

Introdução à Análise de Dados com Python

Paulo Henrique Sales Guimarães

paulo.guimaraes@ufla.br



O que é o Python?

O que é o Python?

Python é uma linguagem de programação de alto nível (VHLL - *Very High Level Language*) criada pelo holandês Guido Van Rossum.

O que é o Python?

Python é uma linguagem de programação de alto nível (VHLL - *Very High Level Language*) criada pelo holandês Guido Van Rossum.

- *Software* gratuito

O que é o Python?

Python é uma linguagem de programação de alto nível (VHLL - *Very High Level Language*) criada pelo holandês Guido Van Rossum.

- *Software* gratuito
- Código aberto

O que é o Python?

Python é uma linguagem de programação de alto nível (VHLL - *Very High Level Language*) criada pelo holandês Guido Van Rossum.

- *Software* gratuito
- Código aberto
- Disponibilidade em Windows, Linux, Mac, Palm, em celulares, e outra infinidade de sistemas

O que é o Python?

Python é uma linguagem de programação de alto nível (VHLL - *Very High Level Language*) criada pelo holandês Guido Van Rossum.

- *Software* gratuito
- Código aberto
- Disponibilidade em Windows, Linux, Mac, Palm, em celulares, e outra infinidade de sistemas
- Linguagem orientada a objetos.

O que é o Python?

O que é o Python?

- A comunidade é grande e ativa, sendo fácil de encontrar suporte

O que é o Python?

- A comunidade é grande e ativa, sendo fácil de encontrar suporte
- Atualmente há 137.000 bibliotecas no gerenciador de pacotes pip (2019);

O que é o Python?

- A comunidade é grande e ativa, sendo fácil de encontrar suporte
- Atualmente há 137.000 bibliotecas no gerenciador de pacotes pip (2019);
- **Curiosidade:** O nome Python não vem da cobra, mas sim de um grupo de comediantes dos anos 70: *Monty Python*.

Instalação

O *download* do Python pode ser feito na página: <https://www.python.org/>

Instalação

O download do Python pode ser feito na página: <https://www.python.org/>



Instalação

Há várias maneiras de se obter o interpretador Python. Ele pode ser baixado diretamente do site *python.org* ou por meio do Anaconda.



Mais detalhes sobre a instalação podem ser obtidas neste link.

IDE

A sigla IDE significa *Integrated Development Environment* - Ambiente de Desenvolvimento Integrado, em tradução livre. Trata-se de um programa que reúne uma série de ferramentas que facilitam a vida do programador.

IDE

A sigla IDE significa *Integrated Development Environment* - Ambiente de Desenvolvimento Integrado, em tradução livre. Trata-se de um programa que reúne uma série de ferramentas que facilitam a vida do programador.

Dentre seus recursos, podemos citar a presença do compilador - no qual você rodará seu programa; o editor, em que você escreverá seu código; e o depurador - debugger, que você provavelmente usará para entender por que o seu código não está funcionando.

IDE

A sigla IDE significa *Integrated Development Environment* - Ambiente de Desenvolvimento Integrado, em tradução livre. Trata-se de um programa que reúne uma série de ferramentas que facilitam a vida do programador.

Dentre seus recursos, podemos citar a presença do compilador - no qual você rodará seu programa; o editor, em que você escreverá seu código; e o depurador - debugger, que você provavelmente usará para entender por que o seu código não está funcionando.

Muitas IDEs permitem personalização do seu ambiente de trabalho, customizando desde sua aparência, com temas e cores, ou até alterar totalmente seu código fonte.

