In [3]:
```
import pandas as pd
import numpy as np
```

Descrições sobre os dados

In [4]:
```
# Criando Series
nums = range(5)
ser = pd.Series(nums)
ser
```

Out[4]:
```
0    0
1    1
2    2
3    3
4    4
dtype: int64
```

In [56]:
```
# Exibe os index (como é estamos usando um range para criar ele exibe onde começa e termina a sequencia)
ser.index
```

Out[56]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [19]:
```
# Exibir tipos de dados
ser.dtype
```

Out[19]:
```
dtype('int64')
```

In [22]:
```
# modificando o tipo de dado de 'int64' para 'float64'
ser = pd.Series(nums, dtype='float')
ser
```

Out[22]:
```
0    0.0
1    1.0
2    2.0
3    3.0
4    4.0
dtype: float64
```

In [24]:
```
# Saber a soma total
ser.sum()
```

Out[24]:
```
10.0
```

In [26]:
```
# Maior Valor
ser.max()
```

Out[26]:
```
4.0
```

In [27]:
```
# Menor Valor
ser.min()
```

Out[27]:
```
0.0
```

In [35]:
```
# Média dos valores
ser.mean()
```

Out[35]:
```
2.0
```

In [40]:
```
# Em caso seja necessário aplicar múltiplas funções de agregação podemos passar uma lista com todas elas para a função agg
ser.agg(['mean', 'median', 'sum', 'count', 'min', 'max'])
```

Out[40]:
```
mean       2.0
median     2.0
sum       10.0
count      5.0
min        0.0
max        4.0
dtype: float64
```

In [33]:
```
# retorna n maiores valores da Series
ser.nlargest(n=2)
```

Out[33]:
```
4    4.0
3    3.0
dtype: float64
```

In [41]:
```
# retorna n menores valores da Series
ser.nsmallest(n=2)
```

Out[41]:
```
0    0.0
1    1.0
dtype: float64
```

In [44]:
```
# Exibir Quantidade de linhas
ser.count()
```

Out[44]:
```
5
```

In [5]:
```
# Gerando Series com dados nulos
ser = pd.Series([1, 2, np.nan, 1.5, np.nan, 1, 2, 1])
ser
```

Out[5]:
```
0    1.0
1    2.0
2    NaN
3    1.5
4    NaN
5    1.0
6    2.0
7    1.0
dtype: float64
```

In [6]:
```
# Conta a quantidade de itens nulos
ser.isna().sum()
```

Out[6]:
```
2
```

In [7]:
```
# Retorna o número de ocorrências de cada valor no array Series
ser.value_counts()
```

Out[7]:
```
1.0    3
2.0    2
1.5    1
dtype: int64
```

In [8]:
```
# E finalmente o describe() (quatis)
ser.describe()
```

Out[8]:
```
count    6.000000
mean     1.416667
std      0.491596
min      1.000000
25%      1.000000
50%      1.250000
75%      1.875000
max      2.000000
dtype: float64
```