```
import pandas as pd
```

Exercícios 2

🎯 Parte 1 — Series e DataFrames

1.

Crie uma Series com os números de 10 a 50 (de 10 em 10) e índices personalizados: 'A', 'B', 'C', 'D', 'E'.

- Exiba o valor correspondente ao índice 'D'.
- Multiplique todos os valores por 2.

```
ser1 = list(range(10, 51, 10))
# indices = [{'Valores': 'A', 'B', 'C', 'D', 'E'}] # CRiando a coluna valores já com o nome 'Valores'
indices = ['A', 'B', 'C', 'D', 'E']

df = pd.DataFrame(data=ser1, index=indices)
```

```
df.rename(columns={0: 'Valores'}, inplace = True) # Renomeando a coluna pois estava como 0.
```

```
df
```

border="1" class="dataframe">

	Valores
A	10
B	20
C	30
D	40
E	50

```
# Exiba o valor correspondente ao índice 'D'.
df.loc['D'] # 40
```

```
0    40
Name: D, dtype: int64
```

```
# Multiplique todos os valores por 2.
df['Valores'] * 2
```

```
A     20
B     40
C     60
D     80
E    100
Name: Valores, dtype: int64
```

2.

Crie um dataFrame com os dados abaixo:

>

Aluno	Nota1	Nota2	Nota3
Ana	8	7	6
Bruno	5	6	4
Carla	9	8	10

- Crie uma nova coluna chamada 'Média' com a média das 3 notas.
- Filtre os alunos com média acima de 7.

```
dados = {
    'Aluno': ['Ana', 'Bruno', 'Carla'],
    'Nota1': [8, 5, 9],
    'Nota2': [7, 6, 8],
    'Nota3': [6, 4, 10]
}

df2 = pd.DataFrame(dados)
```

```
df2
```

border="1" class="dataframe">

	Aluno	Nota1	Nota2	Nota3
0	Ana	8	7	6
1	Bruno	5	6	4
2	Carla	9	8	10

```
# Crie uma nova coluna chamada 'Média' com a média das 3 notas.
df2['Media'] = (df2['Nota1'] + df2['Nota2'] + df2['Nota3']) / 3
```

```
df2
```

border="1" class="dataframe">

	Aluno	Nota1	Nota2	Nota3	Media
0	Ana	8	7	6	7.0
1	Bruno	5	6	4	5.0
2	Carla	9	8	10	9.0

```
# Filtre os alunos com média acima de 7.
df2[df2['Media'] > 7]
```

border="1" class="dataframe">

	Aluno	Nota1	Nota2	Nota3	Media
2	Carla	9	8	10	9.0

🎯 Parte 2 — Filtros e Indexação

3.

Usando o DataFrame acima:

- Mostre as notas da Ana usando .loc[].
- Mostre a Nota2 de todos os alunos com .iloc[].
- Substitua todas as notas menores que 6 por 6.

```
df2 # Esse será o dataframe utilizado nesse exercicio
```

border="1" class="dataframe">

	Aluno	Nota1	Nota2	Nota3	Media
0	Ana	8	7	6	7.0
1	Bruno	5	6	4	5.0
2	Carla	9	8	10	9.0

```
# Mostre as notas da Ana usando .loc[].
df2.loc[0]
```

```
Aluno    Ana
Nota1      8
Nota2      7
Nota3      6
Media    7.0
Name: 0, dtype: object
```

```python
# Mostre a Nota2 de todos os alunos com .iloc[].
df2.iloc[:, 2]
```

```
0    7
1    6
2    8
Name: Nota2, dtype: int64
```

```python
# Substitua todas as notas menores que 6 por 6.

colunas_notas = ['Nota1', 'Nota2', 'Nota3'] # Criar uma cópia das colunas de nota

df2[colunas_notas] = df2[colunas_notas].map(lambda x: 6 if x < 6 else x) #
Substituir todas as notas < 6 por 6


"""Entendendo:
- map() aplica uma função em cada célula de um DataFrame.
- A função lambda x: 6 if x < 6 else x diz:
        "Se o valor for menor que 6, substitua por 6; senão, mantenha como está".
"""
```

```python
df2
```

border="1" class="dataframe">

	Aluno	Nota1	Nota2	Nota3	Media
0	Ana	8	7	6	7.0
1	Bruno	6	6	6	5.0
2	Carla	9	8	10	9.0

🎯 Parte 3 — Dados Ausentes

4.

