



Trilha Ciência de dados com Python

Aula 3?



Faísca

Pandas

Séries e Dataframes

V

Opa esses não!

as

GIFS.ICANHASCHEEZBURGER.COM



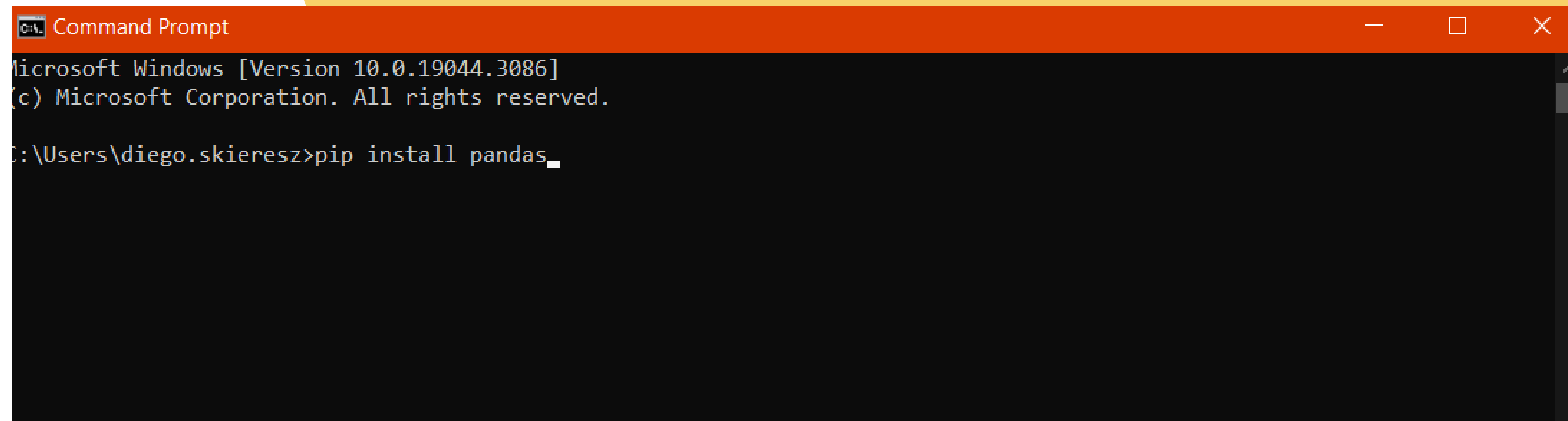
Esse

Pandas

Caso você utilize o **Anaconda**, o **Pandas** já vem instalado por padrão.

Para instalar o Pandas vamos no **terminal**

pip install pandas



```
Command Prompt
Microsoft Windows [Version 10.0.19044.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\diego.skieresz>pip install pandas_
```

Agora só precisamos abrir sua IDE favorita ou notebook **Jupyter** e importar a **biblioteca Pandas**!

O que são as Series?

Podemos ver uma **Serie** do Pandas como um arranjo unidimensional, como uma **Lista**.

Ao aplicar alguma função em uma Serie, ela é aplicada a todos os seus elementos.

Separando ela em 4 partes, temos:

- Dados que a compõe
- Tipo dos dados
- Índice que contém a referência para acessar os elementos
- Um nome

Imagine a série como uma coluna no **Excel**.

Series Parts

counts

name (optional)

index

0	145
1	142
2	38
3	13

values

Separando ela em 4 partes, temos:

- Dados que a compõe
- Tipo
- Índice
- parâmetro
- Um número

IMPORTANTE! Apesar de tecnicamente possível, misturar diferentes tipos de dados numa mesma série isso pode afetar negativamente o desempenho de algumas operações ou análises.

index



1	142
2	38
3	13



values

Saber o tipo de dado é crucial

```
import pandas as pd
```

```
# Criando uma série com tipos diferentes
```

```
data = [1, 2, 'three', 4.0]
```

```
series = pd.Series(data)
```

```
# Exibindo a série e o tipo de dado
```

```
print(series)
```

```
print(f"Tipo da série: {series.dtype}")
```

```
# Converte todos os valores para o tipo "object".
```

saída:

0 1

1 2

2 three

3 4.0

dtype: object

Tipo da série: object

Criando séries com tipos homogêneos

```
integers = pd.Series([1, 2, 3])
```

```
floats = pd.Series([1.0, 2.0, 3.0])
```

