PET COMPUTAÇÃO

Aula 9 - Turma Avançado

Análise de dados com pandas



7 DE NOVEMBRO DE 2026

01.

Conhecendo o pandas



CONTEXTO

- Imagine uma planilha com as notas de 1.000 alunos.
- Frequentemente, é interessante absorver certas informações sobre esses dados, por exemplo: a média das notas, a maior e a menor nota, o perfil dos alunos, dentre outras.

CONTEXTO

- É possível resolver esse problema com estruturas que já estudamos, como o *for*, por exemplo.
- Mas, já existem maneiras mais práticas de trabalhar com essas situações!

DEFINIÇÃO

- O *pandas* é uma biblioteca da linguagem Python para trabalhar com tabelas de dados.
- Biblioteca é um conjunto de códigos prontos, escritos por outras pessoas, que podemos usar nos nossos programas.
- O nome vem de *panel data*, que significa “dados em painel”.
- Pense em uma planilha do Excel dentro do Python.
- Veja mais sobre em: *https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html*

SEM PANDAS x COM PANDAS

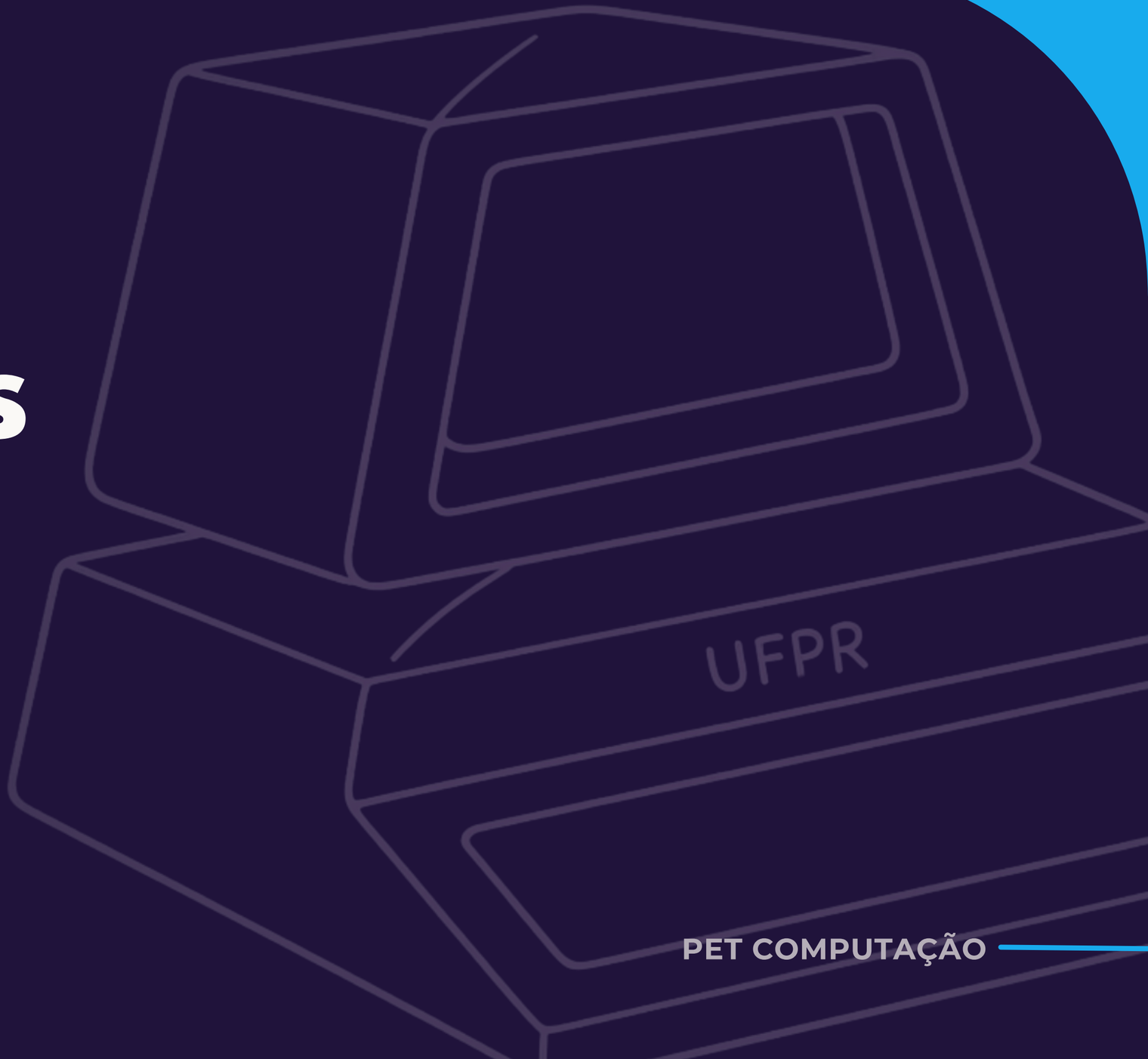
```
notas = [8.5, 7.0, 9.0, 6.5]
soma = 0
for nota in notas:
    soma = soma + nota
media = soma / len(notas)
```

```
media = tabela["nota"].mean()
```

