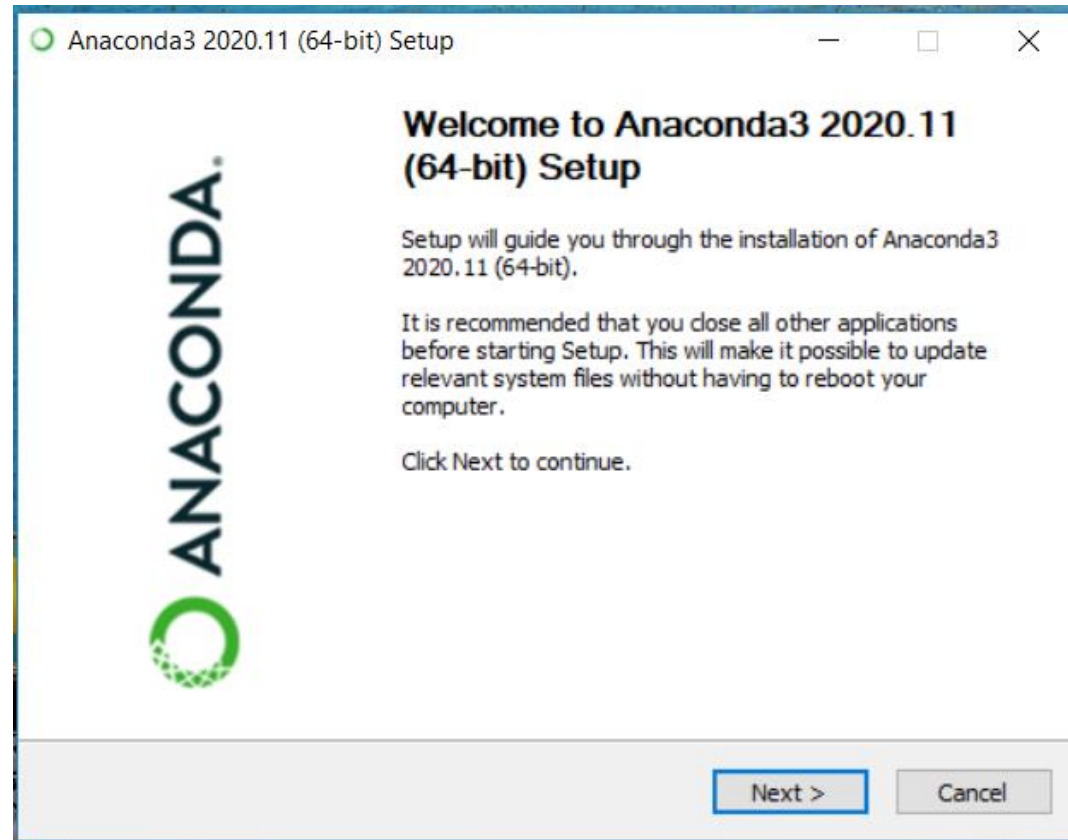
Pacote completo de ferramenta da análise de dados (instala todos os softwares e bibliotecas necessárias, Editor de texto,...)

Introdução a Linguagem Python

- **Instalação:**
- Pode-se instalar somente o Python **OU** o pacote completo (Anaconda) no Windows, Linux, MAC OS,...
- X86-64bits (Windows installer (64-bit)) **OU** para o sistema operacional específico (em formato específico)
<https://python.org.br/instalacao-windows/>
- Anaconda3-2021.05-Windows-x86_64.exe
- https://repo.anaconda.com/archive/Anaconda3-2021.05-Windows-x86_64.exe

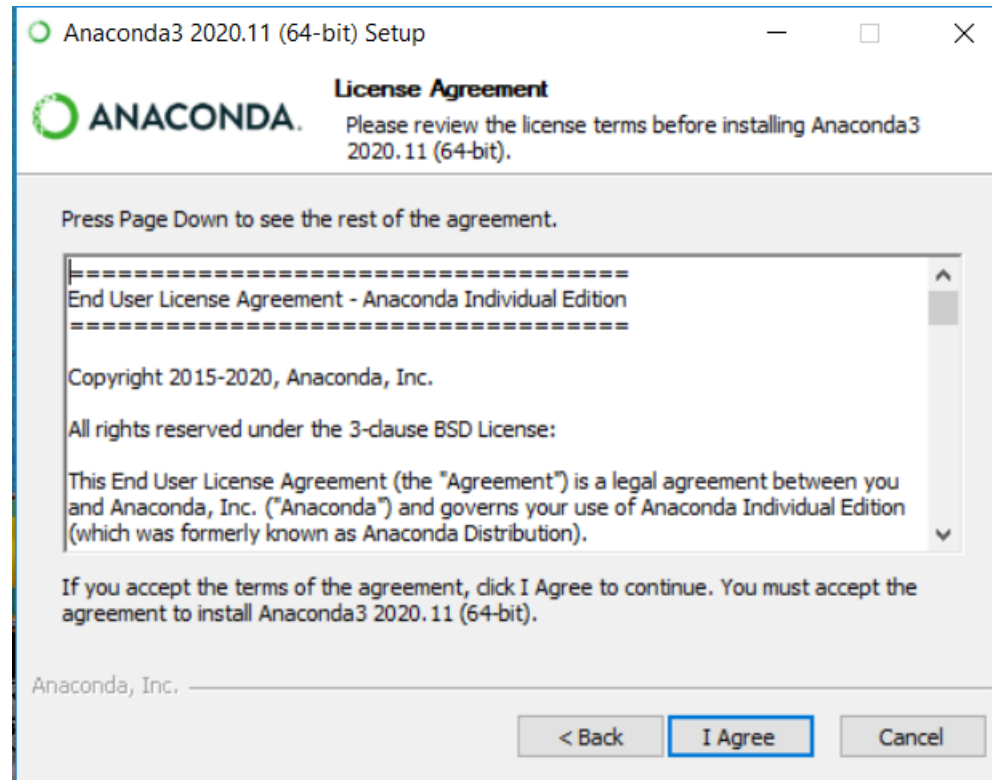
Introdução a Linguagem Python

- Passo a passo:



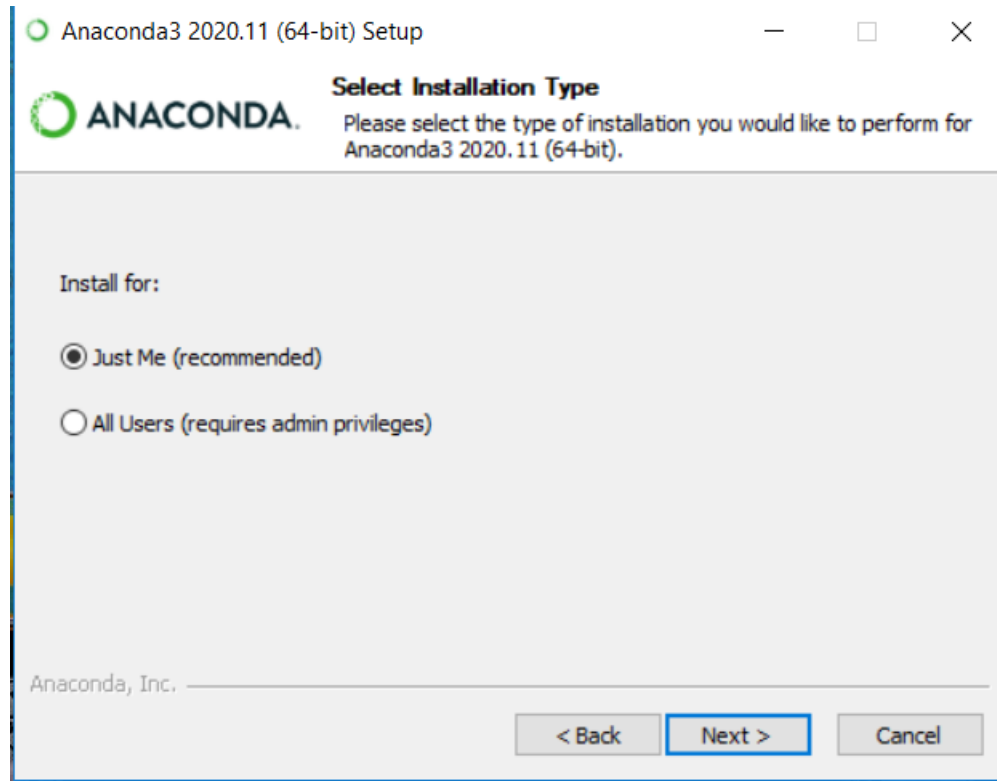
Introdução a Linguagem Python

- Passo a passo:



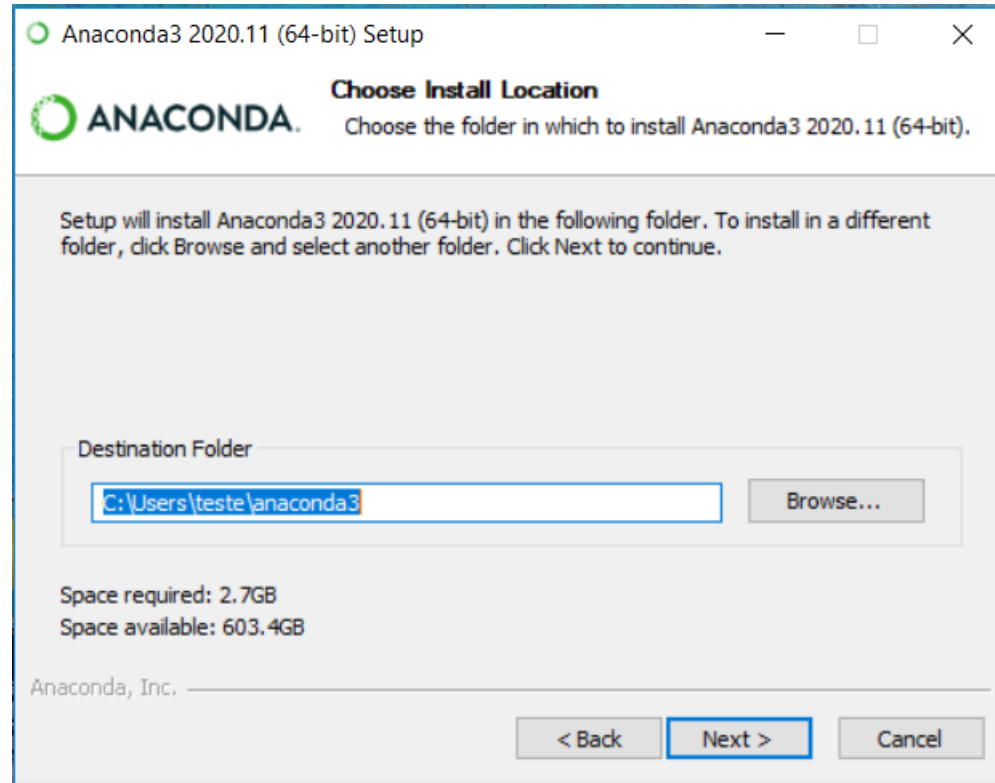
Introdução a Linguagem Python

- Passo a passo:



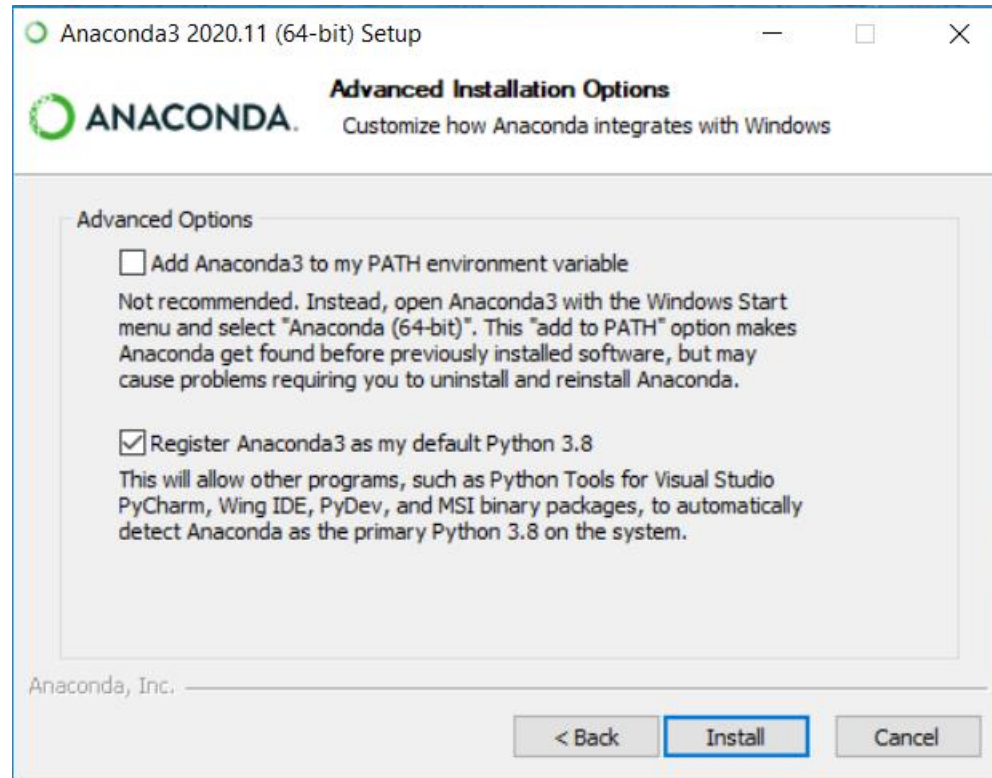
Introdução a Linguagem Python

- Passo a passo:



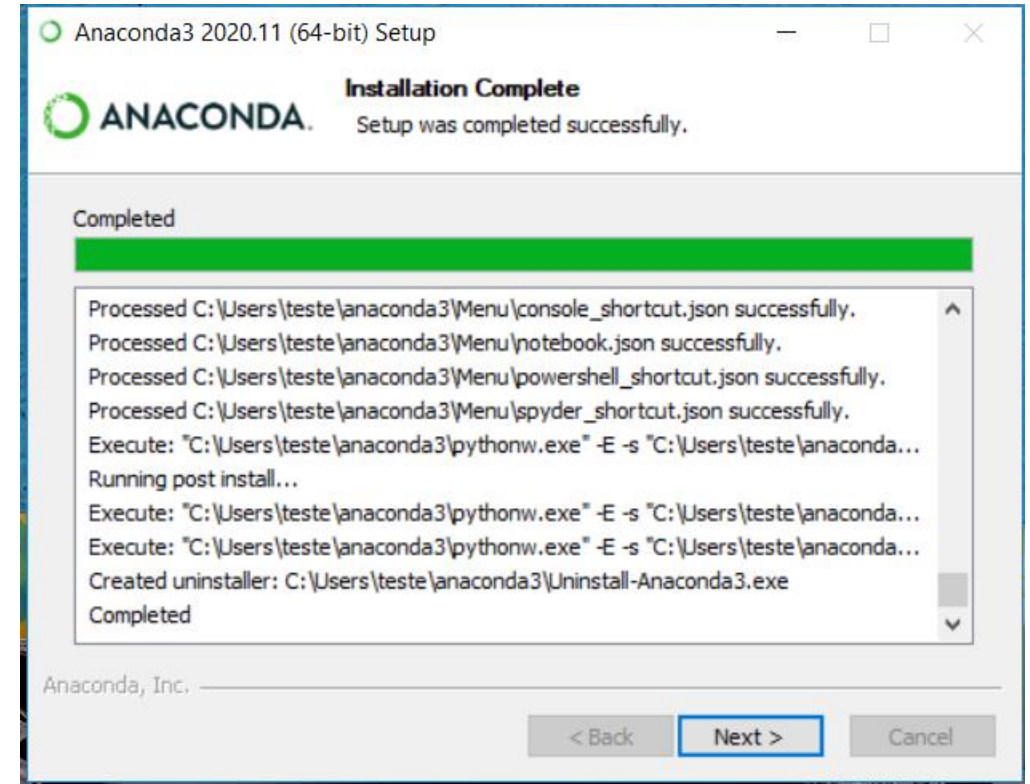
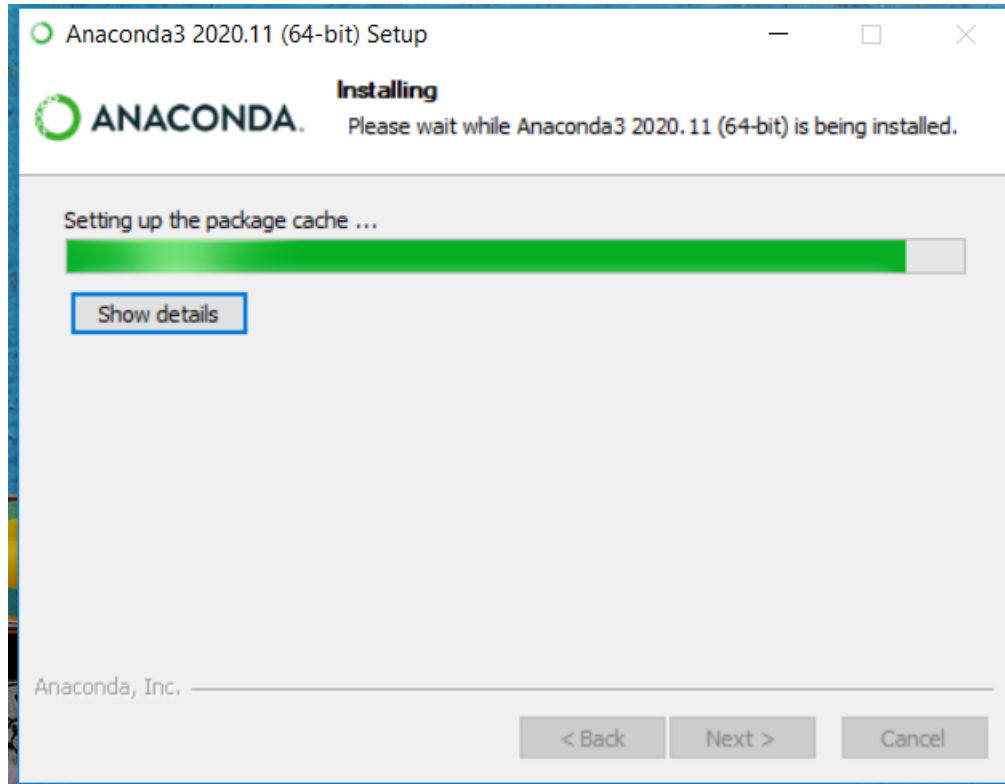
Introdução a Linguagem Python

