DataFrames

cubos academy

DataFrames

Estrutura de dados extremamente utilizada por analistas

- Estrutura bidimensional
  - **linhas e colunas**
- Formato de tabela

Propriedade
(Coluna)

Registro
(Linha)

	Idade	Gênero	Peso	Altura
0	26	F	63	1.67
1	28	M	65	1.60

DataFrame criado no colab

pandas

Biblioteca mais importante para análise de dados em Python

- **Biblioteca:** conjunto de módulos e funções que facilitam a vida do programador
- Devem ser importadas para serem utilizadas
- No colab as principais bibliotecas já vêm instaladas, basta importar

```
import pandas as pd
```

Criando um DataFrame

A partir de um dicionário, as chaves serão as colunas e os valores as linhas.

```
dic = {'Idade' : [26, 28],
       'Gênero' : ['F', 'M'],
       'Peso' : [63, 65],
       'Altura' : [1.67, 1.60]}

data = pd.DataFrame(dic)
```

	Idade	Gênero	Peso	Altura
0	26	F	63	1.67
1	28	M	65	1.60

Criando um DataFrame

A partir de uma lista. Precisaremos também **definir** as **colunas**, seguindo o **padrão** dos dados na **lista**.

```
matrix = [[26,'F', 63, 1.67],[28,'M',65, 1.60]]
cols = ['Idade', 'Gênero', 'Peso', 'Altura']
data = pd.DataFrame(matrix, columns = cols)
```

	Idade	Gênero	Peso	Altura
0	26	F	63	1.67
1	28	M	65	1.60

Se localizando em um DataFrame

Um **DataFrame** é como uma **tabela**, para utilizar um valor armazenado, é necessário saber as suas **coordenadas**

```
lin = 1
col = 2

print(data.iloc[lin][col]) #65
print(data.loc[0]['Idade']) #26

print(data.iat[lin, col]) #65
print(data.at[0,'Idade']) #26
```

	0	1	2	3
	Idade	Gênero	Peso	Altura
0	26	F	63	1.67
1	28	M	65	1.60

1,2

Exercícios

cubos academy

O que podemos fazer com as tabelas?

Métodos em DataFrames

cubos academy

shape

Retorna as dimensões da tabela, ou seja, a **quantidade de linhas e colunas**, no formato **(linhas, colunas)**

	Idade	Gênero	Peso	Altura
0	26	F	63	1.67
1	28	M	65	1.60

```
data.shape
#(2, 4)
```

size

Retorna a **quantidade de elementos** ou células presentes na tabela

	Idade	Gênero	Peso	Altura
0	26	F	63	1.67
1	28	M	65	1.60

```
data.size
#8
```

len()

Retorna a **quantidade de linhas** da tabela

	Idade	Gênero	Peso	Altura
0	26	F	63	1.67
1	28	M	65	1.60

```
len(data)
#2
```

Exercícios

cubos academy

Quais informações
podemos ter da base?

Sumarizando DataFrames

cubos
// academy //

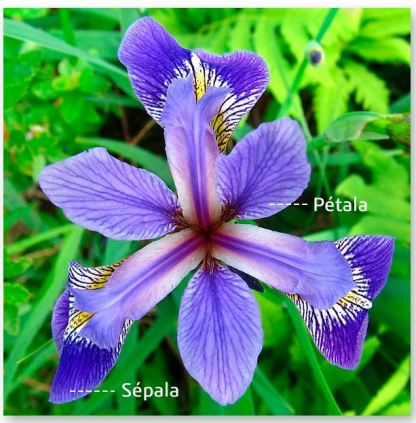
Iris Dataset

Base de dados das Flores de Íris

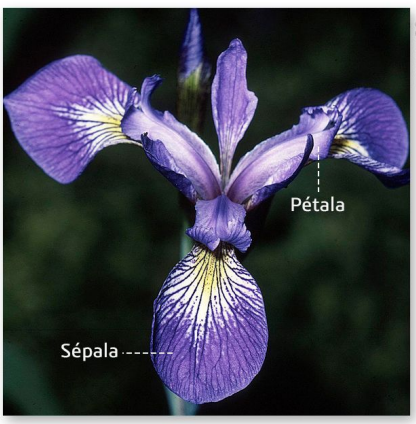
Iris flower dataset

Setosa | Versicolor | Virginica

Tiia Monto, CC BY-SA 4.0, via Wikimedia Commons



Charles de Mille-Isles from Mille-Isles, Canada, CC BY 2.0, via Wikimedia Commons



