PROVAS DE TI
TUDO PARA VOCÊ PASSAR

Bibliotecas Python

Prof. Rodrigo Macedo

Ferramentas Python

Escopo do Curso

- Pandas
- Scikit-Learn
- Matplotlib
- Jupyter Notebook
- NLTK
- Questões de Concursos

Pandas

- Pandas é um biblioteca para manipulação e análise de dados, escrita em Python.
- Essa é a biblioteca perfeita para iniciar suas análises exploratórias de dados, pois ela nos permite ler, manipular, agregar e plotar os dados em poucos passos.

Pandas

- Pandas é uma biblioteca para uso em Python, open-source e de uso gratuito (sob uma licença BSD), que fornece ferramentas para análise e manipulação de dados.
- De acordo com o próprio criador dessa biblioteca, Wes McKinney, o nome Pandas é derivado de panel data (dados em painel), um termo de econometria para conjuntos de dados estruturados.
- O surgimento da biblioteca, no início de 2008, começou devido a insatisfação de McKinney de obter uma ferramenta de processamento de dados de alto desempenho, com recursos flexíveis de manipulação de planilhas e de banco de dados relacionais.

Pandas

O pandas permite trabalhar com diferentes tipos de dados, por exemplo:

- Dados tabulares, como uma planilha Excel ou uma tabela SQL;
- Dados ordenados de modo temporal ou não;
- Matrizes;
- Qualquer outro conjunto de dados, que não necessariamente precisem estar rotulados;

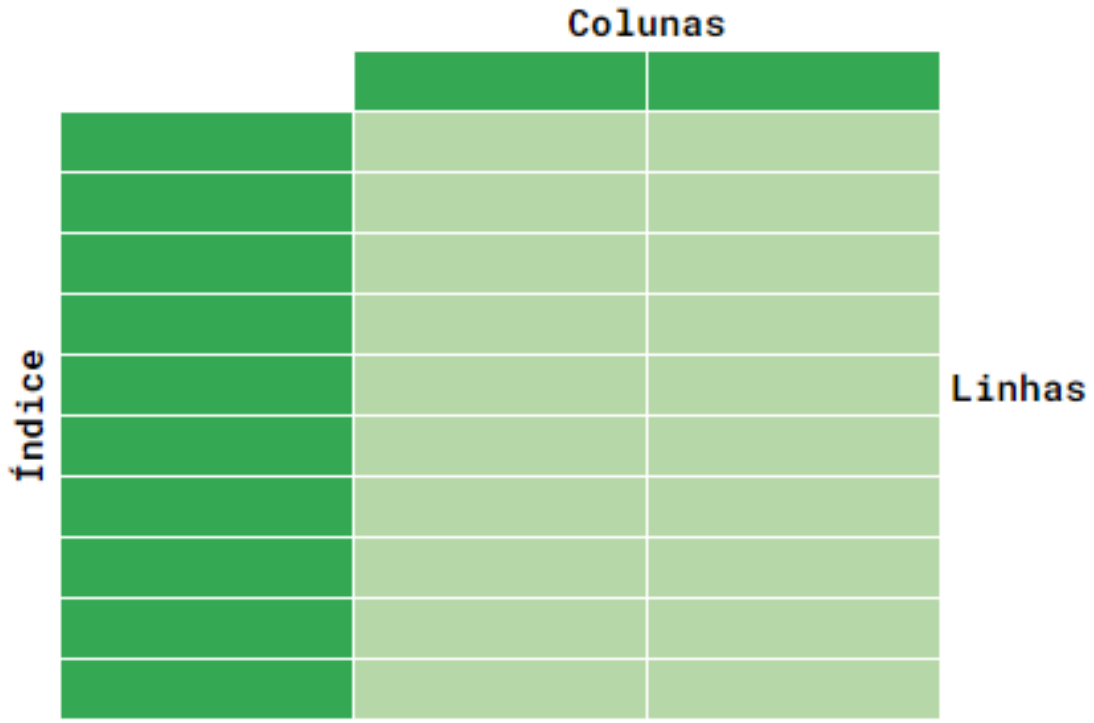
A mágica de ler, manipular, agregar e exibir os dados com poucos comandos explica porque a biblioteca tem se tornado tão popular. Aliás, tudo isso é possível devido às estruturas primárias do Pandas, as famosas Series e DataFrames.

Pandas - Estrutura de Dados

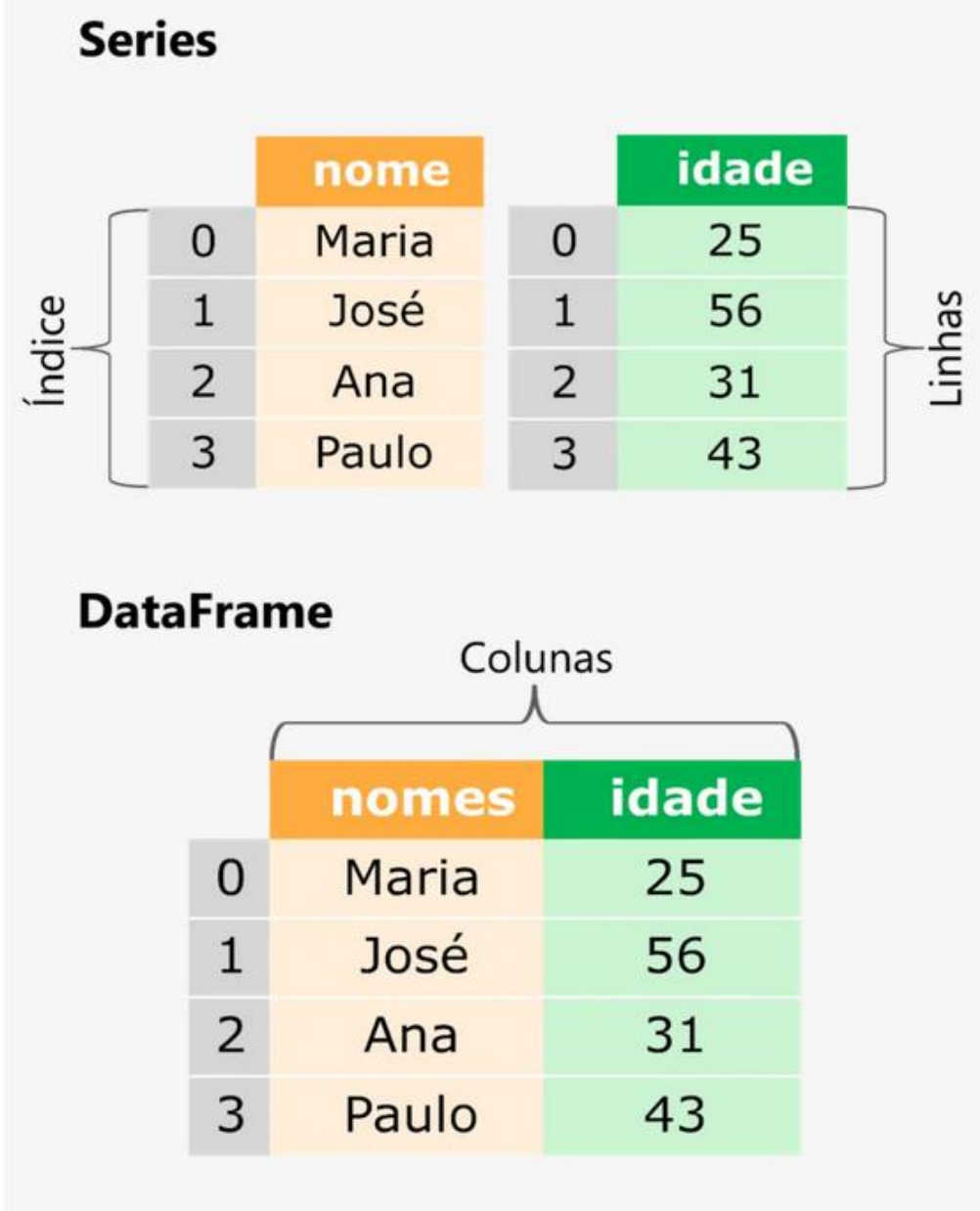
- Os dois principais objetos da biblioteca Pandas são as **Series** e os **DataFrames**. Uma Serie é uma matriz unidimensional que contém uma sequência de valores que apresentam uma indexação (que podem ser numéricos inteiros ou rótulos), muito parecida com uma única coluna no Excel.
- Já o DataFrame é uma estrutura de dados tabular, semelhante a planilha de dados do Excel, em que tanto as linhas quanto as colunas apresentam rótulos.
- A partir dos objetos principais a biblioteca Pandas disponibiliza um conjunto de funcionalidades sofisticadas de indexação, que permite reformatar, manipular, agregar ou selecionar subconjuntos específicos dos dados que estamos trabalhando.

Pandas - DataFrame

- Em uma simples definição, **DataFrame** é como se fosse uma planilha de Excel ou uma tabela de banco de dados.
- É composto por colunas, linhas e índice. Quando nós lemos algum arquivo de dados, ele se torna um DataFrame para o Pandas.