- Passo a passo:



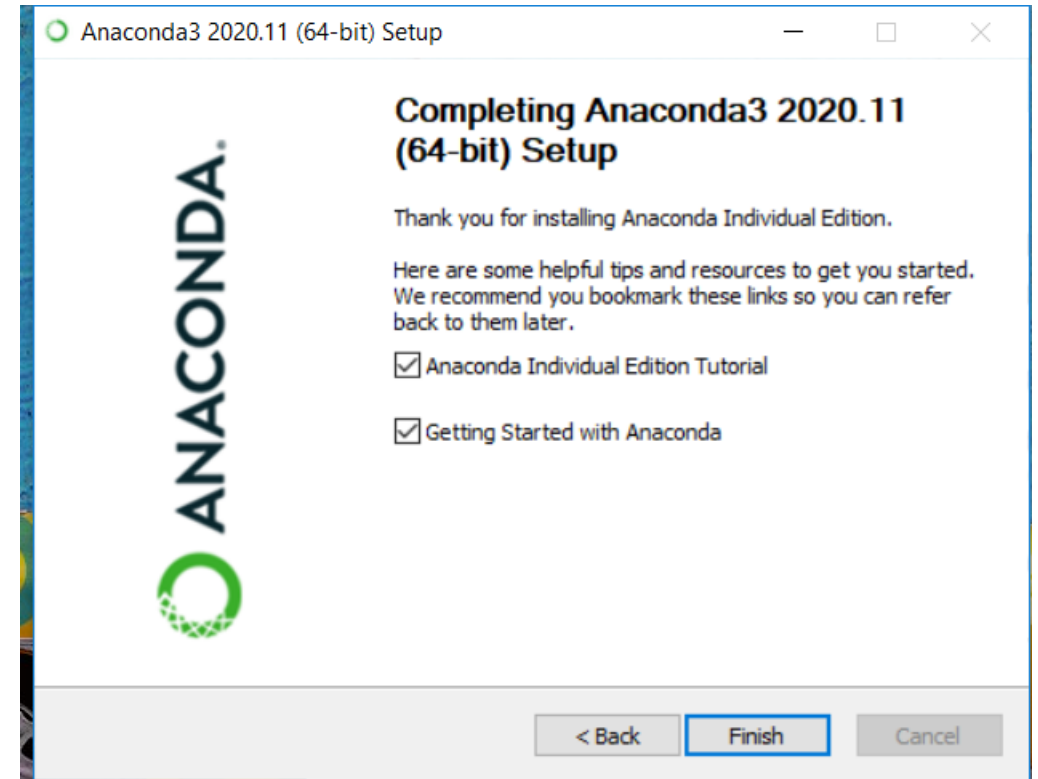
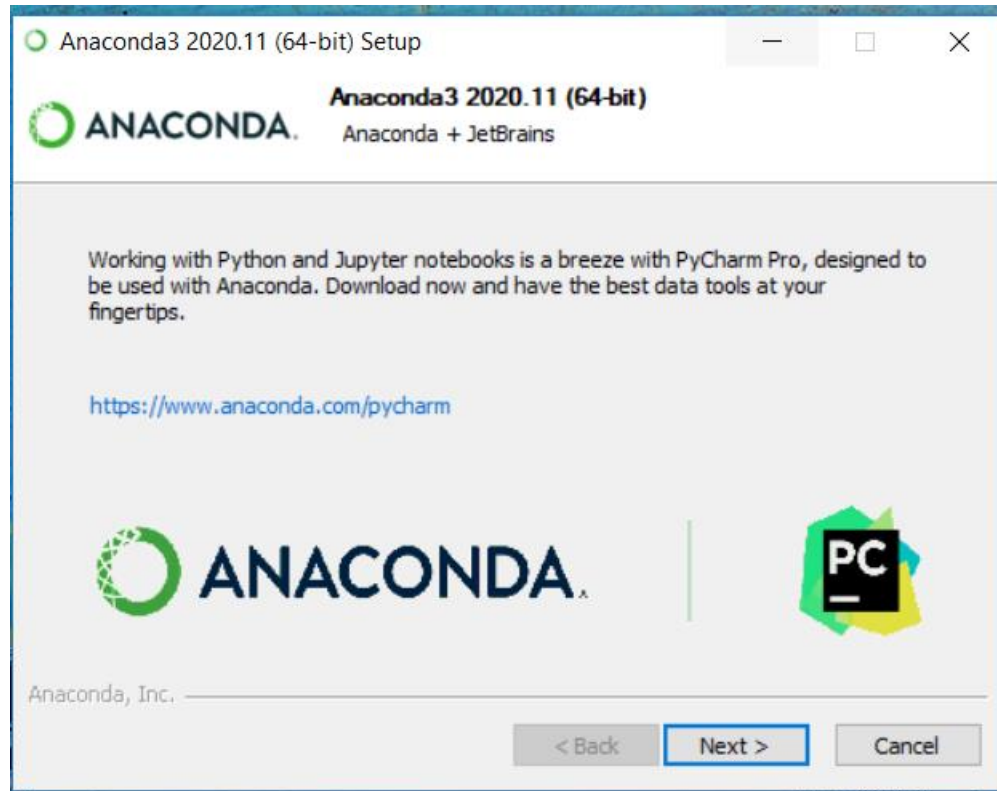
Introdução a Linguagem Python

- Passo a passo:



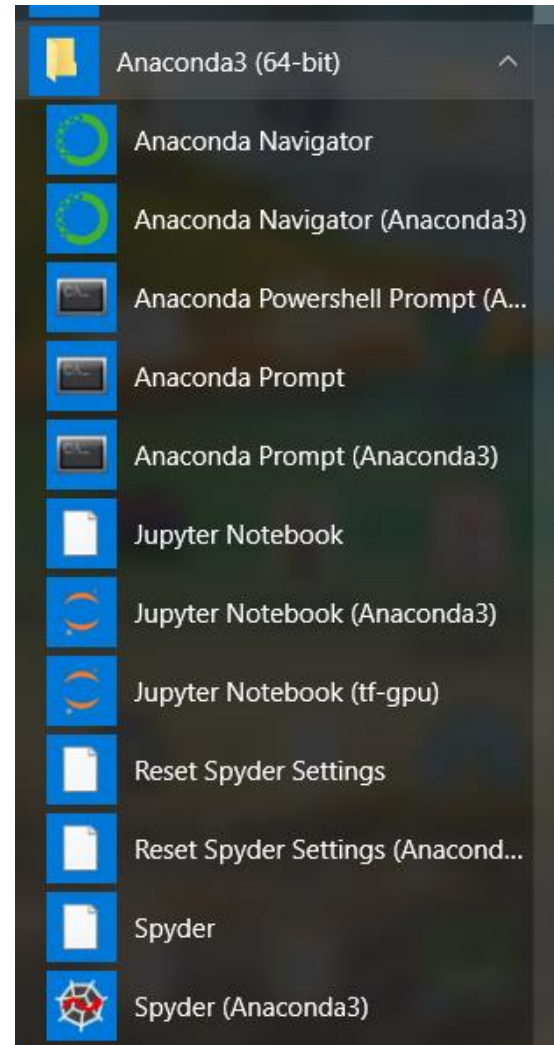
Introdução a Linguagem Python

- Passo a passo:



Introdução a Linguagem Python

- Menu principal:



Introdução a Linguagem Python

- **Editores:**
 - São basicamente dois editores instalados no Anaconda: Jupyter Notebook e Spyder.
 - Pode-se ser utilizado outros editores simples (como o próprio Bloco de Notas do Windows) ou instalar/utilizar outros como Pycharm, Notepad++, Gedit, Sublime,... (<https://python.org.br/ferramentas/>)
- **Execução do código:**
 - Pode-se utilizar o próprio editor *Jupyter* ou *Spyder* (já vincula com o Python e roda o código)
 - Executar via linha de comando (prompt de comando ou shell)

Editores

- Jupyter Notebook
- Spyder
- Dois editores instalados pelo Anaconda.

Editores

- **Jupyter Notebook**

Funcionamento:

- Clicar no ícone: Jupyter Notebook no menu do Anaconda ou via linha de comando: `>> jupyter notebook`
- É Executado por um navegador (como um servidor web local).

<http://localhost:8888>

- Sua execução é interativa, passo a passo (pode ser inserido textos, figuras entre os códigos)

Files

Running

Clusters

Select items to perform actions on them.

Upload

New



0



Name ↓

Last Modified

File size


 codigo1.py

7 dias atrás

810 B


 Dia1.pptx

4 minutos atrás

7.95 MB


 {ECB14B49-2BC7-4AEA-B645-CABBEAFEB2AB}.tmp

21 horas atrás

7.94 MB

Files = Lista dos arquivos no diretório principal

New = novo arquivo Python3

Running = arquivo em aberto (em funcionamento)

Editores

- **Spyder**

Funcionamento:

- Clicar no ícone: Spyder no menu do Anaconda
- Editor mais tradicional, interface completa com console para executar os códigos (a direita)

Spyder (Python 3.7)

Arquivo Editar Pesquisar Código Executar Depurar Consoles Projetos Ferramentas Ver Ajuda

C:\Users\tsbre\OneDrive - Associacao Antonio Vieira\Doutorado\Unisinos\2017-2021\eventos 2019\Curso de Python Unisinos 2019\Dia 1

Editor - C:\Users\tsbre\OneDrive - Associacao Antonio Vieira\Doutorado\Unisinos\2017-2021\eventos 2019\Curso de Python Unisinos 2019\Dia 1\codigo1.py

codigo1.py

```
1#coding: utf-8
2
3
4#início do código
5operações matemáticas
6print (1+2)
7print (2-1)
8print (2/2)
9print (4%2)
10print (8%3)
11print (2**2)
12#funções matemáticas
13print (max(2,4,9))
14print (min(1,5,8))
15print (abs(-22))
16
17#operações com string
18
19#concatenar
20print ("Curso" + "de" + "Python")
21
22
23print ("Caixa d'agua") #correto
24
25#print ('Caixa d'agua') #errado - cuidar as aspas simples
26print ('Caixa d'agua') #correto
27
28#funções com strings
29print (len ("Curso de Python"))
30
31var = "CURSO DE PYTHON"
32print (var.lower())
33
34var = "curso de python"
35print (var.upper())
36
37
38print (2,5)
39
40print (2)
41
42#print ("Data: ", 02/04/2019) #errado
   print ("Data: ", "02/04/2019") #certo
```

Ajuda

Origem Console Objeto

Uso

Neste painel é possível obter a ajuda de qualquer objeto ao pressionar **Ctrl+I** estando na frente do mesmo, tanto no Editor quanto no Console.

Essa ajuda também pode ser mostrada automaticamente depois de escrever um parênteses junto a um objeto. Você pode ativar este comportamento em *Preferências > Ajuda*.

Explorador de variáveis

Explorador de arquivos

Ajuda

Console IPython

Console 1/A

Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.6.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/tsbre/OneDrive - Associacao Antonio Vieira/Doutorado/Unisinos/2017-2021/eventos 2019/Curso de Python Unisinos 2019/Dia 1/codigo1.py', wdir='C:/Users/tsbre/OneDrive - Associacao Antonio Vieira/Doutorado/Unisinos/2017-2021/eventos 2019/Curso de Python Unisinos 2019/Dia 1')

3
1
1.0
0
2
4
9
1
22
CursodePython
Caixa d'agua
Caixa d'agua
15
curso de python
CURSO DE PYTHON

Console IPython

Log do histórico

Permissões: RW Fim de linha: CRLF Codificação: LATIN-1-GUESSED Linha: 11 Coluna: 13 Memória: 44 %



Digite aqui para pesquisar



09:36

10/09/2019

Editores

- **Jupyter Notebook x Spyder**

Sugestão de uso:

- Para executar arquivos *.py: diretamente no cmd (prompt de comando) ou utilize o editor Spyder
- Para executar arquivos passo a passo (linha por linha) com comentários utilize o Jupyter Notebook. Arquivos criados diretamente no Jupyter possuem a extensão *.ipynb
- O bloco de notas pode ser utilizado como editor, porém não possui as cores dos códigos e todos os recursos de um editor para Python

Introdução a Linguagem Python

- **Tipo de arquivo da linguagem:**
- Os arquivos de código devem ser salvos com o tipo *.py
- Nota: O editor Jupyter para executar os códigos diretamente no editor utiliza outro tipo de arquivo (*.ipynb).

Introdução a Linguagem Python

- **Instalação de pacotes (módulos ou bibliotecas) adicionais:**

Duas formas (linha de comando):

Utilizar o **pip** (diretamente no python, mais utilizado no Linux):

`pip install <nome_pacote>`, `pip uninstall <nome_pacote>`, `pip list`

<https://pip.pypa.io/en/stable/quickstart/>

Utilizar o **conda** (diretamente no anaconda):

`conda install -n <nome_pacote>`, `conda remove -n <nome_pacote>`,
`conda list -n`, `conda info`

<https://docs.conda.io/projects/conda/en/latest/commands/remove.html>

Os dois formatos funcionam e são compatíveis entre si (no mesmo computador)

Introdução a Linguagem Python

- **Instalação de pacotes (módulos ou bibliotecas) adicionais**
- Pode ser utilizado pelo gerenciador (aplicativo) do Anaconda:
- Environments (ambientes): Ambiente para desenvolvimento das aplicações (códigos). Cada ambiente trabalha independente (separa os códigos e bibliotecas). base(root) é o principal
- Create, Clone, Import e Remove Environments
- Para o Clone e Import, o tipo de arquivo é *.yml



Home

Environments

Learning

Community

Documentation

Developer Blog

Applications on

base (root)

Channels

Refresh



JupyterLab

1.0.2

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch



Notebook

6.0.0

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch



Spyder

3.3.6

Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Launch



Glueviz

0.13.3

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Install



Orange 3

3.19.0

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Install



RStudio

1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Install



VS Code

1.37.1

Streamlined code editor with support for development operations like debugging, task running and version control.

Install



Home

Environments

Learning

Community

Documentation

Developer Blog



Search Environments

base (root)

tf-gpu

Installed

Channels

Update index...

Search Packages

Name

Description

✓ _libgcc_mutex	
✓ _tflow_select	
✓ absl-py	Abseil python common libraries, see https://github.com/abseil/
✓ astor	Read, rewrite, and write python asts nicely
✓ attrs	Attrs is the python package that will bring back the joy of writi
✓ backcall	Specifications for callback functions passed in to an api
✓ blas	
✓ bleach	Easy, whitelist-based html-sanitizing tool
✓ ca-certificates	Certificates for use with other packages.
✓ certifi	Python package for providing mozilla's ca bundle.
✓ cloudpickle	Extended pickling support for python objects
✓ colorama	Cross-platform colored terminal text.
✓ cudatoolkit	
✓ cudnn	Nvidia's cudnn deep neural network acceleration library
✓ cycler	Composable style cycles.
✓ cytoolz	Cython implementation of toolz. high performance functional utilities.
✓ dask-core	Parallel python with task scheduling
✓ decorator	Better living through python with decorators.
✓ defusedxml	Xml bomb protection for python stdlib modules

154 packages available

Pesquisar pacotes

Installed



Installed

Not installed

Updatable

Selected

All



Create



Clone



Import



Remove

Introdução a Linguagem Python

- **Instalação de pacotes (módulos ou bibliotecas) adicionais**

- Pode ser instalado o pacote diretamente do github:

```
pip install git+<endereço_do_site>
```

```
pip install -r requirements.txt (instalar as dependências do pacote em questão)  
(para criar dentro do ambiente: pip freeze > requirements.txt)
```

- Exemplo:

```
pip install  
git+https://github.com/ceddlyburge/python_world#egg=python_world
```

```
pip install git+https://github.com/fact-project/smart_fact_crawler
```

Introdução a Linguagem Python

- **Estrutura básica de um código em Python:** em 3 partes
 - Definir/importar as bibliotecas (**Não obrigatório)
 - Código (sintaxe) para acessar as bibliotecas (input, processamento)
 - Saída (Mostrar/visualizar na tela ou um arquivo)

** conforme utilização

Pacotes (Módulo ou Bibliotecas)

(**import**) Importam para o código bibliotecas ou módulos extras instalados no Python (ou Anaconda), como suporte para execução de tarefas específicas (não suportadas pela biblioteca padrão do Python).

Exemplos:

```
import numpy
```

```
import pandas
```

```
import matplotlib
```

```
import random
```

Após importar as bibliotecas é possível utilizar as funções vinculadas a cada biblioteca.

Estrutura da Linguagem Python – Sintaxe Inicial

- **Comandos básicos (não necessitam import):**

= -> atribuição

print () -> mostrar alguma informação na tela

-> comentário de uma linha no código

""" """ ou ''' ''' -> comentário de várias linhas no código

Operadores matemáticos:

+, -, *, /, % (resto da divisão), ** (potência)

Funções matemáticas:

max() -> máximo

min() -> mínimo abs() -> retorna valor absoluto

Estrutura da Linguagem Python – Sintaxe Inicial

- **Comandos básicos (não necessitam import):**

Operações com strings

+ ->(concatenar), \'s ->(Escape para \')

Funções com strings

Len() -> tamanho da string

Lower () ->minúsculo

Upper () -> maiúsculo

Str() -> converter em string

Isalpha() -> retorna *false* se a string contiver caracter que não seja letra.