02.

Primeiros passos

Como instalar e importar a biblioteca e trabalhar com tabelas



ONDE VAMOS PROGRAMAR

- O pandas não vem junto com o Python por padrão.
- Vamos utilizar o site Google Colab: *https://colab.research.google.com*

Importando o pandas

```
import pandas as pd
```

- *import* traz a biblioteca para o nosso programa;
- *as pd* dá um apelido ao pandas;
- A convenção recomendada é sempre usar *pd*.

O QUE É UM DATAFRAME?

- *DataFrame*: a tabela do pandas. *Data* = dados, *frame* = quadro.
- O objetivo é converter os dados numa tabela fácil de ser consultada.

	nome	país	idade
0	Ana	Brasil	23
1	Carlos	Portugal	31
2	Maria	Argentina	28

Criando um DataFrame

```
dados = {
    "nome": ["Ana", "Carlos", "Maria"],
    "pais": ["Brasil", "Portugal", "Argentina"],
    "idade": [23, 31, 28],
}
tabela = pd.DataFrame(dados)
print(tabela)
```

Saída

```
     nome        pais  idade
0     Ana      Brasil     23
1  Carlos    Portugal     31
2   Maria   Argentina     28
```

- Cada chave do dicionário vira uma coluna da tabela;
- Os valores das colunas são armazenados nas listas;
- **Atenção:** todas as listas precisam ter o mesmo tamanho!

Lista de dicionários

```python
produtos = [
    {"nome": "Arroz", "preco": 25.9, "quantidade": 40},
    {"nome": "Feijão", "preco": 8.5, "quantidade": 35},
]
estoque = pd.DataFrame(produtos)
print(estoque)
```

```
     nome  preco  quantidade
0   Arroz   25.9          40
1  Feijão    8.5          35
```

- Lembram do projeto do estoque?
- Cada item é um dicionário e a lista guarda os dicionários.
- O pandas transforma tudo isso em tabela de uma vez!

03.

Lendo arquivos CSV



O QUE É UM CSV?

```
nome,país,idade,nota,turma
Ana,Brasil,23,8.5,Manhã
Carlos,Portugal,31,7.0,Noite
Maria,Argentina,28,9.0,Manhã
José,Venezuela,35,6.5,Noite
Jean,Haiti,32,8.0,Manhã
Amina,Síria,40,7.5,Noite
Luna,Colômbia,22,9.5,Manhã
Ali,Angola,30,5.5,Noite
```

- CSV = *Comma-Separated Values*: valores separados por vírgula.
- É um arquivo de texto (.csv): cada linha do arquivo é uma linha da tabela.

read_csv()

```python
alunos = pd.read_csv(
    "alunos.csv",
    encoding="utf-8")
print(alunos)
```