Pandas - Dataframe x Series



Pandas - Vantagens

- **A facilidade de aprender e de utilizar a biblioteca:** É muito mais fácil trabalhar com um objeto do Pandas, do que reunir informações por meio de interações de listas e dicionários Python. Para colaborar ainda mais, no seu site a biblioteca ainda disponibiliza uma lista de comandos para que desenvolvedores que utilizam outras linguagens, como R, SQL, SAS, entre outros, possam encontrar os comandos equivalentes, com a mesma funcionalidade no Pandas;

Querying, filtering, sampling

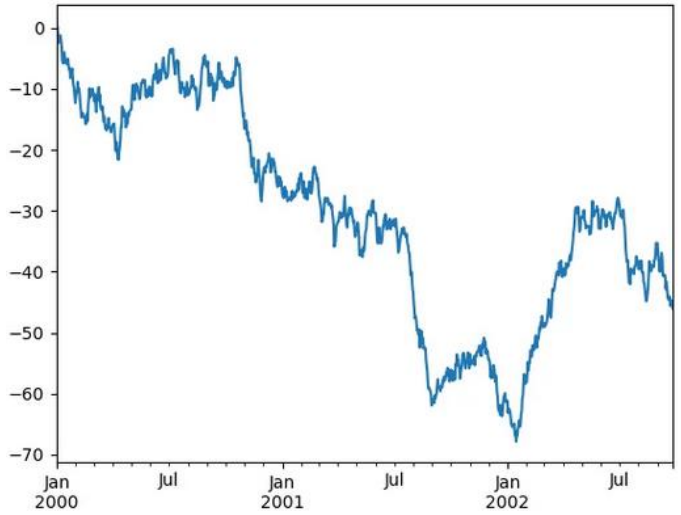
R	pandas
`dim(df)`	`df.shape`
`head(df)`	`df.head()`
`slice(df, 1:10)`	`df.iloc[:9]`
`filter(df, col1 == 1, col2 == 1)`	`df.query('col1 == 1 & col2 == 1')`
`df[df$col1 == 1 & df$col2 == 1,]`	`df[(df.col1 == 1) & (df.col2 == 1)]`

Pandas - Vantagens

- **Comunidade crescente e muito ativa:** Na última pesquisa do Stack Overflow ‘2020 Developer Survey’, o Pandas aparece como quarto colocado na lista de ferramentas e pacotes que os desenvolvedores profissionais utilizam.
- **Suporte para alinhamento automático ou explícito dos dados:** Os objetos no Pandas podem ser explicitamente alinhados com eixos nomeados, que o usuário pode ou não especificar. Esse alinhamento evita erros comuns de dados desalinhados e possibilita trabalhar com dados que apresentem indexações diferentes (que podem ser provenientes de origens distintas);
- **Tratamento flexível e simplificado de dados ausentes:** Possibilita de forma simplificada a substituição ou exclusão de dados ausentes que o conjunto de dados que estamos trabalhando pode apresentar;

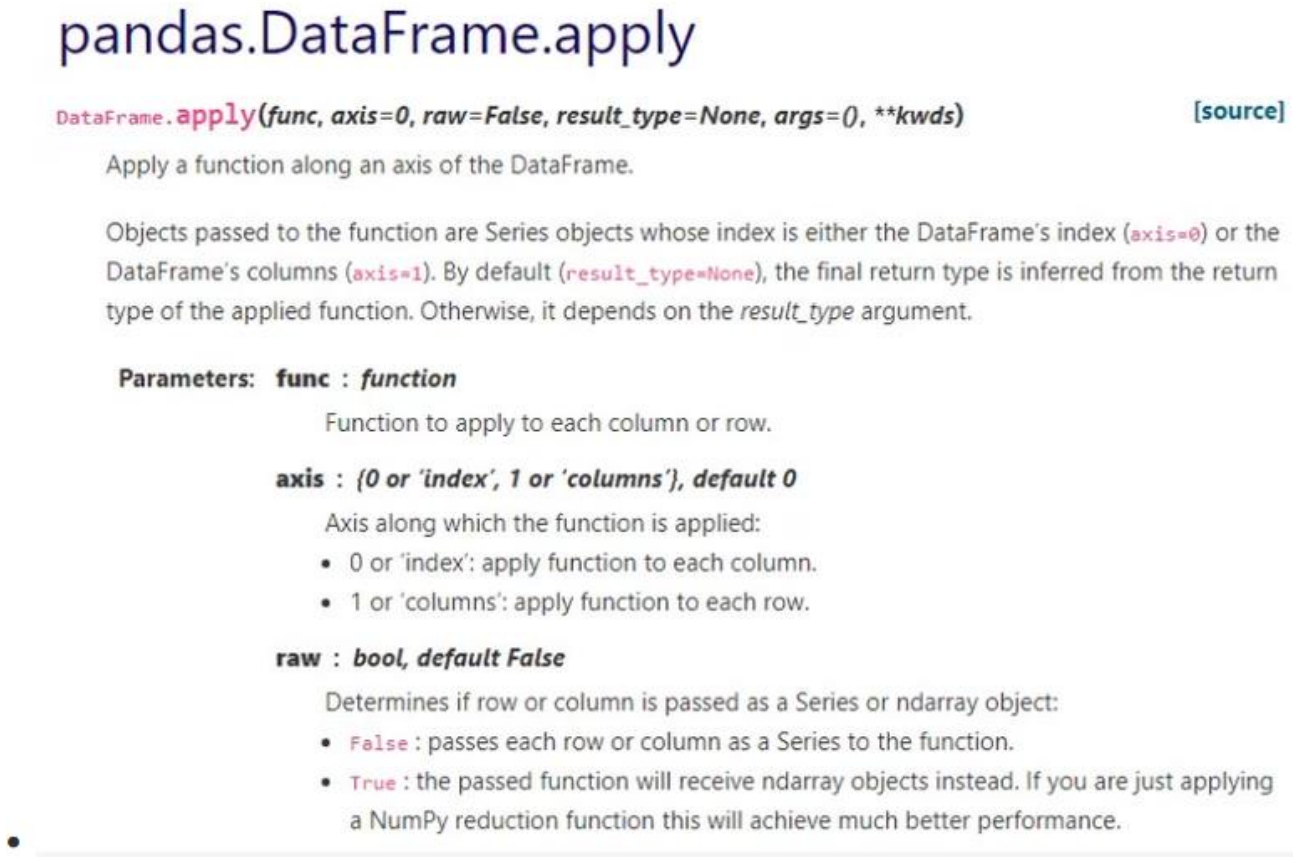
Pandas - Vantagens

- **Uso de operações:** A biblioteca permite a utilização de operações aritméticas para agregar ou transformar os dados que se encontram em suas estruturas principais (Series e DataFrames);
- **Combinações e operações relacionais:** O Pandas disponibiliza métodos para facilitar a combinação de conjuntos de dados, além de permitir selecionar subconjuntos dos nossos dados originais, com base em determinados filtros;
- **Visualização de dados:** Como o Pandas engloba algumas funcionalidades da biblioteca matplotlib, ela permite que os usuários criem visualizações simplificadas dos dados.



Pandas - Vantagens

- **Documentação:** A documentação fornecida pela biblioteca Pandas fornece uma rica explicação das estruturas e métodos possíveis de serem aplicadas, descrevendo as funcionalidades e parâmetros existentes. Também, a biblioteca apresenta exemplos de utilização, além de métodos que podem estar relacionados ao método pesquisado.



pandas.DataFrame.apply

`DataFrame.apply(func, axis=0, raw=False, result_type=None, args=(), **kwds)` [source]

Apply a function along an axis of the DataFrame.

Objects passed to the function are Series objects whose index is either the DataFrame's index (`axis=0`) or the DataFrame's columns (`axis=1`). By default (`result_type=None`), the final return type is inferred from the return type of the applied function. Otherwise, it depends on the *result_type* argument.

Parameters: **func** : *function*

Function to apply to each column or row.

axis : *{0 or 'index', 1 or 'columns'}, default 0*

Axis along which the function is applied:

- 0 or 'index': apply function to each column.
- 1 or 'columns': apply function to each row.

raw : *bool, default False*

Determines if row or column is passed as a Series or ndarray object:

- `False` : passes each row or column as a Series to the function.
- `True` : the passed function will receive ndarray objects instead. If you are just applying a NumPy reduction function this will achieve much better performance.

Pandas - Instalação e Configuração

- A biblioteca Pandas pode ser instalada via gerenciador de pacotes Pip, com o comando: **pip install pandas**.
- Para saber se o pacote foi instalado no ambiente, pode ser executado o comando: **pip freeze**.
- Obs: Caso esteja usando o Anaconda, o Pandas, bem como outras bibliotecas já vem instalados.
- Uma vez que foi instalado o Pandas, o comando para importá-lo a fim de utilizá-lo na prática é o **import pandas**. Geralmente a importação pode ser dado um nome de apelido à biblioteca, exemplo: **import pandas as pd.**

Pandas - Funções

PYTHON FOR DATA SCIENCE CHEAT SHEET

Python Pandas

What is Pandas?

It is a library that provides easy to use data structure and data analysis tool for Python Programming Language.

Import Convention

import pandas as pd – Import pasdas

Pandas Data Structure

- Series:

```
s = pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])
```

- Data Frame:

```
data_mobile = {'Mobile': ['iPhone', 'Samsung', 'Redmi'], 'Color': ['Red', 'White', 'Black'], 'Price': [High, Medium,Low]}
df = pd.DataFrame(data_mobile, columns=['Mobile', 'Color', 'Price'])
```

Importing Data

- pd.read_csv(filename)
- pd.read_table(filename)
- pd.read_excel(filename)
- pd.read_sql(query, connection_object)
- pd.read_json(json_string)

Exporting Data

- df.to_csv(filename)
- df.to_excel(filename)
- df.to_sql(table_name, connection_object)
- df.to_json(filename)

Create Test/Fake Data

- pd.DataFrame(np.random.rand(4,3)) - 3 columns and 4 rows of random floats
- pd.Series(new_series) - Creates a series from an iterable new_series

Plotting

- Histogram: df.plot.hist()
- Scatter Plot: df.plot.scatter(x='column1',y='column2')

Operations

View DataFrame Contents:

- df.head(n) - look at first n rows of the DataFrame.
- df.tail(n) – look at last n rows of the DataFrame.
- df.shape() - Gives the number of rows and columns.
- df.info() - Information of Index, Datatype and Memory.
- df.describe() -Summary statistics for numerical columns.