Crie um novo DataFrame com dados faltantes:

>

Nome	Idade	Salário
Ana	25	3000
Bruno	NaN	2500
Carla	30	NaN
Daniel	NaN	NaN

- Preencha as idades faltantes com 28.
- Preencha os salários faltantes com a média da coluna.
- Remova as linhas onde ambos os valores são NaN.

```
import numpy as np

dados2 = {
    'Nome': ['Ana', 'Bruno', 'Carla', 'Daniel'],
    'Idade': [25, np.nan, 30, np.nan],
    'Salario': [3000, 2500, np.nan, np.nan]
}

df3 = pd.DataFrame(dados2)
```

```
df3
```

border="1" class="dataframe">

	Nome	Idade	Salario
0	Ana	25.0	3000.0
1	Bruno	NaN	2500.0
2	Carla	30.0	NaN
3	Daniel	NaN	NaN

```
# Preencha as idades faltantes com 28.
df3['Idade'] = df3['Idade'].fillna(28)
```

```
df3
```

border="1" class="dataframe">

	Nome	Idade	Salario
0	Ana	25.0	3000.0
1	Bruno	28.0	2500.0
2	Carla	30.0	NaN
3	Daniel	28.0	NaN

```
# Preencha os salários faltantes com a média da coluna.
df3['Salario'] = df3['Salario'].fillna(df3['Salario'].mean())
```

```
df3
```

border="1" class="dataframe">

	Nome	Idade	Salario
0	Ana	25.0	3000.0
1	Bruno	28.0	2500.0
2	Carla	30.0	2750.0
3	Daniel	28.0	2750.0

```
# Remova as linhas onde ambos os valores são NaN.
df3.dropna()
```

border="1" class="dataframe">

	Nome	Idade	Salario
0	Ana	25.0	3000.0

🎯 Parte 4 — GroupBy Avançado

5.

Crie este DataFrame

>

Departamento	Funcionário	Salário
RH	Ana	3000
RH	Bruno	3200
TI	Carla	4000
TI	Daniel	4500
Vendas	Eduardo	3500

- Calcule o salário médio por departamento.
- Conte quantos funcionários existem em cada departamento.
- Pegue o maior salário de cada departamento.

```
dados3 = {
    'Departamento': ['RH', 'RH', 'TI', 'TI', 'Vendas'],
    'Funcionario': ['Ana', 'Bruno', 'Carla', 'Daniel', 'Eduardo'],
    'Salario': [3000, 3200, 4000, 4500, 3500]
}

df4 = pd.DataFrame(dados3)
```

```
df4
```

border="1" class="dataframe">

	Departamento	Funcionario	Salario
0	RH	Ana	3000
1	RH	Bruno	3200
2	TI	Carla	4000
3	TI	Daniel	4500
4	Vendas	Eduardo	3500

```python
# Calcule o salário médio por departamento.
df4.groupby('Departamento')['Salario'].mean()
```

```
Departamento
RH        3100.0
TI        4250.0
Vendas    3500.0
Name: Salario, dtype: float64
```

```python
# Conte quantos funcionários existem em cada departamento.
df4.groupby('Departamento')['Funcionario'].count()
```

```
Departamento
RH        2
TI        2
Vendas    1
Name: Funcionario, dtype: int64
```

```python
# Pegue o maior salário de cada departamento.
df4.groupby('Departamento')['Salario'].max()
```

```
Departamento
RH        3200
TI        4500
Vendas    3500
Name: Salario, dtype: int64
```

🎯 Parte 5 — Merge, Join e Concat

6.

Crie dois dataframes:

```python
d1 = pd.DataFrame({
    'ID': [1, 2, 3],
    'Produto': ['A', 'B', 'C']
})
```

```
df_d1 = pd.DataFrame(d1)

d2 = pd.DataFrame({
    'ID': [2, 3, 4],
    'Preço': [10, 20, 15]
})

df_d2 = pd.DataFrame(d2)
```

```
df_d1
```

border="1" class="dataframe">

	ID	Produto
0	1	A
1	2	B
2	3	C

```
df_d2
```

border="1" class="dataframe">

	ID	Preço
0	2	10
1	3	20
2	4	15

- Faça um merge do tipo left.
- Faça um join() usando o ID como índice.
- Faça um concat() horizontal desses dois DataFrames.

```
# Faça um merge do tipo left
pd.merge(df_d1, df_d2, how="left")
```

border="1" class="dataframe">

	ID	Produto	Preço
0	1	A	NaN
1	2	B	10.0
2	3	C	20.0

```
# Faça um join() usando o ID como índice
```

```
# Faça um concat() horizontal desses dois DataFrames.
pd.concat([df_d1, df_d2], axis=1)
```

border="1" class="dataframe">

	ID	Produto	ID	Preço
0	1	A	2	10
1	2	B	3	20
2	3	C	4	15

Parte 6 — Séries Temporais

7.

