

PYTHON - AVANÇADO



python



INTRODUÇÃO AO PANDAS

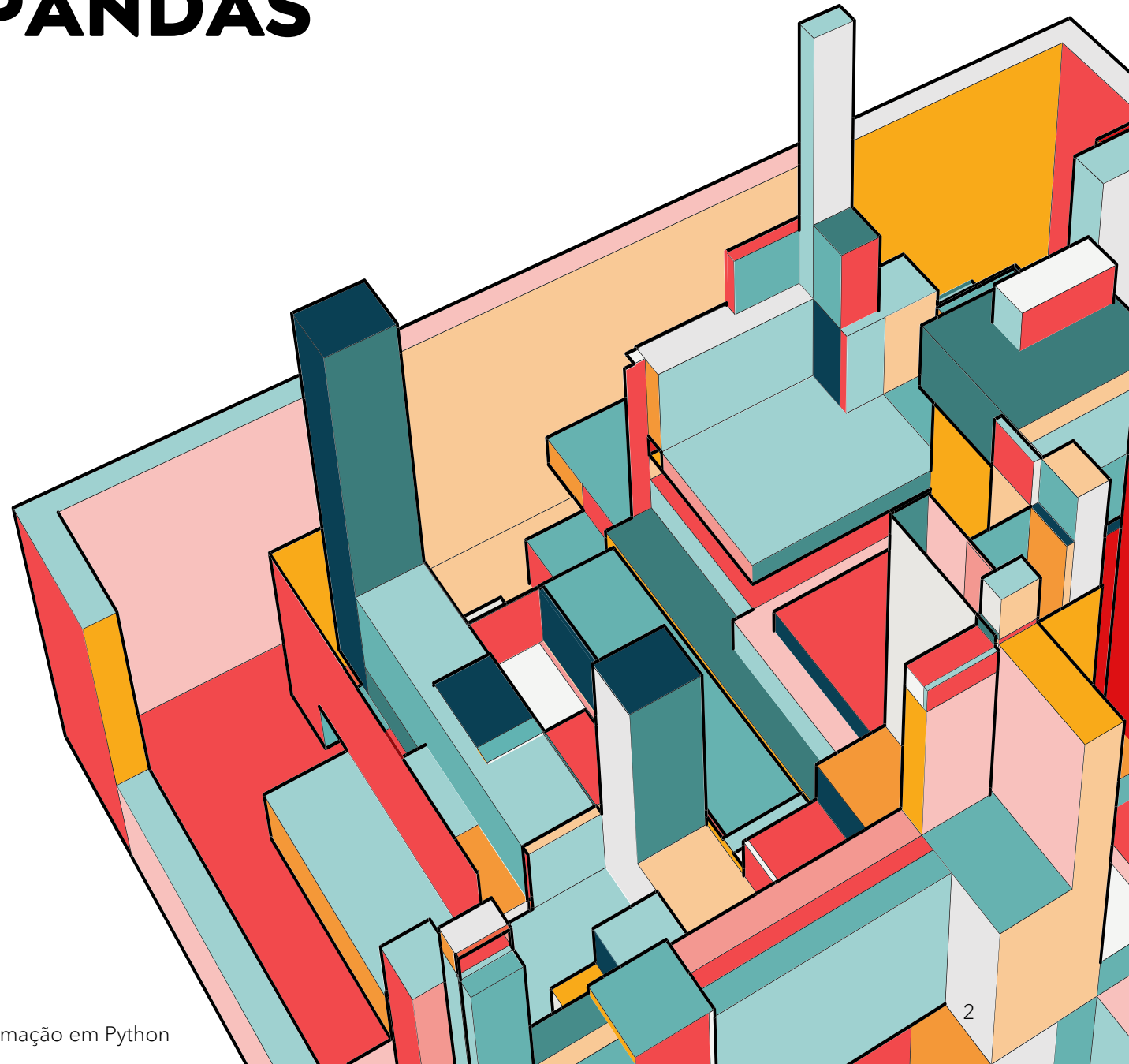
Abrir o **VS Code**

Ou

PyCharm

OU

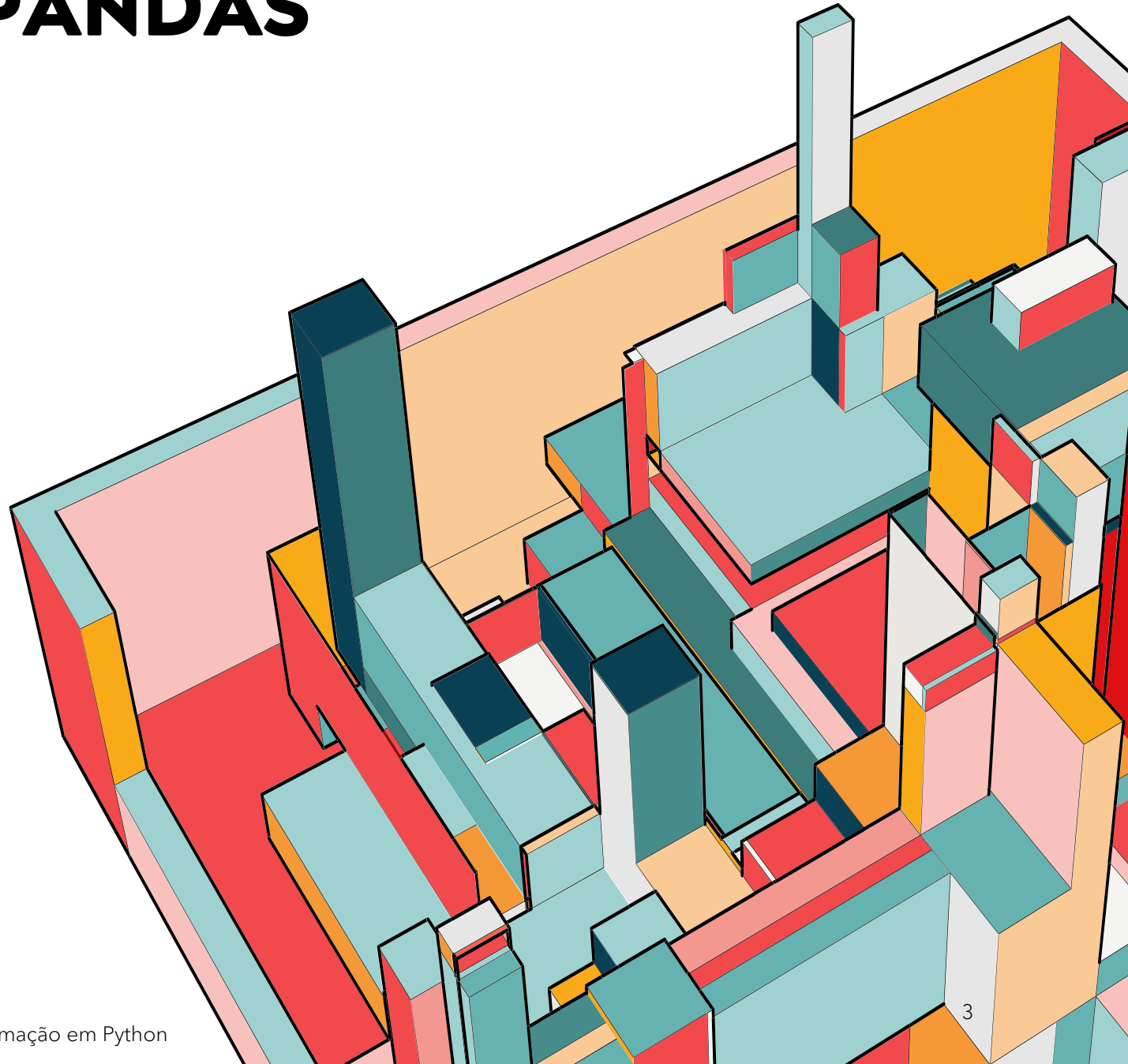
Anaconda



INTRODUÇÃO AO PANDAS

Usaremos a função:

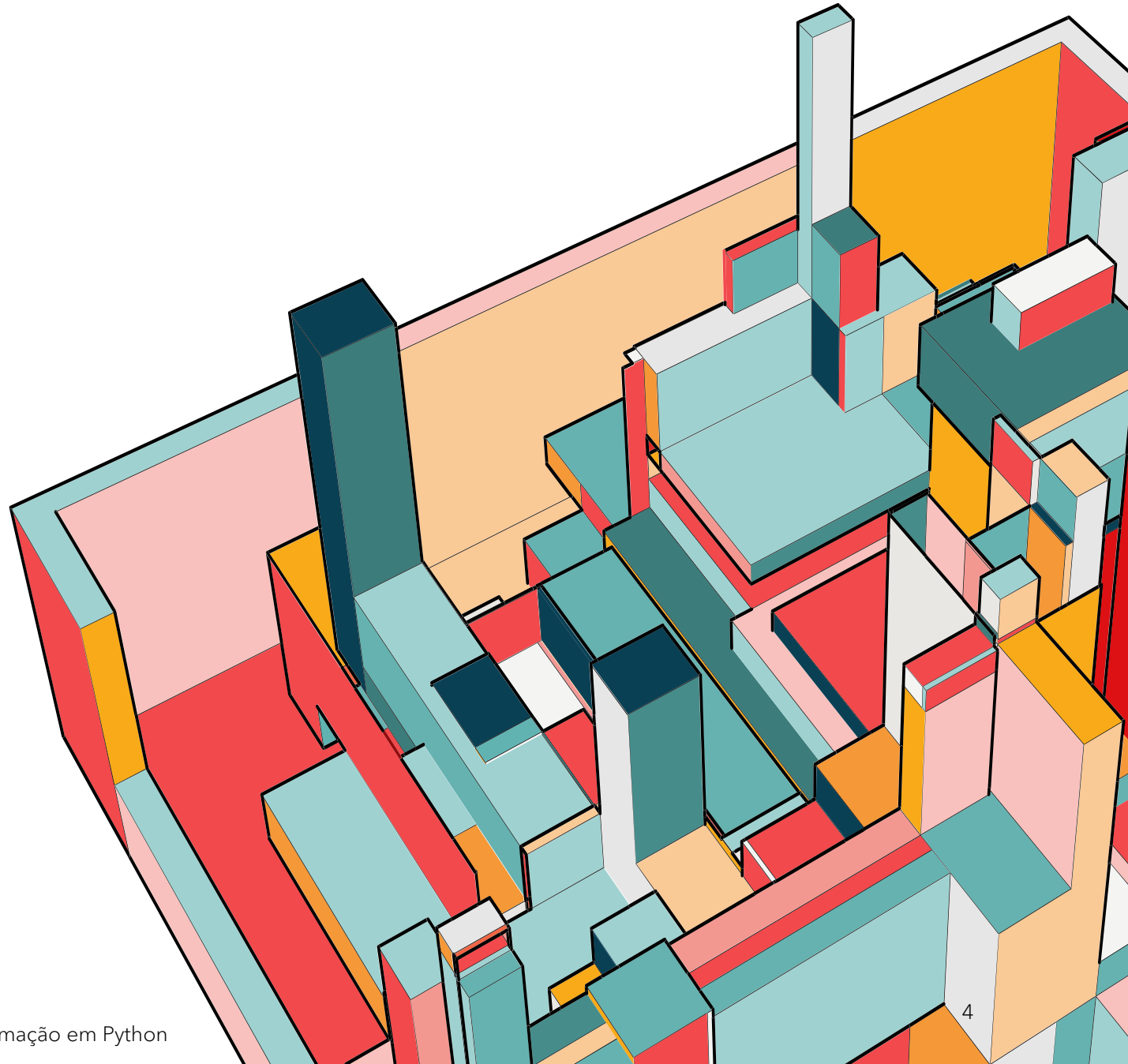
read_html



READ_HTML

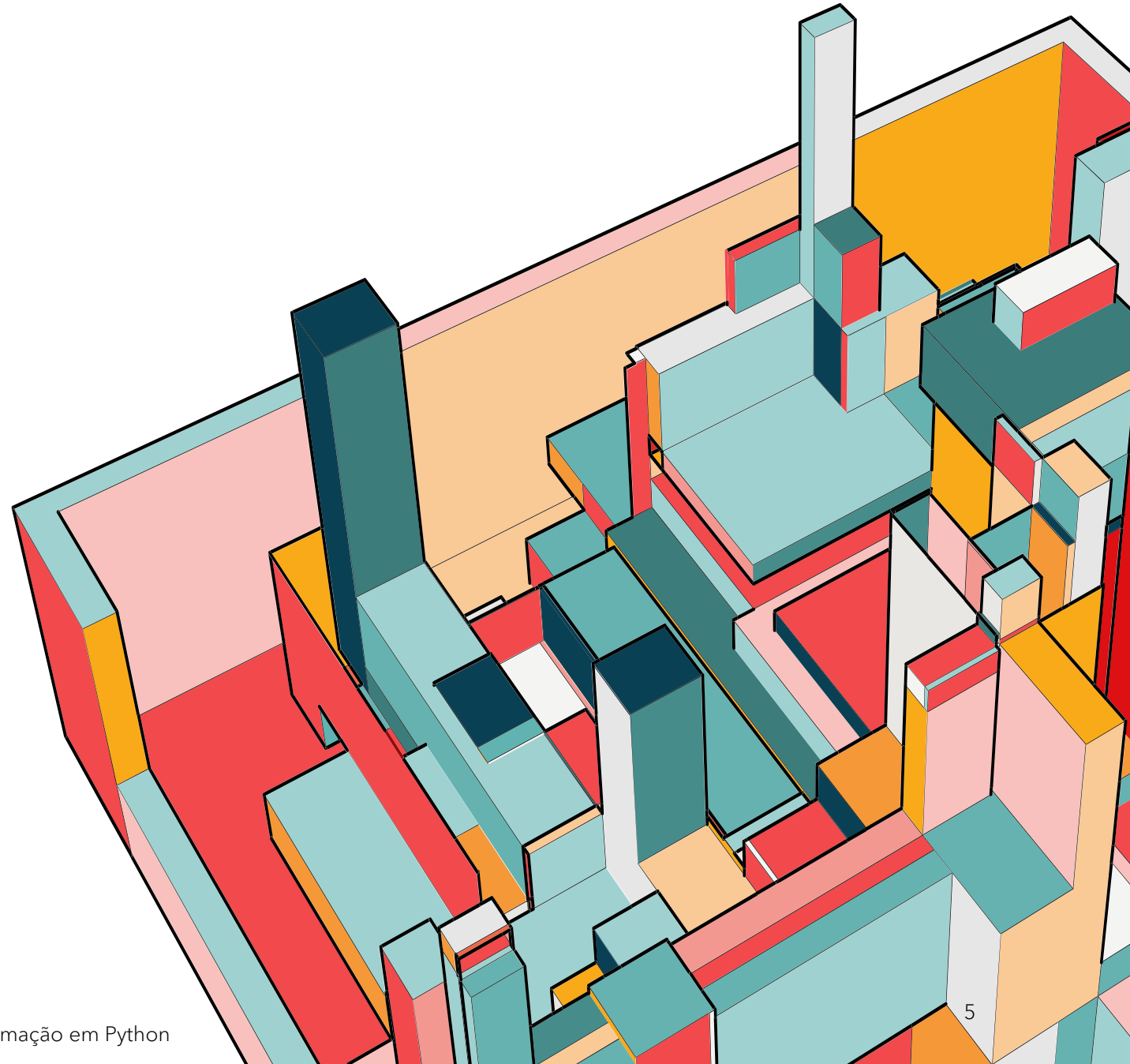
Para ser mais específico, ele vai “varrer” o código HTML inteiro da página em busca de tabelas.

Todas as tabelas que ele encontrar serão adicionadas a uma lista (list). Ou seja, o retorno dessa função é