Estrutura da Linguagem Python – Sintaxe Inicial

- **EXEMPLOS:**

Anaconda Prompt (Anaconda3) - python

```
(tf-gpu) C:\Users\tsbre>
(tf-gpu) C:\Users\tsbre>
(tf-gpu) C:\Users\tsbre>
(tf-gpu) C:\Users\tsbre>
(tf-gpu) C:\Users\tsbre>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Curso de Python")
Curso de Python
>>>
>>>
>>>
>>> 1+2
3
>>> 2-1
1
>>> 2*2
4
>>> 2/2
1.0
>>> 4%2
0
>>> 8%3
2
>>> 2**2
4
>>> max(2,4,9)
9
>>> min(1,5,8)
1
>>>
```

Estrutura da Linguagem Python – Sintaxe Inicial

- **EXEMPLOS:**

```
>>>
>>> print ("Curso" + "de" + "Python")
CursodePython
>>> print ("Caixa d'agua")
Caixa d'agua
>>> print ('Caixa d'agua')
File "<stdin>", line 1
    print ('Caixa d'agua')
          ^
SyntaxError: invalid syntax
>>> print ('Caixa d\'agua')
Caixa d'agua
```

```
>>> var = "CURSO DE PYTHON"
>>> var.lower()
'curso de python'
>>>
>>> var = "curso de python"
>>> var.upper()
'CURSO DE PYTHON'
>>>
>>>
>>> len ("Curso de Python")
15
```

Estrutura da Linguagem Python – Sintaxe Inicial

- **EXEMPLOS:**

```
>>> print (2,5)
2 5
>>> print (2)
2
>>> print ("Data: ", 02/04/2019)
File "<stdin>", line 1
    print ("Data: ", 02/04/2019)
                        ^
SyntaxError: invalid token
>>> print ("Data: ", "02/04/2019")
Data:  02/04/2019
>>>
>>> print ("Temperatura externa: "+ str(25))
Temperatura externa: 25
>>>
>>> var = "Curso de Python"
>>> var.isalpha()
False
>>>
>>> var = "Curso de Python 123"
>>> var.isalpha()
False
>>> var = "Curso"
>>> var.isalpha()
True
>>>
```


Estrutura da Linguagem Python – Sintaxe Inicial

- **EXEMPLOS:**

- Digitar os códigos diretamente na tela preta??
 - Podemos armazenar os códigos em um arquivo *.py
- abrir o arquivo de código: *codigo1.py* utilizando o Bloco de notas
- cuidar a linha `#coding: utf-8` (codificação)

Executar:

```
>> python codigo1.py
```

Editores

Editor

- **Jupyter Notebook**

Funcionamento:

- Clicar no ícone: Jupyter Notebook no menu do Anaconda ou via linha de comando: `>> jupyter notebook`
- É Executado por um navegador (como um servidor web local).

<http://localhost:8888>

Files

Running

Clusters

Select items to perform actions on them.

Upload

New



0



Name ↓

Last Modified

File size


 codigo1.py

7 dias atrás

810 B


 Dia1.pptx

4 minutos atrás

7.95 MB


 {ECB14B49-2BC7-4AEA-B645-CABBEAFEB2AB}.tmp

21 horas atrás

7.94 MB

Files = Lista dos arquivos no diretório principal


New = novo arquivo Python3

Running = arquivo em aberto (em funcionamento)

Jupyter Notebook



- New / Python3:

 jupyter **Untitled** Last Checkpoint: poucos segundos atrás (unsaved changes)



Logout


File Edit View Insert Cell Kernel Widgets Help

- Menu Principal:

Trusted



Python 3

        Run    Code


In []:

- Código:












Jupyter Notebook



- Nome do arquivo

jupyter **Untitled** Last Checkpoint: 22 minutos atrás (unsaved changes)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

           Code

- Formatação do texto / tipo de linha

In []:

In []:

- Shift + enter

- Menu Insert: Insert line Above or Bellow
- Menu Edit: Delete Cells

Jupyter Notebook



jupyter Untitled Last Checkpoint: 28 minutos atrás (unsaved changes)



Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3



In [3]: `print ("Curso de Python")`

Curso de Python

↓ - Shift + enter

In [4]: `numero1 = 1`
`numero2 = 2`

In [5]: `print (numero1 + numero2)`

3

In []:

Estrutura da Linguagem Python – Sintaxe Inicial

Estrutura da Linguagem Python – Sintaxe Inicial

- **VARIÁVEIS:**

- Utilizada para armazenar alguma “informação” conforme um tipo de dado
- Deve ser inicializada/criada antes de ser utilizada.
- Não existe criação automática de variáveis em Python

- Por exemplo:

```
>>> soma = numero1 + numero2
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'numero1' is not defined

Estrutura da Linguagem Python – Sintaxe Inicial

- **VARIÁVEIS:**

Tipo de dados em Python

- **inteiro:**

`a = 123` #decimal `a = 017` #Octal inicia em zero

`a = 0xAF` #Hexadecimal inicia em 0x

- **Float:**

`a = 0.024`

- **Long:** #números inteiros longos

`a = 145897896254`

Estrutura da Linguagem Python – Sintaxe Inicial

- **VARIÁVEIS:**

Tipo de dados em Python

- **bool:** #valores booleanos True ou False (Operadores lógicos, and-or)

a=True

b=False

- **None type:** #tipo None, ausência de valores, simular a null

a=None

- **String(str):** #entre aspas simples, duplas ou triplas.

a="Curso"

Estrutura da Linguagem Python – Sintaxe Inicial

- **VARIÁVEIS:**

EXEMPLOS PRÁTICOS(1):

`a=2.25 #tipo float`

`b=55 #tipo inteiro`

`c=0o740 #tipo inteiro octal`

`e=0xFFAB #tipo inteiro hexadecimal`

`f="Curso de Python" #tipo string`

`type(a) #mostra o tipo da variável`

Estrutura da Linguagem Python – Sintaxe Inicial

- **VARIÁVEIS:**

EXEMPLOS PRÁTICOS(2):

```
meu_nome = "Carlos"
```

```
meu_sobrenome = 'Santini'
```

```
print ("Nome: %s, Sobrenome: %s" % (meu_nome.upper(), meu_sobrenome))
```

```
print (f'Nome: {meu_nome.upper()}, Sobrenome: {meu_sobrenome}') #Formatted string literals ou f-string
```

```
print("Nome: {0}, Sobrenome: {1}".format(meu_nome.upper(), meu_sobrenome)) #new-style string formatting
```

```
print ("Meu nome começa com a letra ", meu_nome[0])
```

```
print ("Meu nome começa com a letra ", meu_nome[0].lower())
```

```
print ("Meu primeiro nome é ", meu_nome[0:6])
```

```
print ("Meu sobrenome é ", meu_sobrenome[0:7])
```

Estrutura da Linguagem Python – Sintaxe Inicial

- **VARIÁVEIS:**

Conversão de tipos em python

```
a = float(21/4)
```

```
b = int(4.8)
```

```
c = int(4.9)
```

```
d = int(0xff500)
```

```
e = float(int(3.9))
```

```
f = int(float(3.9))
```

```
g = int(float(3))
```

```
h = round(3.9)
```

```
i = round(3)
```

```
j = int(round(3.9))
```

```
print (a,b,c,d,e,f,g,h,i,j)
```


Estrutura da Linguagem Python – Sintaxe Inicial

- **VARIÁVEIS:**

Palavras reservadas – não utilizar como uma variável:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Código estruturado com função

```
def main():      #função com o nome main()
    a = 3
     b = 4
    soma = a + b
    print("A soma de a + b e igual a: ", soma)
```

```
main()      # executa a função
```

- Cuidar a indentação;
- Revisar aspas, parênteses, pontos, vírgulas,..., programação é revisar o texto várias e várias vezes

Entrada de dados – input()

- **Responsável por receber os dados que o usuário fornece via teclado.**

Formato:

```
variável = input ("prompt")
```

Exemplo:

```
nome = input ("Qual é o seu nome? ")  
print ("O seu nome é:", nome)
```

Entrada de dados – input()

```
def main():
```

```
    a = input("Digite o primeiro numero: ")
```

```
    b = input("Digite o segundo numero: ")
```

```
    soma = int(a) + int(b)
```

```
    print("A soma de", a, "+", b, "e igual a", soma)
```

```
main()
```

Exemplos práticos:

- **1) Operações matemáticas com 2 variáveis**
- **2) Operações matemáticas com 3 variáveis**
- **3) Operações com String**
- **4) Conversão de tipos**
- **5) Operações com booleanos**
- **6) Operação com String - avançado**

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

- Utilizada para decidir qual bloco de código deve ser executado ou não, através de uma condição lógica em linguagem de programação

- -----

Se estiver chovendo: #expressão verdadeira

 Levarei guarda-chuva

Senão: #expressão falsa

 Não levarei

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

- Utiliza o bloco de comando se..senão ou se..senãose
- Traduzindo para a linguagem Python: if ...else ou if..elif

```
a = 10
if a > 3:  #verdadeiro
    print("Número maior que 3")
else:     #falso
    print("Número menor que 3")
```

Saída:::

```
>> Número maior que 3
```

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

```
valor_entrada = 10
if valor_entrada == 1:
    print("a entrada era 1")
elif valor_entrada == 2:
    print("a entrada era 2")
elif valor_entrada == 3:
    print("a entrada era 3")
elif valor_entrada == 4:
    print("a entrada era 4")
else:
    print("o valor de entrada não era esperado em nenhum if")
```

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Operadores utilizados na estrutura de controle:

`==` #igual

`>` #maior

`<` #menor

`>=` #maior igual

`<=` #menor igual

`!=` #diferente

`and-or` # operadores lógicos

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Exemplo utilizando função:

```
def main():
```

```
    ← idade= int(input ("Quantos anos voce tem?"))
```

```
        if idade >= 16:
```

```
            ← print ("Você já tem idade para votar.")
```

```
        elif idade > 10 and idade < 16:
```

```
            print ("Você é adolescente")
```

```
        else:
```

```
            print("Você ainda é um garoto(a). Aproveite para brincar...")
```

```
main()
```

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Exemplo:

```
def main():
```

```
    ← Numero1= int(input ("Digite o número 1:"))
    Numero2= int(input ("Digite o Número 2:"))
    if Numero1 > Numero2:
        ← print ("Primeiro número é maior que o segundo número")
    elif Numero2 > Numero1:
        print ("Segundo número é maior que o primeiro número")
    else:
        print("Os números são iguais")
```

```
main()
```

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Vetor simples unidirecional:

Lista de dados de um mesmo tipo. No Python é possível armazenar em vetor dados de vários 'tipos', a qual chamamos de Lista:

```
Notas = [5.0, 8.5, 7.8, 9.3]  
lista = ['A', 1, 2, 'Casa', 2.3]
```

```
Notas = [5.0, 8.5, 7.8, 9.3]  
índice: 0    1    2    3
```

Para acessar, basta indicar a posição(índice) do vetor (iniciando sempre na posição zero)

```
Notas[0] -> 5.0
```

```
Lista[2] -> 2
```

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Exemplo utilizando vetor simples:

```
Numeros = [] #inicio do vetor
```

```
def main():
```

```
    Numeros.append(input ("Digite o número 1:"))
```

```
    Numeros.append(input ("Digite o Número 2:"))
```

```
    if Numeros[0] > Numeros[1]:
```

```
        print ("Primeiro número é maior que o segundo número")
```

```
    elif Numeros[1] > Numeros[0]:
```

```
        print ("Segundo número é maior que o primeiro número")
```

```
    else:
```

```
        print("Os números são iguais")
```

```
main()
```

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Erro comum – acessar posição do vetor que não existe

```
alunos = ['Andre', 'Lucas', 'Antonio', 'Maria']  
print(alunos[4])
```

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Mais algumas funções aplicados sobre um vetor de dados:

```
alunos = ['Andre', 'Lucas', 'Antonio', 'Maria']
```

```
len (alunos) #tamanho do vetor alunos-4
```

```
append() #método responsável por adicionar um novo elemento na próxima posição do vetor
```

Concatenar valores em vetor

```
lista = [1,2,3]
```

```
lista = lista + [4]
```

```
print (lista)
```

```
lista = lista + [4,5,6]
```

```
print (lista)
```

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Mais algumas funções aplicados sobre um vetor de dados:

Multiplicação de vetores (duplicar conteúdo do vetor):

```
lista = [1,2,3]
```

```
lista * 3          #repete 3x o conteúdo da lista
```

Utilizado para inicializar um vetor com zero. Ex.:

```
tamanho = 10
```

```
lista = [0]
```

```
lista * tamanho
```

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Exemplos práticos. Armazenar os valores em vetor:

- 1) Escreva um código Python que, dados 2 números diferentes (a e b), encontre o menor e o maior.
- 2) Leia um número e imprima a raiz quadrada do número caso o número seja positivo ou igual a zero.
- 3) Leia dois números e efetue a adição. Caso o valor somado seja maior que 20, este deverá ser apresentado somando-se a ele mais 8; caso o valor somado seja menor ou igual a 20, este deverá ser apresentado subtraindo-se 5.

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Exemplos práticos. Armazenar os valores em vetor:

4) Faça um algoritmo que leia dois vetores de 3 posições e realize a soma dos valores nas mesmas posições.

5) Faça um algoritmo que leia 3 números float, armazene em vetor e calcule a soma dos valores, a média dos valores

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Matriz = linha x coluna

O Python não possui em sua estrutura (de forma nativa da linguagem) um tipo de dado matriz.

Como resolver?

- Lista de listas
- Biblioteca Numpy (será visto com detalhes)

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Matriz = linha x coluna

Lista de listas

```
>>> m = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]
```

Operações: acessar posição, remover item, apagar lista, operações matemáticas.

Estrutura de controle e Estrutura de Repetição

- **Estrutura de controle:**

Exemplos práticos – matriz com lista

1) Faça um código Python para criar uma matriz com 3 linhas e 5 colunas. Inicializar com valores aleatórios (tipo float). Realizar a soma dos valores por linha e por coluna. Mostrar os valores.

Estrutura de controle e Estrutura de Repetição

- **Estrutura de repetição (loop ou laço):**

Estrutura de controle e Estrutura de Repetição

- **Estrutura de repetição (loop ou laço):**

Dois comandos utilizados para estrutura de repetição: for e while

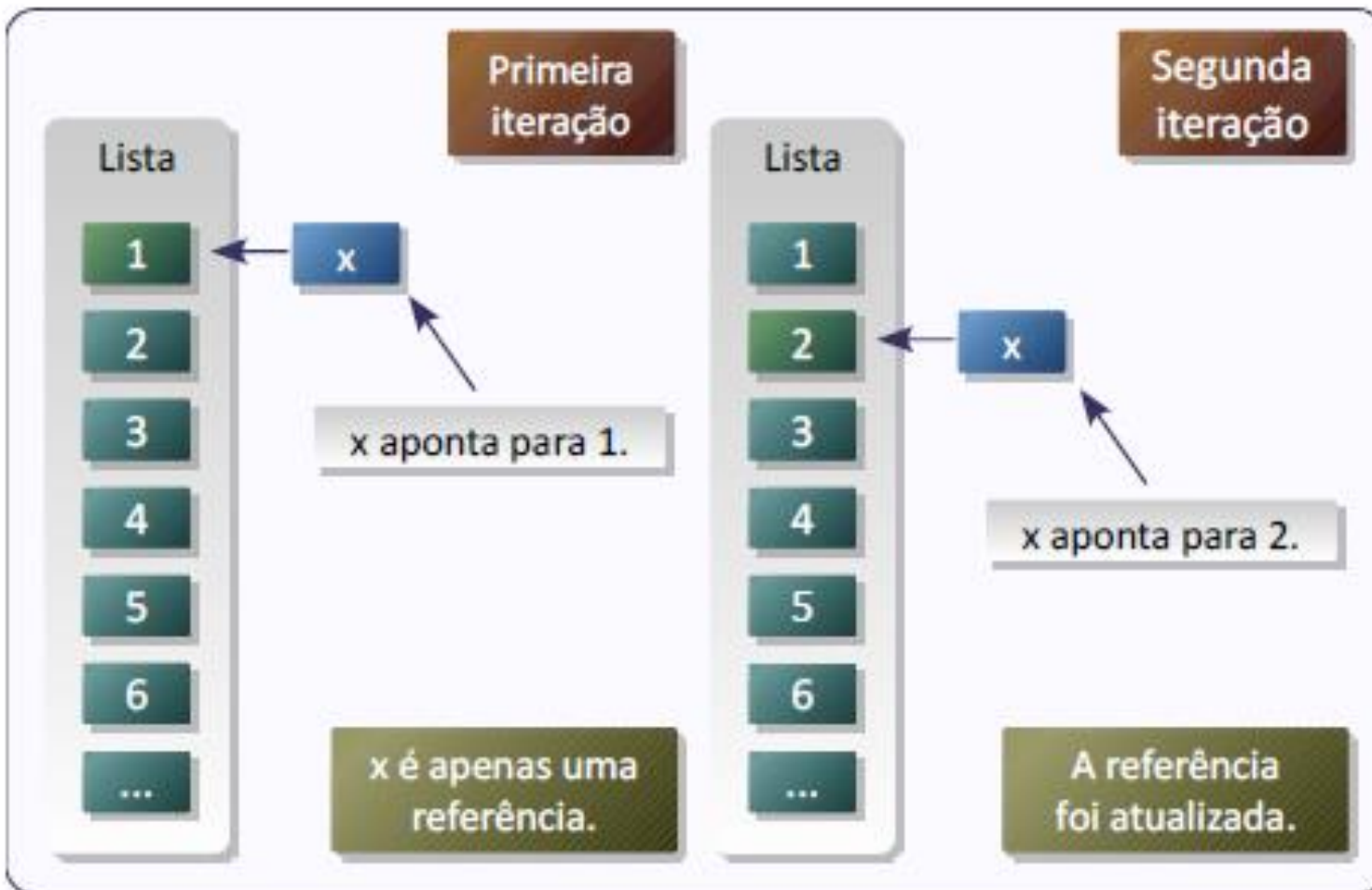
--- Utilizada quando queremos que um bloco de código seja executado várias vezes----

O for é usado quando se quer iterar sobre um bloco de código um número determinado de vezes.

O while é usando quando queremos que o bloco de código seja repetido até que uma condição seja satisfeita, neste caso, utilizando uma expressão booleana (true ou false)

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição (loop ou laço):



Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **for**