Robert H. Mohlenbrock. Courtesy of USDA NRCS, Public domain, via Wikimedia Commons

Sépala: Semelhantes a **folhas** que envolvem o botão da flor. Localizam-se em sua parte inferior. Elas se dobram e **protegem o botão** fechado contra o clima ou lesões durante o desenvolvimento.

Pétala: A principal tarefa das pétalas é **atrair** colibris e insetos para que possa ocorrer a **polinização**. As pétalas também protegem o estame e o pistilo, que são as partes das plantas necessárias para a reprodução.

Iris Dataset

```
import seaborn as sns

iris = sns.load_dataset('iris')
iris
```

Documentação Seaborn **Datasets Seaborn**

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

length: altura | **width:** largura

head()

Retorna as **primeiras 5 linhas** de um DataFrame. Usado para ter uma amostra do conteúdo, sem ter que carregar toda a base

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

***Caso queira mais de 5 linhas, adicionar o tamanho desejado entre os parênteses**

tail()

Retorna as **últimas 5 linhas** de um DataFrame. Usado para ter uma amostra do conteúdo, sem ter que carregar toda a base

```
iris.tail()
```

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

***Caso queira mais de 5 linhas, adicionar o tamanho desejado entre os parênteses**

sort_values()

Ordena o dataframe de acordo com **uma coluna** desejada. Podemos indicar se a ordem é crescente ou decrescente

```
iris.sort_values('sepal_length', ascending = False)
```

	sepal_length	sepal_width	petal_length	petal_width	species
131	7.9	3.8	6.4	2.0	virginica
135	7.7	3.0	6.1	2.3	virginica
122	7.7	2.8	6.7	2.0	virginica
117	7.7	3.8	6.7	2.2	virginica
118	7.7	2.6	6.9	2.3	virginica
...	...	...	...	...	...
41	4.5	2.3	1.3	0.3	setosa
42	4.4	3.2	1.3	0.2	setosa

dtypes

Retorna o **tipo** dos **dados** de **cada coluna** no dataframe

```
iris.dtypes

#sepal_length    float64
#sepal_width     float64
#petal_length    float64
#petal_width     float64
#species          object
#dtype: object
```

count()

Retorna a **quantidade** de observações **não-nulas** em cada **coluna**

```
iris.count()

#sepal_length    150
#sepal_width     150
#petal_length    150
#petal_width     150
#species         150
#dtype: int64
```

min()

Retorna o **valor mínimo** de cada **coluna** da base de dados

```
iris.min()

#sepal_length        4.3
#sepal_width         2.0
#petal_length        1.0
#petal_width         0.1
#species          setosa
#dtype: object
```

max()

