Python Fundamentos – Biblioteca Pandas e NumPy – Parte 02

Disciplina: Estatística Aplicada à Engenharia de Software

Prof. Me. Max Gabriel Steiner

- Para utilizar a biblioteca NumPy precisamos importar ela para o Python

In [42]:
```
# Importando o NumPy
import numpy as np
```

In [43]:
```
# Usando o NumPy para alimentar uma das colunas do dataframe
frame2['Débito'] = np.arange(5.)
```

In [44]:
```
frame2
```

Out[44]:

	Ano	Estado	População	Débito
um	2002	Santa Catarina	1.5	0.0
dois	2003	Paraná	1.7	1.0
três	2004	Goiás	3.6	2.0
quatro	2005	Bahia	2.4	3.0
cinco	2006	Minas Gerais	2.9	4.0

- Utilizando o código a seguir é possível visualizar a variedades de elementos contidos no DataFrame

```
In [45]: frame2.values

Out[45]: array([[2002, 'Santa Catarina', 1.5, 0.0],
                [2003, 'Paraná', 1.7, 1.0],
                [2004, 'Goiás', 3.6, 2.0],
                [2005, 'Bahia', 2.4, 3.0],
                [2006, 'Minas Gerais', 2.9, 4.0]], dtype=object)
```

- Vamos agora utilizar o método describe()
- Este método gera um resumo do nosso DataFrame, com as principais informações matemáticas dos dados contidos nele.

```
In [46]: # Resumo do Dataframe com todas as informações contidas
frame2.describe()
```

Out[46]:

	Ano	População	Débito
count	5.000000	5.000000	5.000000
mean	2004.000000	2.420000	2.000000
std	1.581139	0.864292	1.581139
min	2002.000000	1.500000	0.000000
25%	2003.000000	1.700000	1.000000
50%	2004.000000	2.400000	2.000000
75%	2005.000000	2.900000	3.000000
max	2006.000000	3.600000	4.000000

- A 1ª linha **count**: apresenta o número de elementos contidos em cada coluna do DataFrame;

Out[46]:

	Ano	População	Débito
count	5.000000	5.000000	5.000000
mean	2004.000000	2.420000	2.000000
std	1.581139	0.864292	1.581139
min	2002.000000	1.500000	0.000000
25%	2003.000000	1.700000	1.000000
50%	2004.000000	2.400000	2.000000
75%	2005.000000	2.900000	3.000000
max	2006.000000	3.600000	4.000000

- A 2ª linha **mean**: apresenta a média dos elementos em cada coluna;
- A 3ª linha **std**: apresenta o desvio-padrão;
- A 4ª linha **min**: apresenta o valor mínimo;
- As linhas 5, 6 e 7 **25%**, **50%** e **75%**: apresentam os quartis;
- A 8ª linha **max**: apresenta os valores máximos de cada coluna.
- Portanto, essa aplicação é uma forma rápida de mostrar um resumo do DataFrame.

- Utilizando o código a seguir é possível visualizar os tipos de valores presentes no Dataset:

```
In [47]: # Tipos de dados no DataFrame
         frame2.dtypes

Out[47]: Ano            int64
         Estado        object
         População    float64
         Débito       float64
         dtype: object
```

- Essa função é interessante, pois um DataFrame é um conjunto de elementos que contém tipos distintos de dados e variáveis.
- Por exemplo neste caso temos que:
- Ano: contém valores do tipo inteiros;
- Estado: contém valores do tipo string;
- População: contém valores do tipo float.

- A função abaixo permite observar quais são os índices do meu DataFrame

```
In [48]: frame2.index
Out[48]: Index(['um', 'dois', 'três', 'quatro', 'cinco'], dtype='object')
```

- A próxima função permite observar quais são as colunas do meu DataFrame

```
In [49]: frame2.columns
Out[49]: Index(['Ano', 'Estado', 'População', 'Débito'], dtype='object')
```

- E por fim, podemos ver quais são os valores do nosso DataFrame

```
In [50]: frame2.values
Out[50]: array([[2002, 'Santa Catarina', 1.5, 0.0],
                [2003, 'Paraná', 1.7, 1.0],
                [2004, 'Goiás', 3.6, 2.0],
                [2005, 'Bahia', 2.4, 3.0],
                [2006, 'Minas Gerais', 2.9, 4.0]], dtype=object)
```