Exibindo as séries e os tipos de dados

```
print("Série de inteiros:")
```

```
print(integers)
```

```
print(f"Tipo da série de inteiros: {integers.dtype}")
```

```
print("\nSérie de floats:")
```

```
print(floats)
```

```
print(f"Tipo da série de floats: {floats.dtype}")
```

saída

Série de inteiros:

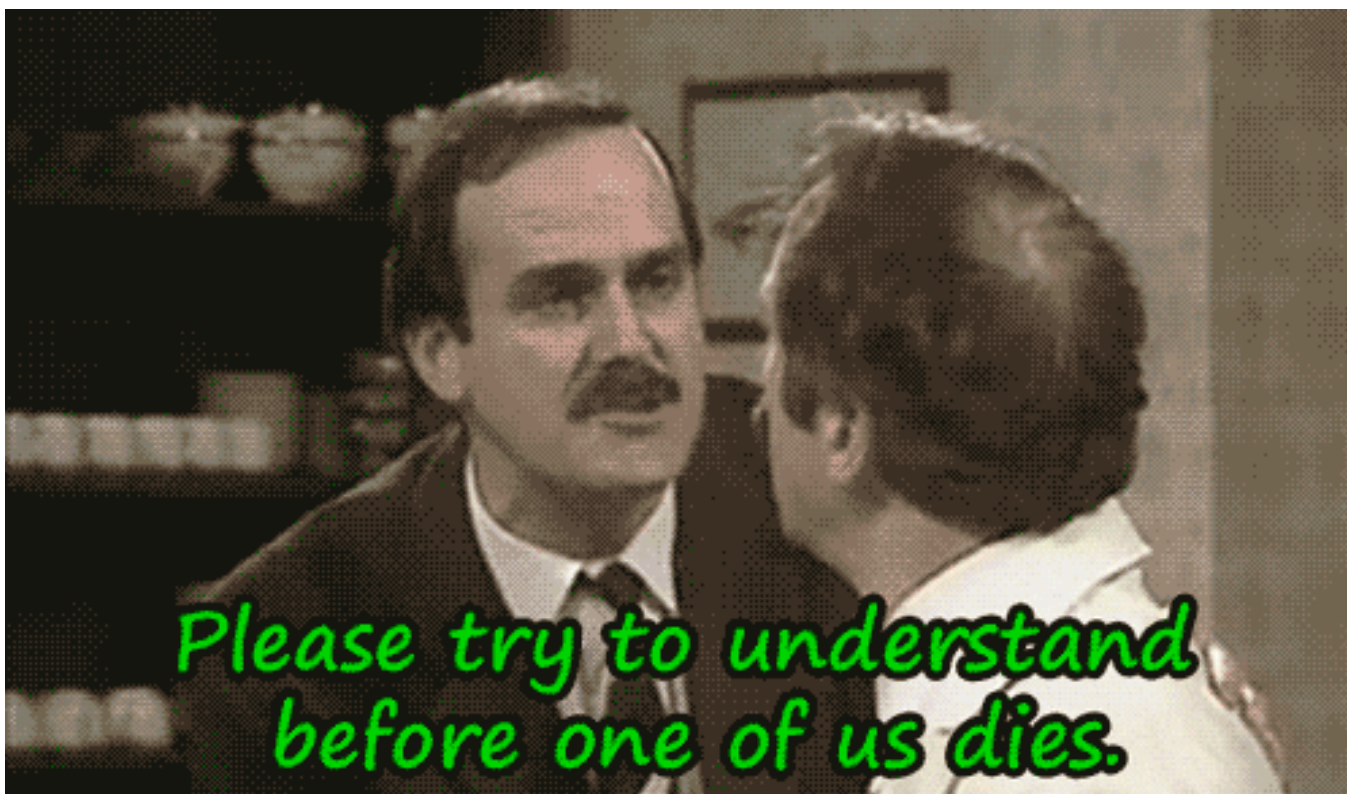
0 1

1 2

2 3

dtype: int64

Tipo da série de
inteiros: int64



No Python acessamos listas usando

✦ `lista[0]`, `lista[1]`, ...

Na série é só usar

✦ `serie[0]`, `serie[1]`, ...