Selection:

- iloc
  - df.iloc[0] - Select first row of data frame
  - df.iloc[1] - Select second row of data frame
  - df.iloc[-1] - Select last row of data frame
  - df.iloc[:,0] - Select first column of data frame
  - df.iloc[:,1] - Select second column of data frame
- loc
  - df.loc[0], [column labels])- Select single value by row position & column labels
  - df.loc['row1':'row3', 'column1':'column3']- Select and slicing on labels

Sort:

- df.sort_index() - Sorts by labels along an axis
- df.sort_values(by='Column label') - Sorts by the values along an axis
- df.sort_values(column1) - Sorts values by column1 in ascending order
- df.sort_values(column2,ascending=False) - Sorts values by column2 in descending order

Operations - GroupBy

from one column

- df.groupby([column1,column2]) - Returns a groupby object values from multiple columns
- df.groupby(column1)[column2].mean() - Returns the mean of the values in column2, grouped by the values in column1
- df.groupby(column1)[column2].median() - Returns the mean of the values in column2, grouped by the values in column1

Functions

Mean:

- df.mean() - mean of all columns

Median

- df.median() - median of each column

Standard Deviation

- df.std() - standard deviation of each column

Max

- df.max() - highest value in each column

Min

- df.min() - lowest value in each column

Count

- df.count() - number of non-null values in each DataFrame column

Describe

- df.describe() - Summary statistics for numerical columns

FURTHERMORE:
Python for Data Science Certification Training Course

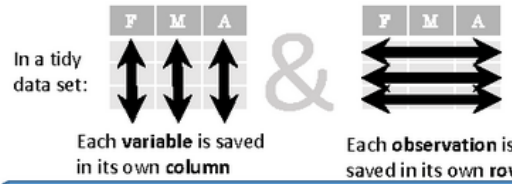
IntelliPaat

Pandas - Operações

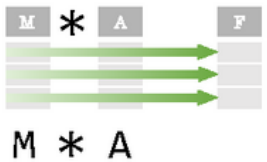
Data Wrangling with pandas Cheat Sheet

http://pandas.pydata.org

Tidy Data – A foundation for wrangling in pandas



Tidy data complements pandas's vectorized operations. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.



Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
       index = [1, 2, 3])
```
Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```
Specify values for each row.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

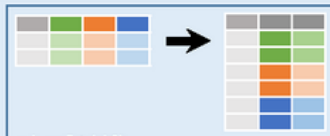
```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
          names=['n','v'])))
```
Create DataFrame with a MultiIndex

Method Chaining

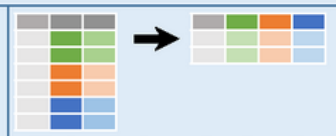
Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
    )
```

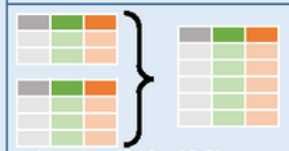
Reshaping Data – Change the layout of a data set



`pd.melt(df)`
Gather columns into rows.



`df.pivot(columns='var', values='val')`
Spread rows into columns.



`pd.concat([df1,df2])`
Append rows of DataFrames

`pd.concat([df1,df2], axis=1)`
Append columns of DataFrames

`df.sort_values('mpg')`
Order rows by values of a column (low to high).

`df.sort_values('mpg',ascending=False)`
Order rows by values of a column (high to low).

`df.rename(columns = {'y':'year'})`
Rename the columns of a DataFrame

`df.sort_index()`
Sort the index of a DataFrame

`df.reset_index()`
Reset index of DataFrame to row numbers, moving index to columns.

`df.drop(columns=['Length','Height'])`
Drop columns from DataFrame

Subset Observations (Rows)

`df[df.Length > 7]`
Extract rows that meet logical criteria.

`df.drop_duplicates()`
Remove duplicate rows (only considers columns).

`df.head(n)`
Select first n rows.

`df.tail(n)`
Select last n rows.

`df.sample(frac=0.5)`
Randomly select fraction of rows.

`df.sample(n=10)`
Randomly select n rows.

`df.iloc[10:20]`
Select rows by position.

`df.nlargest(n, 'value')`
Select and order top n entries.

`df.nsmallest(n, 'value')`
Select and order bottom n entries.

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

Subset Variables (Columns)

`df[['width','length','species']]`
Select multiple columns with specific names.

`df['width']` *or* `df.width`
Select single column with specific name.

`df.filter(regex='regex')`
Select columns whose name matches regular expression *regex*.

regex (Regular Expressions) Examples

'\.'	Matches strings containing a period '.'
'Length$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
'''^(?!Species$).*'	Matches strings except the string 'Species'

`df.loc[:,'x2':'x4']`
Select all columns between x2 and x4 (inclusive).

`df.iloc[:,[1,2,5]]`
Select columns in positions 1, 2 and 5 (first column is 0).

`df.loc[df['a'] > 10, ['a','c']]`
Select rows meeting logical condition, and only the specific columns .

Pandas - Hands On

- Criação de um Dataframe:

```
df = pd.DataFrame({'Aluno' : ["Wilfred", "Abbie", "Harry", "Julia",
"Carrie"],
                   'Faltas' : [3,4,2,1,4],
                   'Prova' : [2,7,5,10,6],
                   'Seminário': [8.5,7.5,9.0,7.5,8.0]})
df
```

	Aluno	Faltas	Prova	Seminário
0	Wilfred	3	2	8.5
1	Abbie	4	7	7.5
2	Harry	2	5	9.0
3	Julia	1	10	7.5
4	Carrie	4	6	8.0

Tipos de Dados

```
df.dtypes


Aluno          object
Faltas          int64
Prova           int64
Seminário     float64
dtype: object
```

Pandas - Hands On

- Acessando valores por coluna:

```
df["Seminário"]


0    8.5
1    7.5
2    9.0
3    7.5
4    8.0
Name: Seminário, dtype: float64
```

Consulta pelo index

```
df.loc[3]


Aluno         Julia
Faltas            1
Prova            10
Seminário       7.5
Name: 3, dtype: object
```

```
df.sort_values(by="Seminário")
```

	Aluno	Faltas	Prova	Seminário
1	Abbie	4	7	7.5
3	Julia	1	10	7.5
4	Carrie	4	6	8.0
0	Wilfred	3	2	8.5
2	Harry	2	5	9.0

Filtragem e Indexação de Dados

```
df[df["Seminário"] > 8.0]
```

	Aluno	Faltas	Prova	Seminário
0	Wilfred	3	2	8.5
2	Harry	2	5	9.0

Pandas - Hands On

- Apresentando os primeiros e últimos registros:

```
df.head()
```

	condominio	quartos	suites	vagas	area	bairro	preco	pm2
0	350	1	0.0	1.0	21	Botafogo	340000	16190.48
1	800	1	0.0	1.0	64	Botafogo	770000	12031.25
2	674	1	0.0	1.0	61	Botafogo	600000	9836.07
3	700	1	1.0	1.0	70	Botafogo	700000	10000.00
4	440	1	0.0	1.0	44	Botafogo	515000	11704.55

Contagem de valores

```
df["bairro"].value_counts()
```

```
Copacabana    346
Tijuca        341
Botafogo      307
Ipanema       281
Leblon        280
Grajaú        237
Gávea         205
Name: bairro, dtype: int64
```

```
df.tail()
```

Valores únicos em uma coluna

```
df["bairro"].unique()
```

```
array(['Botafogo', 'Copacabana', 'Gávea', 'Grajaú', 'Ipanema',
'Leblon',
       'Tijuca'], dtype=object)
```

	condominio	quartos	suites	vagas	area	bairro	preco	pm2
1992	1080	3	1.0	1.0	80	Tijuca	680000	8500.00
1993	750	3	0.0	1.0	82	Tijuca	650000	7926.83
1994	700	3	1.0	1.0	100	Tijuca	629900	6299.00
1995	1850	3	1.0	2.0	166	Tijuca	1600000	9638.55
1996	800	3	1.0	1.0	107	Tijuca	540000	5046.73

Matplotlib

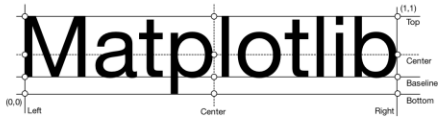
- O pacote matplotlib é voltado para criação de gráficos de alta qualidade e é muito usado hoje no meio científico, além de ser software livre.
- Matplot é uma biblioteca que facilita a visualização de dados em gráficos. O objetivo da biblioteca é facilitar a criação de gráficos. Permitindo que o programador gaste o seu tempo na busca e analise de dados.
- Com a biblioteca matplotlib é possível criar gráfico com poucas linhas de código.

