## **Para carregar uma base de dados usando pandas devemos dar o comando**

## **pd.read\_csv("local do arquivo.csv")**

## **Dessa maneira estará sendo lido um arquivo csv da cotação do Bitcoin em relação ao Dola BTU-USD que foi extraída do** [**Yahoofinance**](https://finance.yahoo.com/quote/BTC-USD?p=BTC-USD&.tsrc=fin-srch) **e salva como df.**

## **No “local\_do\_arquivo.csv” colocamos o caminho da base de dados que pode ser local ou na web.**

## **#carregar dados**

## **df = pd.read\_csv("https://finance.yahoo.com/quote/BTC-USD/history/") (ou copiar do drive do Prof. Sovat)**

## **print(df)**

## **como instalar pandas python**

## **Visualizando os dados**

## **Antes de começarmos a trabalhar com nossos dados primeiro temos que entender como eles estão se comportando e com o que estamos trabalhando.**

## **O primeiro comando que vamos observar é df.head(). Com ele conseguiremos mostrar um determinado numero de linhas.**

## **Se executamos esse codigo será exibido as 5 primeiras linhas da base de dados.**

## **#Mostrar os 5 primeiros linhas de dados**

## **df.head()**

## **biblioteca pandas python**

## **#Mostrar os 7 primeiras linhas dados**

## **df.head(7)**

## **vizualizando dados com pandas**

## **Com o comando df.tail() ele mostra as 5 ultimas linhas da nossa base de dados. Caso queira um número maior basta fazer como no exemplo acima e indicar a quantidade**

## **# Mostrar os 5 ultimos**

## **df.tail()**

## **vizualização de dados com python**

## **O comando df.shape, como o proprio nome diz, nos mostra a forma da nossa base de dados indicando o numero de linhas e colunas (linha,coluna).**

## **# mostrar o tamanho dos dados (linhas,colunas)**

## **df.shape**

## **(367, 7)**

## **Outro comnando muito interessante é o df.columns ele nos mostra o nome das colunas da nossa base de dados.**

## **#mostrar os nomes das colunas**

## **df.columns**

## **Index([‘Date’, ‘Open’, ‘High’, ‘Low’, ‘Close’, ‘Adj Close’, ‘Volume’], dtype=’object’)**

## **Neste ponto mudaremos o nome das colunas e para isso usaremos comando o df.columns e igualar aos nomes escolhidos respeitando a ordem.**

## **#Mudar o nome das colunas**

## **df.columns = ['Data','Abertura','Máxima','Mínima','Fechamento','adjFechamento','Volume']**

## **#mostrar os nomes das colunas**

## **df.columns**

## **Index([‘Data’, ‘Abertura’, ‘Máxima’, ‘Mínima’, ‘Fechamento’, ‘adjFechamento’, ‘Volume’], dtype=’object’)**

## **Trabalhando com as colunas**

## **O primeiro comando que iremos estudar seleciona uma coluna especifica. Para isso basta executar o comando df["nome\_da\_coluna'].**

## **#selecionar uma coluna**

## **df['Data']**

## **analise de dados com python e pandas**

## **Então com esse comando conseguimos observar todos os dados daquela coluna.**

## **Vamos imaginar que não seja necessário todos os valores e sim um intervalo especifico:**

## **df['Data'][10:20]**

## **python para analise de dados**

## **Outro comando é o df.iloc[:,"numero\_da\_coluna"], com o iloc colocamos primeiro os “:” para selecionar todas as linhas e depois colocamos o numero da coluna que queremos trabalhar.**

## **# usando o iloc**

## **df.iloc[:,0]**

## **criar dataframes pandas**

## **Caso queria selecionar uma linha em especifico de uma coluna basta colocar no lugar dos “:” anterior.**

## **# usando o iloc**

## **df.iloc[:,0]**

## **‘2019-07-01’**

## **Nesse caso selecionamos a primeira linha da primeira coluna (lembrando que a contagem na linguagem Python começa com 0).**

## 

## **Para retirarmos uma coluna basta usar o comando df.drop(columns=["nome\_da\_coluna], inplace=True).**

## **Nele colocamos .drop e o nome da coluna que queremos retirar da base de dados, então primeiro vamos vasculhar a base original.**

