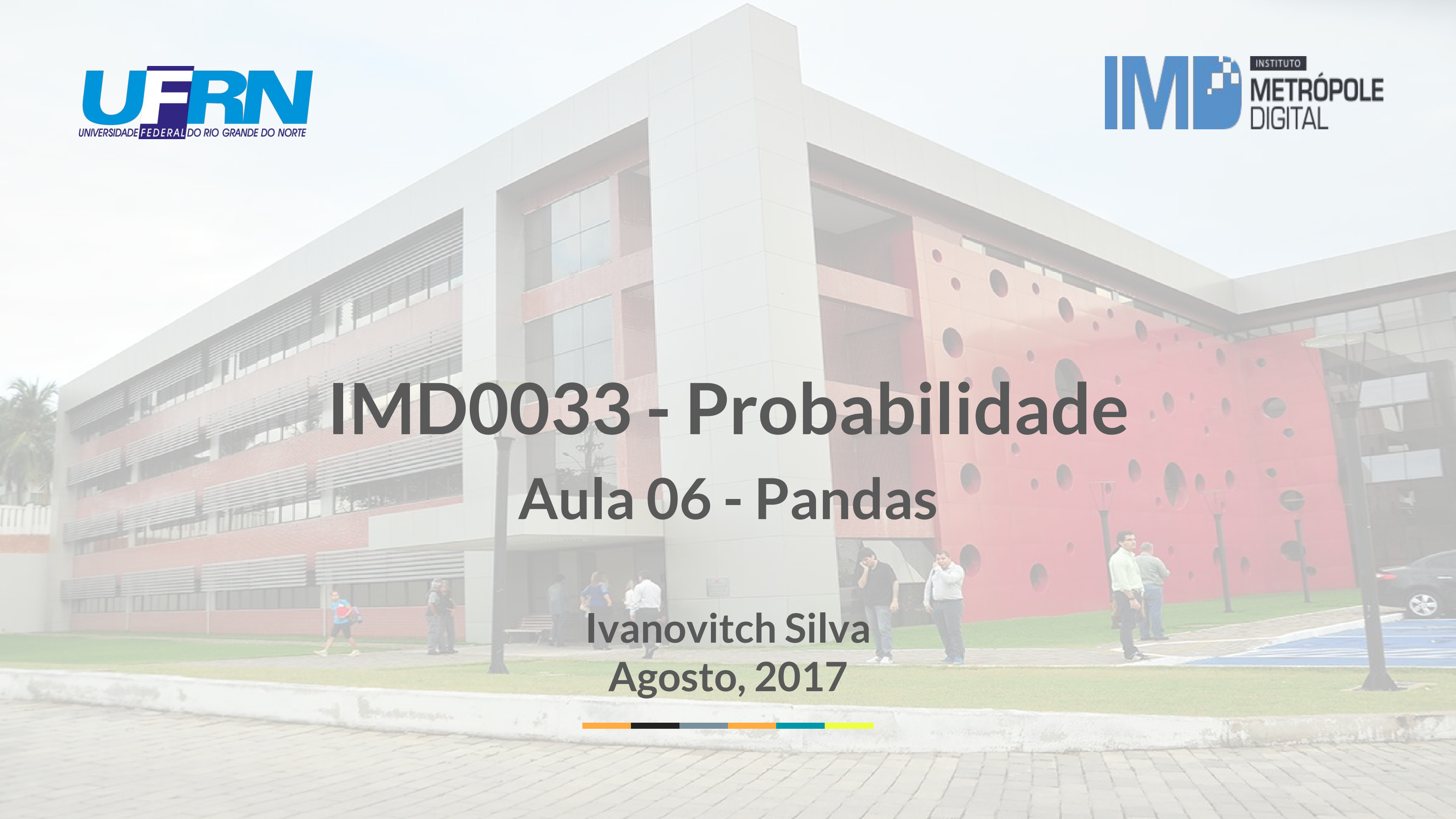


UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

IMD INSTITUTO METRÓPOLE DIGITAL

IMD0033 - Probabilidade

Aula 06 - Pandas

Ivanovitch Silva
Agosto, 2017

Agenda

- Motivação
- Introdução sobre Pandas
- Lendo CSV
- Listando as colunas do dataset
- Dimensões
- Acessando os dados
- Series vs Dataframe
- Selecionando linhas e colunas
- Manipulando dados com Pandas

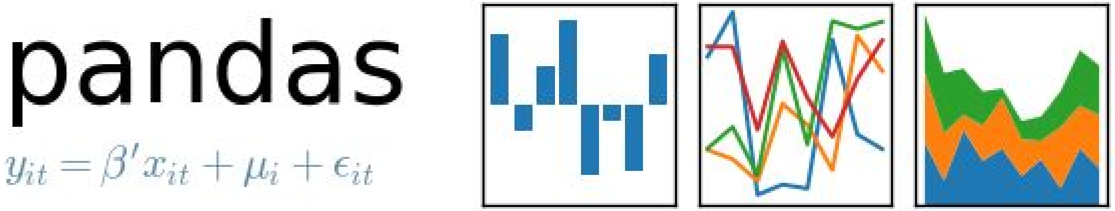
Atualizar o repositório

```
git clone https://github.com/ivanovitchm/IMD0033_Probabilidade.git
```

