Workshop Python Basic {

[Aula 3]

```
print(“Leitura e escrita em arquivos \n
Tratamento de Dados \n
Mexendo com diretórios”)
```

}

Aula 3.py

01 {

[Leitura e Escrita]

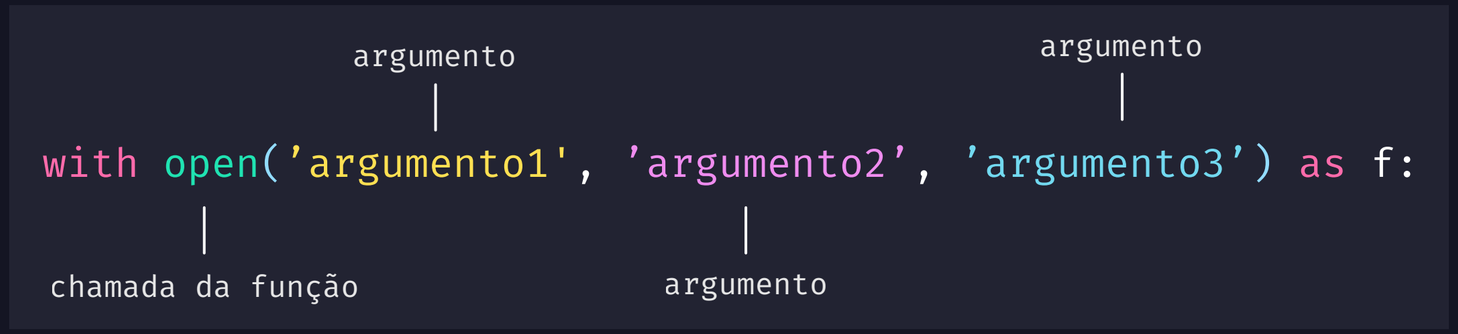
Como manipular arquivos

}

Aula 3.py

Leitura {

função open()

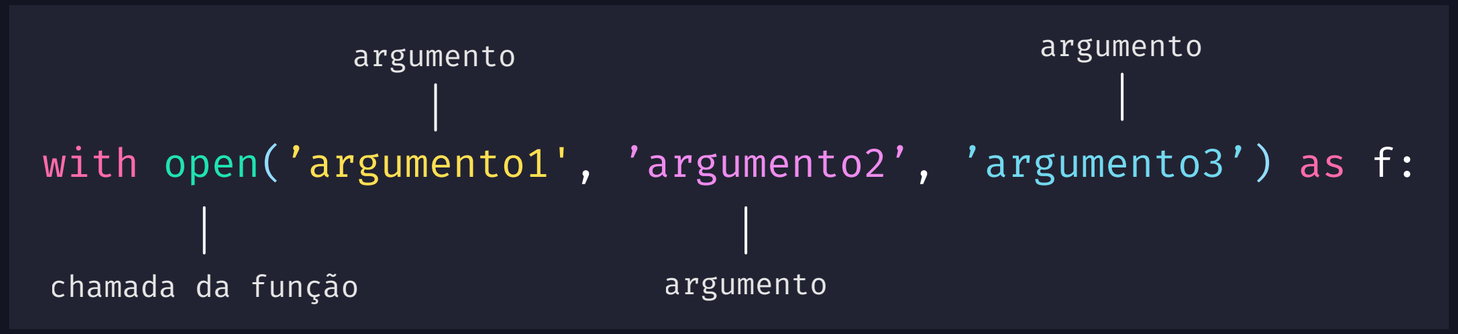



```
argumento1: Nome do arquivo: ‘arquivo.txt’
argumento2: Modo de operação: ‘r’
argumento3: Interpretação dos bytes: encoding=”utf-8”
with ... as ... → Gerenciador de Contexto

linhas = f.read()
linhas = f.readline()
linhas = f.readlines()[2]
}
```

Aula 3.py

Escrita {

função open()



argumento1: Nome do arquivo: ‘arquivo.txt’
argumento2: Modo de operação: ‘w’ ou 'a'
argumento3: Interpretação dos bytes: encoding=”utf-8”
with ... as ... → Gerenciador de Contexto

'w' - Write → Apaga todo o conteúdo existente
'a' - Append → Acrescenta apenas

}

Aula 3.py

Escrever no final {

```
with open('argumento1', 'a', encoding="utf-8") as f:
    f.write("TEXTO")
    f.write("\n")
```