```
# Aqui repetimos o print 3 vezes
```

```
for n in list(range(0, 3)):
```

```
    print("Número: " , n)
```

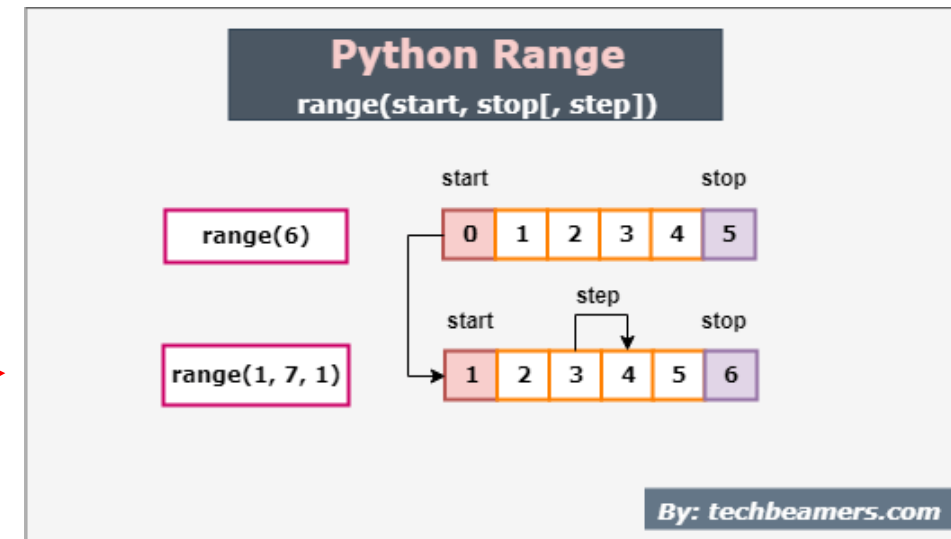
```
#####
```

```
# Para iterar(todos os elementos) sobre um vetor
```

```
v = [1, 2, 3, 4, 10]
```

```
for numero in v:
```

```
    print(numero ** 2)
```



Estrutura de controle e Estrutura de Repetição

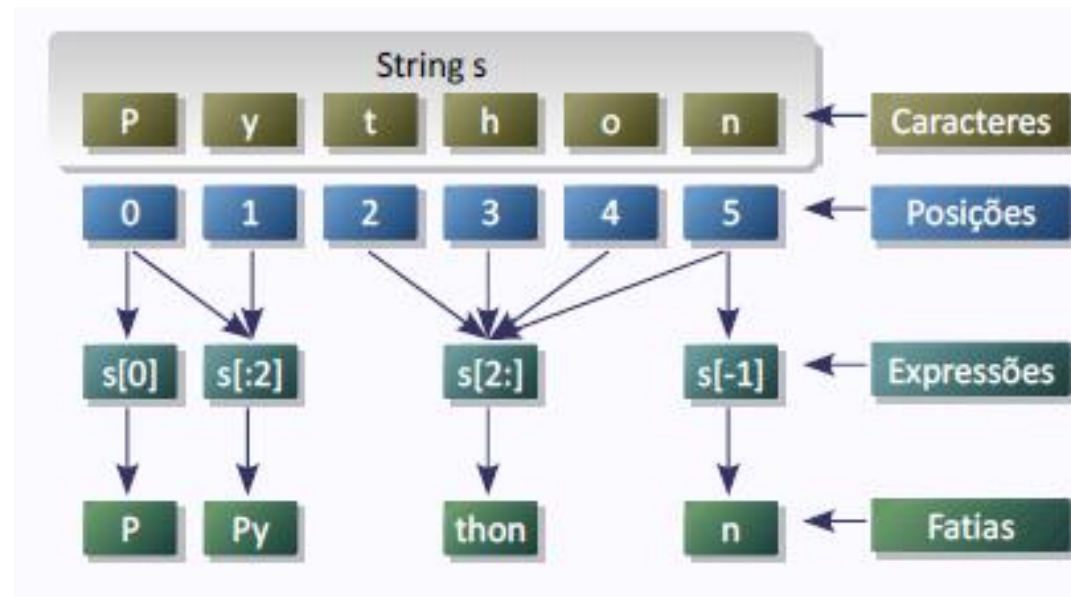
- Estrutura de repetição: **for**

Muito utilizado em strings (já usamos)

```
palavra = "unisinos"
```

```
for letra in palavra:
```

```
    print(letra)
```



IMPORTANTE: para auxiliar as estruturas de repetição, inclui-se dois comandos importantes: break e continue. break utilizado para sair da execução (loop). continue finalizar a execução daquele trecho e inicia o próximo (dentro do mesmo loop)

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **for**

```
for n in range(0,3):  
    string_digitada = input("Digite uma palavra: ")  
    if string_digitada.lower() == "quit":  
        print("Finalizou a execução!")  
        break  
    if len(string_digitada) <= 3:  
        print("Palavra muito pequena")  
        continue  
    if len(string_digitada) > 3:  
        print("Palavra digitada está correta..")
```

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **for**

Exercícios práticos de for

- 1) Mostrar todos os valores de 1 até 10
- 2) Mostrar os valores pares de 1 até 10
- 3) Encontrar o maior e o menor valor do intervalo de 1 até 10
- 4) Imprimir na tela somente os valores armazenados num vetor, onde o índice do vetor é par.

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **while**

O while é usado quando queremos que o bloco de código seja repetido até que uma condição seja satisfeita, neste caso, utilizando uma expressão booleana (true ou false)

while <condição_lógica>:

↔ #linhas de código

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **while**

Exemplo:

Inicia-se o n em 0, e repetimos o print até que seu valor seja maior ou igual a 3:

```
n = 0
```

```
while n <= 3:    #condição for verdadeira – teste lógico
```

```
    print(n)
```

```
    n += 1
```

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a c //= a is equivalent to c = c // a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **while**

```
n=0
```

```
n = int (input("Digite um número inteiro (-1 para sair ou quit após execução)))
```

```
while n != -1:
```

```
    string_digitada = input("Digite uma palavra: ")
```

```
    if string_digitada.lower() == "quit":
```

```
        print("Finalizou a execução!")
```

```
        break
```

```
    if len(string_digitada) <= 3:
```

```
        print("Palavra muito pequena")
```

```
        continue
```

```
    if len(string_digitada) > 3:
```

```
        print("Palavra digitada está correta..")
```

```
print ("código finalizado")
```

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **while**

Exemplo prático com vetor:

```
seq = []  
i = 0  
while i < 5:  
    novo_elemento = i  
    seq.append( novo_elemento )  
    i = i + 1  
  
print (seq)
```

Estrutura de controle e Estrutura de Repetição

- Estrutura de repetição: **while**

Exercícios práticos de while:

- 1) Mostrar todos os valores de 1 até 10
- 2) Mostrar todos os valores de 10 até 1
- 3) Digitar 3 números inteiros, armazenar em vetor, realizar a soma e média.
- 4) Utilizando vetor, receber 3 números float.

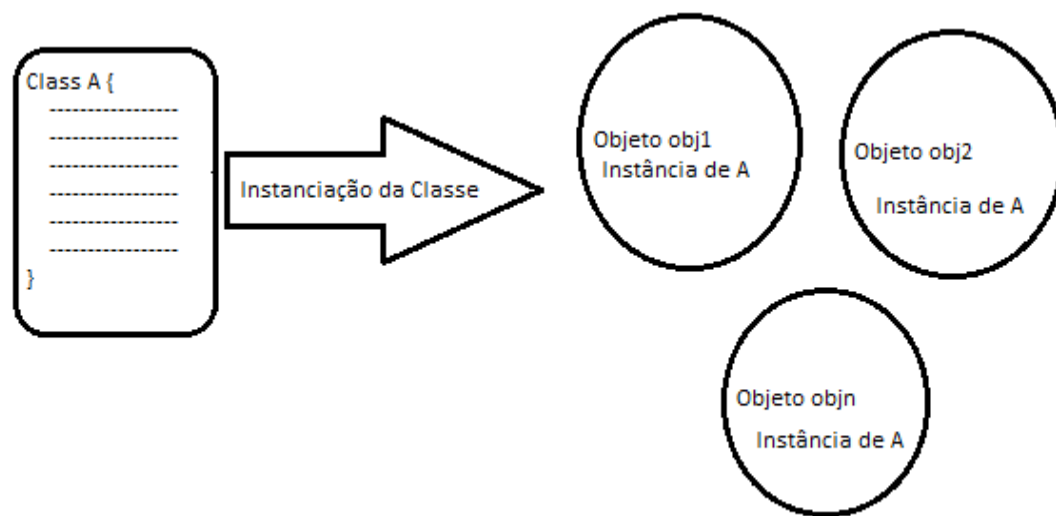
Trabalhando com classes e métodos (funções) no Python



Trabalhando com classes e métodos (funções) no Python

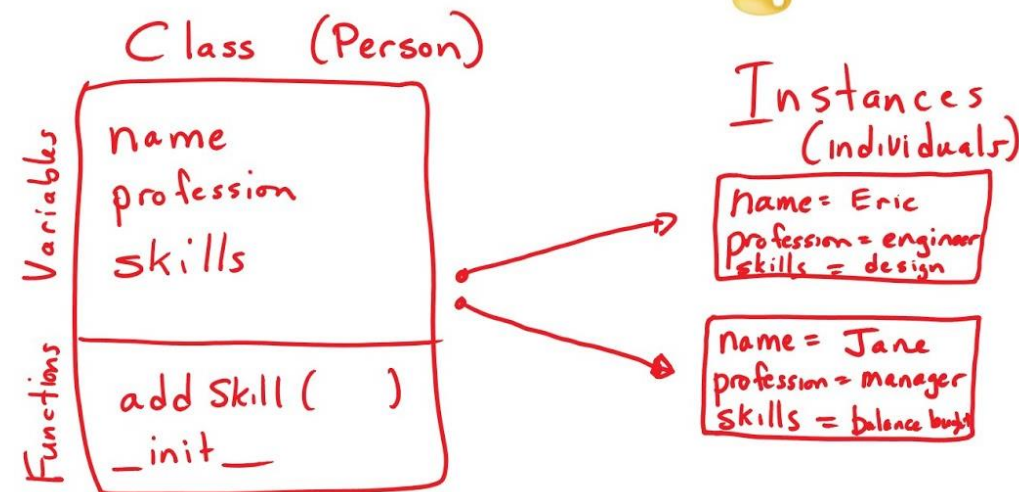
- Classes e Funções são blocos de códigos que realizem determinada tarefa programada, que normalmente precisam ser usadas várias vezes (ou seja, são reutilizadas).
- Uma classe associa dados (atributos) e operações (métodos) numa só estrutura. Um objeto é uma instância de uma classe. Ou seja, uma representação da classe. Por exemplo, Regis é uma instância de uma classe chamada Pessoa, mas a Pessoa é a classe que o representa de uma forma genérica. Se você criar um outro objeto chamado Fabio, esse objeto também será uma instancia da classe Pessoa.
- Tudo isso é chamado em programação de orientação a objetos.
- Objetivo principal é a REUTILIZAÇÃO DE CÓDIGO

Trabalhando com classes e métodos (funções) no Python



<https://i.stack.imgur.com/9aGX2.png>

Introduction to Python Classes



<https://www.youtube.com/watch?v=LwOg0b0ZwCM>



Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções):

```
def main ():
```

```
    #linha de código1
```

```
    #linha de código2
```

```
    #linha de código3
```

```
>> main()
```

Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções):

Palavra reservada **def** realiza a definição da função

main é o nome da função (pode ser alterado)

Entre () define o(s) parâmetro(s) do método. Pode ser vazio.

: ao final indica que a próxima linha deve ser indentada.

Para executar a função, basta chamar o nome (**main()**), se existir, passar os parâmetros dentro dos parênteses.

Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções):

```
def teste (nome):
```

```
    print("Meu nome é: ", nome)
```

```
>> teste("Thiago")
```

Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções): Mais de um parâmetro

```
def teste (nome, idade):
```

```
    print("Meu nome é: ", nome, " Idade: ", idade)
```

```
>> teste("Thiago", 28)
```

Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções): podem receber valores e realizar cálculos.

```
def calcula_media (soma, qtd):
```

```
    media = soma/qtd
```

```
    print(" A média é: ", round(media, 2))
```

```
num1= float(input("Digite o valor do 1º número: "))
```

```
num2= float(input("Digite o valor do 2º número: "))
```

```
soma = num1+num2
```

```
calcula_media(soma, 2)
```


Trabalhando com classes e métodos (funções) no Python

- Criando e utilizando métodos (funções): usando vetor

```
def calcular_media(Numeros):  
    soma = 0  
    for n in list(range(0, len(Numeros))):  
        soma = soma + Numeros[n]
```

```
    media = soma / len(Numeros)  
    print("A média é: ", media)
```

```
Numeros = []  
num = int(input("Quantos números gostaria de digitar?"))
```

```
for n in range(0, num):  
    Numeros.append(eval(input("Digite o número : ")))
```

```
calcular_media(Numeros)
```

Trabalhando com classes e métodos (funções) no Python

- CLASSES

```
class NomeClasse :
```

```
    def metodo(self, args):  
        pass
```

Exemplo:



```
class NomeClasse :
```

```
    def metodo(self, nome):  
        print ("Nome: ", nome)
```

NomeClasse – nome da classe (não tem espaço)

def método () – nome do método da classe: nome_da_classe

#self – palavra reservada obrigatória

args – argumentos ou parâmetros do método

#pass – significa que o método tem conteúdo vazio.

Bibliotecas(módulos) para processamento

Bibliotecas(módulo) para processamento

- **Utilização das bibliotecas**

<https://docs.python.org/3/library/>

- **Import <nome_biblioteca>**

importa todo o módulo especificado

Import <nome_biblioteca> from <pacote>

importa apenas o pacote especificado

Ao tentar *importar* um módulo que não existe, um erro será reportado:
ImportError na tela

Bibliotecas(módulo) para processamento

- **Utilização das bibliotecas**
- Criar bibliotecas – organização do código.
- Example:

Bibliotecas(módulo) para processamento

- **Utilização das bibliotecas**
- **Exemplo:**

```
import math      Biblioteca      https://docs.python.org/3/library/math.html
```

```
print (math.sqrt(0))
```

```
print (math.sqrt(4))
```

```
print (math.sqrt(8))
```

Bibliotecas(módulo) para processamento

- **Utilização das bibliotecas**
- **Exemplo:**

```
import math
```

```
print (math.factorial(8))
```

```
print (math.isnan(4))
```

```
print (math.cos(8))
```

```
print (math.sin(8))
```

Bibliotecas(módulo) para processamento

- **Utilização das bibliotecas**
- **Exemplo:**

```
import string
```

<https://docs.python.org/3/library/string.html>

```
s = 'curso de python'
```

```
print (s)
```

```
print (string.capwords(s))
```


Bibliotecas(módulo) para processamento

- **Utilização das bibliotecas**
- **Exemplo:**

```
import os
```

<https://docs.python.org/3/library/os.html>

```
dir = []
```

```
patch = "
```

```
for filename in os.listdir(patch):
```

```
    dir.append(filename)
```

Bibliotecas(módulo) para processamento

- **Utilização das bibliotecas**
- **Exemplo:**

```
import os  
# mostrar diretorio atual  
path = os.getcwd()  
print ("O Diretorio atual é %s" % path)
```

Bibliotecas(módulo) para processamento

- **Utilização das bibliotecas**
- **Exemplo:**