Ou ….

```
git pull
```

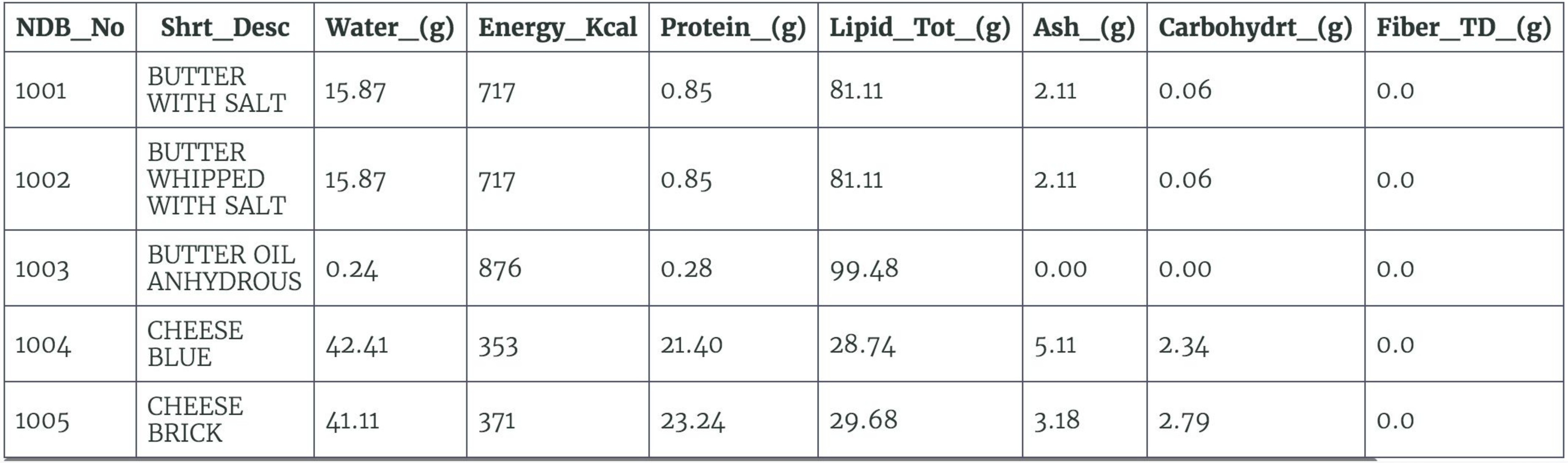
Motivação



- Estrutura de dados que preenche as lacunas deixadas pelo NumPy
  - Armazena tipos diferentes
  - Acesso (index) é mais flexível
- Armazenamento em formato de tabela (dataframe)

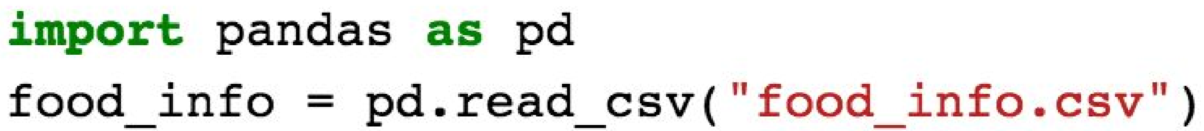
Conjunto de Dados a ser Estudado

USDA National Nutrient Database for Standard Reference



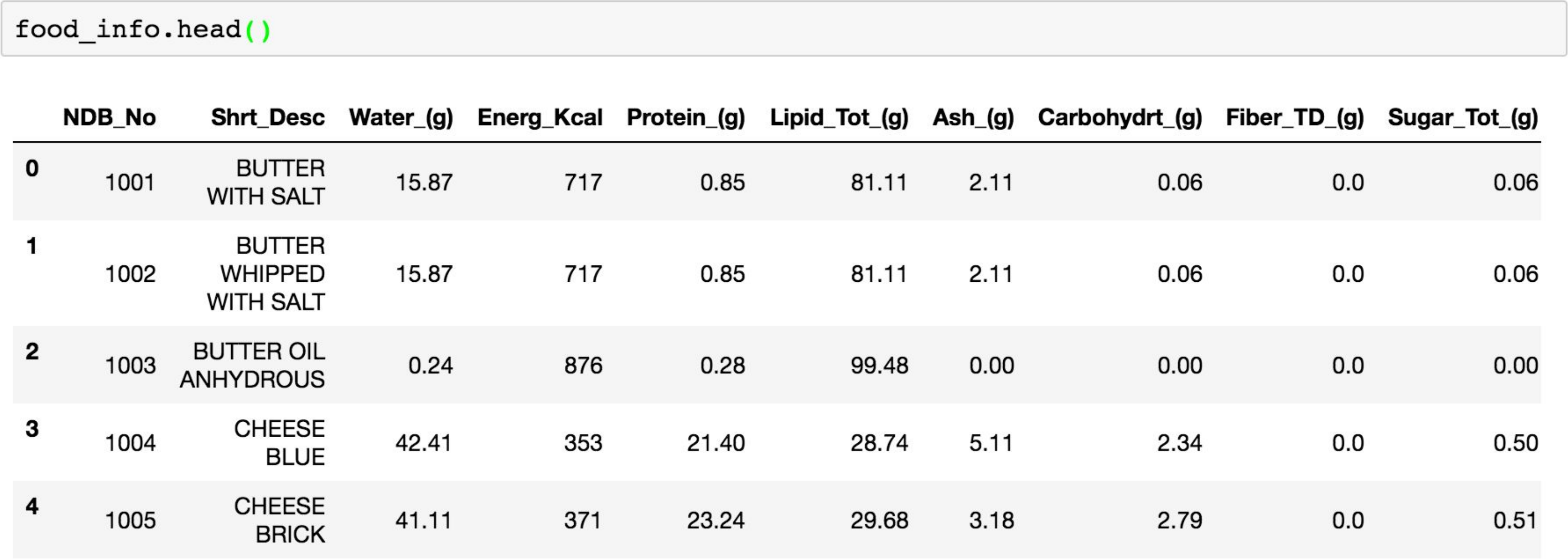
NDB_No	Shrt_Desc	Water_(g)	Energy_Kcal	Protein_(g)	Lipid_Tot_(g)	Ash_(g)	Carbohydrt_(g)	Fiber_TD_(g)
1001	BUTTER WITH SALT	15.87	717	0.85	81.11	2.11	0.06	0.0
1002	BUTTER WHIPPED WITH SALT	15.87	717	0.85	81.11	2.11	0.06	0.0
1003	BUTTER OIL ANHYDROUS	0.24	876	0.28	99.48	0.00	0.00	0.0
1004	CHEESE BLUE	42.41	353	21.40	28.74	5.11	2.34	0.0
1005	CHEESE BRICK	41.11	371	23.24	29.68	3.18	2.79	0.0