IDE

- Spyder

IDE

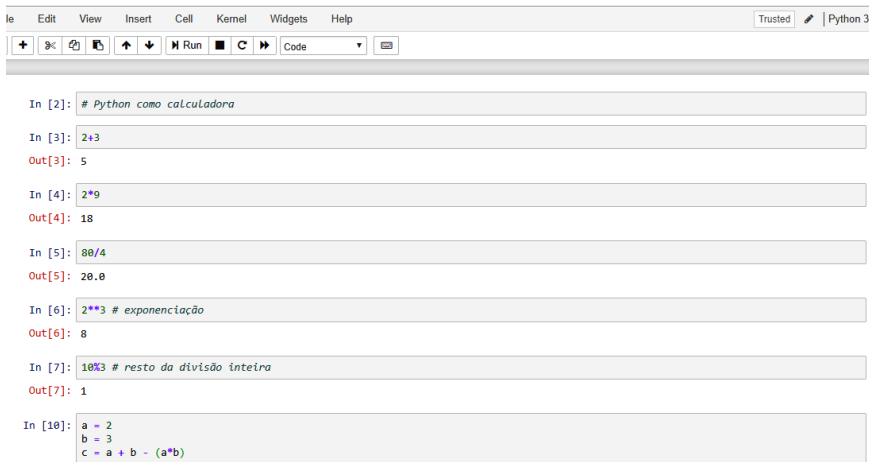
- Spyder
- Pycharm

IDE

- Spyder
- Pycharm
- Jupyter Notebook



Python como calculadora



The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for adding, saving, and running code. The notebook contains several input and output cells:

```
In [2]: # Python como calculadora
```

```
In [3]: 2+3
```

```
Out[3]: 5
```

```
In [4]: 2*9
```

```
Out[4]: 18
```

```
In [5]: 80/4
```

```
Out[5]: 20.0
```

```
In [6]: 2**3 # exponenciação
```

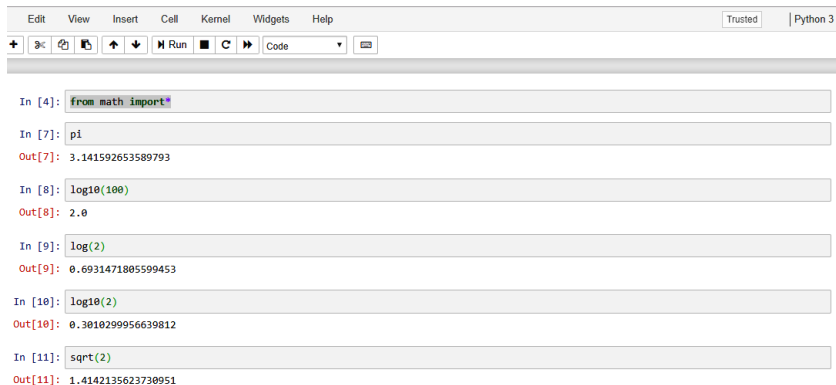
```
Out[6]: 8
```

```
In [7]: 10%3 # resto da divisão inteira
```

```
Out[7]: 1
```

```
In [10]: a = 2  
b = 3  
c = a + b - (a*b)
```

Python como calculadora



The screenshot displays a Jupyter Notebook interface with a menu bar (Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The notebook is running Python 3. The following code cells and their outputs are shown:

```
In [4]: from math import *
```

```
In [7]: pi
```

```
Out[7]: 3.141592653589793
```

```
In [8]: log10(100)
```

```
Out[8]: 2.0
```

```
In [9]: log(2)
```

```
Out[9]: 0.6931471805599453
```

```
In [10]: log10(2)
```

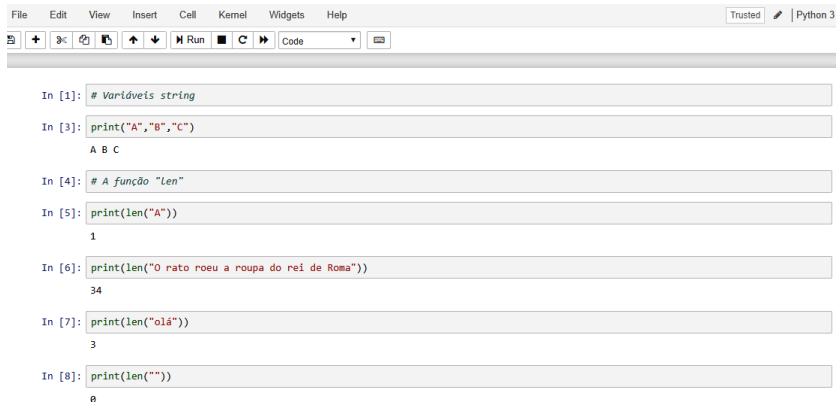
```
Out[10]: 0.3010299956639812
```

```
In [11]: sqrt(2)
```

```
Out[11]: 1.4142135623730951
```

Variáveis string

Variáveis do tipo string armazenam cadeias de caracteres como nomes e textos em geral.