uma lista de DataFrames, em que cada DataFrame contém uma tabela do site



READ_HTML

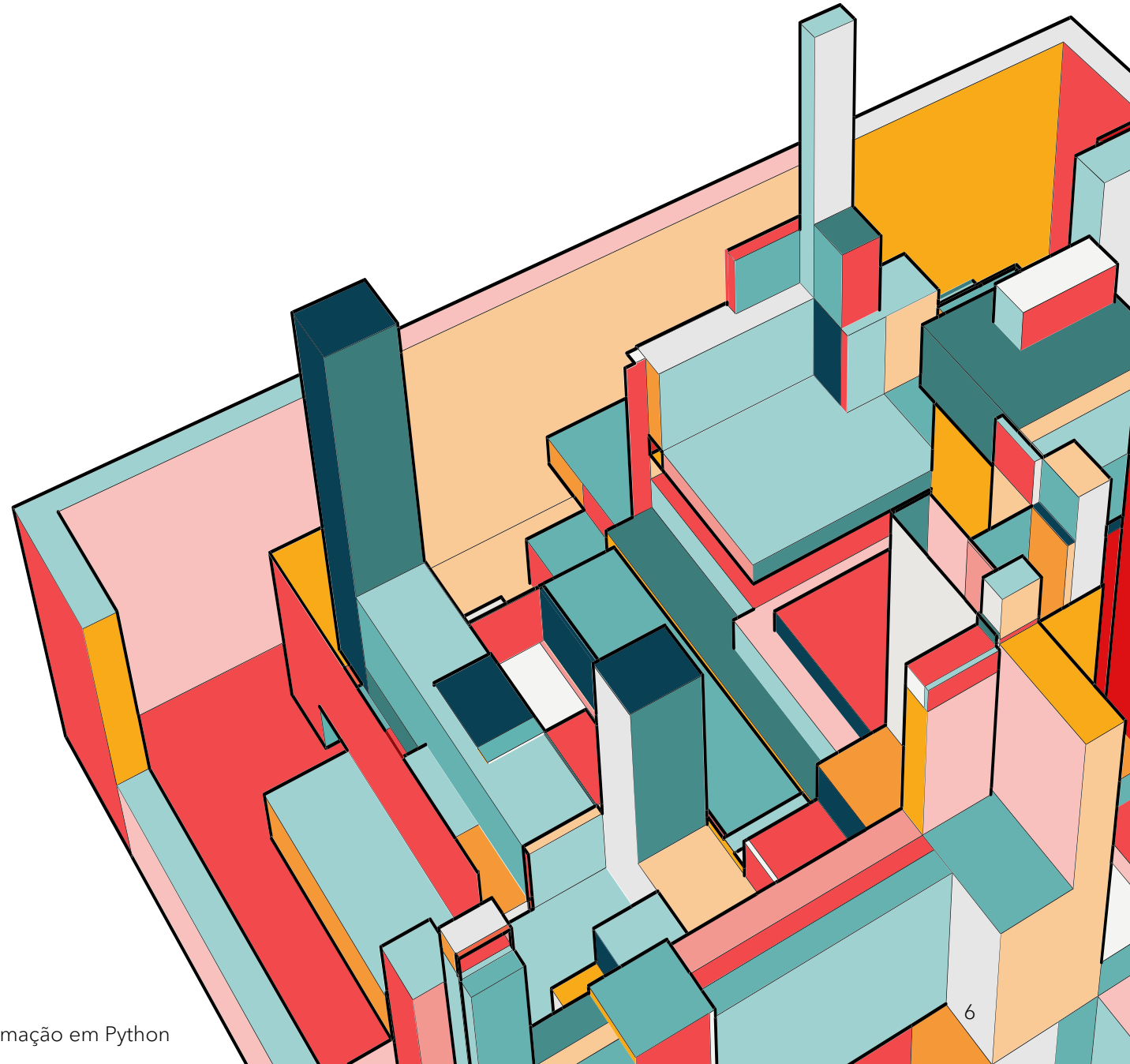
Isso significa que, mesmo que o site contenha apenas uma tabela, o retorno será uma lista com apenas um elemento: a única tabela da página, que já estará armazenada em um DataFrame.