Lendo arquivos CSV



```
import pandas as pd
food_info = pd.read_csv("food_info.csv")
```

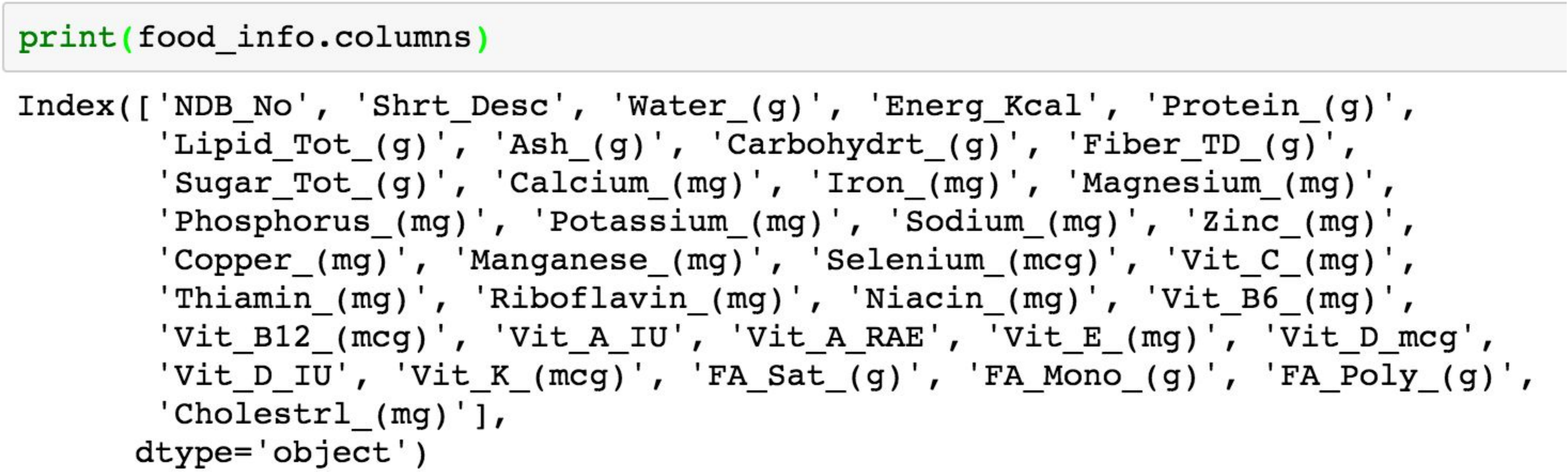
Análise inicial



```
food_info.head()
```

	NDB_No	Shrt_Desc	Water_(g)	Energ_Kcal	Protein_(g)	Lipid_Tot_(g)	Ash_(g)	Carbohydrt_(g)	Fiber_TD_(g)	Sugar_Tot_(g)
0	1001	BUTTER WITH SALT	15.87	717	0.85	81.11	2.11	0.06	0.0	0.06
1	1002	BUTTER WHIPPED WITH SALT	15.87	717	0.85	81.11	2.11	0.06	0.0	0.06
2	1003	BUTTER OIL ANHYDROUS	0.24	876	0.28	99.48	0.00	0.00	0.0	0.00
3	1004	CHEESE BLUE	42.41	353	21.40	28.74	5.11	2.34	0.0	0.50
4	1005	CHEESE BRICK	41.11	371	23.24	29.68	3.18	2.79	0.0	0.51

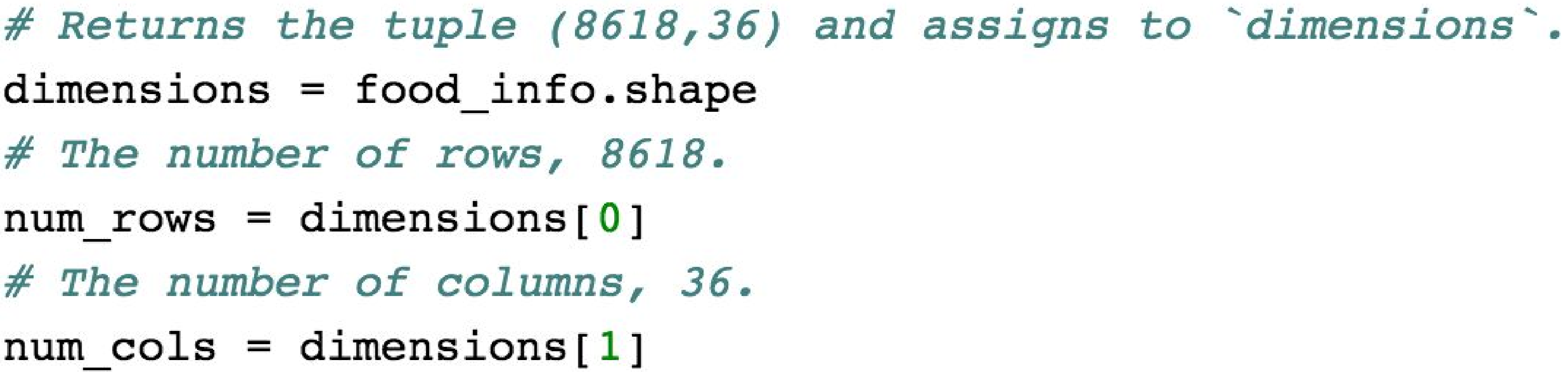
Quais colunas?



```
print(food_info.columns)
```