- *read_csv()* lê o arquivo e já cria o DataFrame.
- O arquivo só precisa estar na mesma pasta do programa.

```
      nome        país  idade  nota  turma
0      Ana      Brasil     23   8.5  Manhã
1   Carlos    Portugal     31   7.0  Noite
2    Maria   Argentina     28   9.0  Manhã
3     José   Venezuela     35   6.5  Noite
4     Jean       Haiti     32   8.0  Manhã
5    Amina       Síria     40   7.5  Noite
6     Luna    Colômbia     22   9.5  Manhã
7      Ali      Angola     30   5.5  Noite
```

Alguns dados sobre o DataFrame

```
print(alunos.head(3))
print(alunos.shape)
print(list(alunos.columns))
```

```
     nome       país  idade  nota  turma
0     Ana     Brasil     23   8.5  Manhã
1  Carlos   Portugal     31   7.0  Noite
2   Maria  Argentina     28   9.0  Manhã
(8, 5)
['nome', 'pais', 'idade', 'nota', 'turma']
```

- *head()*: seleciona as primeiras linhas da tabela (*head(3)* mostra as 3 primeiras);
- *tail()* seleciona as últimas linhas da tabela;
- *Head* = cabeça e *tail* = cauda;
- *shape*: o formato da tabela (8 linhas e 5 colunas);
- *columns*: os nomes das colunas.

describe()

```
print(alunos.describe())
```

- *Describe* = descrever. Mostra resumo dos números de cada coluna.
- *count*: quantos valores. *mean*: a média. *min* e *max*: menor e maior.
- As outras são propriedades da Estatística: desvio padrão (*std*), mediana (*50%*), primeiro quartil (*25%*) e terceiro quartil (*75%*).

```
              idade      nota
count      8.000000  8.000000
mean      30.125000  7.687500
std        5.938675  1.334635
min       22.000000  5.500000
25%       26.750000  6.875000
50%       30.500000  7.750000
75%       32.750000  8.625000
max       40.000000  9.500000
```

04.

Consultando a tabela



Escolhendo uma coluna

```
notas = alunos["nota"]
print(notas)
```

- Colchetes e o nome da coluna entre aspas.
- Assim como no dicionário, o nome da coluna é a chave!
- Uma coluna sozinha se chama *Series*.

```
0    8.5
1    7.0
2    9.0
3    6.5
4    8.0
5    7.5
6    9.5
7    5.5
Name: nota, dtype: float64
```

Escolhendo várias colunas

```
print(alunos[["nome", "nota"]])
```

- **Para dois pares de colchetes:**
- Por fora: o acesso à tabela;
- Por dentro: uma lista com os nomes das colunas.

```
     nome  nota
0     Ana   8.5
1  Carlos   7.0
2   Maria   9.0
3    José   6.5
4    Jean   8.0
5   Amina   7.5
6    Luna   9.5
7     Ali   5.5
```

Escolhendo linhas: iloc

```
print(alunos.iloc[0:3])
```

```
     nome        pais  idade  nota  turma
0     Ana      Brasil     23   8.5  Manhã
1  Carlos    Portugal     31   7.0  Noite
2   Maria   Argentina     28   9.0  Manhã
```

- *iloc* escolhe linhas pela posição, usando os índices delas.
- *0:3* pega as posições 0, 1 e 2. O 3 não entra! Mesma lógica do *range*.
- *iloc[0]* pega só a primeira linha.

Fazendo perguntas

```
print(alunos["nota"] >= 8)
```

- Operadores de comparação, o resultado é *True* ou *False*;
- O pandas seleciona as linhas que têm nota maior ou igual a 8.

```
0     True
1    False
2     True
3    False
4     True
5    False
6     True
7    False
Name: nota, dtype: bool
```

Filtrando linhas

```
bons = alunos[alunos["nota"] >= 8]
print(bons)
```

```
    nome        pais  idade  nota  turma
0    Ana      Brasil     23   8.5  Manhã
2  Maria   Argentina     28   9.0  Manhã
4   Jean       Haiti     32   8.0  Manhã
6   Luna    Colômbia     22   9.5  Manhã
```

- A pergunta vai dentro dos colchetes da tabela.
- O pandas guarda só as linhas *True*.
- Note que os índices continuam como 0, 2, 4 e 6! Eles não são alterados.