The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The notebook contains eight code cells, each with an input prompt (In [n]:) and its output. The code cells are as follows:

```
In [1]: # Variáveis string
```

```
In [3]: print("A","B","C")
A B C
```

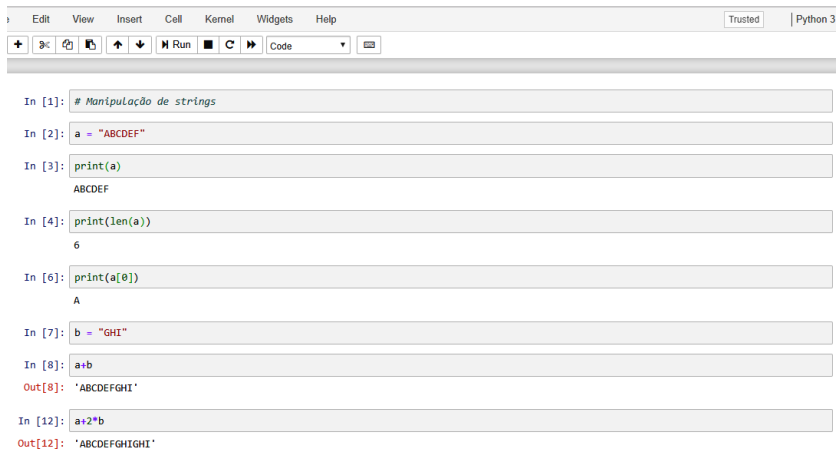
```
In [4]: # A função "len"
In [5]: print(len("A"))
1
```

```
In [6]: print(len("O rato roeu a roupa do rei de Roma"))
34
```

```
In [7]: print(len("olá"))
3
```

```
In [8]: print(len(""))
0
```

Manipulação de string



The image shows a Jupyter Notebook interface with a menu bar (Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for adding, undo, redo, and running code. The notebook contains several input and output cells demonstrating string operations in Python.

```
In [1]: # Manipulação de strings
```

```
In [2]: a = "ABCDEF"
```

```
In [3]: print(a)
```

```
ABCDEF
```

```
In [4]: print(len(a))
```

```
6
```

```
In [6]: print(a[0])
```

```
A
```

```
In [7]: b = "GHI"
```

```
In [8]: a+b
```

```
Out[8]: 'ABCDEFghi'
```

```
In [12]: a+2*b
```

```
Out[12]: 'ABCDEFghighi'
```


Condições

Em Python, a estrutura de decisão é o **if**.



The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and a dropdown menu set to 'Code'. The notebook contains four input cells:

```
In [1]: # Condição
```

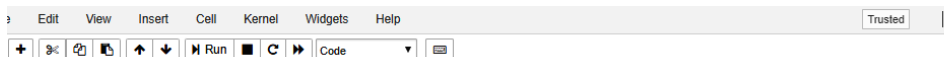
```
In [2]: a = int(input("Primeiro valor:"))
Primeiro valor:10
```

```
In [3]: b = int(input("Segundo valor:"))
Segundo valor:7
```

```
In [7]: if a > b:
        print("O primeiro número é o maior")
        if b > a:
            print("O segundo número é o maior")

O primeiro número é o maior
```

Condições



Condições

```
In [1]: x = int(input("Digite um número:"))
        if x > 0:
            print(x, "é um número positivo")
```

Digite um número:22
22 é um número positivo

```
In [2]: x = int(input("Digite um número:"))
        if x > 0:
            print(x, "é um número positivo.")
        else:
            print(x, "é um número negativo.")
```

Digite um número:-12
-12 é um número negativo.

```
In [3]: # Para o problema de múltiplos "ifs" aninhados podemos utilizar "else if = elif"
```

Repetições

Repetições são utilizadas para executar a mesma parte de um programa várias vezes, normalmente dependendo de uma condição.