READ_HTML

**E se não houver tabelas
na página?**

**Nesse caso, o programa
retornará uma exceção para
você: *ValueError: No tables
found.***



READ_HTML

Mas, como você deve utilizar essa função. É super simples! O único parâmetro que você precisa especificar é o *url* da página que contém a tabela que você deseja obter. Vejamos o exemplo abaixo:

```
import pandas as pd
```

```
df_list = pd.read_html('https://www.fdic.gov/resources/resolutions/bank-failures/failed-bank-list/')
```

READ_HTML

- ❑ Observe que **df_list** é a variável que receberá o retorno da função **read_html**.
- ❑ Esse retorno será uma lista de **DataFrames**.
- ❑ A função **read_html** contém apenas uma string que consiste na *url* da página.

READ_HTML

- ❑ Essa página contém uma tabela com algumas informações sobre bancos que já faliram.

<https://www.fdic.gov/resources/resolutions/bank-failures/failed-bank-list/>

- ❑ Ao executar esse linha de código, e imprimir o conteúdo da variável **df_list**, você obterá o resultado abaixo:

READ HTML

```
In [8]: import pandas as pd
```

```
df_list = pd.read_html('https://www.fdic.gov/bank/individual/failed/banklist.html')
```

```
df_list
```

```
Out[8]: [
      Bank Name      City  ST  CERT \
0  The First State Bank  Barboursville  WV  14361
1  Ericson State Bank    Ericson    NE  18265
2  City National Bank of New Jersey  Newark  NJ  21111
3  Resolute Bank        Maumee    OH  58317
4  Louisa Community Bank  Louisa    KY  58112
..
556  Superior Bank, FSB    Hinsdale  IL  32646
557  Malta National Bank  Malta    OH  6629
558  First Alliance Bank & Trust Co.  Manchester  NH  34264
559  National State Bank of Metropolis  Metropolis  IL  3815
560  Bank of Honolulu     Honolulu  HI  21029

      Acquiring Institution      Closing Date
0  MVB Bank, Inc.      April 3, 2020
1  Farmers and Merchants Bank  February 14, 2020
2  Industrial Bank    November 1, 2019
3  Buckeye State Bank  October 25, 2019
4  Kentucky Farmers Bank Corporation  October 25, 2019
..
556  Superior Federal, FSB    July 27, 2001
557  North Valley Bank      May 3, 2001
558  Southern New Hampshire Bank & Trust  February 2, 2001
559  Banterra Bank of Marion  December 14, 2000
560  Bank of the Orient    October 13, 2000

[561 rows x 6 columns]]
```

READ_HTML

Note que o conteúdo dessa variável é uma lista, que possui apenas um elemento, inclusive (experimente dar o comando `len(df_list)` para comprovar isso). Portanto, imprimindo a posição 0 dessa lista, chegaremos ao DataFrame:

READ_HTML

```
In [9]: import pandas as pd

df_list = pd.read_html('https://www.fdic.gov/bank/individual/failed/banklist.html')

df_list[0].head()
```

Out[9]:

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date
0	The First State Bank	Barboursville	WV	14361	MVB Bank, Inc.	April 3, 2020
1	Ericson State Bank	Ericson	NE	18265	Farmers and Merchants Bank	February 14, 2020
2	City National Bank of New Jersey	Newark	NJ	21111	Industrial Bank	November 1, 2019
3	Resolute Bank	Maumee	OH	58317	Buckeye State Bank	October 25, 2019
4	Louisa Community Bank	Louisa	KY	58112	Kentucky Farmers Bank Corporation	October 25, 2019

READ_HTML

- ❑ Excelente! O DataFrame contém exatamente o mesmo que estava presente na tabela do site. Agora, vamos testar para uma notícia do G1, por exemplo (você pode acessá-la nesse link):

<https://g1.globo.com/bemestar/coronavirus/noticia/2020/05/26/casos-de-coronavirus-e-numero-de-mortes-no-brasil-em-26-de-maio.ghtml>

READ_HTML

```
In [12]: import pandas as pd

df_list = pd.read_html('https://g1.globo.com/bemestar/coronavirus/noticia/2020/05/26/casos-de-coronavirus-e-numero-')

df_list[0].head()
```

Out[12]:

	0	1	2
0	Estado	Nº de testes	Data de divulgação
1	Acre	11.119	25/5
2	Alagoas	2.594	27/4
3	Amapá	12.165	25/5
4	Amazonas	6.183	27/4

READ_HTML

Note que, dessa vez, **o cabeçalho da tabela do site não se tornou o label das colunas do DataFrame**. Por que isso aconteceu? O fato é que a tabela do site que utilizamos no primeiro exemplo tem uma **tag thead**, que define o cabeçalho de uma tabela no HTML. A função **read_html** utiliza essa **tag** para saber quais serão os labels atribuídos às colunas do **DataFrame** (como foi o caso do primeiro exemplo). Quando essa **tag** não está presente na tabela (o caso do segundo exemplo), ela criará um label automático (0, 1, 2, ...).

```
In [12]: import pandas as pd

df_list = pd.read_html('https://g1.globo.com/bemestar/coronavirus/noticia/2020/05/26/casos-de-coronavirus-e-numero-c')

df_list[0].head()
```

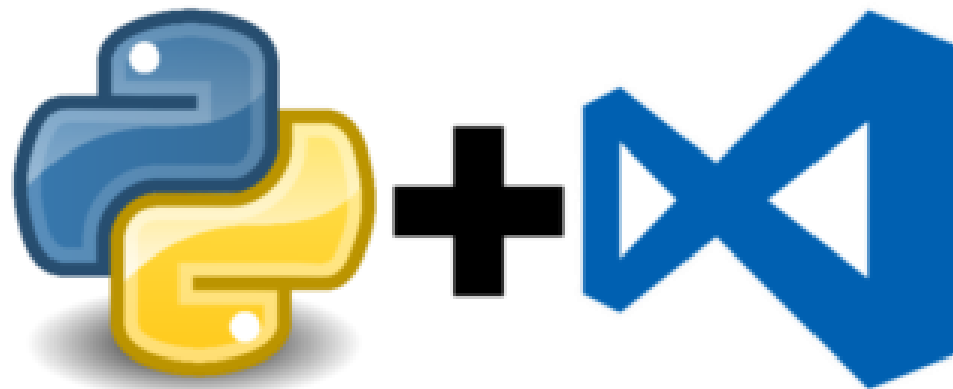
Out[12]:

	0	1	2
0	Estado	Nº de testes	Data de divulgação
1	Acre	11.119	25/5
2	Alagoas	2.594	27/4
3	Amapá	12.165	25/5
4	Amazonas	6.183	27/4

VAMOS PRATICAR???