}

Escrever no índice 2 {

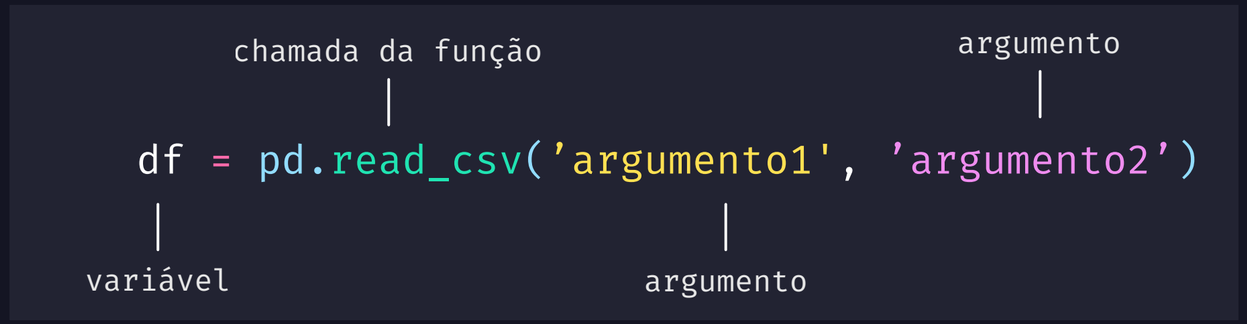
```
with open('argumento1', 'r', encoding="utf-8") as f:
    linha = f.readlines()

linha.insert(2, "TEXTO, TEXTO\n")

with open('argumento1', 'w', encoding="utf-8") as f:
    f.writelines(linha)
```

}

Aula 3.py

Arquivos tabulares {

```
Import pandas as pd
```



argumento1: Nome do arquivo: ‘arquivo.csv’
argumento2: Interpretação dos bytes: encoding=”utf-8”

}

Aula 3.py

Salvar dados {

- pd.loc

```
df.loc[len(df)] = [‘valor1’]
```

- pd.concat

```
novo = pd.DataFrame([{‘coluna1’: ‘valor1‘}])
df = pd.concat([df, novo], ignore_index=True)
```

}

Salvar arquivo {

- df.to_csv()

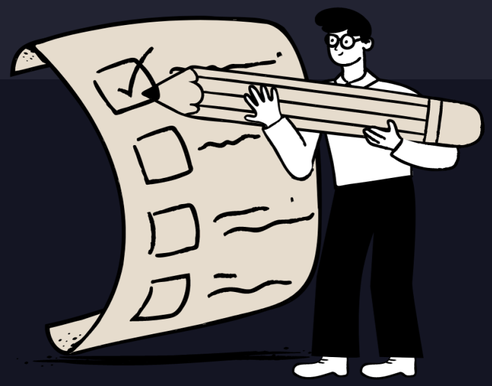
```
df = pd.to_csv([‘novo_arquivo.csv’, index=False)
```

}

Aula 3.py

```
Buscar valores {

  Coluna específica {
    df['idade']
  }

  Valor específico {
    df[df['genero'] == 'Feminino']
  }

  Condição {
    df[(df['idade'] < 14)]
    df[(df['idade'] < 14) & (df['cidade'] == 'Belém')]
  }
}
```

Aula 3.py



01.1 {

[Hora de praticar!]

Praticando que se aprende!

}

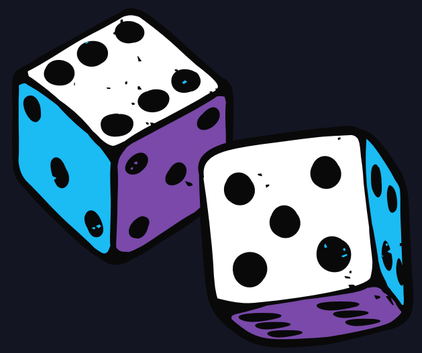
Aula 3.py

02 {

[Tratamento de Dados]

Lidando com erros e valores incorretos

}



Aula 3.py

Dataset ruim! {

Quando temos um arquivo com valores pra extrair informações, devemos sempre organizá-los.

- Verificar todo o dataset
- Padronizar
- Tratar valores nulos
- Tratar valores duplicados
- Tratar valores incorretos

}

Aula 3.py

Dataset ruim! {

Quando temos um arquivo com valores pra extrair informações, devemos sempre organizá-los.

- Verificar todo o dataset
- Padronizar
- Tratar valores nulos
- Tratar valores duplicados
- Tratar valores incorretos

```
for i in df.columns:
  print(f"Valores únicos em '{i}':", df[i].unique())
```

}

Aula 3.py

Dataset ruim! {

Quando temos um arquivo com valores pra extrair informações, devemos sempre organizá-los.

- Verificar todo o dataset
- Padronizar
- Tratar valores nulos
- Tratar valores duplicados
- Tratar valores incorretos

```
for i in df.columns:
  print(f"Valores únicos em '{i}':", df[i].unique())

df = df.map(lambda s: s.lower() if type(s) == str else s)
```

}

Aula 3.py

Dataset ruim! {

Quando temos um arquivo com valores pra extrair informações, devemos sempre organizá-los.

- Verificar todo o dataset
- Padronizar
- Tratar valores nulos
- Tratar valores duplicados
- Tratar valores incorretos

```
for i in df.columns:
  print(f"Valores únicos em '{i}':", df[i].unique())
```

```
df = df.map(lambda s: s.lower() if type(s) == str else s)
```

```
df = df.dropna()
```

}

Aula 3.py

Dataset ruim! {

Quando temos um arquivo com valores pra extrair informações, devemos sempre organizá-los.