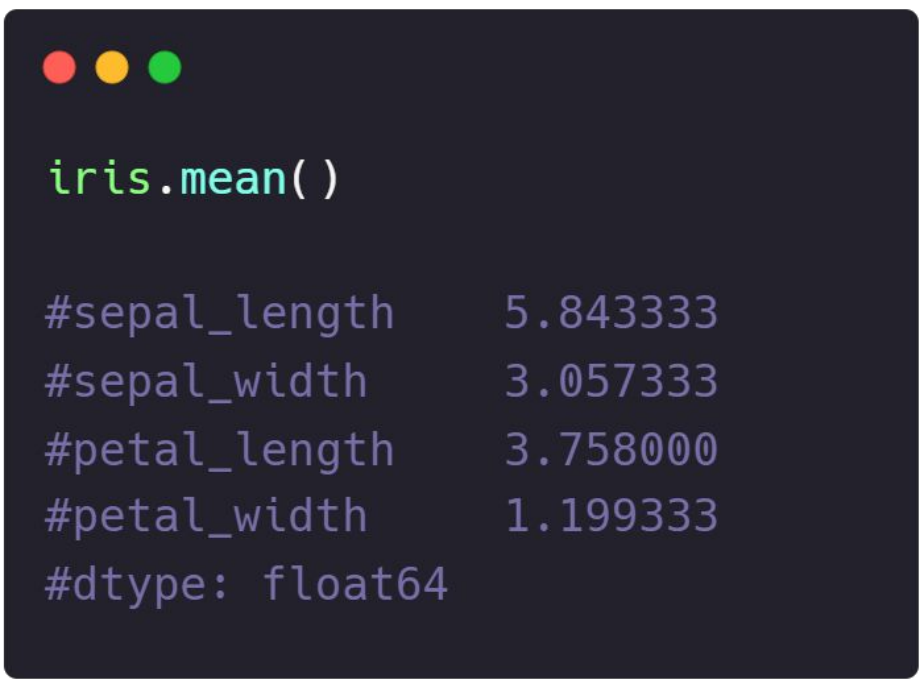
Retorna o **valor máximo** de cada **coluna** da base de dados

```
iris.max()

#sepal_length          7.9
#sepal_width           4.4
#petal_length          6.9
#petal_width           2.5
#species          virginica
#dtype: object
```

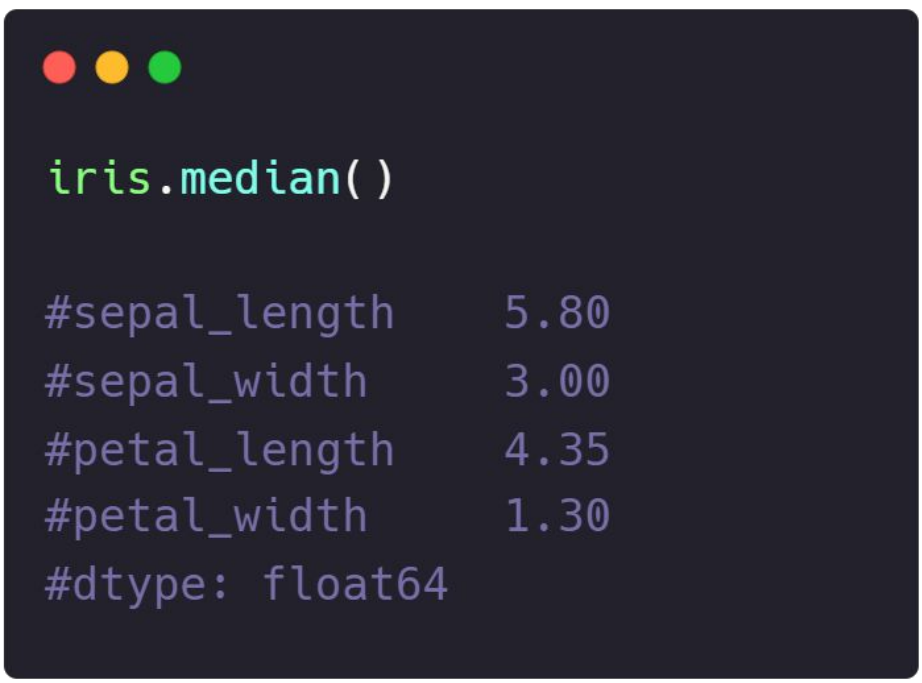
mean()

Retorna a **média** dos valores de cada **coluna** do dataframe



median()

Retorna a **mediana** dos valores de cada **coluna** do dataframe



Exercícios

cubos academy

Selecionando Valores

cubos
// academy //

columns

Retorna um objeto que se comporta como **lista** com **os nomes das colunas** do dataframe

```
iris.columns

#Index(['sepal_length', 'sepal_width',
#'petal_length', 'petal_width',
#       'species'],
#      dtype='object')
```

Selecionando Colunas

Podemos **selecionar** apenas **colunas específicas** de um Dataframe. Isso é muito útil quando temos uma base com muitas colunas mas só precisamos manipular algumas

```
iris[['sepal_length','sepal_width','petal_length',
'petal_width']]
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Selecionando Linhas e Colunas

O comando **loc()** é utilizado para **selecionar linhas e colunas** específicas. pode ser feita de duas formas:

- Por intervalo
- Por índice

```
df.loc[selecao_linhas, selecao_colunas]
```

Seleção por intervalo

Suponha que quero selecionar as **5 primeiras linhas** das **3 primeiras colunas** do dataframe **isis**.