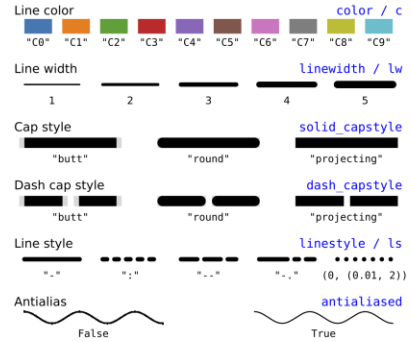
Matplotlib - Instalação e Configuração

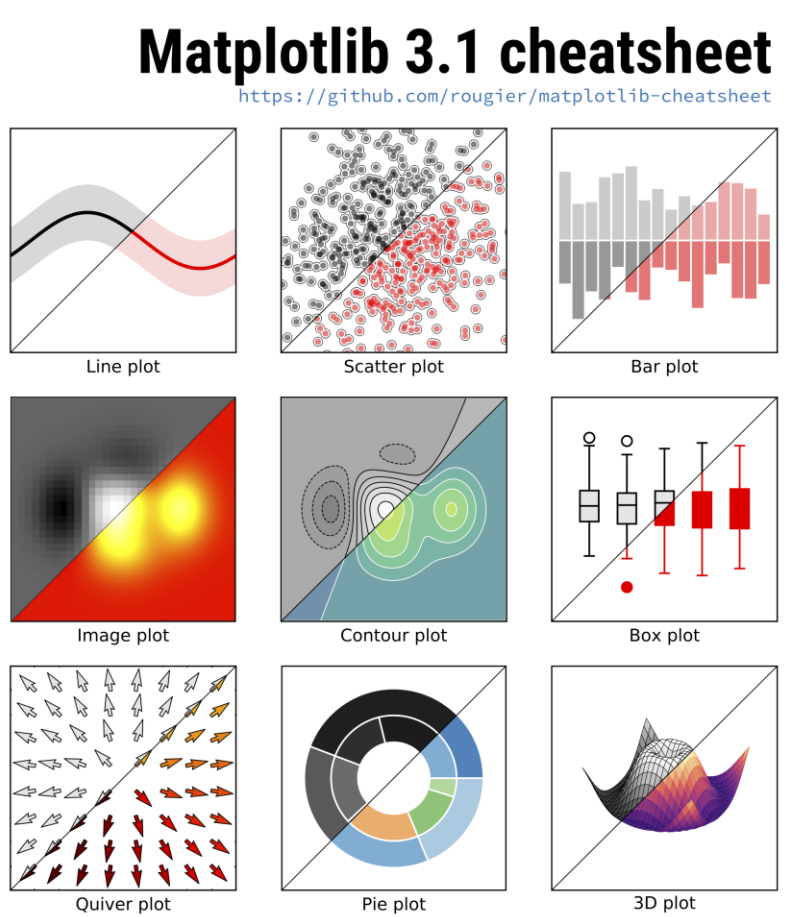
- A biblioteca Matplotlib pode ser instalada via gerenciador de pacotes Pip, com o comando: **pip install matplotlib**.
- Para saber se o pacote foi instalado no ambiente, pode ser executado o comando: **pip freeze**.
- Obs: Caso esteja usando o Anaconda, o Matplotlib , bem como outras bibliotecas já vem instalados.
- Uma vez que foi instalado o Matplotlib, o comando para importá-lo a fim de utilizá-lo na prática é o **import matplotlib**. Geralmente, por questões de praticidade o matplotlib costuma ser importado assim:

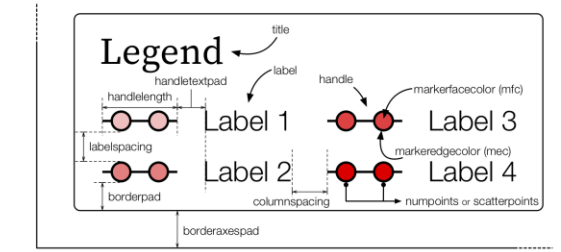
import matplotlib.pyplot as plt

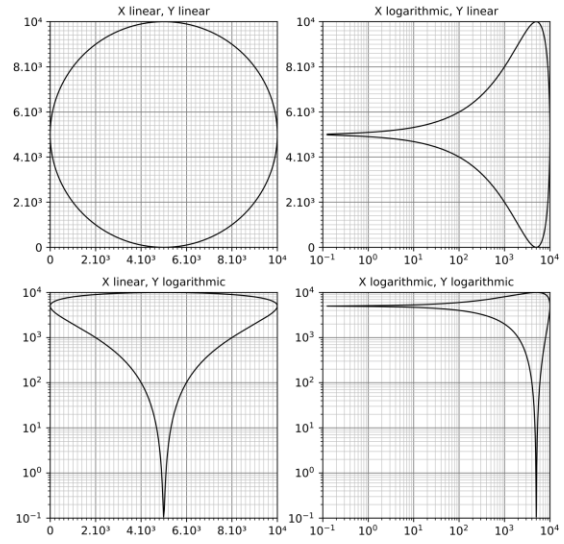
Matplotlib - Operações

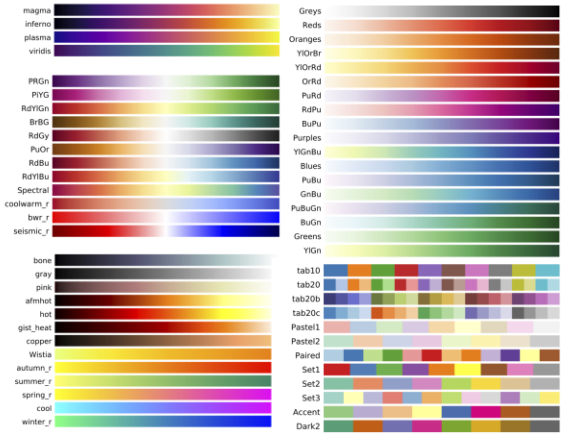


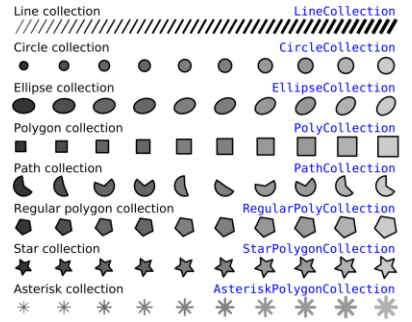


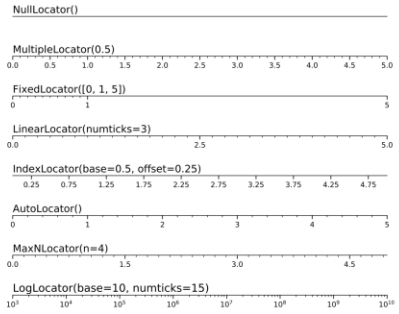


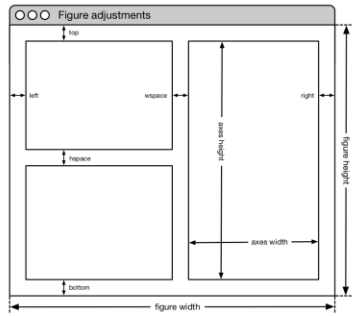




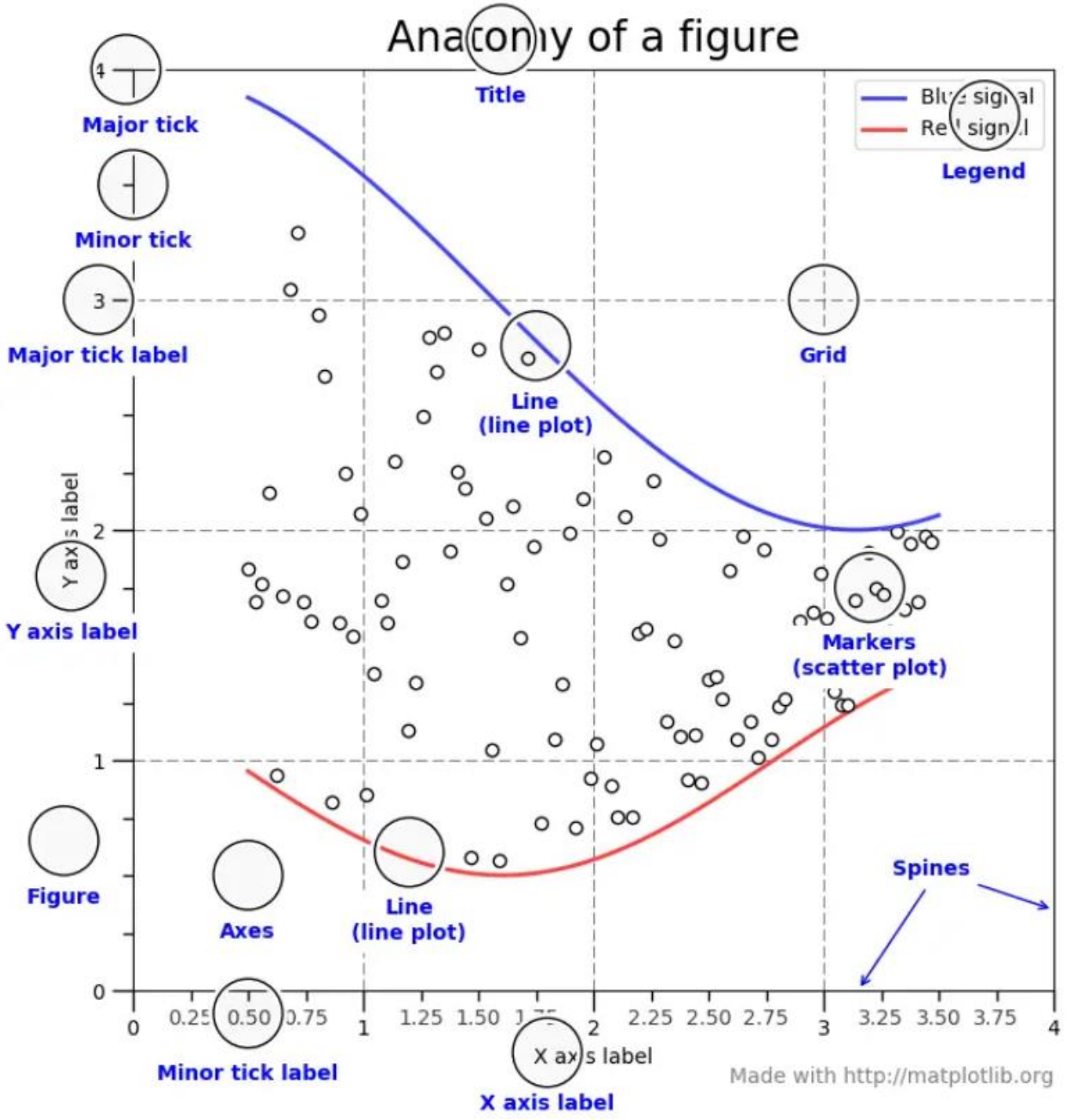






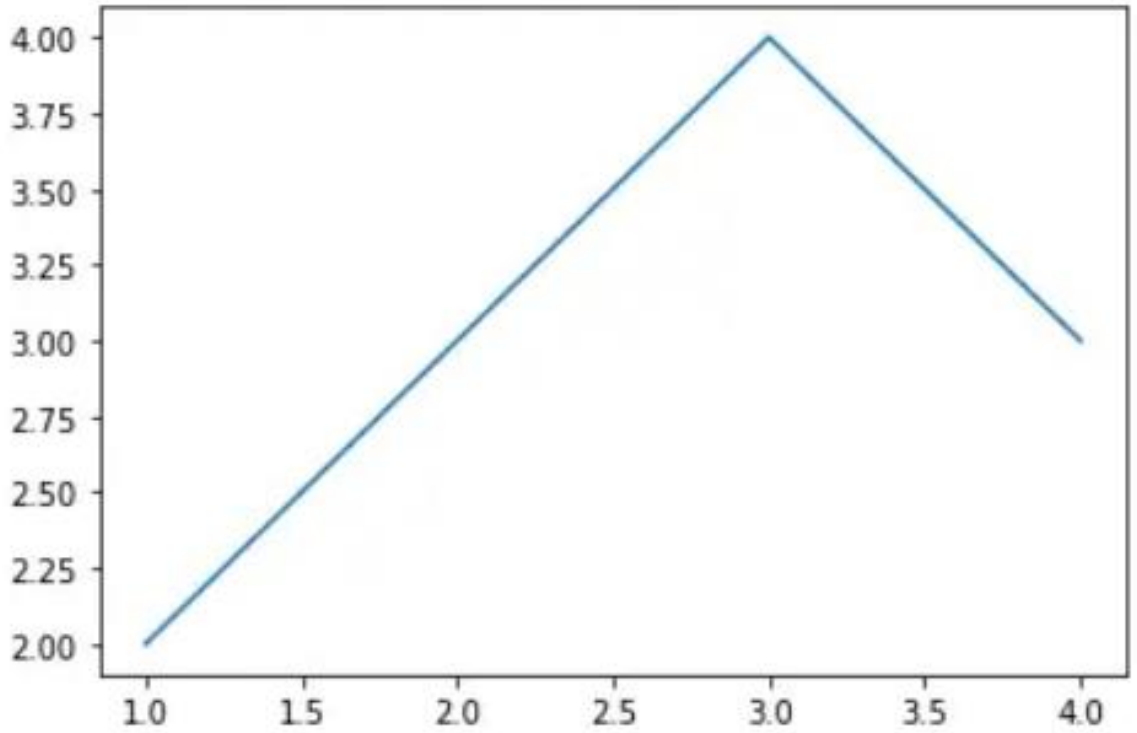


Matplotlib - Anatomia Figura

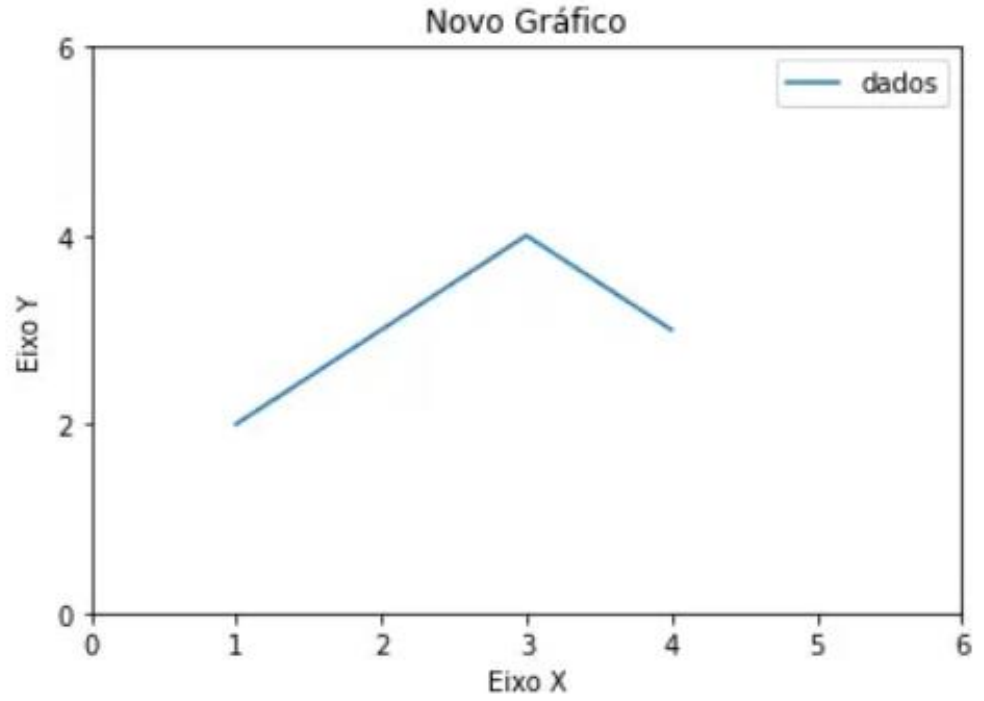


Matplotlib - Hands On

```
x = [1, 2, 3, 4]
y = [2, 3, 4, 3]

plt.plot(x, y)
plt.show()
```



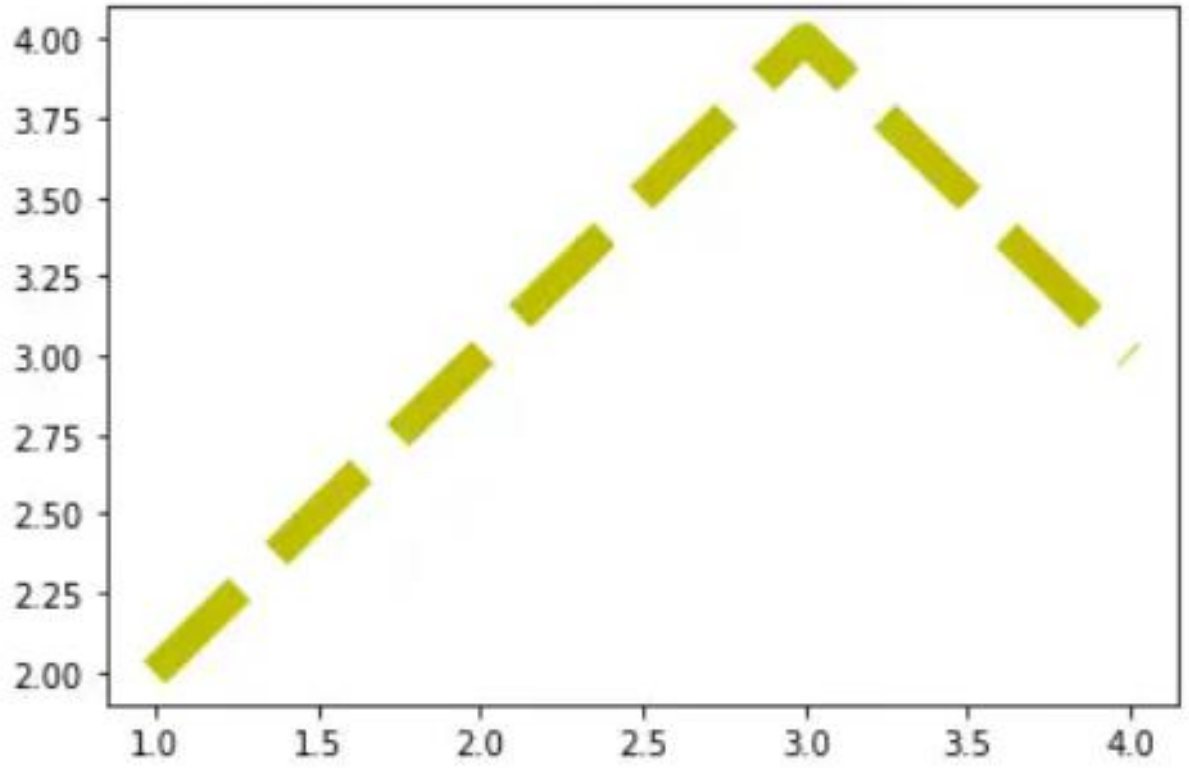

```
plt.plot(x, y, label='dados')
plt.ylabel('Eixo Y')
plt.xlabel('Eixo X')
plt.title('Novo Gráfico')
plt.xticks([0,1,2,3,4,5,6])
plt.yticks([0,2,4,6])
plt.legend()
plt.show()
```



Matplotlib - Hands On

```
plt.plot(x, y, linestyle='--',color='y',lw=10.0)
```

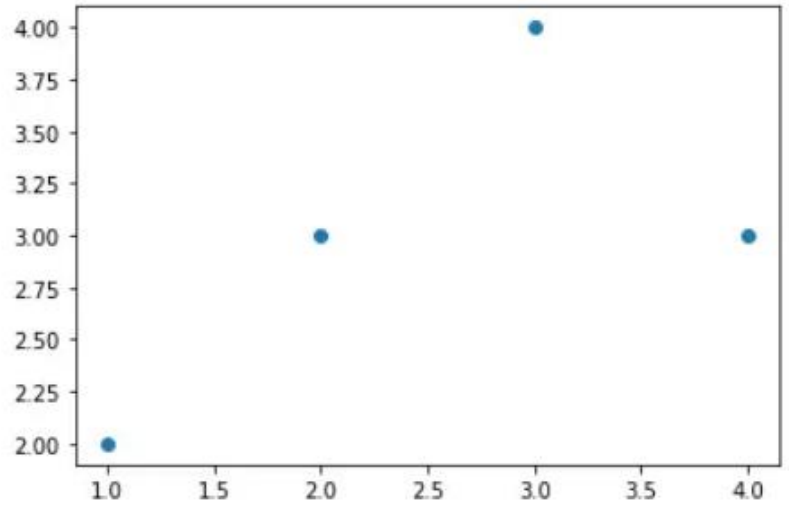
```
[<matplotlib.lines.Line2D at 0x25061f62d90>]
```