Como IDE, utilizaremos:

- Google Colab
- Anaconda
- Jupyter Notebook
- Ou o VS Code com a extensão do Jupyter Notebook ativada.



VAMOS PRATICAR???

- ❖ Então, abra o Google;
- ❖ Use a Janela Anônima;
- ❖ Faça login em sua conta do Gmail;
- ❖ Abra o Google Colab.
- ❖ Vá ao Menu *Arquivo – Novo Notebook*;
- ❖ Dê o nome de: *usando_read_html*;
- ❖ E siga as Instruções definidas nos próximos slides.



DIGITE CADA # (TEXTO) E OS COMANDOS (CÓDIGO)

```
# importando a biblioteca Pandas  
import pandas as pd
```



DIGITE CADA # (TEXTO) E OS COMANDOS (CÓDIGO)

A linha de comandos DEVERÁ SER DIGITADA SEM USAR O ENTER

```
# Lendo arquivos HTML  
df =  
pd.read_html('https://gist.github.com/armgilles/194b  
cfff35001e7eb53a2a8b441e8b2c6')[0]
```



DIGITE CADA # (TEXTO) E OS COMANDOS (CÓDIGO)

```
# Digite a variável usada anteriormente  
df
```



A PROVÁVEL SAÍDA, APÓS, A INSERÇÃO DOS CÓDIGOS.

Unnamed: 0		#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	NaN	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	NaN	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	NaN	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	NaN	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	NaN	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
...
795	NaN	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True
796	NaN	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	True
797	NaN	720	HoopaaHoopaa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	True
798	NaN	720	HoopaaHoopaa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	True
799	NaN	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True



AS PRÓXIMAS INSTRUÇÕES SÃO CHAMADAS DE “**CRITÉRIOS PARA FILTROS**” NA EXIBIÇÃO DOS DADOS

```
# selecionando uma coluna do DataFrame.  
coluna = df['Name']
```



AS PRÓXIMAS INSTRUÇÕES SÃO CHAMADAS DE “**CRITÉRIOS PARA FILTROS**” NA EXIBIÇÃO DOS DADOS

```
# selecionando várias colunas do DataFrame  
colunas = df[['Name', 'Type 1', 'Attack']]
```



AS PRÓXIMAS INSTRUÇÕES SÃO CHAMADAS DE “**CRITÉRIOS PARA FILTROS**” NA EXIBIÇÃO DOS DADOS

```
# selecionando linhas do DataFrame com base em  
condições, ex >, <, <=, >=, ==.  
linhas = df[df['Attack'] > 100]
```



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
#Inspecionando informações  
df.info()
```



PROVÁVEL SAÍDA DE INFORMAÇÕES (OS DADOS ESTÃO ONLINE E PODEM SOFRER ALTERAÇÕES)

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 800 entries, 0 to 799  
Data columns (total 14 columns):  
#   Column      Non-Null Count  Dtype  
---  -  
0   Unnamed: 0    0 non-null     float64  
1   #            800 non-null   int64  
2   Name         800 non-null   object  
3   Type 1       800 non-null   object  
4   Type 2       414 non-null   object  
5   Total        800 non-null   int64  
6   HP           800 non-null   int64  
7   Attack       800 non-null   int64  
8   Defense      800 non-null   int64  
9   Sp. Atk      800 non-null   int64  
10  Sp. Def      800 non-null   int64  
11  Speed        800 non-null   int64  
12  Generation   800 non-null   int64  
13  Legendary    800 non-null   bool  
dtypes: bool(1), float64(1), int64(9), object(3)  
memory usage: 82.2+ KB
```



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
#Inspecionando a base  
df.tail()
```



PROVÁVEL SAÍDA DE INFORMAÇÕES (OS DADOS ESTÃO ONLINE E PODEM SOFRER ALTERAÇÕES)

	Unnamed: 0	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
795	NaN	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True
796	NaN	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	True
797	NaN	720	HoopaaHoopaa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	True
798	NaN	720	HoopaaHoopaa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	True
799	NaN	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
#Inspecionando o shape  
df.shape
```



PROVÁVEL SAÍDA DE INFORMAÇÕES (OS DADOS ESTÃO ONLINE E PODEM SOFRER ALTERAÇÕES)

```
(800, 14)
```



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
#Outro critério de Filtro para as informações  
df[(df["Attack"] >= 80) & (df["Defense"] >= 150)]
```



PROVÁVEL SAÍDA DE INFORMAÇÕES (OS DADOS ESTÃO ONLINE E PODEM SOFRER ALTERAÇÕES)

	Unnamed: 0	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legend	
	98	NaN	91	Cloyster	Water	Ice	525	50	95	180	85	45	70	1	Fall
	223	NaN	208	Steelix	Steel	Ground	510	75	85	200	55	65	30	2	Fall
	224	NaN	208	SteelixMega Steelix	Steel	Ground	610	75	125	230	55	95	30	2	Fall
	268	NaN	248	TyranitarMega Tyranitar	Rock	Dark	700	100	164	150	95	120	71	2	Fall
	332	NaN	306	Aggron	Steel	Rock	530	70	110	180	60	60	50	3	Fall
	333	NaN	306	AggronMega Aggron	Steel	NaN	630	70	140	230	60	80	50	3	Fall
	413	NaN	376	MetagrossMega Metagross	Steel	Psychic	700	80	145	150	105	110	110	3	Fall
	414	NaN	377	Regirock	Rock	NaN	580	80	100	200	50	100	50	3	Tr
	424	NaN	383	GroudonPrimal Groudon	Ground	Fire	770	100	180	160	150	90	90	3	Tr
	749	NaN	680	Doublade	Steel	Ghost	448	59	110	150	45	49	35	6	Fall
	789	NaN	713	Avalugg	Ice	NaN	514	95	117	184	44	46	28	6	Fall
	795	NaN	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	Tr



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

A linha de comandos DEVERÁ SER DIGITADA **SEM USAR O ENTER**

```
#Operação entre colunas
```

```
df["Índice de força"] = df['Attack'] + df['Defense']  
+ df['Speed']
```



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

A linha de comandos DEVERÁ SER DIGITADA **SEM USAR O ENTER**

