Lendo os Dados:
Importação e Criação de Objetos

Objetivos da aula

- Nesta aula vamos realizar a leitura de dados de Filmes, transformando informações de um arquivo CSV em objetos Python.
- Ao final da aula você será capaz de:
  - Ler arquivos CSV usando **csv.reader**
  - Converter dados brutos em tipos adequados
  - Tratar dados ausentes ou inconsistentes
  - Criar objetos **Filme** a partir de dados reais
  - Armazenar objetos em listas para análise futura.

Por que usar CSV?

- CSV é um formato simples e amplamente utilizado para dados.
- Arquivos CSV são comuns em bases de dados reais e fáceis de manipular em Python.

Estrutura do Arquivo filmes.csv

- Cada linha representa um filme e cada coluna um atributo.

titulo,orcamento,receita,gênero

Matrix,63000000,465000000,Ficção

Titanic,200000000,2200000000,Drama

	A	B	C	D
1	titulo	orcamento	receita	genero
2	Matrix	63000000	465000000	Ficção
3	Titanic	200000000	2200000000	Drama

Importando o módulo csv

- **import** csv
  - O módulo csv permite ler arquivos CSV de forma estruturada.
  - Ele trata automaticamente **separadores, aspas e linhas**, facilitando a manipulação de dados tabulares no Python.

Abrindo o arquivo CSV

- Abrimos o arquivo para leitura dos dados.

```
import csv

arquivo = open("filmes.csv", encoding="utf-8")
```

Usando csv.reader

- O csv.**reader** transforma o arquivo em um iterador de linhas.

```
import csv

arquivo = open("filmes.csv", encoding="utf-8")

leitor = csv.reader(arquivo)
```

Usando delimiter no csv.reader

- O parâmetro delimiter define qual caractere separa os dados no arquivo CSV.
- Por padrão, o csv.reader usa vírgula (,).

```
import csv

arquivo = open("filmes.csv", encoding="utf-8")
leitor = csv.reader(arquivo, delimiter=';')
```

Pulando o cabeçalho

- next(leitor)
  - Ignoramos a primeira linha, que contém apenas os nomes das colunas.

```
import csv

arquivo = open("filmes.csv", encoding="utf-8")
leitor = csv.reader(arquivo, delimiter=';')
next(leitor)
```

Criando a lista de filmes

- filmes = []
  - A lista armazenará todos os objetos Filme criados.

Percorrendo as linhas do CSV

- Cada linha do CSV é lida como uma lista de strings.

```python
import csv

arquivo = open("filmes.csv", encoding="utf-8")
leitor = csv.reader(arquivo, delimiter=';')
next(leitor)

filmes = []
for linha in leitor:
    print(linha)
```

```
(3.9.9) (base) joaopauloaramuni@MacBook-Pro-de-Joao testes % python main.py
['Matrix', '63000000', '465000000', 'Ficção']
['Titanic', '200000000', '2200000000', 'Drama']
```

Extraindo os campos da linha

- Cada posição da lista corresponde a uma coluna do CSV.

```
titulo = linha[0]
orcamento = linha[1]
receita = linha[2]
genero = linha[3]
```

Strings vindas do CSV

- print(**type**(orcamento))
  - Dados lidos do CSV chegam como strings.

```
(3.9.9) (base) joaopauloaramuni@MacBook-Pro-de-Joao testes % python main.py
['Matrix', '63000000', '465000000', 'Ficção']
['Titanic', '200000000', '2200000000', 'Drama']
<class 'str'>
```

```python
for linha in leitor:
    print(linha)

titulo = linha[0]
orcamento = linha[1]
receita = linha[2]
genero = linha[3]

print(type(orcamento))
```

Convertendo orçamento para número

- Convertendo string para número para permitir cálculos.

```
print(type(orcamento))
orcamento = float(orcamento)
```

Convertendo receita para número

- A conversão é essencial para análises financeiras.

```
print(type(receita))
receita = float(receita)
```

Tratando gênero vazio

- Valores ausentes são representados por **None**.

```
if genero == "":
    genero = None
```

Criando um objeto Filme

- Cada linha do CSV gera um objeto Filme.

```
from filme import Filme

filme = Filme(titulo, orcamento, receita, genero)
```

Adicionando o filme à lista

- O objeto criado é armazenado na lista de filmes.

```
filme = Filme(titulo, orcamento, receita, genero)

filmes.append(filme)
```

Estrutura completa do loop

- O loop converte cada linha do CSV em um objeto.

```
for linha in leitor:
    titulo = linha[0]
    orcamento = float(linha[1])
    receita = float(linha[2])
    genero = linha[3] or None
    filmes.append(Filme(titulo, orcamento, receita, genero))
```

Fechando o arquivo

- arquivo.close()
  - Fechar o arquivo libera recursos do sistema.