.Scatter()

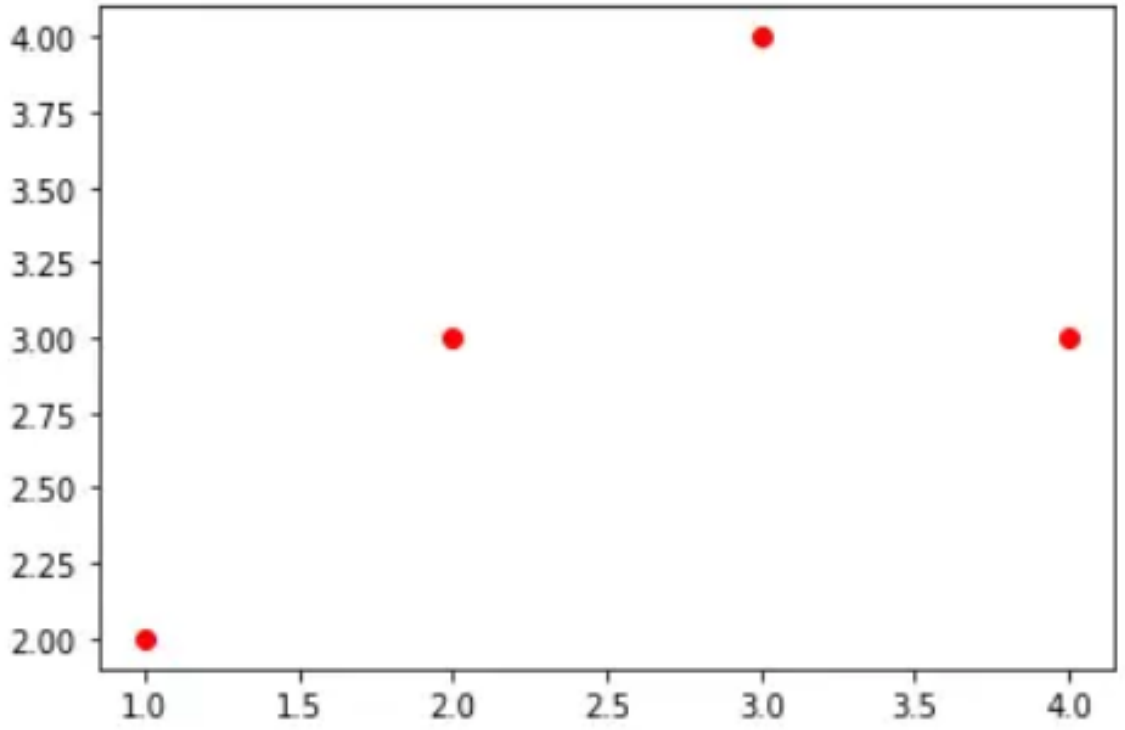
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html#matplotlib-pyplot-scatter

```
plt.scatter(x, y)
plt.show()
```

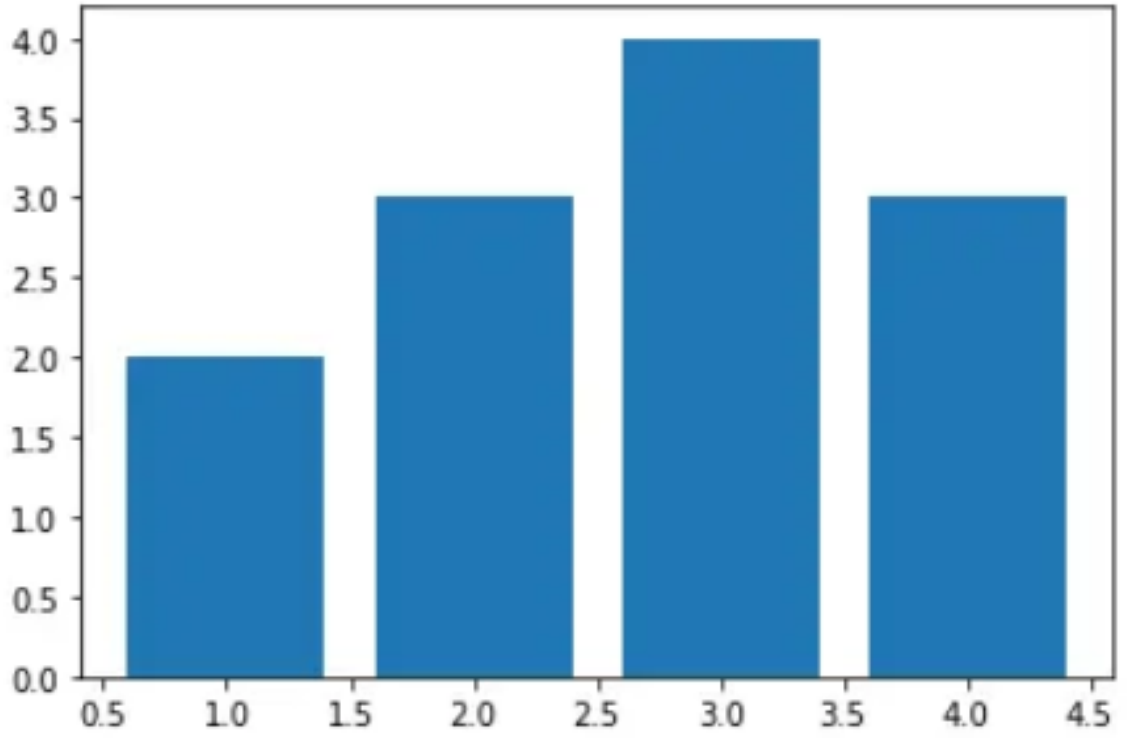


Matplotlib - Hands On

```
plt.plot(x, y, 'ro')
plt.show()
```




```
plt.bar(x,y)
plt.show()
```



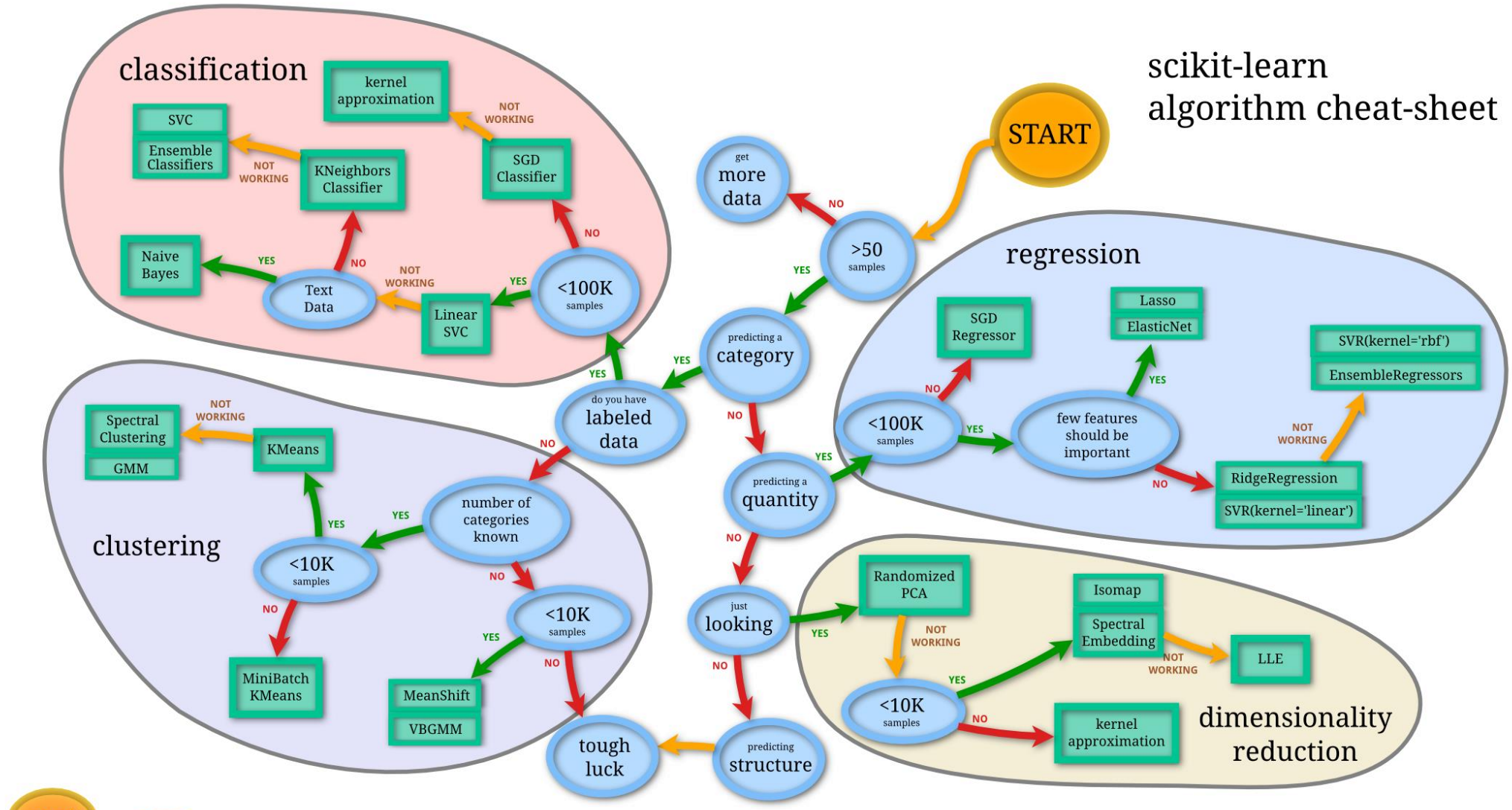
Scikit-Learn

- A Scikit-Learn é uma biblioteca Python para trabalhar com Machine Learning, com ela já estão implementados diversos métodos, algoritmos e técnicas bem interessantes que simplificam a vida do desenvolvedor.
- Biblioteca Python de código aberto que implementa uma variedade de algoritmos de aprendizado de máquina, pré-processamento, validação cruzada e visualização usando uma interface unificada.
- Antes de instalar o scikit-learn, certifique-se de ter o NumPy e o SciPy instalados. Depois de instalar o NumPy e o SciPy em funcionamento.
- Construído sobre NumPy, SciPy e matplotlib.
- Código aberto, utilizável comercialmente - licença BSD.