```
import os
```

```
# caminho e nome do diretório a ser criado
```

```
path = "..."
```

```
try:
```

```
    os.mkdir(path)
```

```
except OSError:
```

```
    print (" Erro ao criar o diretório %s " % path)
```

```
else:
```

```
    print (" Diretório criado com sucesso. %s " % path)
```

Github data Python 2018



Library Name	Type	Commits	Contributors	Releases	Watch	Star	Fork	Commits/ Contributors	Commits/ Releases	Star/ Contributors
matplotlib	Visualization	25 747	725	70	498	7 292	398	36	368	10
Bokeh	Visualization	16 983	294	58	363	7 615	2 000	58	293	26
plotly	Visualization	2 906	48	8	198	3 444	850	61	363	72
Seaborn	Visualization	2 044	83	13	205	4 856	752	25	157	59
<i>pydot</i>	Visualization	169	12	12	17	193	80	14	14	16
learn	Machine learning	22 753	1 084	86	2 114	28 098	14 005	21	265	26
XGBoost	Machine learning	3277	280	9	868	11 991	5 425	12	364	43
LightGBM	Machine learning	1083	79	14	363	5 488	1 467	14	77	69
CatBoost	Machine learning	1509	61	20	157	2 780	369	25	75	46
eli5	Machine learning	922	6	22	39	672	89	154	42	112
SciPy	Data wrangling	19 150	608	99	301	4 447	2 318	31	193	7
NumPy	Data wrangling	17 911	641	136	390	7 215	2 766	28	132	11
pandas	Data wrangling	17 144	1 165	93	858	14 294	5 788	15	184	12
statsmodels	Statistics	10 067	153	21	234	2 868	1 240	66	479	19
TensorFlow	Deep learning	33 339	1 469	58	7 968	99 664	62 952	23	575	68
PYTORCH	Deep learning	11 306	635	16	816	15 512	3 483	18	707	24
Keras	Deep learning	4 539	671	41	1 673	29 444	10 964	7	1111	44
dist-keras	Distributed deep learning	1125	5	7	41	431	106	225	161	86
elephas	Distributed deep learning	170	13	5	97	913	189	13	34	70
spark-deep-learning	Distributed deep learning	67	11	3	116	920	206	6	22	84
Natural Language Toolkit	NLP	13 041	236	24	467	6 405	1 804	55	543	27
spaCy	NLP	8 623	215	56	425	9 258	1 446	40	154	43
gensim	NLP	3 603	273	52	415	6 995	2 689	13	69	26
Scrapy	Data scraping	6 625	281	81	1 723	27 277	6 469	24	82	97

Last reviewed: 13.02.2018

Created by ActiveWizards

- Matplotlib
- Scipy
- NumPy
- pandas

Importação de arquivos

- **Dataset**

Importação de arquivos

- **Datasets disponíveis**

Muitos sites fornecem datasets livres para utilização:

- <https://www.kaggle.com/datasets>
- <http://web.iodp.tamu.edu/LORE/> <- <- <- <- <-
- <http://dados.gov.br/>
- <http://www.portaldatransparencia.gov.br/>
- <https://www.usgs.gov/products/data-and-tools/science-datasets>

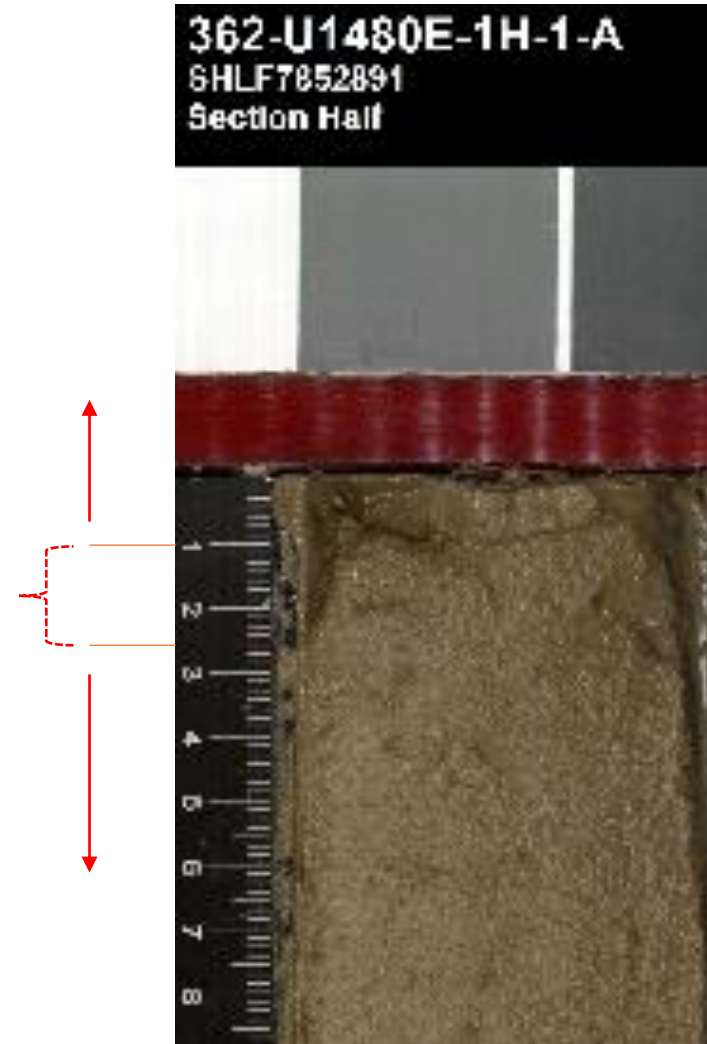
Bibliotecas para processamento

- **Baixar material da IODP**
- <http://web.iodp.tamu.edu/LORE/>
- **Vamos utilizar a propriedade GRA, Expedition 362, site U1480**
- **Arquivo: *.csv**
- **Separador: ,**

Bibliotecas para processamento

- **Dataset – leitura IODP**

Resolução – Intervalo de leitura



Bibliotecas para processamento

- **Baixar material da IODP**
- **Importante:**
- Antes de importar, ajustar nome das colunas. Eliminar espaços ou caracteres especiais.

Bibliotecas para processamento

- **Pandas:** <https://pandas.pydata.org/>

Bibliotecas para processamento

- **Pandas:** <https://pandas.pydata.org/>
- Biblioteca muito utilizada para análise, manipulação e formação de dados
- Necessita ser instalado no Python (pode ser instalado no ambiente selecionado): pip install pandas ou conda install pandas ou pelo gerenciador: Anaconda Navigator
- Necessita ser importada antes da utilização:
- `import pandas as pd`

Bibliotecas para processamento

- **Pandas: consultar instalação**

```
import pandas as pd
```

```
pd.__version__
```

Bibliotecas para processamento

- **Pandas: dataframe**

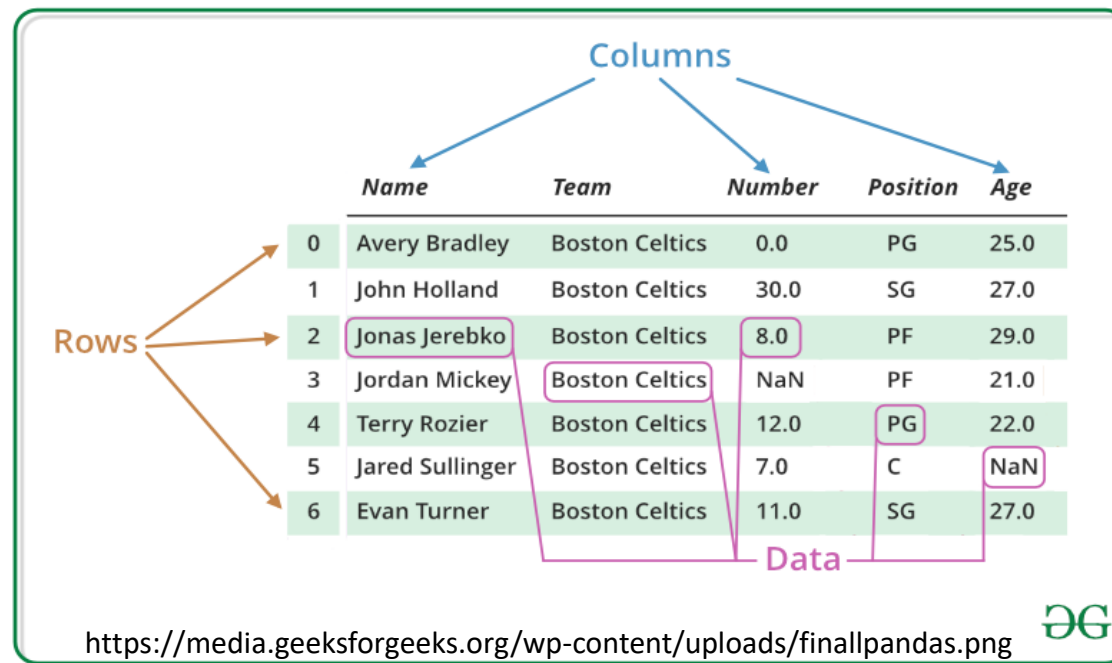
Forma uma estrutura de dados bidimensional, suporte a dados heterogêneos, com eixo rotulados (linhas e colunas). Os dados são tabulados conforme organização entre linhas e colunas formando uma tabela de dados.

0 based indexing

	0	1	2	
	col1	col2	col3	
0	1	a	b	c
1	2	d	e	f
2	3	g	h	i

Column labels

Index labels



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Bibliotecas para processamento

- **Pandas: dataframe**

Por exemplo:

```
import pandas as pd
```


```
#lista de cidades
```

```
idades = ['Porto Alegre', 'Curitiba', 'Fortaleza', 'Maceio',  
          'Santiago', 'Brasília', 'São Paulo']
```

```
#DataFrame constructor
```

```
df = pd.DataFrame(cidades)  
print(df)
```

Saída:



	0
0	Porto Alegre
1	Curitiba
2	Fortaleza
3	Maceio
4	Santiago
5	Brasília
6	São Paulo

Bibliotecas para processamento

- **Pandas: dataframe**

Por exemplo:

```
import pandas as pd
```

```
#lista de cidades
```

```
idades = {"Cidade": ['Porto Alegre', 'Curitiba', 'Fortaleza', 'Maceio',  
                    'Santiago', 'Brasília', 'São Paulo'], "Estado": ['RS','PR','CE','AL','RS','DF','SP']}
```

```
#DataFrame constructor
```

```
df = pd.DataFrame(cidades)
```

```
print(df)
```

Saída:

	Cidade	Estado
0	Porto Alegre	RS
1	Curitiba	PR
2	Fortaleza	CE
3	Maceio	AL
4	Santiago	RS
5	Brasília	DF
6	São Paulo	SP

Bibliotecas para processamento

- **Pandas: dataframe**

Importante sempre consulta a documentação:

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

```
class pandas.DataFrame(data=None, index=None, columns=None,
dtype=None, copy=False)
```

```
>>> d = {'col1': [1, 2], 'col2': [3, 4]}
>>> df = pd.DataFrame(data=d)
>>> df
   col1  col2
0     1     3
1     2     4
```


Bibliotecas para processamento

- **Pandas: dataframe**

```
>>>type(df)
```

```
-> pandas.core.frame.DataFrame
```

```
>>>df.dtypes
```

```
Cidade  object
```

```
Estado  object
```

```
dtype: object
```

Bibliotecas para processamento

- **Pandas: dataframe**

```
>>>df["Cidade"]
```

```
0    Porto Alegre
```

```
1      Curitiba
```

```
2    Fortaleza
```

```
3      Maceio
```

```
4    Santiago
```

```
5    Brasília
```

```
6    São Paulo
```

```
Name: Cidade, dtype: object
```

Conversão de tipo (exemplo):
`df["A"] =pd.to_numeric(df["A"])`

Bibliotecas para processamento

- **Pandas: dataframe**

```
>>>df.head()    #retorna um número n de linhas (resumo do dataset).  
Padrão: 5
```

Out[18]:

	Cidade	Estado
0	Porto Alegre	RS
1	Curitiba	PR
2	Fortaleza	CE
3	Maceio	AL
4	Santiago	RS

```
>>> df          #listar todo o dataset – cuidado  
                #com o tamanho!!!
```

Out[21]:

	Cidade	Estado
0	Porto Alegre	RS
1	Curitiba	PR
2	Fortaleza	CE
3	Maceio	AL
4	Santiago	RS
5	Brasília	DF
6	São Paulo	SP

Bibliotecas para processamento

- **Pandas: dataframe**

```
>>> df.rename(columns = {'Cidade': 'Cid'})
```

```
>>> df.drop(['Cidade'], axis=1)
```

```
>>> df.drop([0], axis=0)
```

```
>> df.iloc[0:2] #selecionar intervalo de linhas no dataset
```

```
>> df.sample(frac=0.5) #acesso aleatório a uma quantidade de dados
```

Importação de arquivos .xlsx e .csv

- Criando o dataset de dados com o Pandas DataFrame
- **Dataset a partir de arquivo *.xlsx**

```
import pandas as pd
```

```
#nome do arquivo
```

```
file = "... "
```

```
df = pd.read_excel(file, sheet_name = '...')
```

Importação de arquivos .xlsx e .csv

- Criando o dataset de dados com o Pandas DataFrame
- **Dataset a partir de arquivo *.csv com delimitador: ,:**

```
import pandas as pd
```

```
#nome do arquivo
```

```
file = "..."
```

```
#ler arquivos .csv com delimitador informado, neste caso a ,
```

```
df = pd.read_csv(file,sep=",")
```

Importação de arquivos .xlsx e .csv

- Atividades sobre os dados importados:

- `df.describe()`

Resumo da estatística com média, maior, menor,...

- `df["<column>"].max()`
- `df["<column>"].min()`
- `df["<column>"].mean()`
- `df.count()`
- `df.value_counts()`

Importação de arquivos .xlsx e .csv

- Atividades sobre os dados importados:

- `df.sort_values(by= "<column>", ascending=...)`

Ordenar os dados por ordem crescente ou decrescente.

Seleção/Consulta :

- `df["<column>"]` #column
- `df[...:..]` #line por exemplo: `df[0:2]` # usa o index

Importação de arquivos .xlsx e .csv

- Atividades sobre os dados importados:

Seleção/Consulta :

- `df[df["<column>"] > 0]`
- Consulta por coluna. Retorna dataframe completo pelo parâmetro da consulta.
- `df.filter(["Exp", "Site"])`
- Mostar na tela apenas as colunas selecionadas.

Importação de arquivos .xlsx e .csv

- Filtro nos dados:

#dados em branco ou ausentes - consultar usando os operadores

Logic in Python (and pandas)			
<	Less than	!=	Not equal to
>	Greater than	df.column.isin(values)	Group membership
==	Equals	pd.isnull(obj)	Is NaN
<=	Less than or equals	pd.notnull(obj)	Is not NaN
>=	Greater than or equals	&, , ~, ^, df.any(), df.all()	Logical and, or, not, xor, any, all

Importação de arquivos .xlsx e .csv

- Filtro nos dados:

#dados em branco ou ausentes

None ou NaN (Not a Number) # numpy nan (np.nan)

0 ou “ ” ou ‘ ’ não são dados em branco.

Importação de arquivos .xlsx e .csv

- Filtro nos dados:

#dados outliers

Valores fora do padrão, dados com ruídos ou desproporcionais

- Visualizado através de um box plot, ou um gráfico de pontos simples
- Ajustar esses valores com uma função específica, por exemplo, `scipy.stats.zscore`)
- Uma forma simples é comparar os valores com um intervalo conhecido.

```
dataset[(dataset["valor"] > 1) & (dataset["valor"] < 2)]
```

Bibliotecas para processamento

- **Pandas: dataframe**

Acessar o conteúdo do dataframe via estrutura de repetição (for):

```
...  
for line in range(0,len(dataset)):  
    print (dataset["..."][line])  
...
```

Bibliotecas para processamento

- **Criar arquivos .csv**
- Utiliza a biblioteca pandas

```
import pandas as pd
```

```
df = pd.DataFrame({'nome': ['João', 'Carlos'],  
                  'Cor': ['red', 'Blue'],  
                  'Filmes': ['Fast and Furious', 'Bruxa de Blair']})
```