Repetições

```
In [3]: x=1  
while x<=3:  
    print(x)  
    x = x+1
```

```
1  
2  
3
```

```
In [15]: contador = 0  
while (contador < 5):  
    print(contador)  
    contador = contador + 1  
else:  
    print('O loop while foi encerrado com sucesso!')
```

```
0  
1  
2  
3  
4  
O loop while foi encerrado com sucesso!
```

Funções

```
In [1]: # Funções
```

```
In [4]: def f(x):  
        return x*x
```

```
In [5]: f(10)
```

```
Out[5]: 100
```

```
In [7]: def subtract(a, b):  
        return a - b  
        subtract(1, 2)
```

```
Out[7]: -1
```

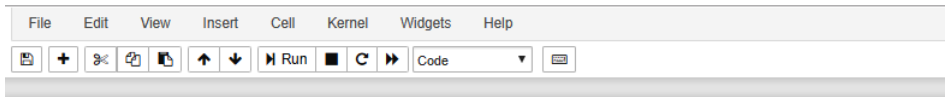
```
In [9]: import math
```

```
In [17]: def soma(a, b):  
         return a + b
```

```
In [21]: def area_triangulo(x,y):  
         return ((x*y)/2)  
         print (area_triangulo(8,5))
```

```
20.0
```

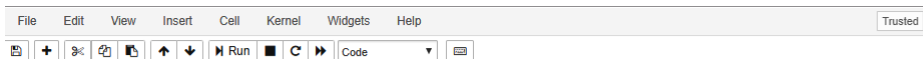
Dados externos



```
In [1]: # Dados no formato csv
```

```
In [2]: import csv
with open('iris.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    cabecalho = True
    for row in csv_reader:
        if cabecalho:
            print(f'Nomes das colunas: {", ".join(row)}')
            cabecalho = False
        else:
            print(f'{"", ".join(row)}')
```

Dados externos



```
In [1]: # Dados no formato csv
```

```
In [10]: import pandas as pd

dados = pd.read_csv("iris.csv", delimiter = ";", na_values=[" "])
```

```
In [8]: dados
```

```
Out[8]:
```

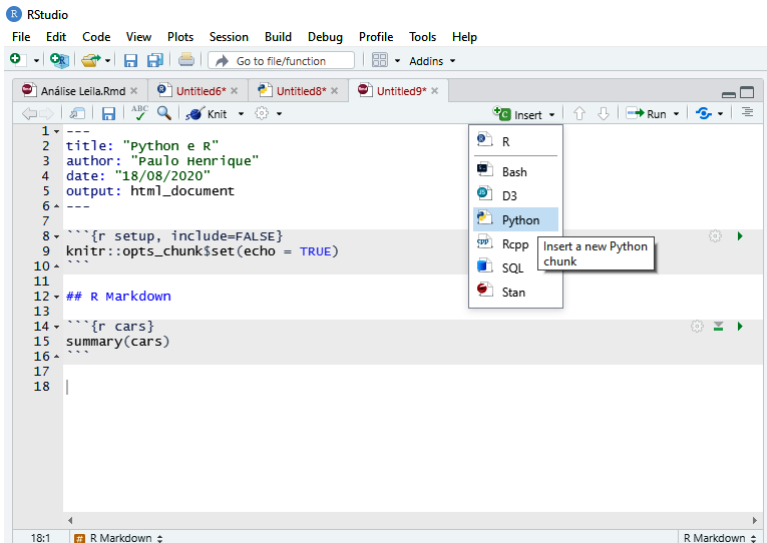
	sepal_length	sepal_width	petal_length	petal_width	iris
0	5,1	3,5	1,4	0,2	Iris-setosa
1	4,9	3	1,4	0,2	Iris-setosa
2	4,7	3,2	1,3	0,2	Iris-setosa
3	4,6	3,1	1,5	0,2	Iris-setosa
4	5	3,6	1,4	0,2	Iris-setosa
5	5,4	3,9	1,7	0,4	Iris-setosa
6	4,6	3,4	1,4	0,3	Iris-setosa
7	5	3,4	1,5	0,2	Iris-setosa
8	4,4	2,9	1,4	0,2	Iris-setosa
9	4,9	3,1	1,5	0,1	Iris-setosa
10	5,4	3,7	1,5	0,2	Iris-setosa

Python no RStudio



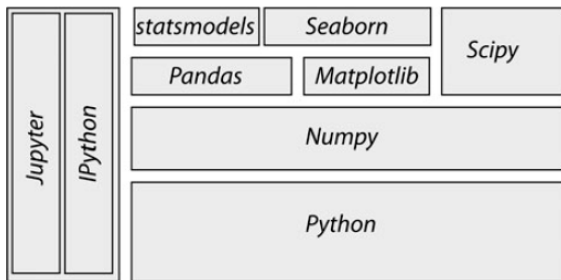
Neste caso, precisamos instalar o pacote "reticulate"
(`install.packages('reticulate')`).