Usando with para abrir arquivo

- O with garante que o arquivo será fechado automaticamente.

```
with open("filmes.csv", encoding="utf-8") as arquivo:
    leitor = csv.reader(arquivo, delimiter=';')
```

Tratando erros de conversão

- Tratamos dados inconsistentes sem interromper o programa.

```
try:
    orcamento = float(linha[1])
except ValueError:
    orcamento = 0
```

Ignorando linhas inválidas

- Linhas incompletas são descartadas.

```
if len(linha) < 4:
    continue
```

For completo

- Com a validação de tamanho da linha e tratamento de erro na conversão.

```
for linha in leitor:
    if len(linha) < 4:
        continue

    titulo = linha[0]

    try:
        orcamento = float(linha[1])
    except ValueError:
        orcamento = 0

    try:
        receita = float(linha[2])
    except ValueError:
        receita = 0

    genero = linha[3] or None

    filmes.append(Filme(titulo, orcamento, receita, genero))
```

Exibindo filmes carregados

- O método __str__ facilita a visualização dos dados.

```
for filme in filmes:
    print(filme)
```

```
# Classe Filme
def __str__(self):
    return f"{self.titulo} ( {self.genero} )"
```

Quantidade de filmes lidos

- print(**len**(filmes))
  - Verificamos quantos objetos foram criados.
  - Total: 2.

Acessando atributos dos objetos

- print(filmes[0].titulo)
  - Os dados agora são acessados como atributos.

Função para carregar filmes

- Encapsulamos a leitura em uma função reutilizável.

```
def carregar_filmes(caminho):
    filmes = []
    return filmes
```

Função completa de leitura

- A função retorna uma lista de objetos Filme.
- Crie um novo arquivo **leitura_csv.py** para mover esta função.

```python
def carregar_filmes(caminho):
    filmes = []
    with open(caminho, encoding="utf-8") as arquivo:
        leitor = csv.reader(arquivo, delimiter=';')
        next(leitor)
        for linha in leitor:
            filmes.append(Filme(linha[0], float(linha[1]), float(linha[2]), linha[3] or None))
    return filmes
```

Usando a função no main.py

- O código principal fica mais limpo e organizado.

```python
from leitura_csv import carregar_filmes

filmes = carregar_filmes("filmes.csv")
```

Testando a leitura de dados

```python
from leitura_csv import carregar_filmes
from filme import Filme

def test_carregar_filmes():
    filmes = carregar_filmes("filmes.csv")

    # Verifica se retornou uma lista
    assert isinstance(filmes, list)

    # Verifica se há pelo menos um objeto Filme
    assert len(filmes) > 0

    # Verifica se todos os elementos são instâncias de Filme
    for filme in filmes:
        assert isinstance(filme, Filme)

    # Verifica um atributo específico de exemplo
    assert filmes[0].titulo != ""
```

Dados prontos para análise

- Os dados agora estão convertidos, organizados e acessíveis em listas de objetos, prontos para cálculos, visualizações e modelagem preditiva.

```
# Os objetos permitem cálculos diretos.
lucros = [ f.lucro() for f in filmes ]

# Extraímos dados para visualização.
orcamentos = [ f.orcamento for f in filmes ]

# Os dados numéricos estão prontos para modelos matemáticos.
receitas = [ f.receita for f in filmes ]
```

Encerramento

- Você aprendeu:
  - Como ler dados reais de filmes a partir de um arquivo CSV
  - Converter strings em valores numéricos
  - Tratar dados ausentes usando **None**
  - Criar listas de objetos **Filme**
  - Separar leitura de dados do restante do sistema
  - Preparar a base para análises, visualizações e predições

PUC Minas