Armazenando dados

Muitos de seus programas pedirão aos usuários que forneçam determinados tipos de informação. Você pode permitir que os usuários armazenem suas preferências em um jogo ou forneçam dados para uma visualização.

Qualquer que seja o foco de seu programa, você armazenará as informações fornecidas pelos usuários em estruturas de dados como listas e dicionários. Quando os usuários fecham um programa, quase sempre você vai querer salvar as informações que eles forneceram. Uma maneira simples de fazer isso envolve armazenar seus dados usando o **módulo json** ou o **módulo csv**.

JSON

É um formato leve de intercâmbio de dados, baseado em texto, usado para armazenar e transferir informações comumente entre um servidor e uma aplicação.

CSV (Comma-Separated Values)

É um outro formato de arquivo de texto simples que armazena dados estruturados em tabela, onde cada linha é um registro e as colunas são separadas por vírgulas (como o nome sugere).

Módulo JSON

No Python existe um módulo que nos auxilia a tratar arquivos JSON. Assim podemos descaregar estruturas de Dados Python simples em um arquivo e carregar os dados desse arquivo em uma próxima vez que o programa executar.

> O formato de dados JSON não é específico de Python, portanto podemos compartilhar dados armazenados em formato JSON com pessoas que trabalhem com várias outras linguagens de programação. É um formato útil e portável, além de ser fácil de aprender.

Exemplo:

```
{
  "nome": "João",
  "idade": 20,
  "ativo": true
}
```

Vamos começar a prática, ou seja, salvar dados em JSON:

```
import json # Importando a biblioteca para trabalhar com JSON
```

json.dump

Escrever em arquivos json

```
dados = {
    "nome": "João",
    "idade": 20,
    "cursos": ["Python", "SQL"]
}

with open('dados.json', 'w', encoding='utf-8') as arquivo:
  json.dump(dados, arquivo, ensure_ascii=False, indent=4)

# indent=4 → deixa o JSON legível
# ensure_ascii=False → mantém acentos
```

json.load

Ler arquivos json

```
# Lendo o arquivo

with open('dados.json', 'r', encoding='utf-8') as arquivo:
  dados_recebidos = json.load(arquivo)


print(dados_recebidos)
```

```
print(dados_recebidos['nome'])
print(dados_recebidos['idade'])
print(dados_recebidos['cursos'])
```

```
{'nome': 'João', 'idade': 20, 'cursos': ['Python', 'SQL']}
João
20
['Python', 'SQL']
```

Salvando e lendo dados gerados pelo usuário

Salvar dados com json é conveniente quando trabalhamos com dados gerados pelo usuário porque, se você não salvar as informações de seus usuários de algum modo, elas serão perdidas quando o programa parar de executar.

O exemplo abaixo fara com que, caso seja a primeira vez do programa rodando, perguntará ao usuário seu nome, e apresentará uma mensagem especifica. Caso o programa já tenha sido rodado, ele apresentará outra mensagem contendo o nome do usuário anteriormente armazenada:

```
import json

filename = 'username.json'

try:
  with open(filename) as f_obj:
    username = json.load(f_obj)
except FileNotFoundError:
  username = input("Qual o seu nome? ")
  with open(filename, 'w') as f_obj:
    json.dump(username, f_obj)
    print(f" Irei lembrar do seu nome, {username}!")
else:
  print(f"Bem vindo de volta {username}")
```

```
Bem vindo de volta Kaio
```

Fazendo um outro exemplo

```
import json

with open('exemplo3.json', 'r', encoding='utf8') as arquivo:
  pessoa = json.load(arquivo)
  print(f'{pessoa}\n')

  # Aqui iremos percorrer o arquivo JSON que contém diferentes formatos
  # de armazenamento de dados
  for c, v in pessoa.items():
    print(f'{c}:{v}')
```

```
{'nome': 'Luiz Otávio 2', 'sobrenome': 'Miranda', 'enderecos': [{'rua': 'R1', 'numero': 32}, {'rua': 'R2', 'numero': 55}], '

nome:Luiz Otávio 2
sobrenome:Miranda
enderecos:[{'rua': 'R1', 'numero': 32}, {'rua': 'R2', 'numero': 55}]
altura:1.8
numeros_preferidos:[2, 4, 6, 8, 10]
dev:True
nada:None
```

Módulo CSV

Como já vimos anteriormente, CSV é outro jeito que podemos trabalhar com arquivos em Python.

CSV significa Comma Separated Values (valores separados por vírgula).

Exemplo de um CSV:

```
nome,idade,cidade
João,20,São Paulo
Maria,22,Rio de Janeiro
```

```
# importando o módulo CSV
import csv
```

Escrevendo em CSV (writer)

```
with open("pessoas.csv", "w", newline="", encoding="utf-8") as arquivo:
    escritor = csv.writer(arquivo)
    escritor.writerow(["nome", "idade", "cidade"])
    escritor.writerow(["João", 20, "São Paulo"])
    escritor.writerow(["Maria", 22, "Rio de Janeiro"])

# newline="" evita linhas em branco no Windows
# Cada writerow é uma linha do CSV
```

Lendo CSV (reader)

```
with open("pessoas.csv", "r", encoding="utf-8") as arquivo:
    leitor = csv.reader(arquivo)
    for linha in leitor:
        print(linha)

# A saída será toda em string!
```

```
['nome', 'idade', 'cidade']
['João', '20', 'São Paulo']
['Maria', '22', 'Rio de Janeiro']
```

CSV com dicionários (DictWriter e DictReader)

DictReader e DictWriter foram projetadas para ler e escrever arquivos CSV mapeando dados diretamente para dicionários, em vez de listas.

Eles facilitam a manipulação de dados baseados em colunas, usando os cabeçalhos como chaves, o que torna o código mais legível e robusto, especialmente para tabelas grandes.

Escrita

```
import csv

with open("alunos.csv", "w", newline="", encoding="utf-8") as arquivo:
    campos = ["nome", "nota"]
    escritor = csv.DictWriter(arquivo, fieldnames=campos)

    escritor.writeheader()
    escritor.writerow({"nome": "Ana", "nota": 9})
    escritor.writerow({"nome": "Carlos", "nota": 7})
```

- *Fieldnames*: É necessário definir fieldnames (uma lista com os nomes das colunas) para indicar quais chaves do dicionário serão escritas. Se não especificado, usa a primeira linha do CSV como cabeçalho.
- *writeheader()*: Método essencial do DictWriter que escreve a primeira linha de cabeçalho com base nos fieldnames definidos.
- *writerow() / writerows():* Métodos para escrever uma ou múltiplas linhas (dicionários) no arquivo.

Leitura

```
with open("alunos.csv", "r", encoding="utf-8") as arquivo:
    leitor = csv.DictReader(arquivo)
    for linha in leitor:
        print(linha["nome"], linha["nota"])
```

```
Ana 9
Carlos 7
```