```
iris.loc[0:4 ,'sepal_length':'petal_length']
```

	sepal_length	sepal_width	petal_length
0	5.1	3.5	1.4
1	4.9	3.0	1.4
2	4.7	3.2	1.3
3	4.6	3.1	1.5
4	5.0	3.6	1.4

Seleção por índice

Suponha que quero selecionar as **linhas** de índice **0, 3 e 4** das **colunas sepal_length** e **petal_length**.

```
iris.loc[[0,3,4] ,['sepal_length', 'petal_length']]
```

	sepal_length	petal_length
0	5.1	1.4
3	4.6	1.5
4	5.0	1.4

Exercícios

cubos academy

O que podemos fazer
com as tabelas?

Operações Matemáticas

cubos
// academy //

Operações Matemáticas

Tendo como base os seguintes DataFrames:

```
df1
```

	Idade	Peso	Altura
0	26	63	1.67
1	28	65	1.60

```
df2
```

	Idade	Peso	Altura
0	30	70	1.73
1	21	80	1.80

add()

Adicionar um valor a **todos os itens** de uma tabela:

```
df1.add(2)
```

	Idade	Peso	Altura
0	26	63	1.67
1	28	65	1.60

	Idade	Peso	Altura
0	28	65	3.67
1	30	67	3.60

add()

Soma os itens de **duas tabelas** levando em consideração o **número da linha** e o **nome da coluna**

```
df1.add(df2)
```

	Idade	Peso	Altura
0	30	70	1.73
1	21	80	1.80

	Idade	Peso	Altura
0	26	63	1.67
1	28	65	1.60

	Idade	Peso	Altura
0	56	133	3.4
1	49	145	3.4

sub()

Subtrai um valor a **todos os itens** de uma tabela:

```
df1.sub(2)
```

	Idade	Peso	Altura
0	26	63	1.67
1	28	65	1.60

	Idade	Peso	Altura
0	24	61	-0.33
1	26	63	-0.40

sub()

Subtrai os itens de **duas tabelas** levando em consideração o **número da linha** e o **nome da coluna**

```
df1.sub(df2)
```

	Idade	Peso	Altura
0	26	63	1.67
1	28	65	1.60

	Idade	Peso	Altura
0	30	70	1.73
1	21	80	1.80

	Idade	Peso	Altura
0	-4	-7	-0.06
1	7	-15	-0.20

mul()

Multiplica um valor a **todos os itens** de uma tabela:

```
df1.mul(2)
```

	Idade	Peso	Altura
0	26	63	1.67
1	28	65	1.60

	Idade	Peso	Altura
0	52	126	3.34
1	56	130	3.20

mul()

Multiplica os itens de **duas tabelas** levando em consideração o **número da linha** e o **nome da coluna**

```
df1.mul(df2)
```

	Idade	Peso	Altura
0	26	63	1.67
1	28	65	1.60

	Idade	Peso	Altura
0	30	70	1.73
1	21	80	1.80

	Idade	Peso	Altura
0	780	4410	2.8891
1	588	5200	2.8800

div()

Divide um valor a **todos os itens** de uma tabela:

```
df1.div(2)
```

	Idade	Peso	Altura
0	26	63	1.67
1	28	65	1.60

	Idade	Peso	Altura
0	13.0	31.5	0.835
1	14.0	32.5	0.800

div()

Divide os itens de **duas tabelas** levando em consideração o **número da linha** e o **nome da coluna**

```
df1.div(df2)
```

	Idade	Peso	Altura
0	26	63	1.67
1	28	65	1.60

	Idade	Peso	Altura
0	30	70	1.73
1	21	80	1.80

	Idade	Peso	Altura
0	0.866667	0.9000	0.965318
1	1.333333	0.8125	0.888889

Exercícios

cubos academy

Helena Guimarães
Professora Cubos Academy

cubos
// academy //

www.cubos.academy