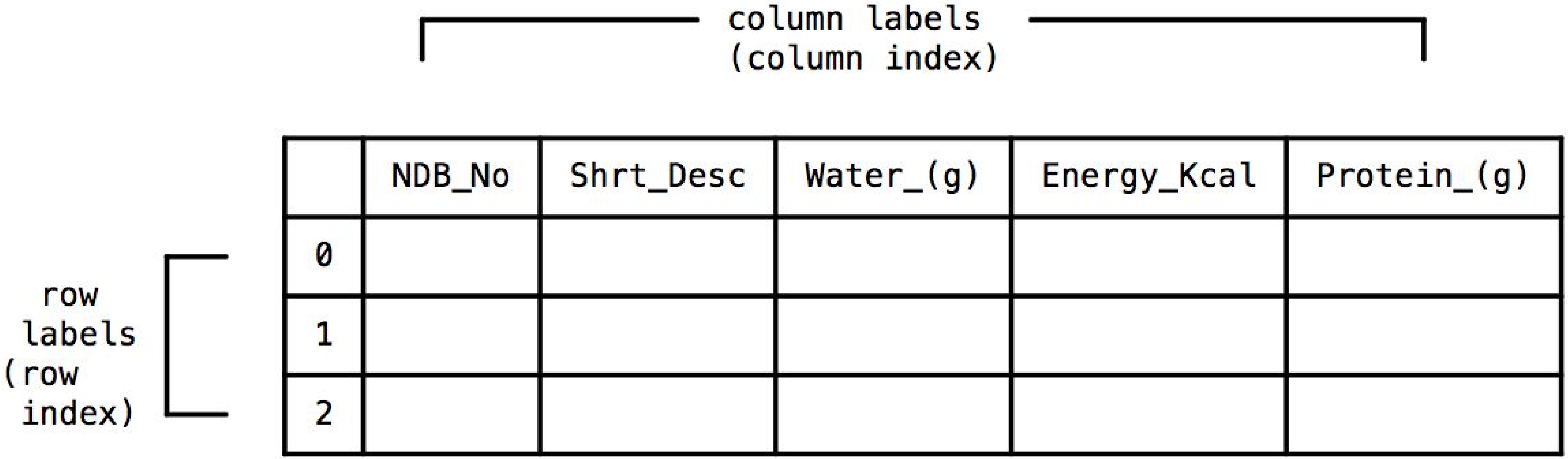
```
Index(['NDB_No', 'Shrt_Desc', 'Water_(g)', 'Energ_Kcal', 'Protein_(g)',
       'Lipid_Tot_(g)', 'Ash_(g)', 'Carbohydrt_(g)', 'Fiber_TD_(g)',
       'Sugar_Tot_(g)', 'Calcium_(mg)', 'Iron_(mg)', 'Magnesium_(mg)',
       'Phosphorus_(mg)', 'Potassium_(mg)', 'Sodium_(mg)', 'Zinc_(mg)',
       'Copper_(mg)', 'Manganese_(mg)', 'Selenium_(mcg)', 'Vit_C_(mg)',
       'Thiamin_(mg)', 'Riboflavin_(mg)', 'Niacin_(mg)', 'Vit_B6_(mg)',
       'Vit_B12_(mcg)', 'Vit_A_IU', 'Vit_A_RAE', 'Vit_E_(mg)', 'Vit_D_mcg',
       'Vit_D_IU', 'Vit_K_(mcg)', 'FA_Sat_(g)', 'FA_Mono_(g)', 'FA_Poly_(g)',
       'Cholestrl_(mg)'],
      dtype='object')
```

E as dimensões do conjunto de dados?



```
# Returns the tuple (8618,36) and assigns to `dimensions`.
dimensions = food_info.shape
# The number of rows, 8618.
num_rows = dimensions[0]
# The number of columns, 36.
num_cols = dimensions[1]
```

Acessando (indexing)