```
#Pegando o maior valor do índice de força.  
df.sort_values(by = "Índice de força", ascending =  
False)
```



PROVÁVEL SAÍDA DE INFORMAÇÕES (OS DADOS ESTÃO ONLINE E PODEM SOFRER ALTERAÇÕES)

Unnamed: 0	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legend	
424	NaN	383	GroudonPrimal Groudon	Ground	Fire	770	100	180	160	150	90	90	3	Tr
333	NaN	306	AggronMega Aggron	Steel	NaN	630	70	140	230	60	80	50	3	Fa
163	NaN	150	MewtwoMega Mewtwo X	Psychic	Fighting	780	106	190	100	154	100	130	1	Tr
413	NaN	376	MetagrossMega Metagross	Steel	Psychic	700	80	145	150	105	110	110	3	Fa
409	NaN	373	SalamenceMega Salamence	Dragon	Flying	700	95	145	130	120	90	120	3	Fa
...	
261	NaN	242	Blissey	Normal	NaN	540	255	10	10	75	135	55	2	Fa
187	NaN	173	Cleffa	Fairy	NaN	218	50	25	28	45	55	15	2	Fa
188	NaN	174	Igglybuff	Normal	Fairy	210	90	30	15	40	20	15	2	Fa
121	NaN	113	Chansey	Normal	NaN	450	250	5	5	35	105	50	1	Fa
488	NaN	440	Happiny	Normal	NaN	220	100	5	5	15	65	30	4	Fa
800 rows × 15 columns														



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

A linha de comandos DEVERÁ SER DIGITADA **SEM USAR O ENTER**

```
#Utilizando o método iloc para selecionar uma linha específica:
```

```
# criando um DataFrame
```

```
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6],  
                   'C': [7, 8, 9]})
```



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
# selecionando a segunda linha (índice 1) usando o  
método iloc  
linha = df.iloc[1]
```



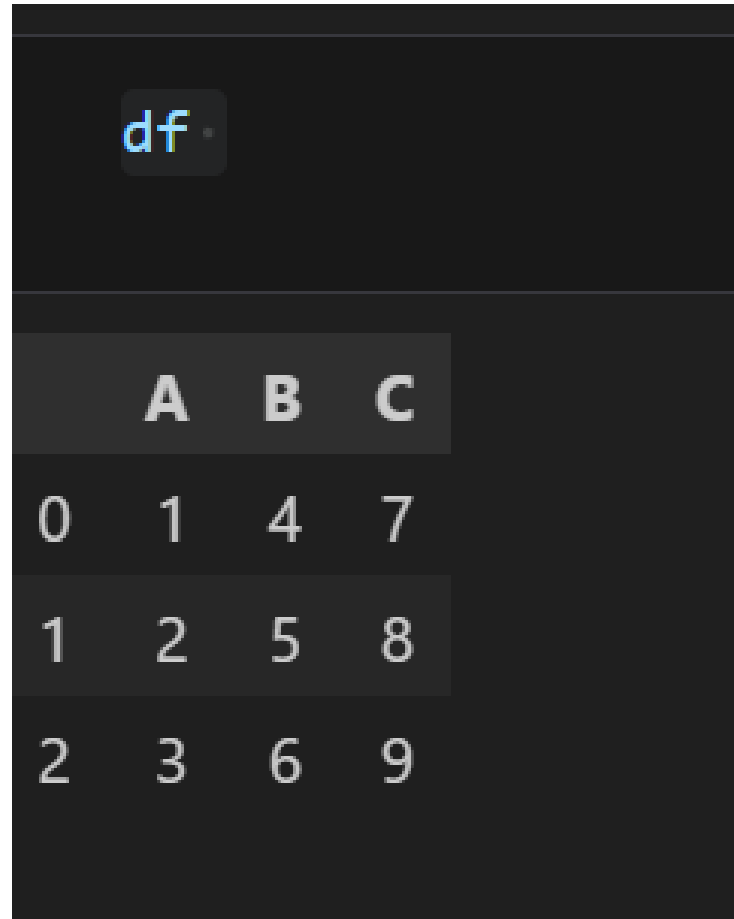
UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
#Digite a variável usada anteriormente, para exibir  
o resultado dos filtros
```

```
df
```



PROVÁVEL SAÍDA DE INFORMAÇÕES (OS DADOS ESTÃO ONLINE E PODEM SOFRER ALTERAÇÕES)



```
df
```

	A	B	C
0	1	4	7
1	2	5	8
2	3	6	9



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
print(linha)
```

SAÍDA

```
A    2  
B    5  
C    8  
Name: 1, dtype: int64
```



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
# selecionando as linhas 0 e 2 usando o método iloc  
linhas = df.iloc[[0, 2]]  
  
print(linhas)
```



PROVÁVEL SAÍDA DE INFORMAÇÕES (OS DADOS ESTÃO ONLINE E PODEM SOFRER ALTERAÇÕES)

	A	B	C
0	1	4	7
2	3	6	9

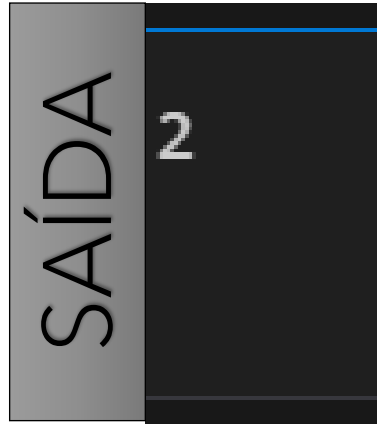


UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
# selecionando a célula da coluna 'A' e linha 1  
usando o método iloc (linha, coluna)  
valor = df.iloc[1, 0]  
  
print(valor)
```



PROVÁVEL SAÍDA DE INFORMAÇÕES (OS DADOS ESTÃO ONLINE E PODEM SOFRER ALTERAÇÕES)



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
#Utilizando o método loc para selecionar uma linha específica:
```

```
# criando um DataFrame com índices  
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6],  
                   'C': [7, 8, 9]}, index=['a', 'b', 'c'])
```



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
# selecionando a linha de índice 'b' usando o método  
loc  
linha = df.loc['b']  
  
print(linha)
```



PROVÁVEL SAÍDA DE INFORMAÇÕES (OS DADOS ESTÃO ONLINE E PODEM SOFRER ALTERAÇÕES)

SAÍDA	A	2
	B	5
	C	8
	Name: b, dtype: int64	



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

df

SAÍDA	A	B	C	
	a	1	4	7
	b	2	5	8
	c	3	6	9



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
# selecionando as linhas de índice 'a' e 'c' usando  
o método loc  
linhas = df.loc[['a', 'c']]  
  
print(linhas)
```



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

SAÍDA	A	B	C	
	a	1	4	7
	c	3	6	9



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