Acessando os dados



Gian

1 second ago

Entretanto, não precisa ser assim, podemos criar um índice próprio que nem precisa ser numérico! <0>



REPLY

Nós mesmos podemos criar os índices do jeito que bem entendermos: **números**, **strings**, **tuplas**.

```
import pandas as pd
```

```
# Criando uma série com índices nomeados
```

```
dados = pd.Series([10, 20, 30, 40, 50], index=['a', 'b', 'c', 'd', 'e'])
```

```
print(dados)
```

```
# Acessando o terceiro elemento (posição 2)
```

```
terceiro_elemento = dados.iloc[2]
```

```
print(terceiro_elemento)
```

```
# Acessando o elemento com índice 'd'
```

```
elemento_c = dados['d']
```

```
print(elemento_c)
```



Fatiamento por posição (do índice 1 ao 3)

```
slice_posicional = dados.iloc[1:4]  
print(slice_posicional)
```

Fatiamento por índices nomeados (do índice 'b' ao 'd')

```
slice_nomeado = dados['b':'d']  
print(slice_nomeado)
```

Acessando elementos maiores que 20

```
elementos_maiores_que_20 = dados[dados > 20]  
print(elementos_maiores_que_20)
```



Diego 1 second ago

Você pode usar o **slicing** para acessar uma faixa de elementos =D



REPLY



Joaquim 1 second ago

Você pode acessar elementos que atendem a uma determinada condição.



REPLY

O que são Dataframes?

Dataframes

✦ Resumidamente, **Séries** são colunas e **Dataframes** são Tabelas!

Até a próxima aula...

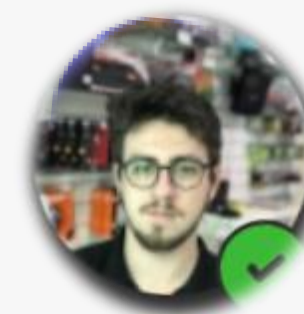
27/09/23

Brincadeira



Imagine uma
planilha no **Exce**

Dataframes



Joaquim 1 second ago

Criação: é simples só instanciar um objeto a partir da Classe
Dataframe `df = pd.DataFrame([2001, 1990, 65])`



REPLY

- ✦ Estrutura **bidimensional** de dados, ou seja, possui o formato de **linhas** e **colunas**. Cada linha representa um registro dos dados, e cada coluna seu atributo ou característica. Pense numa planilha turbinada!



Laura 1 second ago

No Tic temos mais exemplos! ;D



REPLY

Dados dos personagens

```
dados = {  
    'Nome': ['Luke Skywalker', 'Harry Potter', 'Tony Stark'],  
    'Universo': ['Star Wars', 'Harry Potter', 'Marvel'],  
    'Habilidade': ['Jedi', 'Magia', 'Tecnologia'],  
    'Idade': [53, 40, 48]  
}
```

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

```
print(personagens)
```

saida

	Nome	Universo	Habilidade	Idade
0	Luke Skywalker	Star Wars	Jedi	53
1	Harry Potter	Harry Potter	Magia	40
2	Tony Stark	Marvel	Tecnologia	48



Gian

1 second ago

Podemos criar o Dataframe usando um dicionário, onde o nome da coluna é a chave e valor é a lista de dados.



REPLY

Criar Variáveis é
uma boa prática!

```
print(personagens _info)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Nome        3 non-null      object
1   Universo    3 non-null      object
2   Habilidade  3 non-null      object
3   Idade        3 non-null      int64
dtypes: int64(1), object(3)
memory usage: 224.0+ bytes
None
```

- ◆ Possui 4 registros, representados por índices na faixa de 0 a 3
- ◆ Possui 4 colunas
- ◆ A coluna Nome, Universo e Habilidade são do tipo objeto e idade do tipo int64 (8 bytes)
- ◆ O tamanho em memória do DataFrame é de 224 bytes

IDs dos personagens

```
indices = ['linha1', 'linha2', 'linha3']
```

```
personagens = pd.DataFrame(dados,
index=indices)
```

```
print(personagens)
```


Acessando os dados

Uma das formas mais simples acessar todos os dados de uma coluna é usando o operador de índices.



✦ Operador de índices `personagens[coluna]`

```
>>> personagens['Habilidade']
```

```
linha1      'Jedi'
```

```
linha2      'Magia'
```

```
linha3      'Tecnologia'
```

```
Name: 'Habilidade' (%), dtype: object
```

Doidera isso, parece uma série! Não só parece como é uma série.