Mais de uma condição

```
boa_nota = alunos["nota"] >= 8
jovem = alunos["idade"] < 30
print(alunos[boa_nota & jovem])
```

```
     nome        pais  idade  nota  turma
0     Ana      Brasil     23   8.5  Manhã
2   Maria   Argentina     28   9.0  Manhã
6    Luna    Colômbia     22   9.5  Manhã
```

- *&* significa e */* significa *ou*.
- Com tabelas, não usamos *and* e *or*.
- Se escrever as condições direto dentro dos colchetes, coloque cada uma entre parênteses.

05.

Calculando e salvando



Calculando números

```
print(alunos["nota"].mean())   # média
print(alunos["nota"].max())    # maior
print(alunos["nota"].min())    # menor
print(len(alunos))             # quantas linhas
```

```
7.6875
9.5
5.5
8
```

- *.sum()* = soma;
- *.mean()* = média;
- *.max()* = maior;
- *.min()* = menor;
- O *len()* (tamanho) também funciona com a tabela.

Criando uma coluna nova

```
alunos["nota_100"] = alunos["nota"] * 10
print(alunos.head(3))
```

```
     nome        pais  idade  nota  turma  nota_100
0     Ana      Brasil     23   8.5  Manhã      85.0
1  Carlos    Portugal     31   7.0  Noite      70.0
2   Maria   Argentina     28   9.0  Manhã      90.0
```

- Nome novo entre colchetes e =: mesmo procedimento que adicionar uma chave no dicionário.
- A conta é feita para todas as linhas de uma vez.

Ordenando a tabela

```
ranking = alunos.sort_values("nota", ascending=False)
print(ranking.head(3))
```

```
    nome        pais  idade  nota  turma
6   Luna    Colômbia     22   9.5  Manhã
2  Maria   Argentina     28   9.0  Manhã
0    Ana      Brasil     23   8.5  Manhã
```

- *Sort* = ordenar. *sort_values("nota")* ordena pela coluna nota.
- *ascending=False*: do maior para o menor. (*Ascending* = crescente.)
- A tabela original não muda. Mesma situação da função *sorted()*, para dicionários.

Salvando: to_csv()

```python
bons = alunos[alunos["nota"] >= 8]
bons.to_csv(
    "bons_alunos.csv",
    index=False,
    encoding="utf-8")
```

```
nome,pais,idade,nota,turma
Ana,Brasil,23,8.5,Manhã
Maria,Argentina,28,9.0,Manhã
Jean,Haiti,32,8.0,Manhã
Luna,Colômbia,22,9.5,Manhã
```

- *to_csv()* cria o arquivo. Se já existe, é substituído!
- *index=False*: sem isso, o pandas salva também a coluna 0, 1, 2...
- *encoding="utf-8"*: a regra da aula, sempre!

06.

Exercícios

Hora de praticar com o estoque do nosso projeto.



EXERCÍCIO 1

- Crie um programa em Python que:
- Crie uma lista com 4 produtos. Cada produto é um dicionário com nome, preco e quantidade;
- Transforme a lista em um DataFrame e imprima a tabela.

```
importar o pandas como pd
produtos = lista com 4 dicionários (nome, preco, quantidade)
estoque = DataFrame criado a partir de produtos
imprimir estoque
```

EXERCÍCIO 2

- O arquivo estoque.csv já está pronto (colunas: produto, preco, quantidade). Faça um programa que:
- Leia o arquivo com pandas e imprima as 3 primeiras linhas;
- Imprima só os produtos com menos de 20 unidades.

```
estoque = ler "estoque.csv"
imprimir as 3 primeiras linhas de estoque
pouco = estoque onde quantidade < 20
imprimir pouco
```