Scikit-Learn - Instalação e Configuração

- A biblioteca Matplotlib pode ser instalada via gerenciador de pacotes Pip, com o comando: **pip install scikit-learn**.
- Para saber se o pacote foi instalado no ambiente, pode ser executado o comando: **pip freeze**.
- Obs: Caso esteja usando o Anaconda, o Matplotlib , bem como outras bibliotecas já vem instalados.
- Uma vez que foi instalado o Matplotlib, o comando para importá-lo a fim de utilizá-lo na prática é o **import sklearn**.

Scikit-Learn - Algoritmos



Scikit-Learn - Hands On

Animal	Pelo Longo?	Perna curta?	Au Au?	0 cão or 1 porco?
pig A	1	1	0	1
pig B	0	1	1	1
pig C	0	1	0	1
dog A	0	1	1	0
dog B	1	1	1	0
dog C	0	0	1	0

Problema de Classificação:
1 - Porco
0 - Cachorro

X (Entrada) - Temos 3 variáveis de entrada ou preditoras.
Y (Saída) - Temos um problema de uniclasse (uma única classe).

Scikit-Learn - Hands On

```
# features (1 sim, 0 não)
# pelo longo?
# perna curta?
# faz auau?
porco1 = [0, 1, 0]
porco2 = [0, 1, 1]
porco3 = [1, 1, 0]

cachorro1 = [0, 1, 1]
cachorro2 = [1, 0, 1]
cachorro3 = [1, 1, 1]

# 1 => porco, 0 => cachorro
dados = [porco1, porco2, porco3, cachorro1, cachorro2, cachorro3]
classes = [1,1,1,0,0,0]
```

```
from sklearn.svm import LinearSVC

model = LinearSVC()
model.fit(dados, classes)
```

```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
```

Scikit-Learn - Hands On

```
animal_misterioso = [1,1,1]
model.predict([animal_misterioso])
```

Retorno -> array([0])

```
misterio1 = [1,1,1]
misterio2 = [1,1,0]
misterio3 = [0,1,1]

testes = [misterio1, misterio2, misterio3]
model.predict(testes)
```

Retorno -> ([0, 1, 0])

```
misterio1 = [1,1,1]
misterio2 = [1,1,0]
misterio3 = [0,1,1]

testes = [misterio1, misterio2, misterio3]
previsoes = model.predict(testes)

testes_classes = [0, 1, 1]
```

previsoes == testes_classes
array([True, True, False])

Retorno -> 66.66% - indica o valor da acurácia (desempenho do modelo)

```
from sklearn.metrics import accuracy_score

taxa_de_acerto = accuracy_score(testes_classes, previsoes)
print("Taxa de acerto", taxa_de_acerto * 100)
```

NLTK

- NLTK, que significa Natural Language Toolkit, é uma biblioteca disponível na linguagem Python para realizar tarefas de NLP.
- É uma das principais bibliotecas de estudo para quem está ingressando nessa área, pois nela se encontram datasets e alguns algoritmos essenciais para análise e pré-processamento textual.
- O NLP é importante por razões científicas, econômicas, sociais e culturais. O NLP tem sido sujeito a um rápido crescimento decorrência do emprego de suas teorias e métodos em uma variedade de novas tecnologias de linguagem. Por esta razão é importante para uma ampla gama de pessoas ter competências práticas quanto ao NLP. Na indústria, isso inclui as pessoas nos campos da interação homem-máquina, da análise de informações de negócios e do desenvolvimento de software para a web. No meio acadêmico, inclui pessoas em áreas que vão da computação em ciências humanas e da lingüística de corpora aos campos da ciência da computação e da inteligência artificial. (Para muitas pessoas no meio acadêmico, o NLP é conhecido pelo nome de "Lingüística Computacional.")

NLTK - Instalação e Configuração

- A biblioteca Matplotlib pode ser instalada via gerenciador de pacotes Pip, com o comando: **pip install nltk.** Para saber se o pacote foi instalado no ambiente, pode ser executado o comando: **pip freeze**.
- Obs: Caso esteja usando o Anaconda, o NLTK, bem como outras bibliotecas já vem instalados.
- Uma vez que foi instalado o NTLK, o comando para importá-lo a fim de utilizá-lo na prática é o **import nltk**.
- O NLTK vem com inúmeros datasets ou corpus, que podem ser baixados para realizar testes em PLN. Para baixar um dataset, pode ser executado o comando: **nltk.download(dataset)**.

NLTK - Operações

Handling Text

`text='Some words'`	assign string
`list(text)`	Split text into character tokens
`set(text)`	Unique tokens
`len(text)`	Number of characters

Accessing corpora and lexical resources

`from nltk.corpus import brown`	import CorpusReader object
`brown.words(text_id)`	Returns pretokenised document as list of words
`brown.fileids()`	Lists docs in Brown corpus
`brown.categories()`	Lists categories in Brown corpus

Tokenization

`text.split(" ")`	Split by space
`nltk.word_tokenizer(text)`	nltk in-built word tokenizer
`nltk.sent_tokenize(doc)`	nltk in-built sentence tokenizer

Lemmatization & Stemming

`input="List listed lists listing listings"`	Different suffixes
`words=input.lower().split(' ')`	Normalize (lowercase) words
`porter=nltk.PorterStemmer`	Initialise Stemmer
`[porter.stem(t) for t in words]`	Create list of stems
`WNL=nltk.WordNetLemmatizer()`	Initialise WordNet lemmatizer
`[WNL.lemmatize(t) for t in words]`	Use the lemmatizer

Sentence Parsing

`g=nltk.data.load('grammar.cfg')`	Load a grammar from a file
`g=nltk.CFG.fromstring("""...""")`	Manually define grammar
`parser=nltk.ChartParser(g)`	Create a parser out of the grammar
`trees=parser.parse_all(text)`	
`for tree in trees: ... print tree`	
`from nltk.corpus import treebank`	
`treebank.parsed_sents('wsj_0001.mrg')`	Treebank parsed sentences

Text Classification

`from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer`	
`vect=CountVectorizer().fit(X_train)`	Fit bag of words model to data
`vect.get_feature_names()`	Get features
`vect.transform(X_train)`	Convert to doc-term matrix

Entity Recognition (Chunking/Chinking)

`g="NP: {<DT>?<JJ>*<NN>}"`	Regex chunk grammar
`cp=nltk.RegexpParser(g)`	Parse grammar
`ch=cp.parse(pos_sent)`	Parse tagged sent. using grammar
`print(ch)`	Show chunks
`ch.draw()`	Show chunks in IOB tree
`cp.evaluate(test_sents)`	Evaluate against test doc
`sents=nltk.corpus.treebank.tagged_sents()`	
`print(nltk.ne_chunk(sent))`	Print chunk tree

NLTK - Hands On

```
1  # Imports
2  import nltk
3  from nltk.corpus import webtext
4
5  # Fileids
6  print("\n")
7  print(webtext.fileids())
8
9  # Distribuição de frequência de um único arquivo
10 fileid = 'singles.txt'
11 wbt_words = webtext.words(fileid)
12 fdist = nltk.FreqDist(wbt_words)
13
14 # Report
15 print('\nContagem do número máximo de ocorrências do token "',fdist.max(),'" : ', fdist[fdist.max()])
16 print('\nNúmero total de tokens distintos : ', fdist.N())
17 print('\nA seguir estão os 10 tokens mais comuns')
18 print(fdist.most_common(10))
19 print("\n")
```

```
['firefox.txt', 'grail.txt', 'overheard.txt', 'pirates.txt', 'singles.txt', 'wine.txt']

Contagem do número máximo de ocorrências do token " , " :  539

Número total de tokens distintos :  4867

A seguir estão os 10 tokens mais comuns
[(',', 539), ('.', 353), ('/', 110), ('for', 99), ('and', 74), ('to', 74), ('lady', 68), ('-', 66), ('seeks', 60), ('a',
 52)]
```

Nessa análise fazemos uma **distribuição de frequência** a fim de identificar as palavras e / ou caracteres mais recorrentes de um corpus.