E se a gente quiser acessar múltiplas colunas (como um DataFrame)

✦ múltiplas colunas `personagens[['Nome', 'Idade']]`

```
print(personagens[['Nome', 'Idade']])
```

Resultado:

Nome	Idade
Skywalker Luke Skywalker	53
O Escolhido Harry Potter	40
Homem de Ferro Tony Stark	48



Gian

1 second ago

Usando o método nomeado "iloc" ele retorna uma Série com os valores da linha.

👍

👎

REPLY

E agora, para dificultar nossa vida como acessamos as linhas? Podemos usar os índices para nos ajudar!

```
print(personagens.iloc[0])
```

Resultado:

Nome	Luke Skywalker
Universo	Star Wars
Habilidade	Jedi
Idade	53
Name: Skywalker, dtype: object	

Acessando os dados

Alem disso, você pode recortar uma seção usando o slice ou fatiamento.

✦ `print(personagens.loc['linha2:'])`

Onde eu estou pegando todos da segunda linha em diante mas posso tb definir um final

✦ `print(personagens.loc['linha2':'linha3'])`



Joaquim1 1 second ago

Já deu para perceber que tudo que vale com as Séries vale com os dataframes,



REPLY

Também existem 2 funções bastante úteis para visualizar conjuntos de dados maiores, a `'head'`, que exibe os 5 primeiros elementos, e a `'tail'`, que exibe os 5 últimos.



Diego 1 second ago

É importante saber que quando eu estou usando como alvo um label eu utilizo o `"loc"` e quando estou usando um índice eu utilizo o `"iloc"`



REPLY

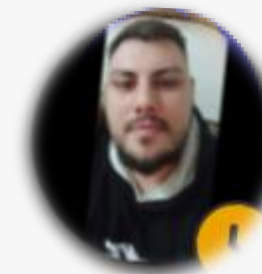
**E se a gente é meio doido e quer
adicionar uma nova coluna?**



Add nova coluna

Simple, criamos uma nova lista e atribuimos esses valores a ela.

```
# Adicionar uma nova coluna com as  
alianças dos personagens  
personagens['Aliança'] = ['Rebeldes',  
    'Ordem da Fênix', 'Vingadores']
```



Gian 1 second ago

E se quiser adicionar uma linha?.



REPLY

Basta usar o “loc”.

```
# Dados da nova personagem  
personagens.loc['linha4'] = ['Leia Organa',  
    'Star Wars', 'Liderança', 54, 'Rebeldes']
```




Foto de Gaelle Marcel,
disponível na Unsplash.
Editada pelo autor.

Mão Na Massa

Exercícios

- ✦ **Exercício 05** -Crie uma lista com 5 nomes de filmes que você goste.
- ✦ Crie uma 2ª lista com o ano de lançamento desses 5 filmes (pode pesquisar, eu deixo). Transforme isso em um dataframe;
- ✦ Adicione uma nova coluna com a nota, de 1 a 10, de quanto você gosta deles;



Exercícios

- ✦ Crie mais 2 dataframes, de séries e anime/desenhos que você gosta, com as mesmas colunas acima.
- ✦ Salve os 3 dataframes em 'csv', 'json' e 'html' e nos envie os arquivos.
- ✦ Concatene horizontalmente os 3 dataframes, ficando 9 colunas no final.



Exercícios

- ✦ Exercício 06 – *miniBoss* – *Coop* –
- ✦ Leia o arquivo do dataset em csv;
- ✦ Visualize as primeiras linhas do dataset;
- ✦ Visualize informações básicas do dataset, como: nomes das colunas, número de linhas, número de campos nulos, etc;



Exercícios

- ✦ Visualize informações sobre os campos numéricos do dataset, como: média, valor mínimo e máximo, etc;
- ✦ Remova informações nulas;
- ✦ Verifique se existem dados duplicados;
- ✦ Remova informações duplicadas;
- ✦ Remova as colunas 'listed_in' e 'duration'.

Análise de dados com Python

Módulo 1:

- ◆ Assistir Aula 4 Criação e Manipulação com Numpy
- ◆ Assistir Aula 5 Pandas Series e Dataframes

Módulo 2:

- ◆ Assistir Aula 1 Explorando Diferentes Formatos de Arquivos e Entrada/Saída de Dados.

Até a próxima aula...

27/09/23