```
df.to_csv("teste1.csv")
```

Bibliotecas para processamento

- **Pandas: dataframe**
- Combinar dados (concatenar)
- `pd.concat([dt1,dt2])`
- `pd.concat([dt1,dt2], axis=1)`

----Exemplo com importação de dados (.xlsx e .csv)

Bibliotecas para processamento

- **Pandas: dataframe**
- Combinar dados (merge - juntar)
- `pd.merge(dt1,dt2, how='left', on='Nome_Rocha')`

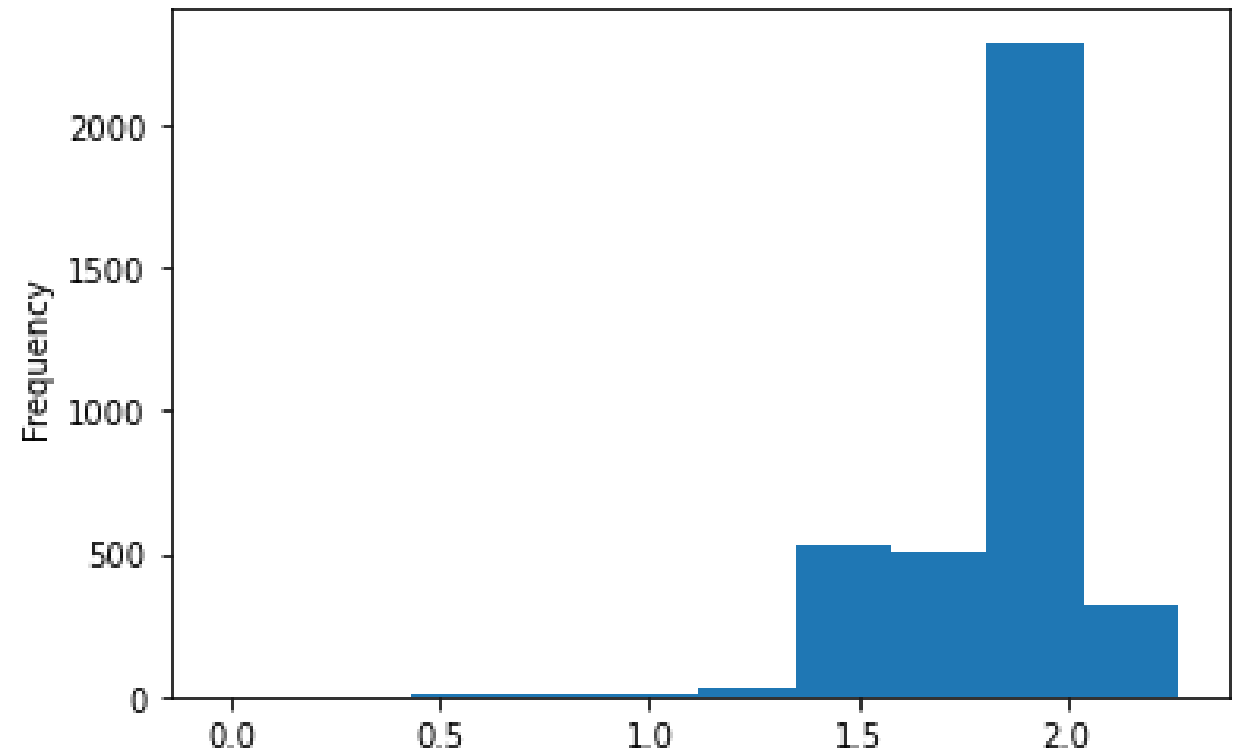
----Exemplo com importação de dados (.xlsx e .csv)

Bibliotecas para processamento

- **Pandas: dataframe**
- Plotting (gráficos simples)
- https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

Bibliotecas para processamento

- **Pandas: dataframe**
- Plotting (gráficos simples)
- Histograma:
`dataset["..."].plot.hist()`



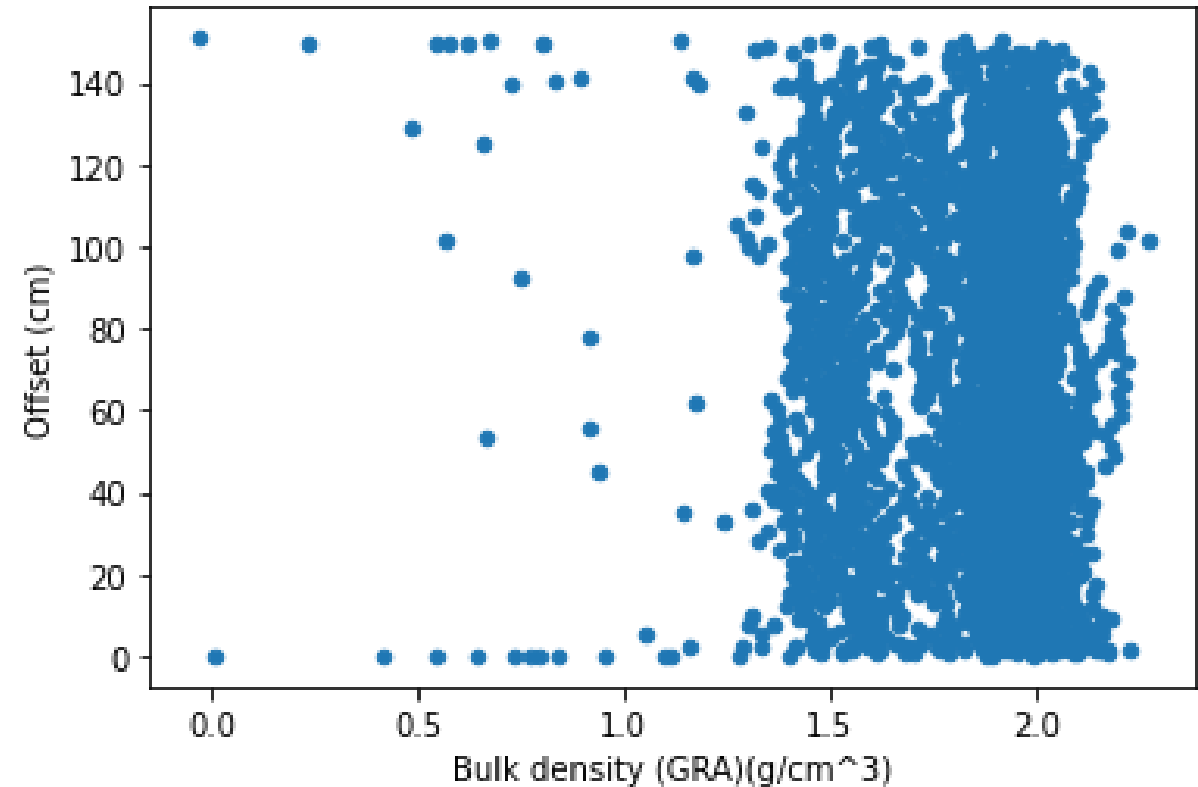
Bibliotecas para processamento

- **Pandas: dataframe**

- Plotting (gráficos simples)

- Scatter plot

```
dataset.plot.scatter(y='Offset(cm)',  
x="Bulkdensity(GRA)")
```



Bibliotecas para processamento

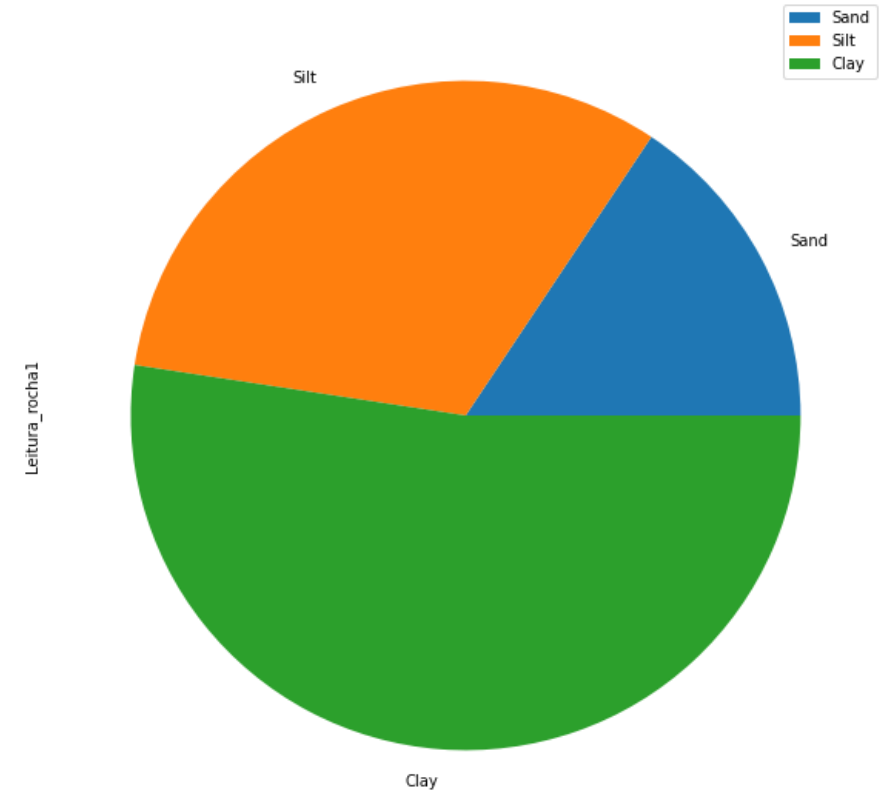
- **Pandas: dataframe**

- Plotting (gráficos simples)

- Pie

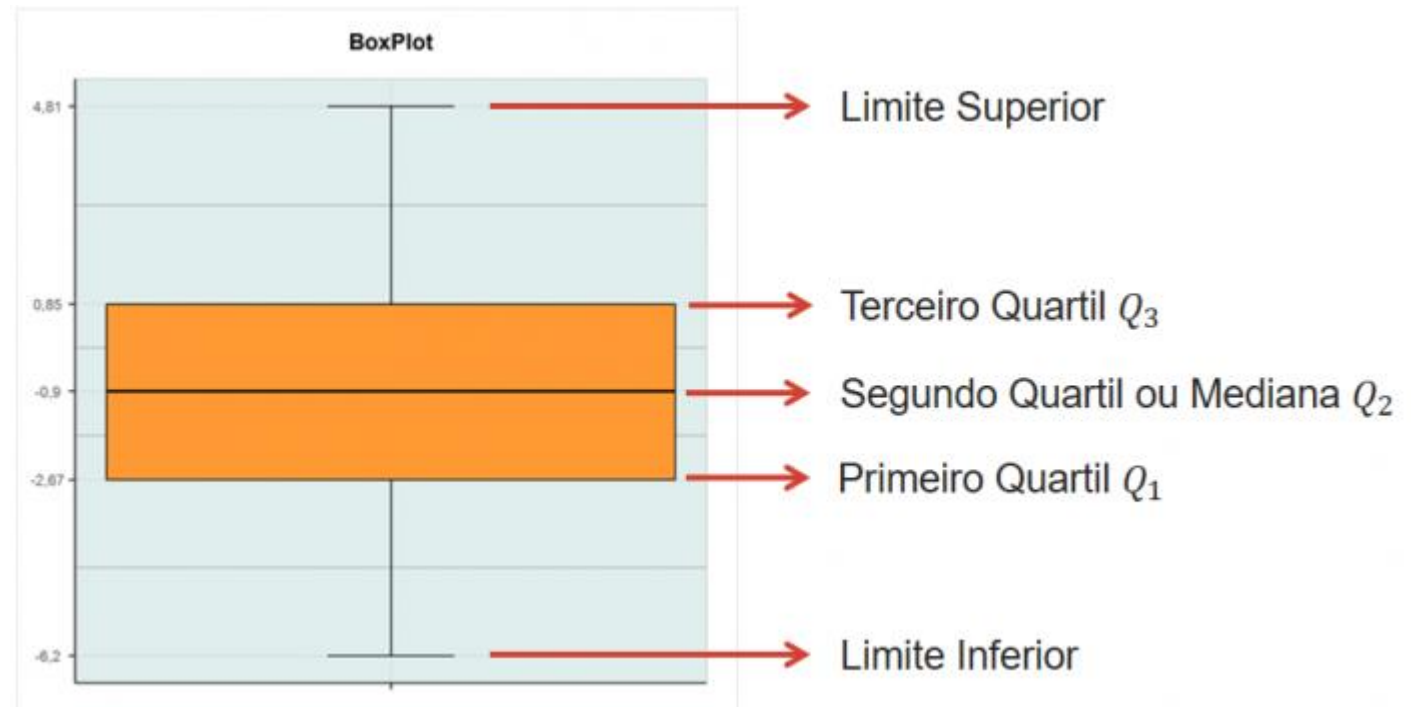
```
df = pd.DataFrame({'Leitura_rocha1': [1.2, 2.45 , 4.02],  
                  'Leitura_rocha2': [550.7, 480.8, 369.5]},  
                  index=['Sand', 'Silt', 'Clay'])
```

```
plot = df.plot.pie(y='Leitura_rocha1', figsize=(10, 10))
```



Bibliotecas para processamento

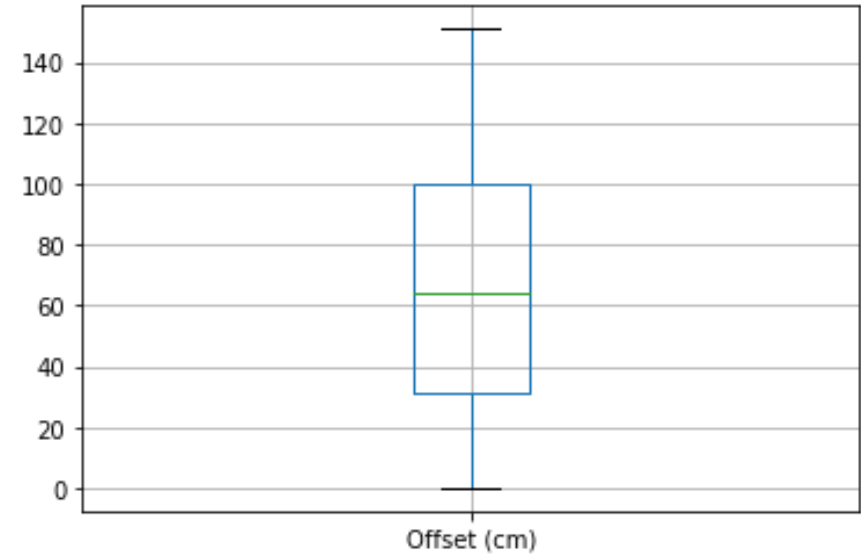
- **Pandas: dataframe**
- Plotting (gráficos simples)
- Boxplot



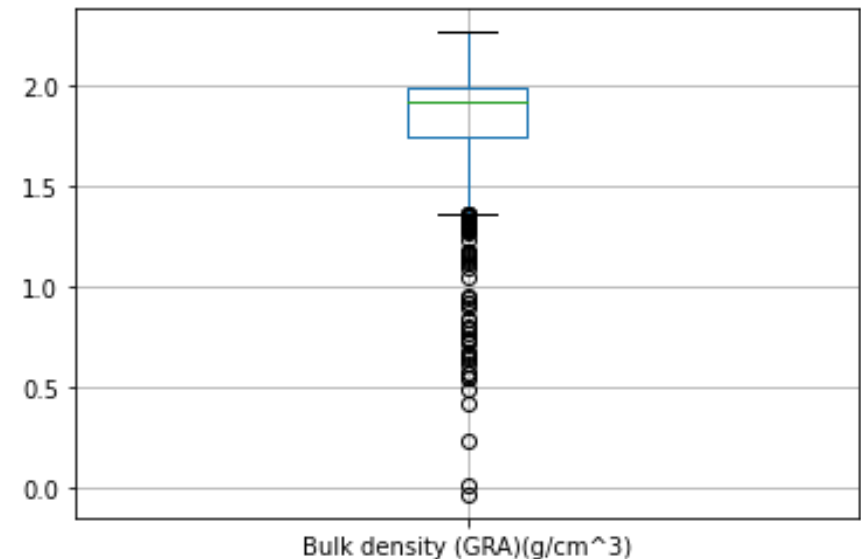
Bibliotecas para processamento

- **Pandas: dataframe**
- Plotting (gráficos simples)
- Boxplot

```
dataset.boxplot(column=['Offset (cm)'])
```



```
dataset.boxplot(column=['Bulk density (GRA)(g/cm^3)'])
```



Bibliotecas para processamento

- **Numpy:** <https://numpy.org/>

Bibliotecas para processamento

- **Numpy: <https://numpy.org/>**
- Principal biblioteca para computação científica em Python
- Biblioteca complementar do Panda
- Necessita ser importada antes da utilização:
- `import numpy as np`
- Suporte a array unidirecionais (vetor) ou multidimensionais(matriz)

Bibliotecas para processamento

- **numpy: consultar instalação**

```
import numpy as np
```

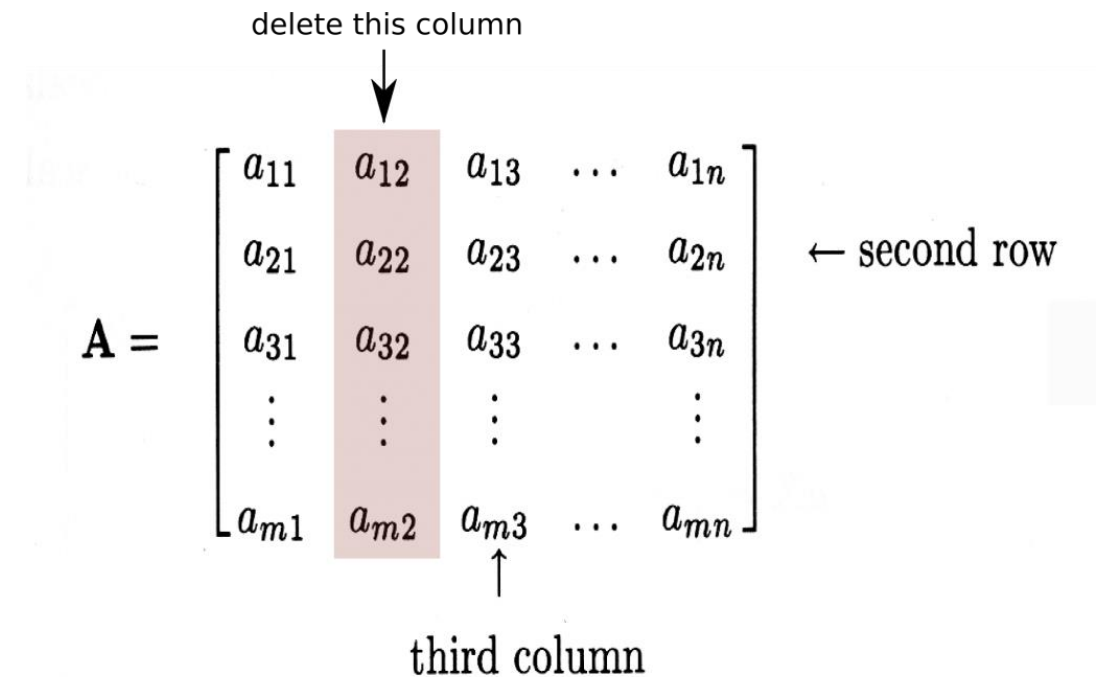
```
np.__version__
```

Bibliotecas para processamento

- **numpy: array**

Arrays NumPy

- tipo de dado chamado ndarray



Bibliotecas para processamento

- **numpy: array**

Exemplo:

```
import numpy as np
```

```
ar_1 = np.arange(10)
```

```
print(ar_1)
```

```
#unidimensional de 0 a 9
```

Bibliotecas para processamento

- **numpy: array**

Exemplo:

```
import numpy as np
```

```
ar_1 = np.array([1, 2, 3, 4, 5])
```

```
print(ar_1)
```

```
print (ar_1.shape)  #dimensões da array
```

```
#array unidimensional de inteiros(igual a uma lista)
```

Bibliotecas para processamento

- **numpy: array**

Exemplo:

```
import numpy as np
```

```
ar_1 = np.array([[1, 2, 3, 4, 5], [10, 21, 35, 46, 500]])
```

```
print(ar_1)
```

```
print (ar_1.shape)
```

```
#matriz de duas linhas e 5 colunas
```

Bibliotecas para processamento

- **numpy: array**

Exemplo:

Acessar elemento por elemento:

```
print(ar_1[0])
```

```
print(ar_1[1])
```

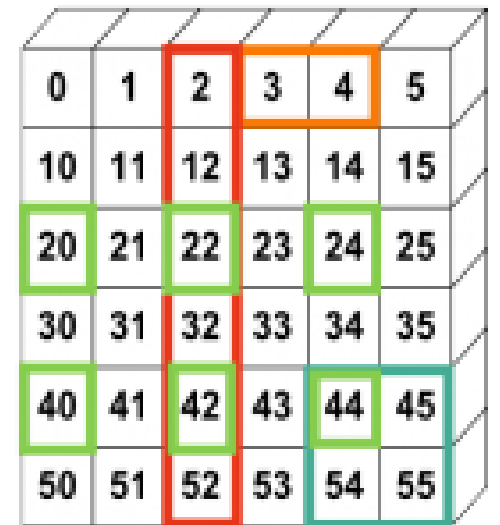
```
print(ar_1[0][0])
```

```
>>> a[0,3:5]  
array([3,4])
```

```
>>> a[4:,4:]  
array([[44, 45],  
       [54, 55]])
```

```
>>> a[:,2]  
array([ 2,12,22,32,42,52])
```

```
>>> a[2::2,::2]  
array([[20,22,24]  
       [40,42,44]])
```



0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

http://www.estruturas.ufpr.br/wp-content/uploads/2015/01/numpy_indexing-300x171.png

Bibliotecas para processamento

- **numpy: array**

Exemplo:

Criar uma array vazia:

```
ar_2 = np.empty([3,2], dtype = int)
```

#3 linhas e 2 colunas – atribui valores inteiros aleatórios na matriz

```
ar_2 = np.zeros((5,2))
```

matriz com valores igual a zero

#Ou com valores aleatórios

```
ar_3 = np.random.random((5,2))
```

Bibliotecas para processamento

- **numpy: array**

Exemplo:

Operações - cálculos matemáticos - entre arrays: +, -, /, *

```
import numpy as np
```

```
a = np.array([[10.0, 2.0], [30.0, 4.0]])
```

```
b = np.array([[50.0, 6.0], [7.0, 80.0]])
```

```
soma = a+b
```

#possui também o operador `sum()`, `prod()`,

Bibliotecas para processamento

- **numpy: array**

Exemplo:

Coeficiente de Correlação Pearson (corrcoef) – R^2

```
import numpy as np  
a = np.array([10.0, 2.0])  
b = np.array([50.0, 6.0])  
np.corrcoef([a,b])
```

Bibliotecas para processamento

- **numpy: array**

Exemplo:

Normalização (linalg.norm)

```
import numpy as np
```

```
a = np.array([ 1.797, 2.7607, 0.4383, 0.7866, 1.8091, 0.1954,  
              3.6307, 4.6599, 0.1065, 2.0508])
```

```
norm = np.linalg.norm(a)
```

```
a' = a/norm
```

Bibliotecas para processamento

- **SciPy:** <https://www.scipy.org/>

Bibliotecas para processamento

- **SciPy:** <https://www.scipy.org/>
- Biblioteca completa para matemática, ciência e engenharia.
- Utiliza como base a biblioteca *Numpy* com suporte para trabalhar com grande quantidade de dados
- Necessita ser importada antes da utilização:
- `import scipy as sc`
- Suporte a estatísticas, processamento de sinais e imagens, solução de equações, funções especiais, polinômios,...

Bibliotecas para processamento

- **SciPy:** <https://www.scipy.org/>
- Principais funções:
- `scipy.stats`, `scipy.interpolate`, `scipy.linalg`, entre outras.

<https://docs.scipy.org/doc/scipy/reference/index.html>

Bibliotecas para processamento

- **SciPy:** stats
- Pearsonr

```
import scipy.stats as sc
```

```
a = np.array([0, 0, 0, 6, 1, 1, 6])
```

```
b = np.array([0, 8, 0, 8, 1, 1, 4])
```

```
sc.pearsonr(a, b)[0]
```

*Pode ser utilizado em dados importados (pandas.dataframes)

Bibliotecas para processamento

- **SciPy:** stats
- Spearmanr

```
import scipy.stats as sc
```

```
a = np.array([0, 0, 0, 6, 1, 1, 6])
```

```
b = np.array([0, 8, 0, 8, 1, 1, 4])
```

```
sc.spearmanr(a, b)
```

*Pode ser utilizado em dados importados (pandas.dataframes)

Bibliotecas para processamento

- **SciPy:** interpolate
- interp1d

```
from scipy.interpolate import interp1d
import numpy as np
x = np.array([10.2, 20.1, np.nan, 40.4, 65.2, np.nan, 31.1])

not_nan = np.logical_not(np.isnan(x))
indice = np.arange(len(x))
interp = interp1d(indice[not_nan], x[not_nan])
interp(indice)
```

*Pode ser utilizado em dados importados (pandas.dataframes)

Bibliotecas para processamento

- **pandas:** interpolation
- `<dataframe>.interpolate(method='...')`

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.interpolate.html>

Bibliotecas para processamento

- **SciPy:** ndimage
 - Analise e processamento de imagens 2D (scipy.ndimage)
 - Extrair características, classificar, interpolar, aplicar filtros.
- OpenCV: cv2
 - Biblioteca para processamento gráfico – visão computacional
 - Editar imagem, criar imagem, converter,...

Bibliotecas para processamento

- **Exemplos Scipy ndimage e cv2**

```
import cv2
```

```
image=cv2.imread('362-u1480e-1h-1-  
a_shlf7852891_20160813075212_trim.jpg')
```

```
type(image)
```

```
print (image)
```

```
Image.shape
```

Bibliotecas para processamento

- **Exemplos Scipy ndimage e cv2**

mostrar os canais RGB

```
red=image[:, :, 0]
```

```
green=image[:, :, 1]
```

```
blue=image[:, :, 2]
```

#imagem – escala de cinza

```
grey = (0.2126 * red) + (0.7152 * green) + (0.0722 * blue)
```

Bibliotecas para processamento

- **Exemplos Scipy ndimage e cv2**

```
# mostrar imagem
import matplotlib.pyplot
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
matplotlib.pyplot.imshow(image)
matplotlib.pyplot.show()
cv2.imwrite('original.jpg', cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

import matplotlib.cm as cm
matplotlib.pyplot.imshow(grey, cmap = matplotlib.cm.Greys_r)
matplotlib.pyplot.show()

#Exportar
cv2.imwrite('grey_test.jpg', grey)
```

Bibliotecas para processamento

- **Exemplos Scipy ndimage e cv2**

aplicar filtro

```
from scipy.ndimage import gaussian_filter  
import cv2
```

```
image=cv2.imread('362-u1480e-1h-1-  
a_shlf7852891_20160813075212_trim.jpg')  
image = gaussian_filter(image,sigma=5)  
cv2.imwrite('filter_test.jpg', cv2.cvtColor(image,  
cv2.COLOR_BGR2RGB))
```

Bibliotecas para processamento

- **Matplotlib:** <https://matplotlib.org/>

Bibliotecas para processamento

- **MatPlotLib:** <https://matplotlib.org/>
- Biblioteca completa para organização e montagem de gráficos em 2D e 3D
- Necessita ser importada antes da utilização:
- `import matplotlib as plt`
- Suporte a criação de gráficos, figuras, gráficos de linhas e multilinhas, colunas, áreas, pizza, Scatter, spectrum, ..., muitos outros gráficos conforme o tipo de dados

Bibliotecas para processamento

- **MatPlotLib:** <https://matplotlib.org/>
- Método: pyplot (método principal para criar gráficos)
- `import matplotlib.pyplot as plt`
- ou
- `import matplotlib.pyplot`

(importante destacar que as demais bibliotecas-numpy, scipy, pandas-
também são utilizadas em conjunto.)

Bibliotecas para processamento

- **MatPlotLib:** Exemplos (gráfico em linha)

```
import matplotlib
```

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

```
import numpy as np
```

```
labels = ['P1', 'P2', 'P3', 'P4', 'P5']
```

```
Valor1 = [20, 34, 30, 35, 27]
```

```
Valor2 = [25, 32, 34, 20, 25]
```

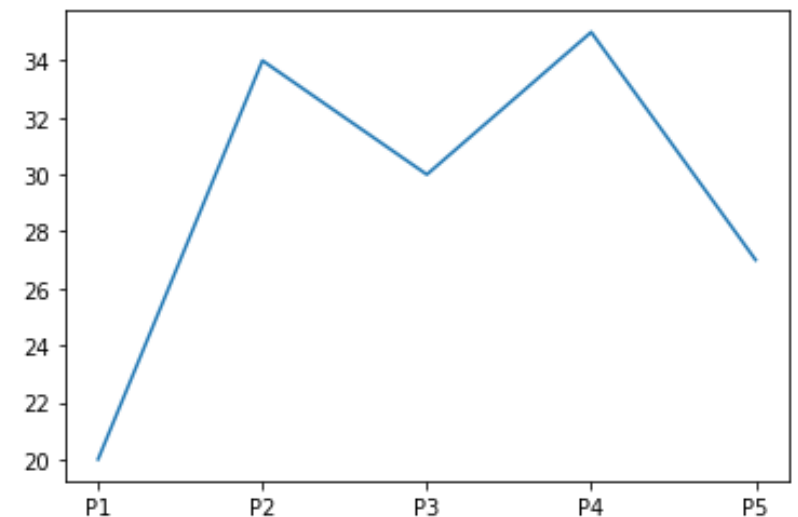
Bibliotecas para processamento

- **MatPlotLib:** Exemplos

#plot – desenha um gráfico de linha

matplotlib.pyplot.plot(labels, Valor1) #labels = x, Valor1 = y

matplotlib.pyplot.show() #mostrar na tela

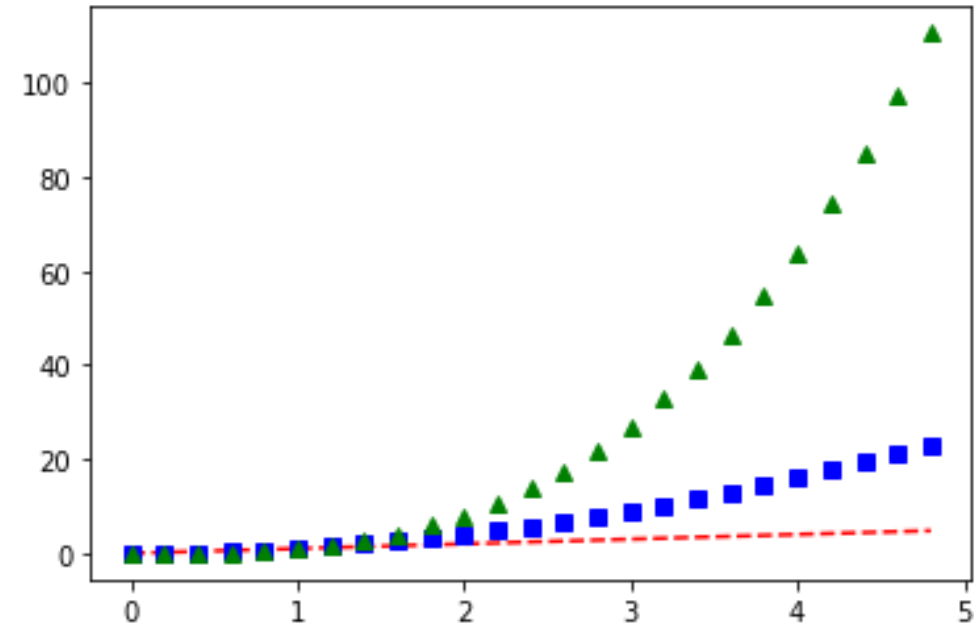


Bibliotecas para processamento

- **Matplotlib:** Exemplos

#plot – configuração

```
t = np.arange(0.0, 5.0, 0.2)
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



Bibliotecas para processamento

- **Matplotlib:** Exemplos

```
matplotlib.pyplot.title('Valor por Propriedade') #adicionar título ao gráfico
```

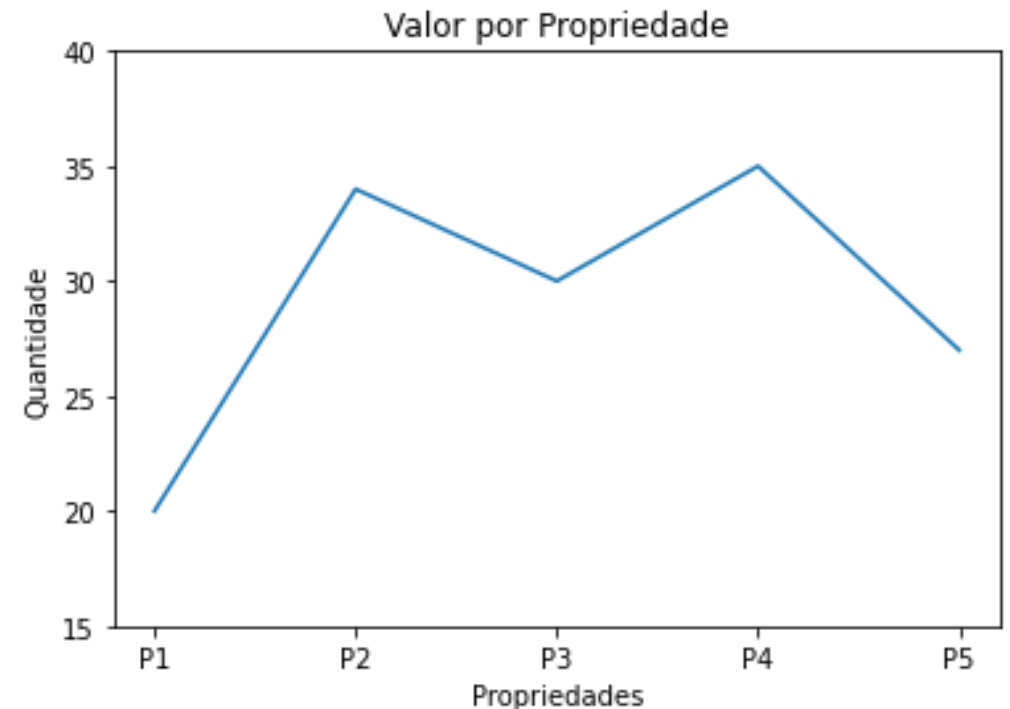
```
matplotlib.pyplot.plot(labels, Valor1) #labels = x, Valor1 = y
```

```
matplotlib.pyplot.ylim(15, 40) #intervalo de valores ao eixo y
```

```
matplotlib.pyplot.xlabel('Propriedades') #título ao eixo x
```

```
matplotlib.pyplot.ylabel('Quantidade') #título ao eixo y
```

```
matplotlib.pyplot.show()
```



Bibliotecas para processamento

- **Matplotlib:** Exemplos - subplots

```
names = ['Pro1', 'Prop2', 'Prop3']
```

```
values = [1, 10, 100]
```

```
plt.figure(figsize=(10, 5))
```

```
plt.subplot(131)
```

```
plt.bar(names, values)
```

```
plt.subplot(132)
```

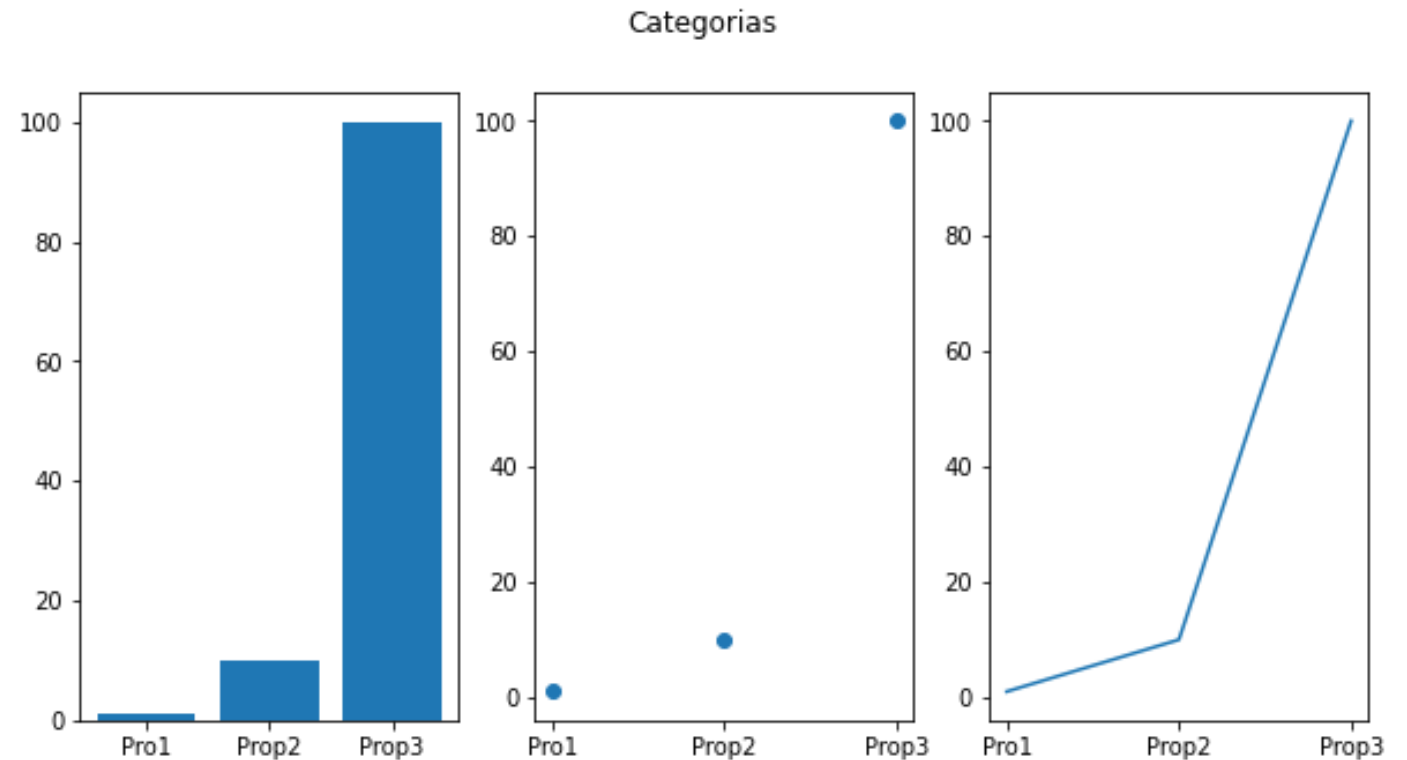
```
plt.scatter(names, values)
```

```
plt.subplot(133)
```

```
plt.plot(names, values)
```

```
plt.suptitle('Categorias')
```

```
plt.show()
```