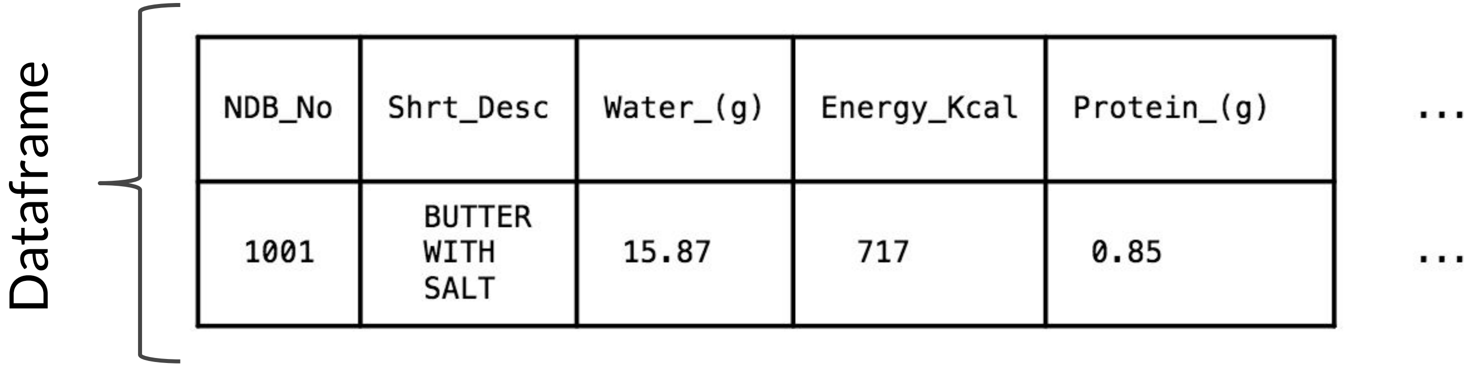


column labels
(column index)

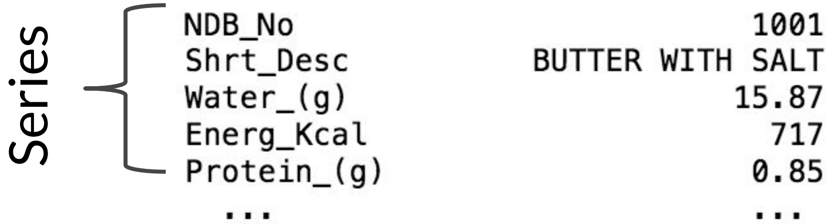
row labels (row index)

	NDB_No	Shrt_Desc	Water_(g)	Energy_Kcal	Protein_(g)
0					
1					
2					

Series vs Dataframe



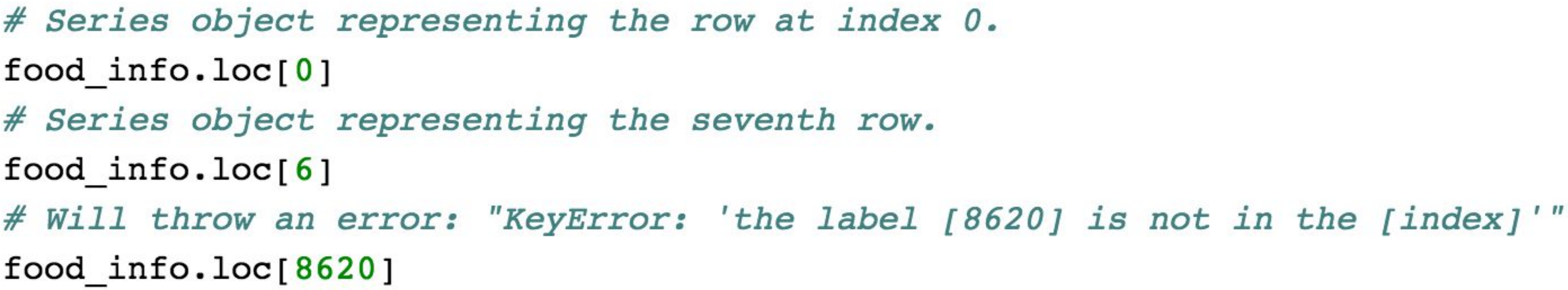
NDB_No	Shrt_Desc	Water_(g)	Energy_Kcal	Protein_(g)	...
1001	BUTTER WITH SALT	15.87	717	0.85	...



```
NDB_No                    1001
Shrt_Desc     BUTTER WITH SALT
Water_(g)                15.87
Energ_Kcal                 717
Protein_(g)               0.85
  ...                      ...
```

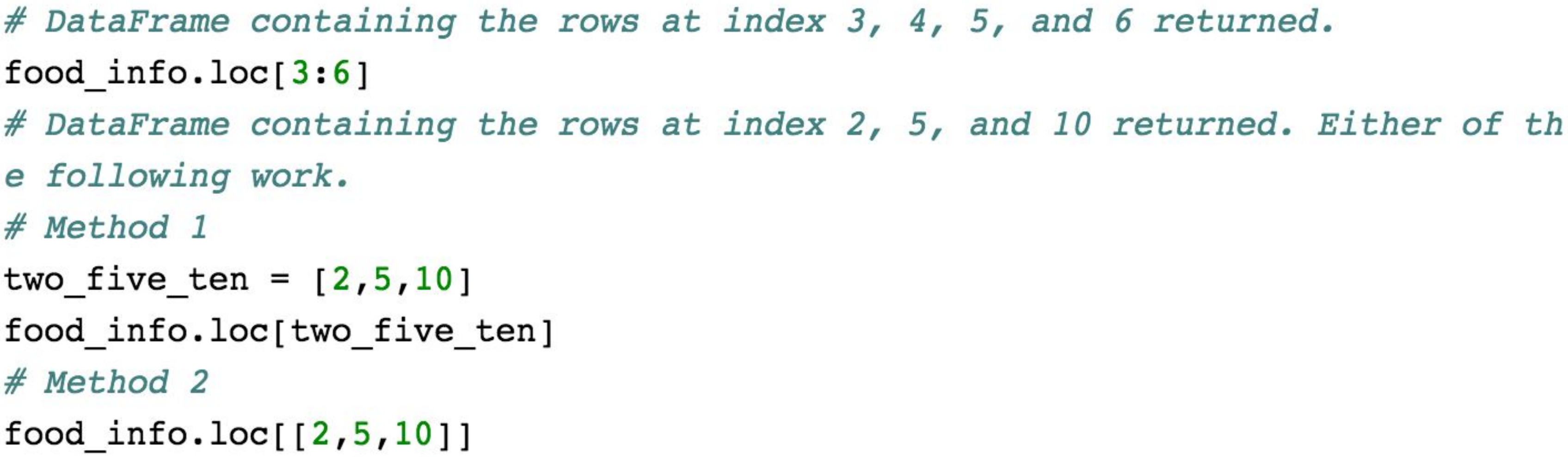
- Series = coleção de valores
- Dataframe = coleção de series