Crie uma série de datas mensais de '2020-01-01' até '2020-12-01':

- Crie uma coluna 'Vendas' com valores de 100 a 1200.
- Calcule o total anual de vendas.
- Crie uma coluna 'Trimestre' indicando o trimestre de cada data.
- Filtre apenas as vendas do segundo trimestre.

```
datas = pd.date_range(start='2020-01-01', end='2020-12-01', freq='MS')
vendas = list(range(100, 1300, 100))

df5 = pd.DataFrame({'Datas': datas, 'Vendas': vendas})
```

```
df5
```

border="1" class="dataframe">

	Datas	Vendas
0	2020-01-01	100
1	2020-02-01	200
2	2020-03-01	300
3	2020-04-01	400
4	2020-05-01	500
5	2020-06-01	600
6	2020-07-01	700
7	2020-08-01	800
8	2020-09-01	900
9	2020-10-01	1000
10	2020-11-01	1100
11	2020-12-01	1200

```
# Calcule o total anual de vendas.
# df5.resample('YE').sum() # OU
```

```
print(f"Total de Vendas Anual: {df5['Vendas'].sum()}")
```

```
Total de Vendas Anual: 7800
```

```
# Crie uma coluna 'Trimestre' indicando o trimestre de cada data.
df5['Trimestre'] = df5['Datas'].dt.quarter
```

Explicando o código:

`.dt` : Esse é um acessador especial do Pandas usado para extrair informações específicas de colunas com datas (tipo datetime64). Você só pode usar .dt se a coluna for do tipo datetime

`.quarter` : É um atributo (sem parênteses!) que devolve o número do trimestre para cada data da Series.

>

Mês	Trimestre (`quarter`)
Jan, Fev, Mar	1
Abr, Mai, Jun	2
Jul, Ago, Set	3
Out, Nov, Dez	4

```
df5
```

border="1" class="dataframe">

	Datas	Vendas	Trimestre
0	2020-01-01	100	1
1	2020-02-01	200	1
2	2020-03-01	300	1
3	2020-04-01	400	2
4	2020-05-01	500	2
5	2020-06-01	600	2
6	2020-07-01	700	3
7	2020-08-01	800	3
8	2020-09-01	900	3
9	2020-10-01	1000	4
10	2020-11-01	1100	4
11	2020-12-01	1200	4

```
# Filtre apenas as vendas do segundo trimestre.
df5[df5['Trimestre'] == 2]
```

border="1" class="dataframe">

	Datas	Vendas	Trimestre
3	2020-04-01	400	2
4	2020-05-01	500	2
5	2020-06-01	600	2

Parte 7 — Operações com DataFrames

8.

Crie o seguinte DataFrame:

>

Produto	Custo	Venda
A	50	80
B	30	60
C	40	90

- Calcule a coluna Lucro = Venda - Custo.
- Calcule a margem de lucro (%).
- Filtre produtos com margem acima de 50%

```
dados5 = {
    'Produto': ['A', 'B', 'C'],
    'Custo': [50, 30, 40],
    'Venda': [80, 60, 90]
}

df6 = pd.DataFrame(dados5)
```

```
df6
```

border="1" class="dataframe">

	Produto	Custo	Venda
0	A	50	80
1	B	30	60
2	C	40	90

```
# Calcule a coluna Lucro = Venda - Custo.
df6['Lucro'] = df6['Venda'] - df6['Custo']
```

```
df6
```

border="1" class="dataframe">

	Produto	Custo	Venda	Lucro
0	A	50	80	30
1	B	30	60	30
2	C	40	90	50

```
# Calcule a margem de lucro (%).
# Calculo para a Margem (%) = (Lucro / Custo) * 100

df6['Margem_Lucro (%)'] = (df6['Lucro'] / df6['Custo']) * 100
```

```
df6
```

border="1" class="dataframe">

	Produto	Custo	Venda	Lucro	Margem_Lucro (%)
0	A	50	80	30	60.0
1	B	30	60	30	100.0
2	C	40	90	50	125.0

```
# Filtre produtos com margem acima de 50%
df6[df6['Margem_Lucro (%)'] > 50]
```

border="1" class="dataframe">

	Produto	Custo	Venda	Lucro	Margem_Lucro (%)
0	A	50	80	30	60.0
1	B	30	60	30	100.0
2	C	40	90	50	125.0

Parte 8 — Entrada/Saída de Dados

9.

Salve o DataFrame acima como:

- CSV: 'lucro.csv'
- Excel: 'lucro.xlsx'

Depois:

- Leia novamente o arquivo CSV.
- Verifique o conteúdo lido.

```
# Salvando o DataFrame (df6) em um arquivo CSV:
df6.to_csv('lucro_csv.csv', index=False)
```

```
# Salvando o DataFrame (df6) em um arquivo EXCEL(xlsx):
df6.to_excel('lucro_excel.xlsx', index=False)
```

```
# Lendo novamente os dados exportandos (em csv)
dados_importados = pd.read_csv('lucro_csv.csv')
```

```
# Verificando o conteúdo
dados_importados
```

border="1" class="dataframe">

	Produto	Custo	Venda	Lucro	Margem_Lucro (%)
0	A	50	80	30	60.0
1	B	30	60	30	100.0
2	C	40	90	50	125.0

BÔNUS — Desafio Final Completo

Crie um programa que:

- Leia um CSV com dados fictícios de clientes: 'clientes.csv' com as colunas ID, Nome, Data de Nascimento, Cidade, Compra Total.
- Calcule a idade dos clientes.
- Filtre clientes com compras acima de R$5000.
- Agrupe por cidade e calcule a soma das compras.
- Salve o resultado como 'relatorio_compras.xlsx'.