- Podemos fazer um “slice” para reportar apenas uma coluna do DataFrame:

```
In [51]: frame2['Ano']

Out[51]: um        2002
         dois      2003
         três      2004
         quatro    2005
         cinco     2006
         Name: Ano, dtype: int64
```

- Apresentando então apenas uma coluna com seus respectivos índices.
- Também é possível utilizar o comando:

```
In [52]: frame2.Ano

Out[52]: um        2002
         dois      2003
         três      2004
         quatro    2005
         cinco     2006
         Name: Ano, dtype: int64
```

- Podemos fazer ainda um “slice” mais detalhado dentro do DataFrame:

```
In [53]: frame2[:2]
```

Out[53]:

	Ano	Estado	População	Débito
um	2002	Santa Catarina	1.5	0.0
dois	2003	Paraná	1.7	1.0

- Neste caso trouxemos todos os elementos que comecem em qualquer posição até a posição 2.

- O conteúdo :número define o fim, por exemplo se aplicarmos :3

```
In [54]: frame2[:3]
```

Out[54]:

	Ano	Estado	População	Débito
um	2002	Santa Catarina	1.5	0.0
dois	2003	Paraná	1.7	1.0
três	2004	Goiás	3.6	2.0

- Neste caso trouxemos todos os elementos que comecem em qualquer posição até a posição 3.

- Vamos ver agora mais um exemplo de criação de DataFrames:

```
In [55]: # Criando um dicionário
         web_stats = {'Dias':[1, 2, 3, 4, 5, 6, 7],
                      'Visitantes':[45, 23, 67, 78, 23, 12, 14],
                      'Taxas':[11, 22, 33, 44, 55, 66, 77]}
```

- Temos um exemplo de dicionário com 3 pares de valores.
- São 3 chaves (Dias, Visitantes e Taxas) e cada chave possui uma lista de dados.
- Após criar o dicionário, vamos criar então o DataFrame, utilizando o dicionário web_stats como parâmetro para a criação deste DataFrame.

```
In [56]: df = pd.DataFrame(web_stats)
```

- Imprimindo nosso dicionário temos:

```
In [57]: print(df)
```

	Dias	Visitantes	Taxas
0	1	45	11
1	2	23	22
2	3	67	33
3	4	78	44
4	5	23	55
5	6	12	66
6	7	14	77

- Podemos perceber que o Pandas automaticamente definiu os índices contidos na 1ª coluna.

- Portanto, com o código a seguir vamos pedir para que o Pandas nos retorne esse DataFrame, considerando que a coluna “Dias” seja na verdade nosso índice:

```
In [58]: # Visualizando uma coluna index (índices)
         print(df.set_index('Dias'))
```

```
      Visitantes  Taxas
Dias
1             45     11
2             23     22
3             67     33
4             78     44
5             23     55
6             12     66
7             14     77
```

- Temos que sempre ficar atentos ao receber DataSets que já tenham sido tratados por algum analista de dados ou por alguma outra ferramenta, pois ao analisar essa modificação que fizemos podemos perceber que os índices naturais em Python começam por zero, entretanto nosso índice agora passou a começar com o número 1.

- Ainda assim é importante perceber que o comando anterior não alterou permanentemente nosso DataFrame, pois através dele apenas realizamos a ação de imprimir o DataFrame utilizando uma das colunas como índice, porém como podemos perceber:

```
In [59]: print(df.head())
```

	Dias	Visitantes	Taxas
0	1	45	11
1	2	23	22
2	3	67	33
3	4	78	44
4	5	23	55

- Nosso DataFrame continua intacto.

➢ Podemos imprimir também nosso DataFrame fatiando ele por colunas, nesse caso vamos fatiar a coluna Visitantes:

```
In [60]: print(df['Visitantes'])

0    45
1    23
2    67
3    78
4    23
5    12
6    14
Name: Visitantes, dtype: int64
```

- Podemos fazer mais um slice (fatiamento) conjunto:

```
In [62]: print(df[['Visitantes', 'Taxas']])
```

```
   Visitantes  Taxas
0          45     11
1          23     22
2          67     33
3          78     44
4          23     55
5          12     66
6          14     77
```