Bibliotecas para processamento

- **Matplotlib:** Exemplos – subplots – outra forma 1(subplot com axes)

```
names = ['Pro1', 'Prop2', 'Prop3']
```

```
values = [1, 10, 100]
```

```
plt.figure(figsize=(10, 5))
```

```
ax1=plt.subplot(1, 3, 2)
```

```
plt.bar(names, values)
```

```
ax2=plt.subplot(1, 3, 1)
```

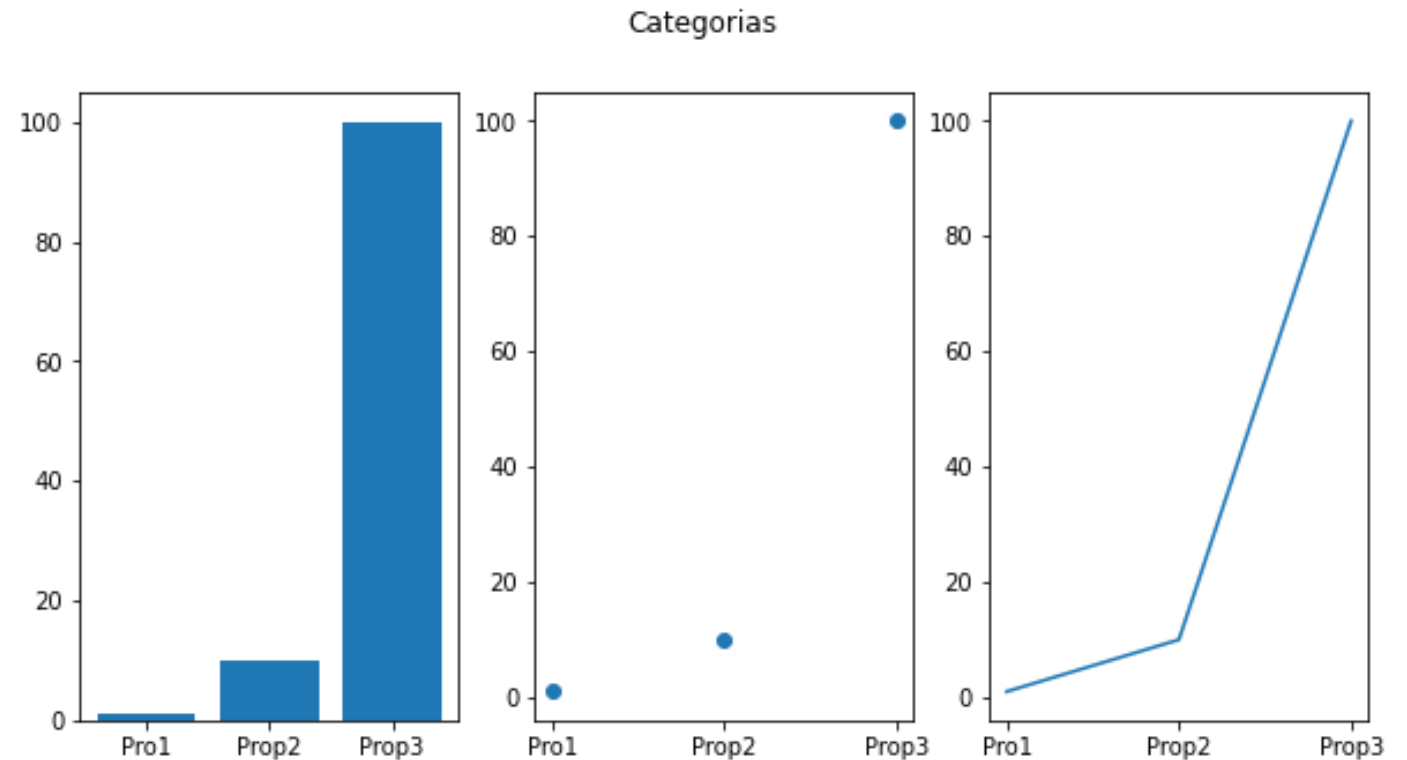
```
plt.scatter(names, values)
```

```
ax2=plt.subplot(1, 3, 3)
```

```
plt.plot(names, values)
```

```
plt.suptitle('Categorias')
```

```
plt.show()
```



Bibliotecas para processamento

- **Matplotlib:** Exemplos – subplots – com 2 plt.Figure

```
names = ['Pro1', 'Prop2', 'Prop3']
```

```
values = [1, 10, 100]
```

```
#-----
```

```
plt.figure(1, figsize=(10, 5))
```

```
ax1=plt.subplot(1, 3, 2)
```

```
plt.bar(names, values)
```

```
ax2=plt.subplot(1, 3, 1)
```

```
plt.scatter(names, values)
```

```
ax2=plt.subplot(1, 3, 3)
```

```
plt.plot(names, values)
```

```
plt.suptitle('Categorias')
```

```
plt.show()
```

```
#-----
```

```
plt.figure(2, figsize=(10, 5))
```

```
ax1=plt.subplot(1, 3, 2)
```

```
plt.bar(names, values)
```

```
ax2=plt.subplot(1, 3, 1)
```

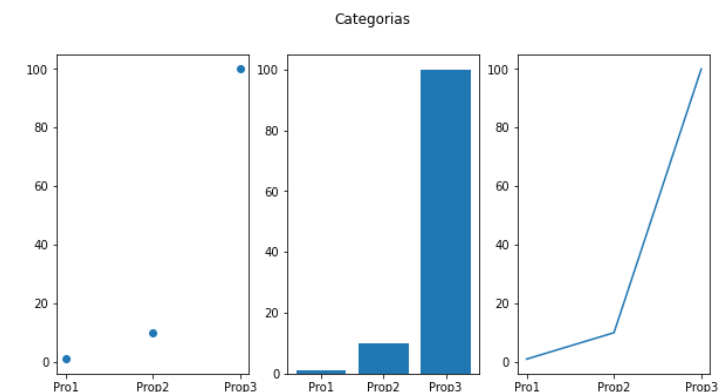
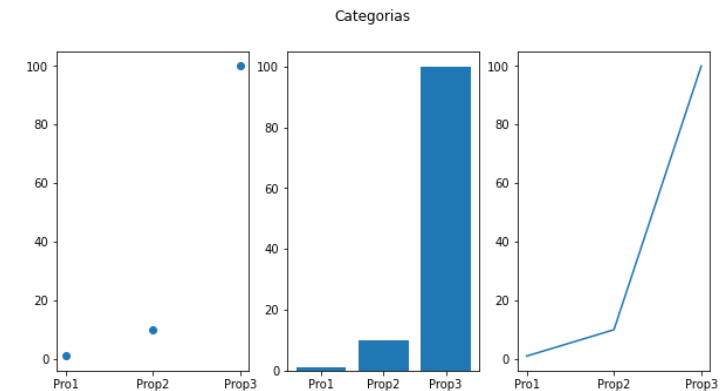
```
plt.scatter(names, values)
```

```
ax2=plt.subplot(1, 3, 3)
```

```
plt.plot(names, values)
```

```
plt.suptitle('Categorias')
```

```
plt.show()
```



Bibliotecas para processamento

- **Matplotlib:** Exemplos (gráfico em barra)

```
x = np.arange(len(labels))
```

```
width = 0.35
```

```
fig, ax = plt.subplots()
```

```
ax.bar(x - width/2, Valor1, width, label='Valor1')
```

```
ax.bar(x + width/2, Valor2, width, label='Valor2')
```

```
ax.set_ylabel('Quantidade')
```

```
ax.set_xlabel('Propriedade')
```

```
ax.set_title('Valor por Propriedade')
```

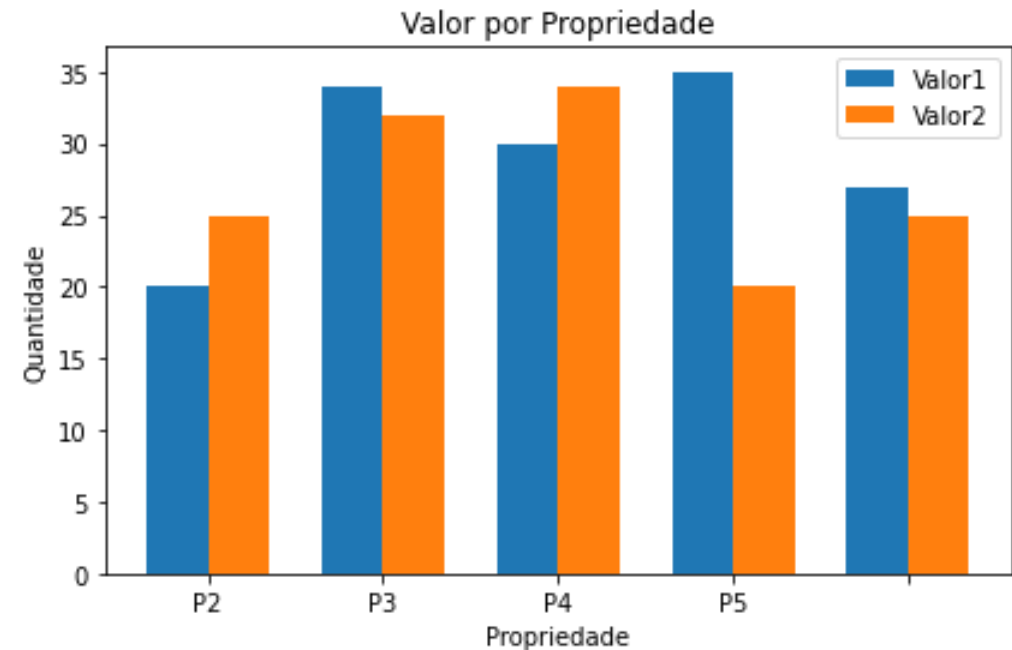
```
ax.set_xticks(x)
```

```
ax.set_xticklabels(labels)
```

```
ax.legend()
```

```
fig.tight_layout()
```

```
plt.show()
```



Bibliotecas para processamento

- **Matplotlib:** Exemplos (gráfico em barra-axes)

```
x = np.arange(len(labels))
```

```
width = 0.35
```

```
fig, ax = plt.subplots(2) #vertical ou horizontal
```

```
ax[0].bar(x - width/2, Valor1, width, label='Valor1')
```

```
ax[0].bar(x + width/2, Valor2, width, label='Valor2')
```

```
ax[0].set_ylabel('Quantidade')
```

```
ax[0].set_xlabel('Propriedade')
```

```
ax[0].set_title('Valor por Propriedade')
```

```
ax[0].set_xticks(x)
```

```
ax[0].set_xticklabels(labels)
```

```
ax[0].legend()
```

```
ax[1].bar(x - width/2, Valor11, width, label='Valor1')
```

```
ax[1].bar(x + width/2, Valor22, width, label='Valor2')
```

```
ax[1].set_ylabel('Quantidade')
```

```
ax[1].set_xlabel('Propriedade')
```

```
ax[1].set_title('Valor por Propriedade')
```

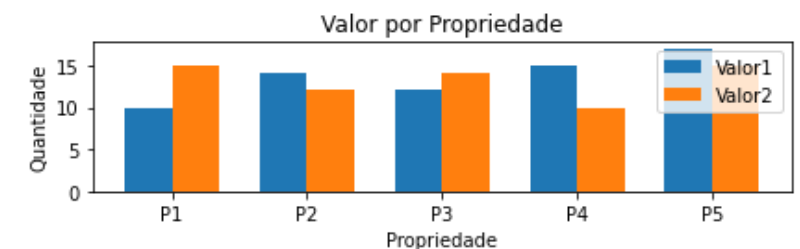
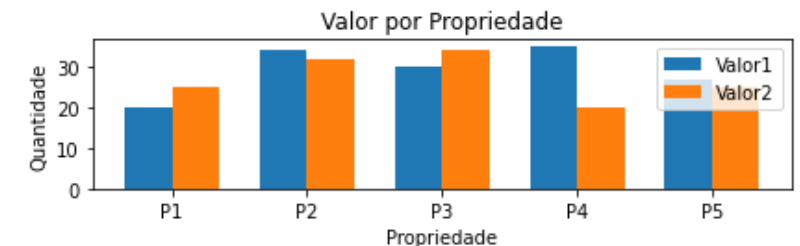
```
ax[1].set_xticks(x)
```

```
ax[1].set_xticklabels(labels)
```

```
ax[1].legend()
```

```
fig.tight_layout()
```

```
plt.show()
```



Bibliotecas para processamento

- **Matplotlib:** Exemplos
- Exemplos com dados do DataFrame

```
file = "csv_merge_properties.csv"
```

```
#ler arquivos .csv com delimitador informado, neste caso a ,
```

```
df_importcsv_merge = pd.read_csv(file,sep=",")
```

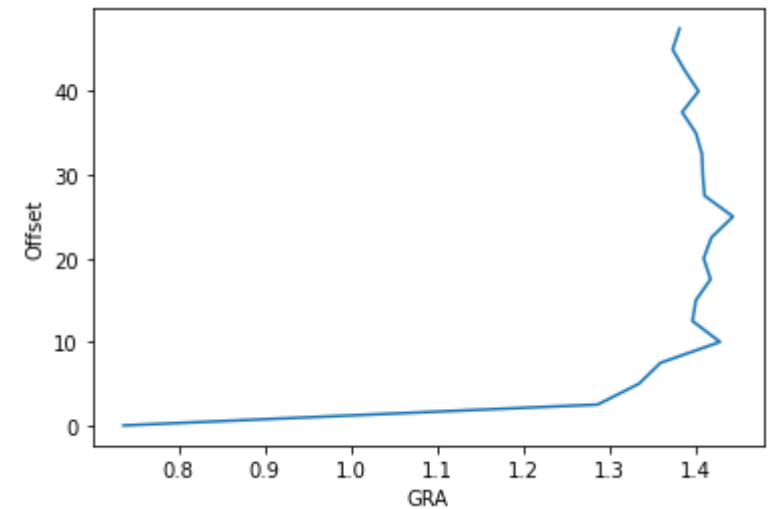
```
df_importcsv_merge1 = df_importcsv_merge[0:20]
```

```
matplotlib.pyplot.plot(df_importcsv_merge1["Bulkdensity(GRA)"], df_importcsv_merge1["Offset(cm)"])
```

```
matplotlib.pyplot.xlabel('GRA')
```

```
matplotlib.pyplot.ylabel('Offset')
```

```
matplotlib.pyplot.show()
```



Bibliotecas para processamento

- **Matplotlib:** Exemplos
- Exemplos com dados do DataFrame

```
df_importcsv_merge1 =  
df_importcsv_merge1.sort_values(by='Offset(cm)', ascending=False)
```

```
fig, ax = plt.subplots(1,2,figsize=(6, 10))
```

```
ax[0].plot(df_importcsv_merge1["Bulkdensity(GRA)"],  
df_importcsv_merge1["Offset(cm)"], 'r--', label="GRA")
```

```
ax[0].set_ylabel('Offset')
```

```
ax[0].set_xlabel('GRA')
```

```
ax[0].set_title('Valor GRA')
```

```
ax[0].set_xlim(1.0, 1.5)
```

```
ax[0].set_ylim([48, 0])
```

```
ax[0].locator_params(axis='y', nbins=20)
```

```
ax[0].grid()
```

```
ax[0].legend()
```

```
ax[1].plot(df_importcsv_merge1["P-wavevelocity"],  
df_importcsv_merge1["Offset(cm)"], label="PWL")
```

```
ax[1].set_ylabel('Offset')
```

```
ax[1].set_xlabel('PWL')
```

```
ax[1].set_title('Valor PWL')
```

```
ax[1].set_xlim(1460, 1500)
```

```
ax[1].set_ylim([48, 0])
```

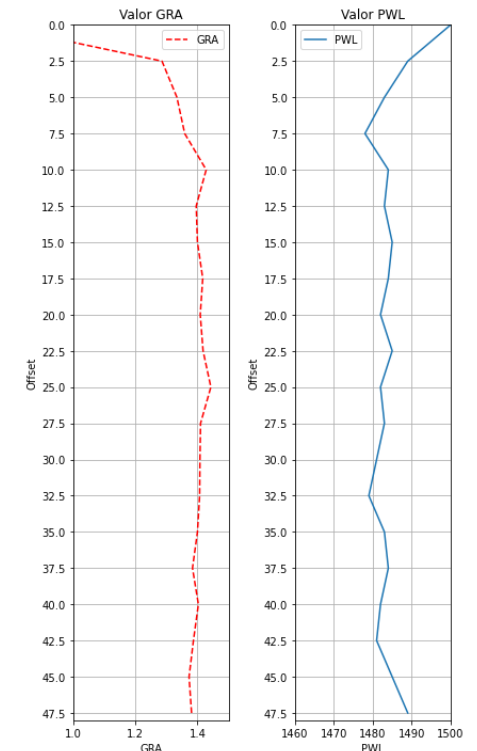
```
ax[1].locator_params(axis='y', nbins=20)
```

```
ax[1].grid()
```

```
ax[1].legend()
```

```
fig.tight_layout()
```

```
plt.show()
```



Bibliotecas para processamento

- **Matplotlib:** Exemplos
- Exemplos prático em código python:
- U1480E – 1H– Depth: 0.0 – 50.0
- Mostrar valor original e valor interpolado (a cada 1 cm).