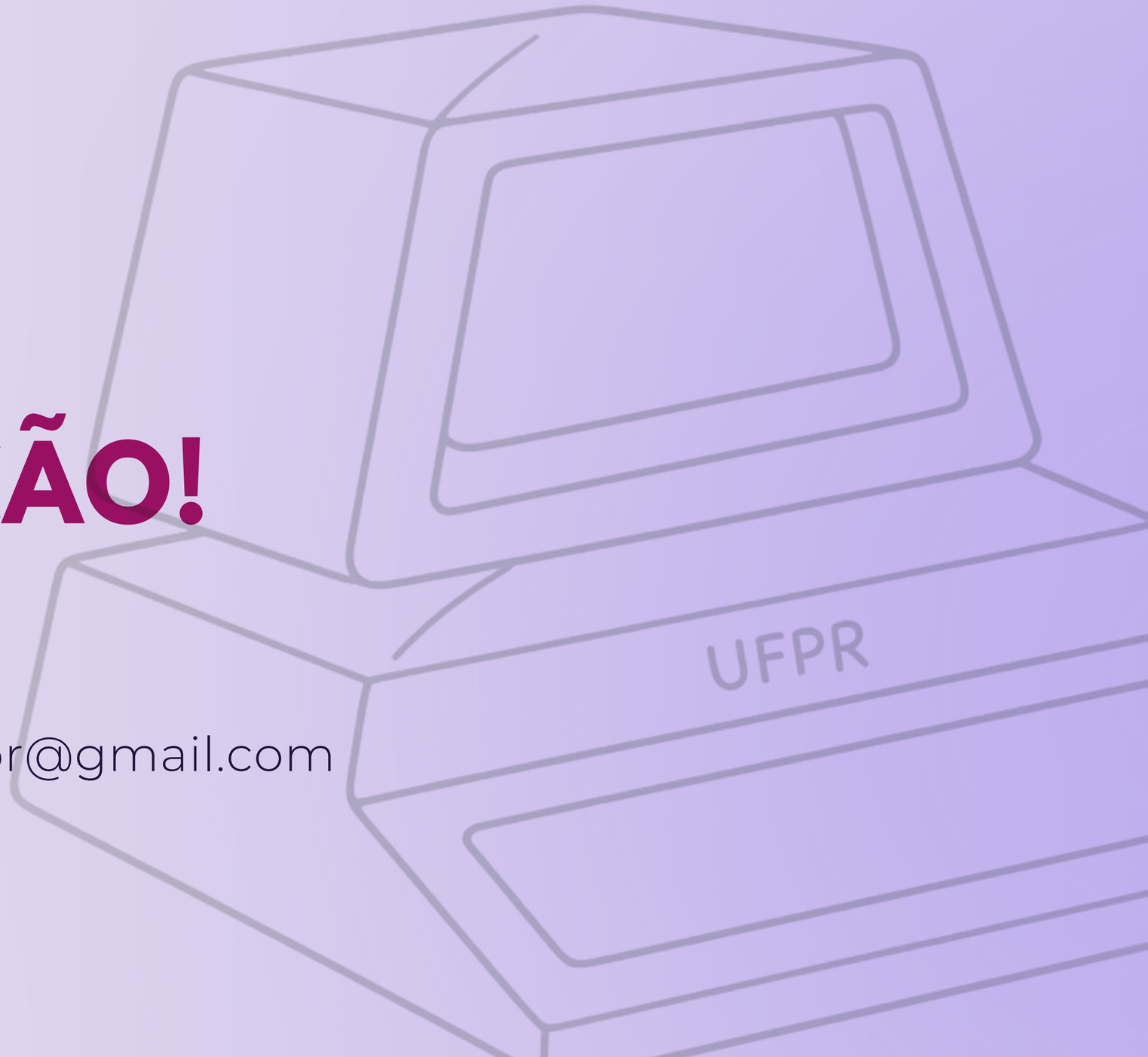
EXERCÍCIO 3

- Continue com o estoque.csv:
- Crie a coluna valor_total (preco x quantidade);
- Ordene do maior para o menor valor_total e salve em estoque_final.csv.
  Dica: *sort_values(..., ascending=False)* e *to_csv(..., index=False)*

```
estoque = ler "estoque.csv"
estoque["valor_total"] = preco vezes quantidade
ordenar estoque por valor_total, do maior para o menor
salvar em "estoque_final.csv" (sem o índice)
```

PET
COMPUTAÇÃO

OBRIGADO PELA ATENÇÃO!



Instagram: @petcompufpr
E-mail: petcomputacaoufpr@gmail.com
Site: petcomp.ufpr.br