## **df.head()**

## **trabalhando com as colunas**

## **Agora com o comando, o inplace = True é somente para não precisarmos redefinir o df.**

## **# RETIRAR A COLUNA**

## **df.drop(columns=['Abertura'], inplace = True)**

## **print(df.head())**

## **tratando as colunas com dados**

## **Outro recurso muito utíl é o calculo de elementos estáticos que se torna muito util com a biblioteca Pandas.**

## 

## **Para calcular a média de uma coluna com Pandas basta usar o comando que foi falado para selecionar uma coluna df["nome\_da\_coluna"] e colocar um complemanto para calcular a média df["nome\_da\_coluna"].mean().**

## **# Calculando a média dos valores de fechamendo**

## **df['Fechamento'].mean()**

## **8836.654677392371**

## **Caso queria outra função estatistica como a mediana basta seguir a mesma logica so que nesse casso usamos .median()**

## **df['Fechamento'].median()**

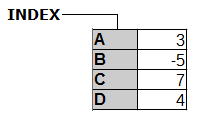
## **8942.808594**

## 

## **Series e DataFrame**

Posso estar sendo meio óbvio falando sobre Series e DataFrame para alguém que já está acostumado a usar o Pandão, mas quero deixar claro para aqueles que estão começando a principal diferença entre esses dois tipos de Estrutura de Dados.

* **Series** nada mais é que um array de 1 dimensão. Você pode considerar um Series também como uma coluna de uma tabela. Exemplo:  
  >>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])



Saída do código acima: um array de valores indexados

* Um **DataFrame** é simplesmente um conjunto de Series. Trata-se de uma estrutura de dados de 2 dimensões — colunas e linhas — que transforma os dados em uma bela tabela. Exemplo:  
  **#Criando um dicionário onde cada chave será uma coluna do DataFrame** >>> data = {  
  'País': ['Bélgica', 'Índia', 'Brasil'],  
  'Capital': ['Bruxelas', 'Nova Delhi', 'Brasília'],  
  'População': [123465, 456789, 987654]  
  }  
  **#Criando o DataFrame**>>> df = pd.DataFrame(data, columns=['País','Capital','População'])



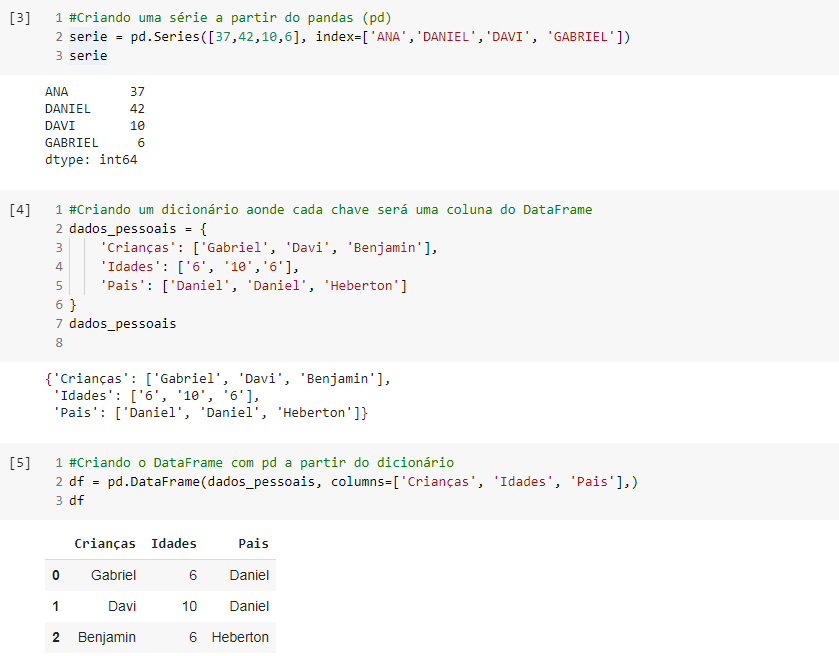
Saída do código: um lindo DataFrame

## **Agora sim! Confira os melhores comandos para utilizar no Pandas**

Sugiro a você salvar esse artigo em seus favoritos para futuras consultas. Para deixar sua leitura mais agradável, prepare-se para enxurrada de GIFs!