- Verificar todo o dataset

```
for i in df.columns:
  print(f"Valores únicos em '{i}':", df[i].unique())
```

- Padronizar

```
df = df.map(lambda s: s.lower() if type(s) == str else s)
```

- Tratar valores nulos

```
df = df.dropna()
```

- Tratar valores duplicados

```
df = df.drop_duplicates()
```

- Tratar valores incorretos

}

Aula 3.py

Dataset ruim! {

Quando temos um arquivo com valores pra extrair informações, devemos sempre organizá-los.

- Verificar todo o dataset

```
for i in df.columns:
  print(f"Valores únicos em '{i}':", df[i].unique())
```

- Padronizar

```
df = df.map(lambda s: s.lower() if type(s) == str else s)
```

- Tratar valores nulos

```
df = df.dropna()
```

- Tratar valores duplicados

```
df = df.drop_duplicates()
```

- Tratar valores incorretos →

}

Aula 3.py

Tratar valores incorretos {

```
correcao = {"valor_errado1": "valor_errado1", ...}

1 - df['coluna'] = df['coluna'].replace(correcao)

2 - df['coluna'] = df['coluna'].map(lambda x: correcao.get(x,x))
```

}

Aula 3.py

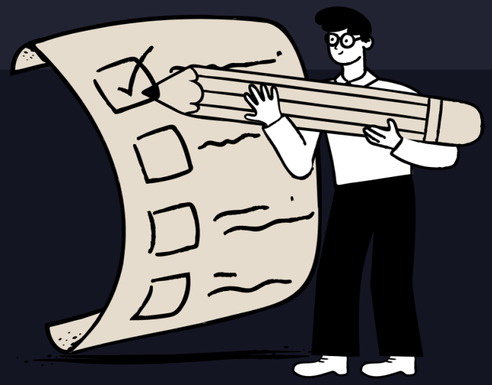
Tratar valores incorretos {

```
correcao = {"valor_errado1": "valor_errado1", ...}

1 - df['coluna'] = df['coluna'].replace(correcao)

2 - df['coluna'] = df['coluna'].map(lambda x: correcao.get(x,x))
```

```
def corrigir(bairro):
    if "maranbaia" in bairro:
        return "marambaia"
    else:
        return bairro

3 - df['coluna'] = df['coluna'].apply(corrigir)
```

}

Aula 3.py



02.1 {

[Hora de praticar!]

Que tal um minuto de cientista de dados?

}

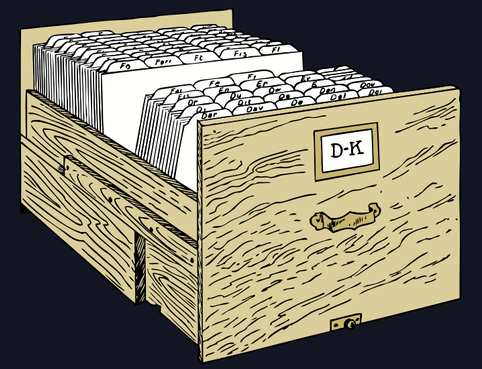
Aula 3.py

03 {

[Mexendo com diretórios]

Listagem e busca

}



Aula 3.py

Árvore de um Diretório {

Aprender a buscar arquivos é super importante em projetos, sejam grandes ou pequenos.

- Listar arquivos de uma pasta
- Processar vários arquivos automaticamente
- Mudar nomes de arquivos
- Salvar em locais próprios
- Encontrar arquivos por extensão ou nome

}

Aula 3.py

Módulos os ou glob {

```python
import os

pasta = "PASTA"

arquivos = os.listdir(pasta)

print("Arquivos encontrados:")
for arq in arquivos:
    print(arq)
```

}

```python
import glob

arquivos = glob.glob("PASTA/*")

print(arquivos)
```

→ **Buscar um só tipo de arquivo (CSV):**

```python
arquivos_csv = glob.glob("PASTA/*.csv")
print(arquivos_csv)
```

→ **Buscar todos os .CSVs**

```python
csvs = glob.glob("PASTA/**/*.csv", recursive=True)
print(csvs)
```

Aula 3.py

Renomeando e movendo com os {

```
import os

→ Renomear:

caminho_antigo = "PASTA/arquivo.txt"
caminho_novo = "PASTA/novo_nome_arquivo.txt"
os.rename(caminho_antigo, caminho_novo)


→ Mover:

os.makedirs(os.path.dirname(caminho_novo), exist_ok=True)

caminho_antigo = "PASTA/arquivo.txt"
caminho_novo = "PASTA/PASTA2/novo_nome_arquivo.txt"
os.rename(caminho_antigo, caminho_novo)

}
```

Aula 3.py

03.1 {

[Hora de praticar!]

Vamos fazer exercícios no diretório

}