Python no RStudio



Pacotes Python

Existem uma infinidade de pacotes disponíveis no Python. Para a Estatística os principais pacotes são: Pandas, Numpy, Scipy, Seaborn, statsmodels, Matplotlib, dentre outros.



Introdução à NumPy

NumPy é um pacote da linguagem Python que foi desenvolvido para computação científica.

Introdução à NumPy

NumPy é um pacote da linguagem Python que foi desenvolvido para computação científica.

Esse pacote é muito útil para análises de dados, no qual possui várias funções que permitem manipular e descrever dados.



Introdução à NumPy

O principal tipo de dados disponibilizado pela biblioteca NumPy é conhecido como *numpy.array*. Um array pode ser instanciado por meio da chamada `numpy.array(lista)`, na qual *lista* é um objeto do tipo *list* contendo apenas valores numéricos.

```
In [13]: import numpy  
        lista = [1, 2, 3, 4, 5]  
        x = numpy.array(lista)  
        print(x)
```

```
[1 2 3 4 5]
```

```
In [14]: y = numpy.array([6,7,8,9,10])  
        print(y)
```

```
[ 6  7  8  9 10]
```

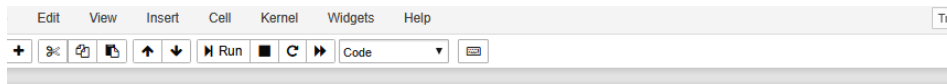
```
In [15]: soma = x + y  
        soma
```

```
Out[15]: array([ 7,  9, 11, 13, 15])
```

```
In [16]: produto = x*y  
        produto
```

```
Out[16]: array([ 6, 14, 24, 36, 50])
```

Introdução à NumPy



```
In [21]: # Array
```

```
In [40]: x = numpy.array([1,2,3,4,5,6,7,8,9,10])
```

```
In [70]: media = numpy.sum(x)/len(x)
media
```

```
Out[70]: 5.5
```

```
In [72]: var = sum((x-media)**2)/(len(x)-1)
var
```

```
Out[72]: 9.166666666666666
```

```
In [69]: import math
desvio = math.sqrt(var)
desvio
```

```
Out[69]: 3.0276503540974917
```

Introdução à NumPy

Vamos calcular agora a mediana e os quartis (percentis).

```
e  Edit  View  Insert  Cell  Kernel  Widgets  Help
+  <  >  ↺  ↻  ⬆  ⬇  ⏪  Run  ⏩  Code  ⌵  ⌂

In [73]: # Array

In [74]: x = numpy.array([1,2,3,4,5,6,7,8,9,10])
         print(x)

         [ 1  2  3  4  5  6  7  8  9 10]

In [75]: mediana = numpy.median(x) # mediana
         print(mediana)

         5.5

In [76]: percentil_15 = numpy.percentile(x, 15)
         print(percentil_15)

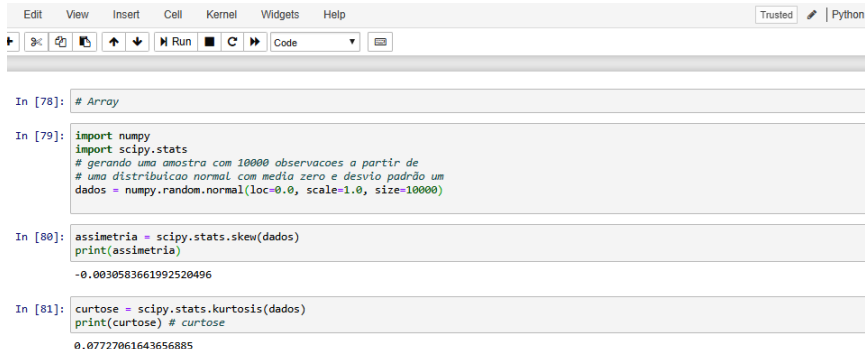
         2.3499999999999996

In [77]: quartil1 = numpy.percentile(x,25)
         quartil1

Out[77]: 3.25
```