* Abrindo e escrevendo arquivos CSV:  
  **#Para ler arquivos CSV codificados em ISO**>>> pd.read\_csv('nome\_do\_arquivo.csv', encoding='ISO-8859-1')  
  **#Para escrever arquivos CSV**>>> pd.to\_csv('nome\_do\_arquivo\_para\_salvar.csv')
* Abrindo arquivos de Excel:  
  >>> xlsx = pd.ExcelFile('seu\_arquivo\_excel.xlsx')  
  >>> df = pd.read\_excel(xlsx, 'Planilha 1')
* Removendo linhas e colunas:  
   **#Removendo linhas pelo index**s.drop([0, 1])  
   **#Removendo colunas utilizando o argumento axis=1**df.drop('País', axis=1)
* Coletando informações básicas sobre o DataFrame:  
   **#Quantidade de linhas e colunas do DataFrame**>>> df.shape  
  **#Descrição do Index**>>> df.index  
   **#Colunas presentes no DataFrame**>>> df.columns  
   **#Contagem de dados não-nulos**>>> df.count()
* Criando uma nova coluna em um DataFrame:  
   >>> df['Nova Coluna'] = 0
* Renomeando colunas de um DataFrame:  
   **#Se seu DataFrame possui 3 colunas, passe 3 novos valores em uma lista**df.columns = ['Coluna 1', 'Coluna 2', 'Coluna 3']
* Resumo dos dados:  
   **#Soma dos valores de um DataFrame**>>> df.sum()  
  **#Menor valor de um DataFrame**>>> df.min()  
  **#Maior valor**>>> df.max()  
  **#Index do menor valor**>>> df.idmin()  
  **#Index do maior valor**>>> df.idmax()  
  **#Resumo estatístico do DataFrame, com quartis, mediana, etc.**>>> df.describe()  
  **#Média dos valores**>>> df.mean()  
  **#Mediana dos valores**>>> df.median()
* Aplicando funções:  
   **#Aplicando uma função que substitui a por b**df.apply(lambda x: x.replace('a', 'b'))
* Ordenando valores:  
  **#Ordenando em ordem crescente**df.sort\_values()  
  **#Ordenando em ordem decrescente**df.sort\_values(ascending=False)
* Operações aritméticas em Series:  
   >>> s = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e'])  
  **#Somando todos os valores presentes na Series por 2**>>> s.add(2)  
  **#Subtraindo 2 de todos os valores**>>> s.sub(2)  
  **#Multiplicando todos os valores por 2**>>> s.mul(2)  
  **#Dividindo valores por 2**>>> s.div(2)
* Indexação por Boolean:  
  **#Filtrando o DataFrame para mostrar apenas valores pares**df[df['População'] % 2 == 0]
* Selecionando valores:  
   **#Selecionando a primeira linha da coluna país**df.loc[0, 'País']

Outros exemplos



**Mais comandos no Pandas**

Agora que você já aprender a importar o Pandas, sabe a principal diferença entre Séries e DataFrame e aprendeu a criá-los, vamos avançar para mais comandos?

Sugiro salvar, copiar para consultas futuras, tenho certeza que será útil para seu aprendizado.

Let's go!

*Importar Arquivos:*

Nos tempos atuais muitos dados são gerados todo o tempo. E diante dessa realidade deparamos com todo tipo de dado, estrutura, não estrutura e semi- estruturado (assunto para um próximo artigo).