```
# selecionando as linhas onde o valor da coluna 'A'
# é maior que 1 usando o método loc
linhas = df.loc[df['A'] > 1]

print(linhas)
```



UMA VEZ QUE OS CRITÉRIOS FORAM DEFINIDOS, AS PRÓXIMAS ETAPAS, SERVIRÃO PARA MOSTRAMOS O QUE EXISTEM NA TABELA, DE ACORDO, COM A SELEÇÃO FEITA.

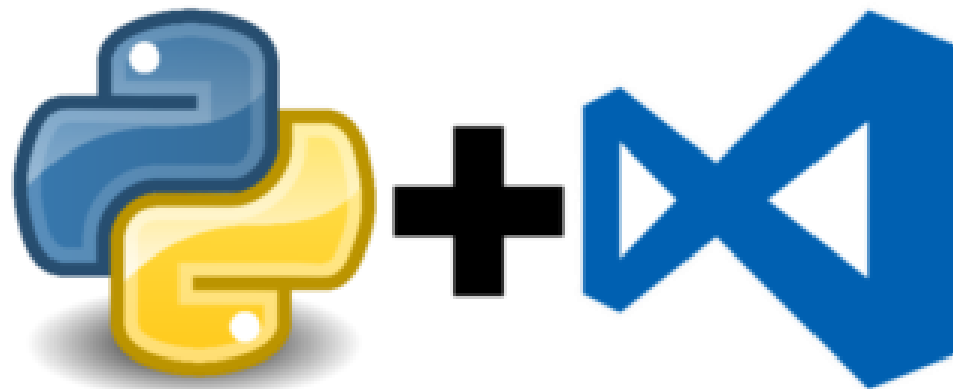
SAÍDA		A	B	C
	b	2	5	8
	c	3	6	9



TRABALHANDO COM PANDAS

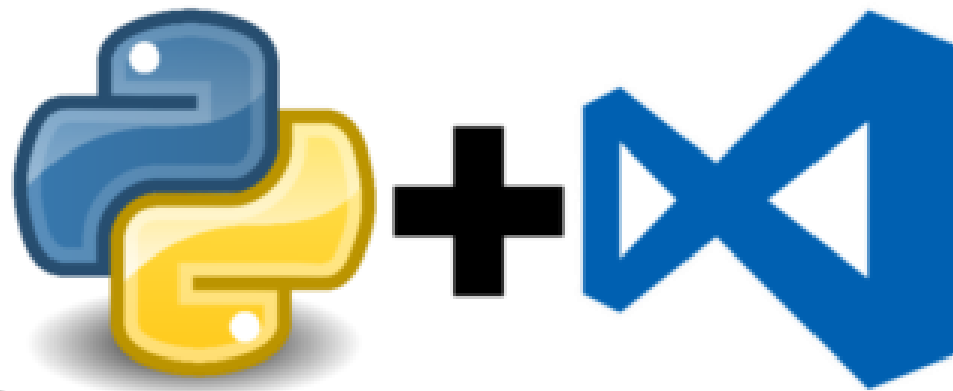
df

	Nome	Idade	Sexo
0	João	21	Masculino
1	Maria	18	Feminino
2	Pedro	34	Masculino
3	Joana	25	Feminino



TRABALHANDO COM PANDAS

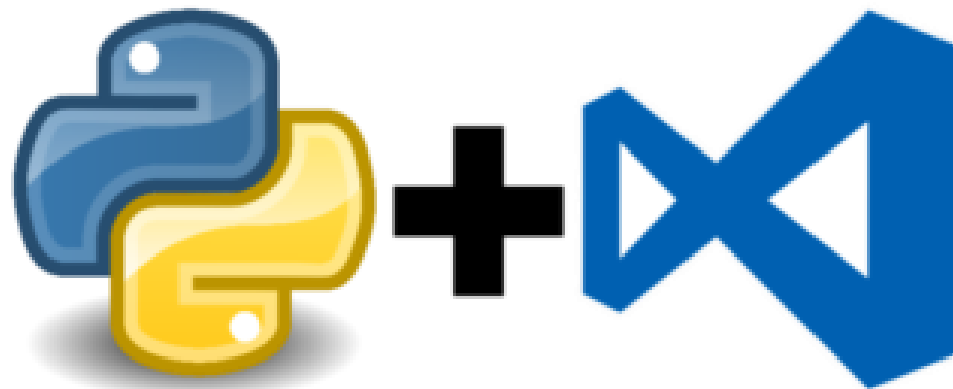
```
# lendo um arquivo Excel  
df = pd.read_excel('meu_arquivo.xlsx')
```



TRABALHANDO COM PANDAS

df

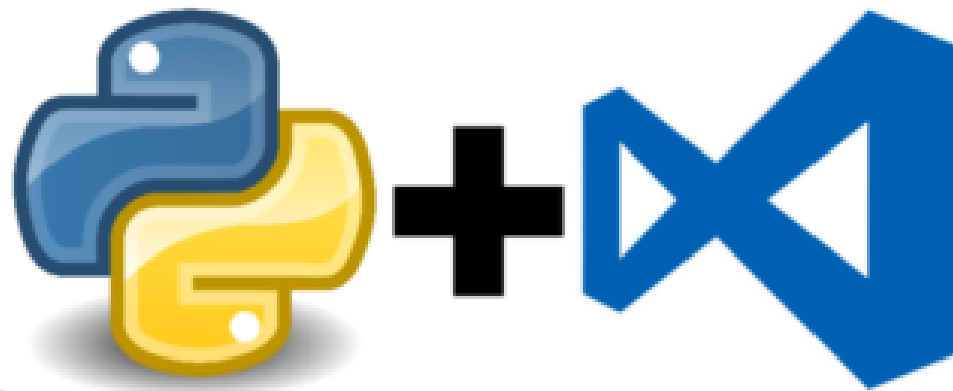
	Nome	Idade	Sexo
0	João	21	Masculino
1	Maria	18	Feminino
2	Pedro	34	Masculino
3	Joana	25	Feminino



TRABALHANDO COM PANDAS

```
# Lendo arquivos HTML
```

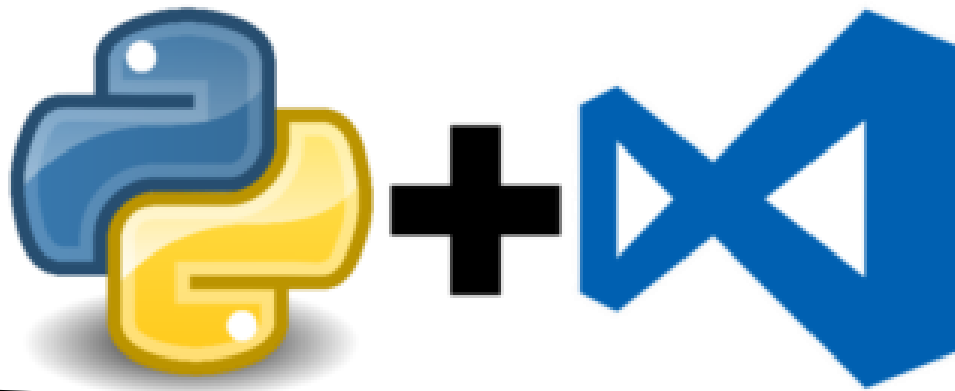
```
tabela_wiki = pd.read_html('https://en.wikipedia.org/wiki/Minnesota')
```



TRABALHANDO COM PANDAS