Selecionando uma linha



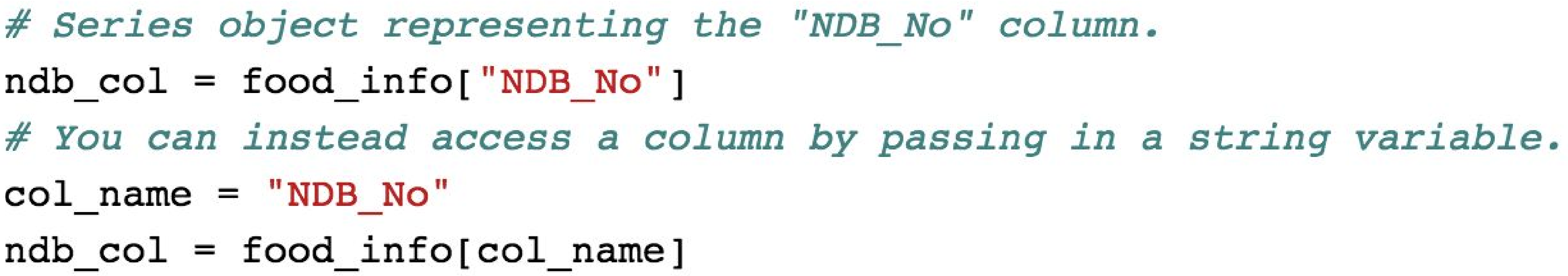
```
# Series object representing the row at index 0.
food_info.loc[0]
# Series object representing the seventh row.
food_info.loc[6]
# Will throw an error: "KeyError: 'the label [8620] is not in the [index]'"
food_info.loc[8620]
```

Selecionando múltiplas linhas



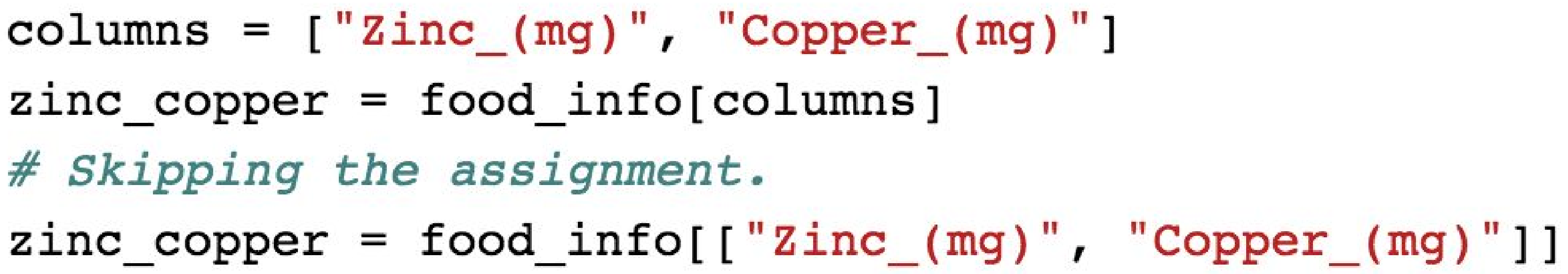
```
# DataFrame containing the rows at index 3, 4, 5, and 6 returned.
food_info.loc[3:6]
# DataFrame containing the rows at index 2, 5, and 10 returned. Either of th
e following work.
# Method 1
two_five_ten = [2,5,10]
food_info.loc[two_five_ten]
# Method 2
food_info.loc[[2,5,10]]
```

Selecionando uma coluna



```
# Series object representing the "NDB_No" column.
ndb_col = food_info["NDB_No"]
# You can instead access a column by passing in a string variable.
col_name = "NDB_No"
ndb_col = food_info[col_name]
```

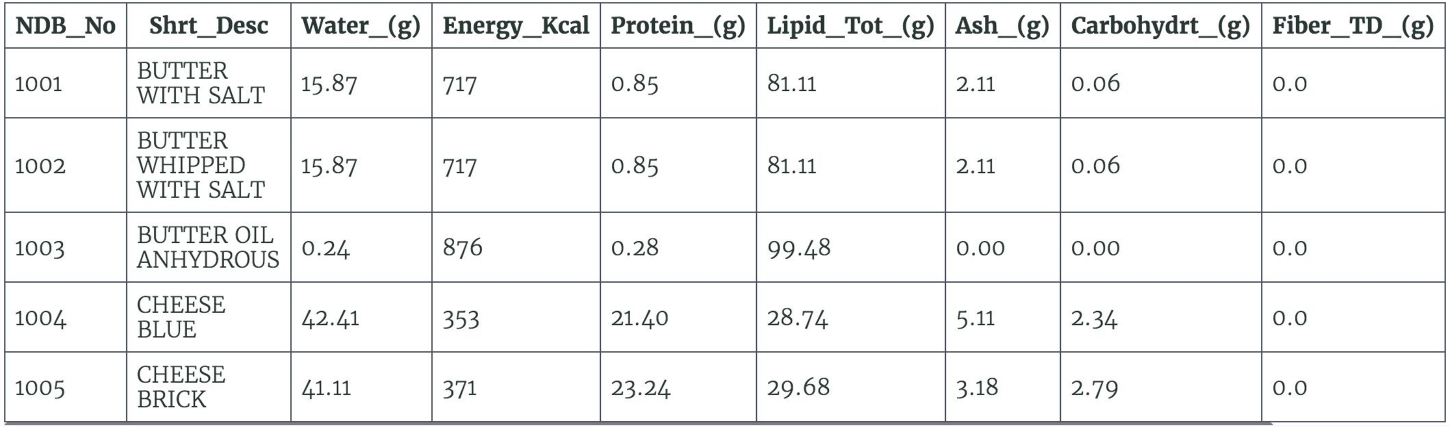
Selecionando múltiplas colunas



```
columns = ["Zinc_(mg)", "Copper_(mg)"]
zinc_copper = food_info[columns]
# Skipping the assignment.
zinc_copper = food_info[["Zinc_(mg)", "Copper_(mg)"]]
```