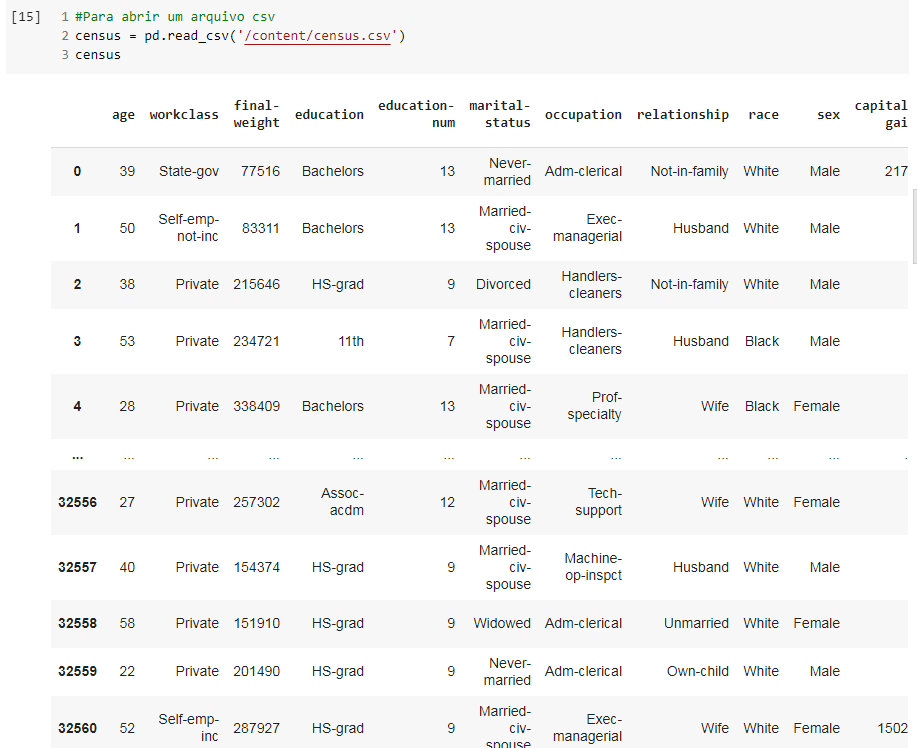
Mas então quais tipos podemos importar para dentro do nosso ambiente de trabalho afim de analisar, extrair, modificar, alterar e explorar para ficarmos apenas com os que nos interessa para responder determinada questão?

Veja alguns:

* CSV;
* Excel;
* txt;
* pdf

e outros.

Abaixo veremos alguns comandos para importar excel e csv:



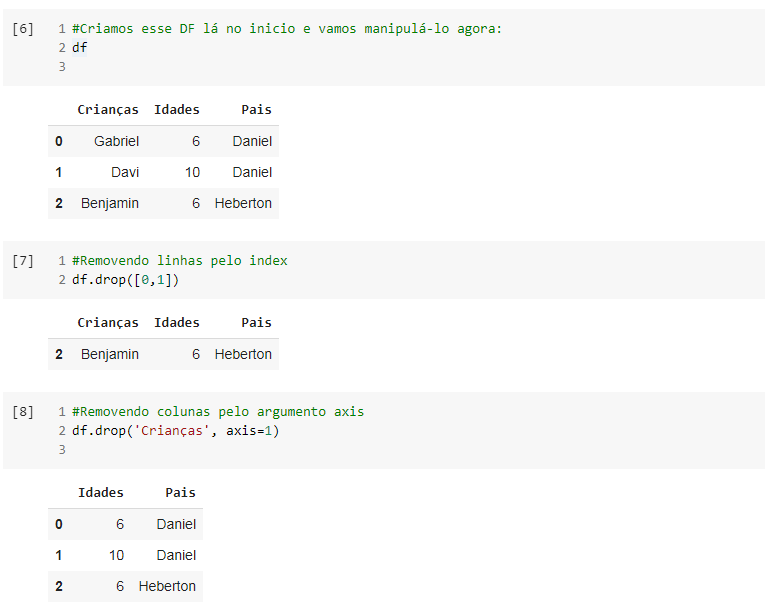


**Métodos adicionais de leitura**

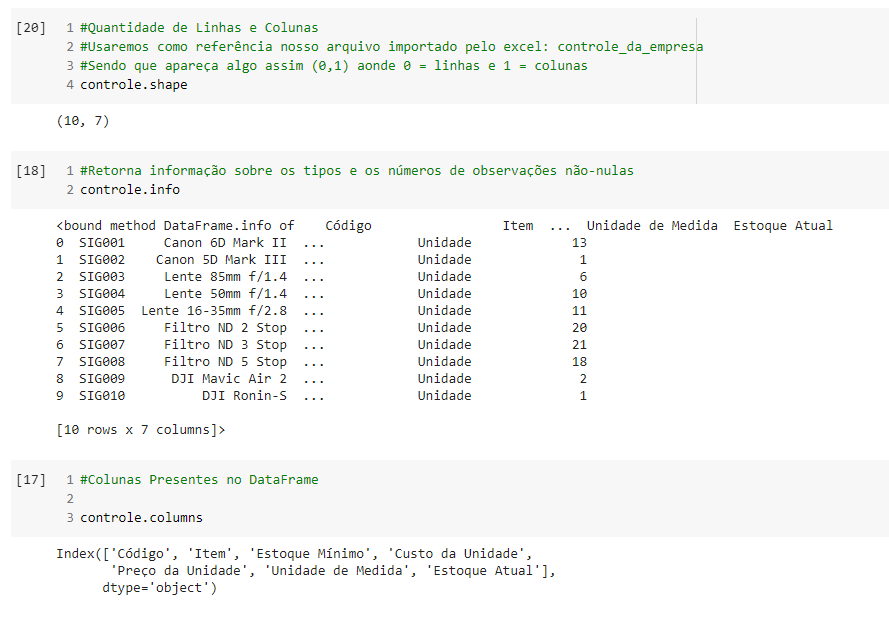
Raramente uso esse métodos, mas também estão implementados no Pandas:

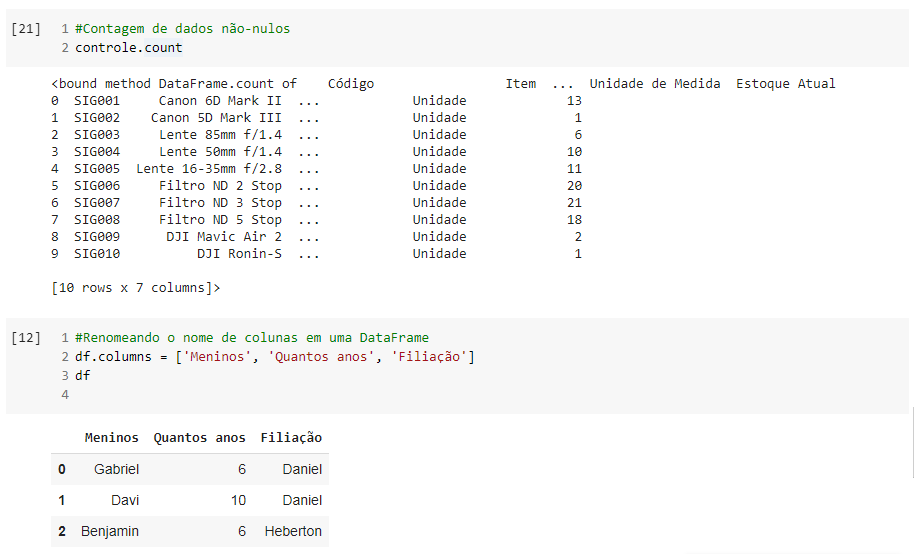
* read\_feather
* read\_fwf
* read\_gbq
* read\_hdf
* read\_html
* read\_json
* read\_msgpack
* read\_parquet
* read\_pickle
* read\_sas
* read\_sql
* read\_sql\_query
* read\_sql\_table
* read\_stata
* read\_table

*Manipulando os Arquivos*: Removendo Linhas e Colunas.



*E agora que tal entendermos como coletar informações relevantes sobre o nosso DataFrame:*

**

**

## **Estruturas de dados no Pandas**

### **Séries**

Uma matriz uni-dimensional capaz de conter qualquer tipo de dado

>>> s = pd.Series([3, -5, 7, 4], index=[‘a’, ‘b’, ‘c’, ‘d’])

### **Frame de dados (Data frame)**

Uma estrutura de dados bi-dimensional com colunas de diferentes tipos

>>> data = {‘País’: [‘Portugal’, ‘Peru’, ‘Chile’], ‘Capital’: [‘Lisboa’, ‘Lima’, ‘Santiago’], ‘População’: [751000, 1120000, 695000]}

>>> df = pd.DataFrame(data, columns=[‘País’, ‘Capital’, ‘População’])

## **Pedindo ajuda**

>>> help(pd.Series.loc)

## **Selecionando**

### **Recuperar elementos**

#Retorna um elemento

>>>s[‘b’]

-5

#Retorna um subconjunto de um DataFrame

>>>df[1:]