Introdução à NumPy

Em Python, a assimetria dos dados contidos em um array pode ser calculada por meio da biblioteca SciPy da seguinte maneira:



The screenshot shows a Jupyter Notebook interface with a menu bar (Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for undo, redo, run, and other actions. The notebook contains four input cells:

```
In [78]: # Array
```

```
In [79]: import numpy
import scipy.stats
# gerando uma amostra com 10000 observacoes a partir de
# uma distribuicao normal com media zero e desvio padrão um
dados = numpy.random.normal(loc=0.0, scale=1.0, size=10000)
```

```
In [80]: assimetria = scipy.stats.skew(dados)
print(assimetria)
-0.0030583661992520496
```

```
In [81]: curtose = scipy.stats.kurtosis(dados)
print(curtose) # curtose
0.07727061643656885
```


Modo *statistics*

```
In [82]: # Modo statistics
```

```
In [95]: x = numpy.array([1,1,2,3,4,5,6,7,8,9,10])
```

```
In [102]: from statistics import mean, median, mode, stdev, variance
```

```
In [103]: mean(x)
```

```
Out[103]: 5
```

```
In [104]: median(x)
```

```
Out[104]: 5
```

```
In [105]: mode(x)
```

```
Out[105]: 1
```

```
In [106]: stdev(x)
```

```
Out[106]: 3.1622776601683795
```

```
In [107]: variance(x)
```

```
Out[107]: 10
```

Matrices

```
In [121]: # Matrices
```

```
In [123]: import numpy as np
A = np.array([[1,2,3],
              [4,5,6]])
```

```
In [124]: print(A)

[[1 2 3]
 [4 5 6]]
```

```
In [125]: B = np.random.random((4,4))
B
```

```
Out[125]: array([[0.68266165, 0.10319303, 0.6548974 , 0.97299364],
                 [0.19370171, 0.77693284, 0.76492865, 0.26065115],
                 [0.69814316, 0.19969897, 0.9186436 , 0.94330884],
                 [0.6076414 , 0.0892387 , 0.87493848, 0.63342834]])
```

Matrizes

```
File Edit View Insert Cell Kernel Widgets Help Trusted Python
[Icons] [Run] [Code]

In [121]: # Matrizes

In [131]: X = np.random.random((3,3))
          Y = np.random.random((3,3))

In [134]: produto = print(X*Y)
          [[0.09860336 0.4407476 0.19096955]
           [0.01442446 0.19891189 0.1273873 ]
           [0.10725645 0.63189118 0.47954408]]

In [135]: print(X+2*Y)
          [[1.20269898 1.96172344 2.15776614]
           [0.89350829 1.69272692 1.85098919]
           [1.09969059 2.3001166 2.36903366]]

In [ ]: |
```

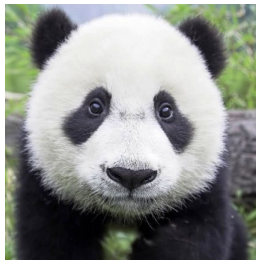
Introdução ao Pandas

A biblioteca Pandas, assim como NumPy, está entre as bibliotecas mais utilizadas no Python.

Introdução ao Pandas

A biblioteca Pandas, assim como NumPy, está entre as bibliotecas mais utilizadas no Python.

Sua utilidade no tratamento de dados é enorme, pois com ela você pode lidar tranquilamente com séries e tabelas, organizando, filtrando e fazendo várias manipulações com seus dados. Por este motivo, muitas vezes será interessante você utilizar o formato *data.frame* da biblioteca.