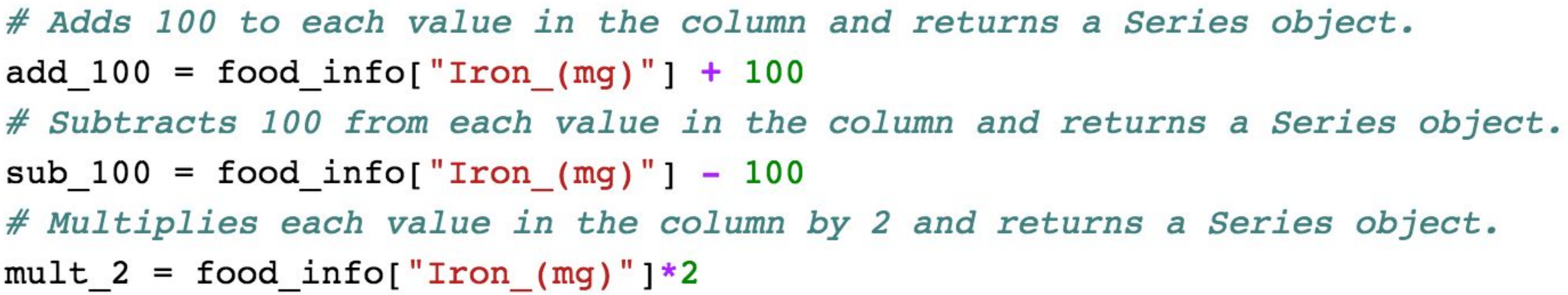
Manipulando dados com Pandas

$$Score = 2 \times (Protein\_(g)) - 0.75 \times (Lipid\_Tot\_(g))$$



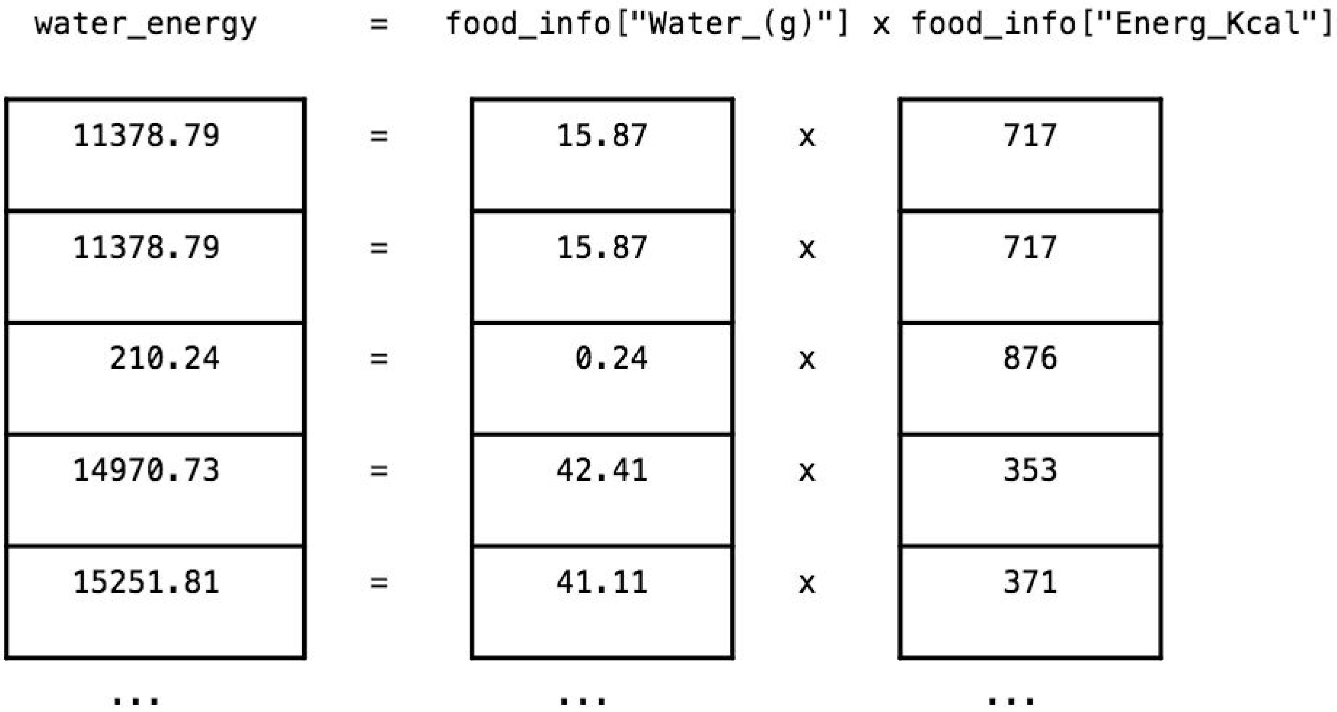
NDB_No	Shrt_Desc	Water_(g)	Energy_Kcal	Protein_(g)	Lipid_Tot_(g)	Ash_(g)	Carbohydrt_(g)	Fiber_TD_(g)
1001	BUTTER WITH SALT	15.87	717	0.85	81.11	2.11	0.06	0.0
1002	BUTTER WHIPPED WITH SALT	15.87	717	0.85	81.11	2.11	0.06	0.0
1003	BUTTER OIL ANHYDROUS	0.24	876	0.28	99.48	0.00	0.00	0.0
1004	CHEESE BLUE	42.41	353	21.40	28.74	5.11	2.34	0.0
1005	CHEESE BRICK	41.11	371	23.24	29.68	3.18	2.79	0.0