```
# Lendo arquivos HTML
```

```
tabela_wiki_temperatura =  
pd.read_html('https://en.wikipedia.org/wiki/Minnesota', match='Average  
daily maximum and minimum temperatures for selected cities in  
Minnesota')
```

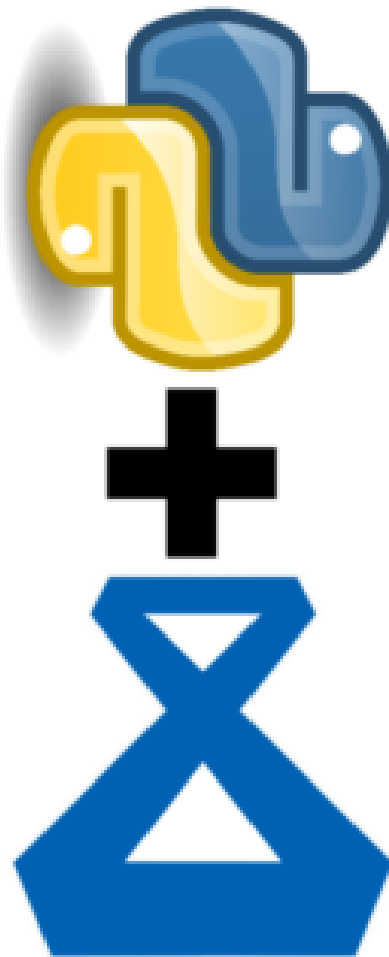


TRABALHANDO COM PANDAS

```
tabela_wiki_temperatura[0]
```

SAÍDA

	Location	July (°F)	July (°C)	January (°F)	January (°C)
0	Minneapolis	83/64	28/18	23/7	-4/-13
1	Saint Paul	83/63	28/17	23/6	-5/-14
2	Rochester	82/63	28/17	23/3	-5/-16
3	Duluth	76/55	24/13	19/1	-7/-17
4	St. Cloud	81/58	27/14	18/-1	-7/-18
5	Mankato	86/62	30/16	23/3	-5/-16
6	International Falls	77/52	25/11	15/-6	-9/-21



TRABALHANDO COM PANDAS

```
#Criando um DataFrame a partir de uma lista
```

```
# criando uma lista de dados
```

```
dados = [[1, 'João', 'São Paulo'], [2, 'Maria', 'Rio de Janeiro'], [3,  
'José', 'Belo Horizonte']]
```

```
# criando um DataFrame a partir da lista
```

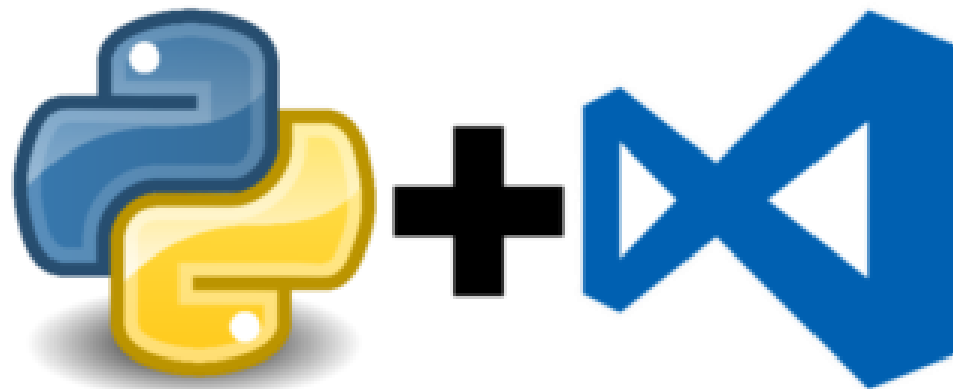
```
df = pd.DataFrame(dados, columns=['id', 'nome', 'cidade'])
```



TRABALHANDO COM PANDAS

df

	id	nome	cidade
0	1	João	São Paulo
1	2	Maria	Rio de Janeiro
2	3	José	Belo Horizonte



TRABALHANDO COM PANDAS

```
#Criando um DataFrame a partir de um dicionário:
```

```
# criando um dicionário de dados
```

```
dados = {'id': [1, 2, 3], 'nome': ['João', 'Maria', 'José'], 'cidade':  
['São Paulo', 'Rio de Janeiro', 'Belo Horizonte']}
```

```
# criando um DataFrame a partir do dicionário
```

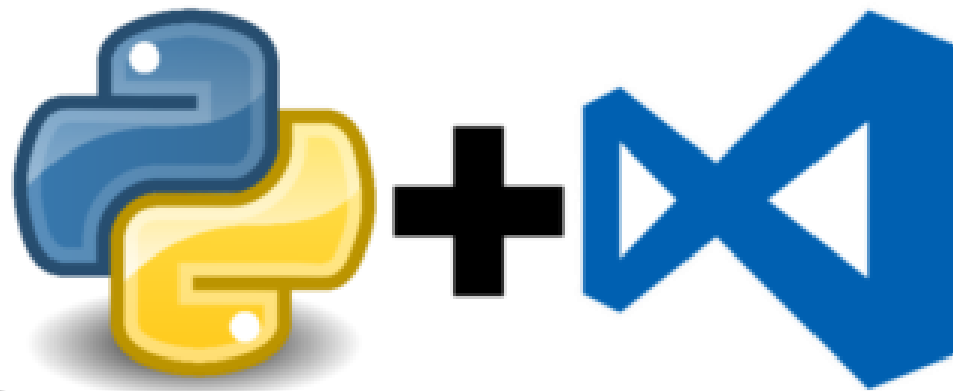
```
df = pd.DataFrame(dados)
```



TRABALHANDO COM PANDAS

```
df.head()
```

	id	nome	cidade
0	1	João	São Paulo
1	2	Maria	Rio de Janeiro
2	3	José	Belo Horizonte



TRABALHANDO COM PANDAS

```
#Criando um DataFrame a partir de uma lista de dicionários:
```

```
# criando uma lista de dicionários
```

```
dados = [{'id': 1, 'nome': 'João', 'cidade': 'São Paulo'}, {'id': 2,  
'nome': 'Maria', 'cidade': 'Rio de Janeiro'}, {'id': 3, 'nome': 'José',  
'cidade': 'Belo Horizonte'}]
```

```
# criando um DataFrame a partir da lista de dicionários
```

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



TRABALHANDO COM PANDAS

```
df.head()
```

	id	nome	cidade
0	1	João	São Paulo
1	2	Maria	Rio de Janeiro
2	3	José	Belo Horizonte