Introdução ao Pandas - Data frame

```
In [136]: # Data frame
```

```
In [147]: import pandas as pd
```

```
df_data = {'pais': ['Brasil', 'Argentina', 'Argentina', 'Brasil', 'Chile', 'Chile'],  
           'ano': [2005, 2006, 2005, 2006, 2007, 2008],  
           'populacao': [170.1, 30.5, 32.2, 172.6, 40.8, 42.0]}  
df = pd.DataFrame(df_data)  
print(df)
```

	pais	ano	populacao
0	Brasil	2005	170.1
1	Argentina	2006	30.5
2	Argentina	2005	32.2
3	Brasil	2006	172.6
4	Chile	2007	40.8
5	Chile	2008	42.0

```
In [ ]: print(df.pais)
```

```
In [148]: print(df.describe()) # descreve o conjunto de dados
```

	ano	populacao
count	6.000000	6.000000
mean	2006.166667	81.366667
std	1.169045	69.853122
min	2005.000000	30.500000
25%	2005.250000	34.350000
50%	2006.000000	41.400000
75%	2006.750000	138.075000
max	2008.000000	172.600000

Introdução ao Pandas - Dados categóricos

```
file Edit View Insert Cell Kernel Widgets Help Trusted Python 3
```

```
In [156]: df = pd.DataFrame({"id": [1, 2, 3, 4, 5, 6],
                             "raw_grade": ['a', 'b', 'b', 'a', 'a', 'e']})

In [157]: print(df)

   id raw_grade
0   1         a
1   2         b
2   3         b
3   4         a
4   5         a
5   6         e

In [158]: df["grade"] = df["raw_grade"].astype("category")

In [159]: df["grade"]

Out[159]: 0    a
          1    b
          2    b
          3    a
          4    a
          5    e
          Name: grade, dtype: category
          Categories (3, object): [a, b, e]

In [161]: df["grade"].cat.categories = ["ótimo", "bom", "ruim"]
          df["grade"]

Out[161]: 0    ótimo
          1    bom
          2    bom
          3    ótimo
          4    ótimo
          5    ruim
          Name: grade, dtype: category
          Categories (3, object): [ótimo, bom, ruim]
```

Visualização de dados

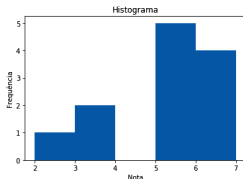
O pacote NumPy fornece uma função para calcular o histograma, que pode ser vista abaixo. Nessa função, bins corresponde ao número de barras verticais. Quando o valor de bins é definido como 'auto', o número de barras é definido automaticamente.

```
In [1]: # Visualização de dados

In [2]: import numpy
        from matplotlib import pyplot

In [3]: notas = numpy.array([2, 5, 7, 3, 5, 6, 5, 6, 6, 5, 5, 3]) # notas avaliação

In [5]: pyplot.hist(notas, bins='auto')
        pyplot.title('Histograma')
        pyplot.ylabel('Frequência')
        pyplot.xlabel('Nota')
        pyplot.show()
```

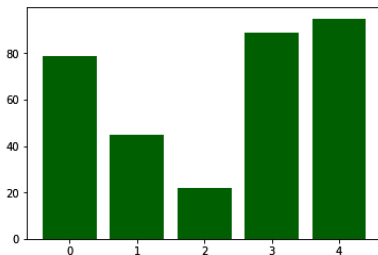


Visualização de dados

```
In [ ]: # Gráfico Barplot
```

```
In [39]: import numpy as np
import matplotlib.pyplot as plt
marks=[79,45,22,89,95]
bars=('Roll 1','Roll 2','Roll 3','Roll 4','Roll 5')
y=np.arange(len(bars))
plt.bar(y,marks,color='g')
```

```
Out[39]: <BarContainer object of 5 artists>
```



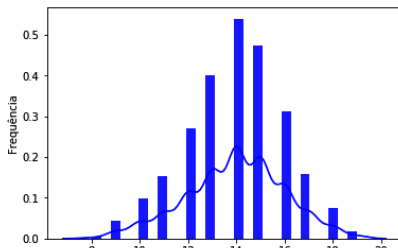
Algumas distribuições

```
In [9]: # Distribuições - Binomial
```

```
In [19]: import scipy.stats # distribuições em Estatística (probabilidade)
import numpy as np
import seaborn
```

```
In [24]: import seaborn
from scipy.stats import binom
data=binom.rvs(n=20,p=0.7,loc=0,size=1000)
ax=seaborn.distplot(data,
                    kde=True,color='blue',
                    hist_kws={"linewidth": 22,'alpha':0.77})
ax.set(xlabel='Binomial',ylabel='Frequência')
```

```
Out[24]: [Text(0, 0.5, 'Frequência'), Text(0.5, 0, 'Binomial')]
```