Operações aritméticas em colunas



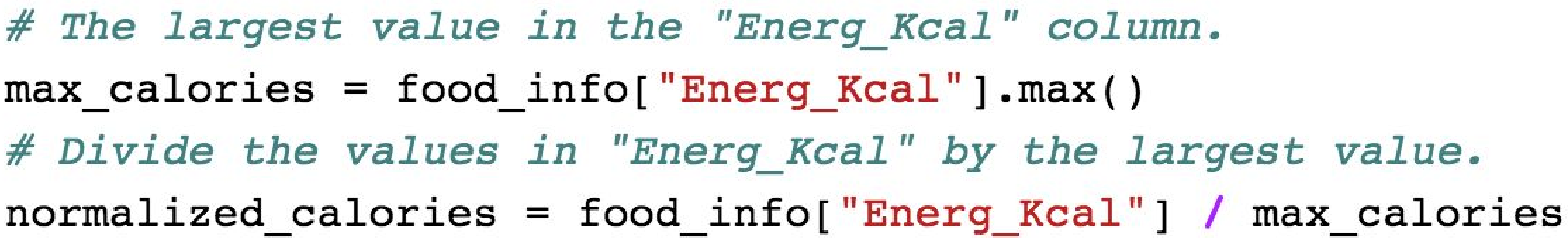
```
# Adds 100 to each value in the column and returns a Series object.
add_100 = food_info["Iron_(mg)"] + 100
# Subtracts 100 from each value in the column and returns a Series object.
sub_100 = food_info["Iron_(mg)"] - 100
# Multiplies each value in the column by 2 and returns a Series object.
mult_2 = food_info["Iron_(mg)"]*2
```

Operações com múltiplas colunas



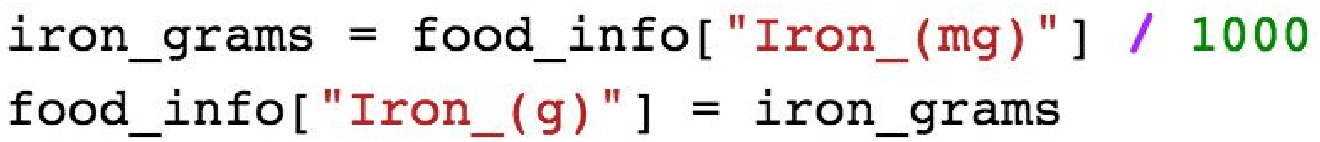
water_energy	=	food_info["Water_(g)"]	x	food_info["Energ_Kcal"]
11378.79	=	15.87	x	717
11378.79	=	15.87	x	717
210.24	=	0.24	x	876
14970.73	=	42.41	x	353
15251.81	=	41.11	x	371
...		...		...

Normalizando dados



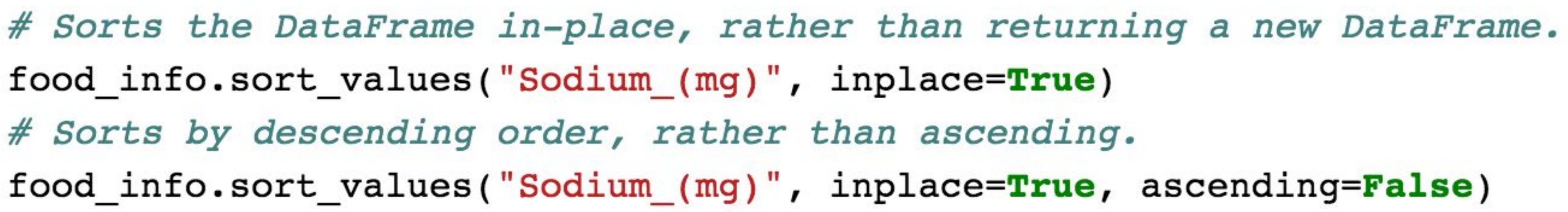
```
# The largest value in the "Energ_Kcal" column.
max_calories = food_info["Energ_Kcal"].max()
# Divide the values in "Energ_Kcal" by the largest value.
normalized_calories = food_info["Energ_Kcal"] / max_calories
```

Criando uma nova coluna



```
iron_grams = food_info["Iron_(mg)"] / 1000
food_info["Iron_(g)"] = iron_grams
```

Ordenando valores de uma coluna



```
# Sorts the DataFrame in-place, rather than returning a new DataFrame.
food_info.sort_values("Sodium_(mg)", inplace=True)
# Sorts by descending order, rather than ascending.
food_info.sort_values("Sodium_(mg)", inplace=True, ascending=False)
```

MacBook Pro