País Capital População

1 Portugal Lisboa 751000

2 Peru Lima 1120000

### **Seleção, indexação booleana & definição**

#Por posição

#Seleciona um único valor por linha & coluna

>>>df.iloc[0][0]

‘Portugal’

>>>df.at([0], [0])

‘Portugal’

#Por rótulo

#Seleciona um único valor por rótulo de linha & coluna

>>>df.iloc[0][‘País’]

‘Portugal’

>>>df.at([0], [‘País’])

‘Portugal’

#Por rótulo/posição

#Seleciona uma única linha de um subconjunto de linhas

>>>df.ix[2]

País Chile

Capital Santiago

População 695000

#Seleciona uma única coluna de um subconjunto de colunas

>>>df.ix[:, ‘Capital’]

0 Lisboa

1 Lima

2 Santiago

#Seleciona linhas e colunas

>>>df.ix[1, ‘Capital’]

'Lima'

#Indexação booleana

#Série s, valor não é > 1

>>>s[~(s > 1)]

#s, valor é < -1 ou >= 6

>>>s[(s < -1) | (s >= 6)]

#Filtro para ajustar o DataFrame

>>>df[df[‘População’] > 1000000]

#Definição

#Define índice a de s = 6

>>>s[‘a’] = 6

## **I/O**

### **Ler e escrever em CSV**

>>> pd.read\_csv(‘arquivo.csv’, header=None, nrows=5)

>>> pd.to\_csv(‘meudf.csv’)

### **Ler e escrever em Excel**

>>> pd.read\_excel(‘arquivo.xls’, sheetname=‘Pasta1’)

>>> pd.to\_excel(‘meudf.csv’, sheetname=‘Pasta1’)

### **Ler múltiplas pastas de um mesmo arquivo**

>>> xlsx = pd.ExcelFile(arquivo.xls’)

>>> df = pd.read\_excel(xlsx, ‘Pasta1’)

### **Ler e escrever query SQL ou tabela de dados**

>>> from sqlalchemy import create\_engine

>>> engine = create\_engine(‘sqlite:///:memory:’)

>>> pd.read\_sql(“SELECT \* FROM m\_tabela;”, engine)

>>> pd.read\_sql\_table(‘m\_tabela’, engine)

>>> pd.read\_sql\_query(“SELECT \* FROM m\_tabela;”, engine)

#read\_sql() é um wrapper conveniente ao redor de read\_sql\_table() e de read\_sql\_query()

>>> pd.to\_sql(meuDf’, engine)

## **Deletando**

#Deleta valores das linhas

>>> s.drop([‘a’, ‘c’])

#Deleta valores das colunas

>>> s.drop(‘País’, axis=1)

## **Ordem e classificação**

#Ordena por linha ou coluna

>>> df.sort\_index(by=’País’)

#Ordena por valores

>>> s.order()

#Classifica as entradas

>>> df.rank()

## **Recuperando informações de séries e dataframes**

### **Informações básicas**

#(linhas, colunas)

>>> df.shape

#Descreve o índice

>>> df.index

#Descreve as colunas

>>> df.columns

#Info sobre o dataframe

>>> df.info()

#Número de valores não-NA

>>> df.count()

### **Resumos**

#Soma de valores

>>> df.sum()

#Soma acumulada

>>> df.cumsum()

#Valores max e min

>>>df.min/df.max()

#Sumário estatístico

>>>df.describe()

#Média dos valores

>>>df.mean()

#Mediana dos valores

>>>df.median()

## **Aplicando funções**

>>> f = lambda x: x\*2

#Aplica a função

>>> df.apply(f)

#Aplica a função por elemento

>>> df.applymap(f)

## **Alinhamento de dados**

### **Alinhamento interno de dados**

>>> s3 = pd.Series([7, -2, 3], index=[‘a’, ’c’, ’d’])

>>> s + s3

a 10.0

b NaN

c 5.0

d 7.0

### **Operações aritméticas com métodos de preenchimento**

#É possível alinhar dados usando métodos de preenchimento.

>>> s.add(s3, fill\_value = 0)

a 10.0

b -5.0

c 5.0

d 7.0

>>> s.sub(s3, fill\_value = 2)

>>> s.div(s3, fill\_value = 4)

>>> s.mul(s3, fill\_value = 3)