Principais bibliotecas para análise de dados

- NumPy;

Principais bibliotecas para análise de dados

- NumPy;
- SciPy - O pacote contém ferramentas que ajudam a resolver álgebra linear, teoria da probabilidade, cálculo integral e muitas outras tarefas;

Principais bibliotecas para análise de dados

- NumPy;
- SciPy - O pacote contém ferramentas que ajudam a resolver álgebra linear, teoria da probabilidade, cálculo integral e muitas outras tarefas;
- Pandas;

Principais bibliotecas para análise de dados

- NumPy;
- SciPy - O pacote contém ferramentas que ajudam a resolver álgebra linear, teoria da probabilidade, cálculo integral e muitas outras tarefas;
- Pandas;
- StatsModels - estimação de modelos estatísticos, a realização de testes estatísticos;

Principais bibliotecas para análise de dados

- NumPy;
- SciPy - O pacote contém ferramentas que ajudam a resolver álgebra linear, teoria da probabilidade, cálculo integral e muitas outras tarefas;
- Pandas;
- StatsModels - estimação de modelos estatísticos, a realização de testes estatísticos;
- Matplotlib e Plotly;

Principais bibliotecas para análise de dados

- NumPy;
- SciPy - O pacote contém ferramentas que ajudam a resolver álgebra linear, teoria da probabilidade, cálculo integral e muitas outras tarefas;
- Pandas;
- StatsModels - estimação de modelos estatísticos, a realização de testes estatísticos;
- Matplotlib e Plotly;
- Scikit-learn - aprendizado de máquina e mineração de dados, como clustering, regressão, classificação, redução de dimensionalidade e seleção de modelo;

Principais bibliotecas para análise de dados

- Keras - é uma biblioteca de alto nível para trabalhar com redes neurais, rodando sobre o TensorFlow, entre outros;

Principais bibliotecas para análise de dados

- Keras - é uma biblioteca de alto nível para trabalhar com redes neurais, rodando sobre o TensorFlow, entre outros;
- Scrapy - raspagem de dados da web (web scraping), bem como a identificação da anatomia de um web site em busca de dados (web crawling);

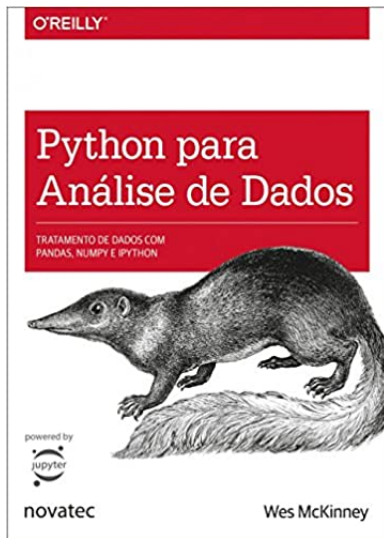
Principais bibliotecas para análise de dados

- Keras - é uma biblioteca de alto nível para trabalhar com redes neurais, rodando sobre o TensorFlow, entre outros;
- Scrapy - raspagem de dados da web (web scraping), bem como a identificação da anatomia de um web site em busca de dados (web crawling);
- PyCaret - biblioteca de aprendizado de máquina de código aberto em Python que permite ir desde a preparação de seus dados até a implantação de seu modelo.

Por onde começar a aprender Python?



Por onde começar a aprender Python?



Por onde começar a aprender Python?



Por onde começar a aprender Python?



Referências

- 1 BEAZLEY, D. ; JONES, B.K. Python Cookbook. Ed. Novatec, 2013.
- 2 HASLWANTER, T. An Introduction to Statistics with Python: With Applications in the Life Sciences. Springer, 2016.
- 3 LABAKI, J. Introdução a python – Módulo A. Ilha Solteira – SP: Universidade Estadual Paulista (UNESP), 2011. (Apostila).
- 4 PYTHON. Python Software Foundation. Disponível em: <<https://www.python.org/>>. Acesso em maio de 2019.