NLTK - Hands On

```python
from nltk.tokenize import sent_tokenize, word_tokenize

# Texto
frase = "Aprendendo processamento linguagem natural. Python e NLTK facilitam nossa vida!"

# Tokenization em sentenças
sent_tokens = sent_tokenize(frase)
print(sent_tokens)
```

```
['Aprendendo processamento linguagem natural.', 'Python e NLTK facilitam nossa vida!']
```

Nessa análise a divisão de palavras por tokens. Essa técnica é chamada de **tokenização.**

```python
from nltk.tokenize import sent_tokenize, word_tokenize

# Texto
frase = "Aprendendo processamento linguagem natural. Python e NLTK facilitam nossa vida!"

# Tokenization em sentenças
word_tokens = word_tokenize(frase)
print(word_tokens)
```

```
['Aprendendo', 'processamento', 'linguagem', 'natural', '.', 'Python', 'e', 'NLTK', 'facilitam', 'nossa', 'vida', '!']
```

NLTK - Hands On

```
1  from nltk.corpus import stopwords
2
3  portuguese_stops = set(stopwords.words('portuguese'))
4
5  print(portuguese_stops)
```

```
{'tenha', 'esteja', 'me', 'houveríamos', 'esses', 'teria', 'na', 'o', 'haja', 'havemos', 'tenhamos', 'e', 'houverá', 'es
tava', 'um', 'pela', 'houver', 'éramos', 'para', 'aquilo', 'eram', 'tiver', 'estiveram', 'vos', 'tivesse', 'no', 'minha'
, 'mas', 'houveriam', 'eu', 'suas', 'tu', 'estivéssemos', 'as', 'estiverem', 'às', 'estivermos', 'estas', 'muito', 'tive
rmos', 'lhes', 'como', 'quem', 'estive', 'houve', 'aos', 'tinham', 'teríamos', 'formos', 'terei', 'uma', 'seu', 'não', '
esta', 'em', 'isto', 'numa', 'fôramos', 'hei', 'este', 'tenham', 'houverem', 'estivessem', 'aquelas', 'vocês', 'do', 'se
ria', 'nos', 'minhas', 'fomos', 'estiver', 'dela', 'fosse', 'estão', 'à', 'qual', 'nossa', 'tive', 'nós', 'estivera', 'q
ue', 'estamos', 'quando', 'tivéramos', 'fui', 'tivera', 'era', 'houvéssemos', 'tenho', 'ele', 'houvéramos', 'houvermos',
 'teus', 'mesmo', 'lhe', 'tiverem', 'num', 'meus', 'aquele', 'os', 'depois', 'tinha', 'tua', 'estavam', 'se', 'hajam', '
houvessem', 'houveremos', 'nosso', 'seja', 'pelo', 'essas', 'sem', 'tivéssemos', 'fôssemos', 'foi', 'você', 'tuas', 'del
as', 'sou', 'houvesse', 'sejam', 'só', 'estejam', 'te', 'houverão', 'terá', 'houvemos', 'dele', 'estivemos', 'tém', 'est
á', 'ao', 'tiveram', 'essa', 'meu', 'ela', 'eles', 'dos', 'seus', 'estávamos', 'terão', 'serei', 'esse', 'houveram', 'se
ríamos', 'são', 'esteve', 'será', 'há', 'também', 'forem', 'foram', 'de', 'somos', 'houveria', 'aqueles', 'tivemos', 'es
tes', 'já', 'nossas', 'aquela', 'houverei', 'sua', 'teriam', 'estivéramos', 'sejamos', 'seremos', 'pelos', 'teu', 'fosse
m', 'for', 'serão', 'pelas', 'estejamos', 'hajamos', 'nossos', 'seriam', 'teve', 'elas', 'temos', 'até', 'tínhamos', 'te
remos', 'estivesse', 'por', 'tivessem', 'mais', 'das', 'da', 'hão', 'nas', 'deles', 'isso', 'houvera', 'nem', 'estou', '
com', 'fora', 'tem', 'ou', 'a', 'entre'}
```

Nessa análise utilizamos o **Stopword** para consultar as palavras que não agregam valor a uma análise textual.

NLTK - Hands On

```python
from nltk.corpus import stopwords
portuguese_stops = set(stopwords.words('portuguese'))
palavras = ["Estou", 'estudando', 'sobre', 'um', 'tema', 'interesante', 'em', 'PLN']
print([palavra for palavra in palavras if palavra not in portuguese_stops])
```

```
['Estou', 'estudando', 'sobre', 'tema', 'interesante', 'PLN']
```

```
PorterStemmer
stude
studi

LancasterStemmer
stud
study
```

```python
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer


# Cria o Stemmer
stemmer = PorterStemmer()

# Aplica o Stemmer
print("\nPorterStemmer")
print(stemmer.stem('studing'))
print(stemmer.stem('studied'))

# Cria o Stemmer
stemmer2 = LancasterStemmer()

# Aplica o Stemmer
print("\nLancasterStemmer")
print(stemmer2.stem('studing'))
print(stemmer2.stem('studied'))
```

Q1) [CESGRANRIO BB 2021] Ao analisar um conjunto de dados com Python, um programador resolveu usar um dataframe Pandas de nome dp para guardá-los. Em um certo momento, ele resolveu que precisaria usar, apenas, quatro colunas de dados do dataframe: “pais“, “ano“, “renda per capita“ e “expectativa de vida“.

Que fragmento de código Python 3 deve ser usado para selecionar, apenas, essas quatro colunas do dataframe dp?

a) dp[“pais“,“ano“,“expectativa de vida“,“renda per capita“]

b) dp[[“pais“,“ano“,“expectativa de vida“,“renda per capita“]]

c) dp(“pais“,“ano“,“expectativa de vida“,“renda per capita“)

d) dp([“pais“,“ano“,“expectativa de vida“,“renda per capita“])

e) dp[dp[“pais“,“ano“,“expectativa de vida“,“renda per capita“]]

Q1) [CESGRANRIO BB 2021] Ao analisar um conjunto de dados com Python, um programador resolveu usar um dataframe Pandas de nome dp para guardá-los. Em um certo momento, ele resolveu que precisaria usar, apenas, quatro colunas de dados do dataframe: “pais“, “ano“, “renda per capita“ e “expectativa de vida“.

Que fragmento de código Python 3 deve ser usado para selecionar, apenas, essas quatro colunas do dataframe dp?

a) dp[“pais“,“ano“,“expectativa de vida“,“renda per capita“]

b) dp[[“pais“,“ano“,“expectativa de vida“,“renda per capita“]]

c) dp(“pais“,“ano“,“expectativa de vida“,“renda per capita“)

d) dp([“pais“,“ano“,“expectativa de vida“,“renda per capita“])

e) dp[dp[“pais“,“ano“,“expectativa de vida“,“renda per capita“]]

Q2) [UFMT UFMT 2021] A respeito da linguagem python, analise as afirmativas.

I- Possui bibliotecas como pandas e numpy.

II- Disponibiliza documentação na Internet.

III- É uma linguagem de baixo nível.

Está correto o que se afirma em

a) I e III, apenas.

b) I e II, apenas.

c) II e III, apenas.

d) III, apenas.

Q2) [UFMT UFMT 2021] A respeito da linguagem python, analise as afirmativas.

I- Possui bibliotecas como pandas e numpy.

II- Disponibiliza documentação na Internet.

III- É uma linguagem de baixo nível.

Está correto o que se afirma em

a) I e III, apenas.

b) I e II, apenas.

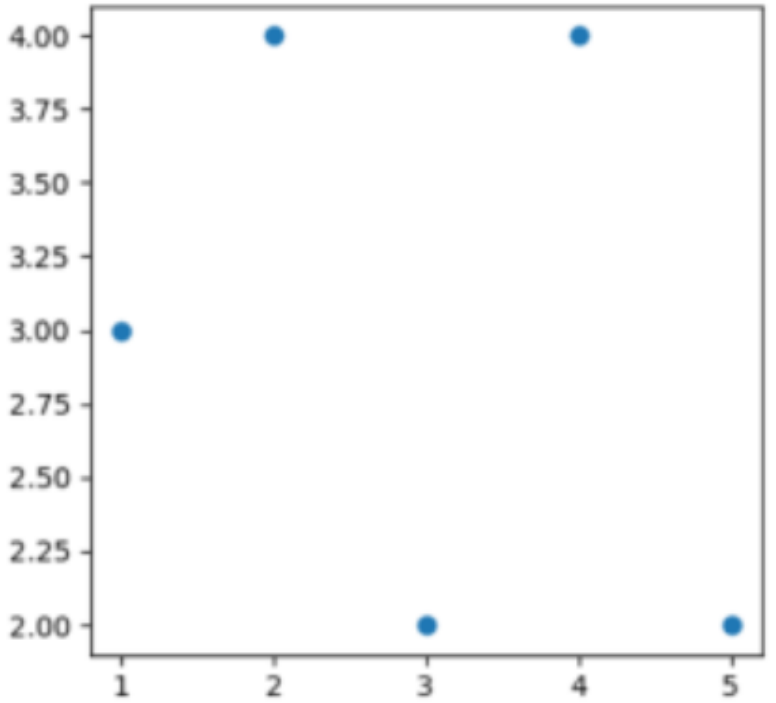
c) II e III, apenas.

d) III, apenas.

Q3) [CESPE BANESE 2021] Com relação aos pacotes Matplotlib e NumPy, julgue o seguinte item.

Considerando o conjunto de dados formado por pares de pontos {(1,3), (2,4), (3,2), (4,4), (5,2)}, o código a seguir produz um gráfico de dispersão na forma ilustrada na figura anterior.

```
import numpy

import matplotlib.pyplot as plt

x = numpy.array([1,2,3,4,5])

y = numpy.array([3,4,2,4,2])

plt.scatter(x, y)

plt.show()
```



Q3) [CESPE BANESE 2021] Com relação aos pacotes Matplotlib e NumPy, julgue o seguinte item.

Considerando o conjunto de dados formado por pares de pontos {(1,3), (2,4), (3,2), (4,4), (5,2)}, o código a seguir produz um gráfico de dispersão na forma ilustrada na figura anterior.

CERTO.

```
import numpy

import matplotlib.pyplot as plt

x = numpy.array([1,2,3,4,5])

y = numpy.array([3,4,2,4,2])

plt.scatter(x, y)

plt.show()
```