

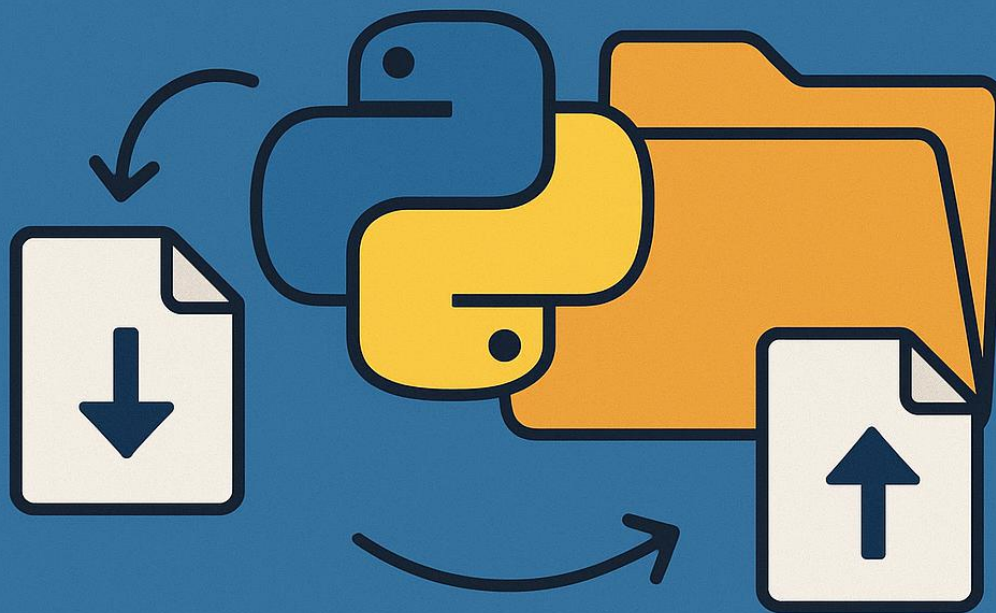
PROGRAMAÇÃO I

Curso Técnico Subsequente em Informática
Lucas Sampaio Leite



Manipulação de arquivos

FILE MANIPULATION PYTHON



Manipulação de arquivos

- Até este momento, os dados utilizados nos programas desenvolvidos na disciplina são inseridos diretamente no código ou digitados pelo usuário durante a execução.
- No contexto real, os dados geralmente estão armazenados em arquivos, disponíveis na web ou em bancos de dados.
- A manipulação de arquivos em Python constitui uma habilidade fundamental e amplamente empregada em diversos tipos de projetos.
- Essa prática permite realizar operações como leitura de dados existentes, gravação de novos conteúdos e atualização das informações já presentes em arquivos.

Manipulação de arquivos

- Em Python, é necessário abrir (open) os arquivos antes de utilizá-los e fechá-los (close) após o término do uso.
- Uma vez aberto, o arquivo passa a ser tratado como um objeto Python, de forma semelhante a outros tipos de dados.
- A leitura de um arquivo consiste em extrair as informações contidas nele para carregá-las na memória.
- Na escrita em arquivos, o processo ocorre de modo inverso: os dados são gravados no arquivo a partir das informações já existentes na memória ou fornecidas pelo usuário durante a execução.

Manipulação de arquivos

Método	Uso	Explicação
open	<code>open(path_arquivo, 'r')</code>	Abre um arquivo chamado <code>nome_arquivo</code> e o usa para leitura. Retorna uma referência para um objeto file.
open	<code>open(path_arquivo, 'w')</code>	Abre um arquivo chamado <code>nome_arquivo</code> e o usa para escrita. Retorna uma referência para um objeto file.
close	<code>ref_arquivo.close()</code>	Fecha um arquivo que foi aberto com a função <code>open()</code> , garantindo que todos os recursos sejam liberados.

Manipulação de arquivos

- Os arquivos são localizados no disco por meio de seu caminho (path).
- Quando o arquivo de dados e o programa Python estão no mesmo diretório, é possível utilizar apenas o nome do arquivo (caminho relativo).
 - Exemplo: `open('arquivo.txt', 'r')`
- Quando o arquivo de dados e o programa Python estão em diretórios diferentes, é necessário utilizar o caminho completo até o arquivo (caminho absoluto).
 - Exemplo: `open('/Users/Lucas/arquivo.txt', 'r')`

Manipulação de arquivos

Parameter Values

Parameter	Description
<i>file</i>	The path and name of the file
<i>mode</i>	<p>A string, define which mode you want to open the file in:</p> <div><p>"r" - Read - Default value. Opens a file for reading, error if the file does not exist</p><p>"a" - Append - Opens a file for appending, creates the file if it does not exist</p><p>"w" - Write - Opens a file for writing, creates the file if it does not exist</p><p>"x" - Create - Creates the specified file, returns an error if the file exist</p></div> <p>In addition you can specify if the file should be handled as binary or text mode</p> <div><p>"t" - Text - Default value. Text mode</p><p>"b" - Binary - Binary mode (e.g. images)</p></div>

Fonte:

[https://www.w3schools.com/python/ref_func_open.asp#:~:text=The%20open\(\)%20function%20opens,our%20chapters%20about%20File%20Handling.](https://www.w3schools.com/python/ref_func_open.asp#:~:text=The%20open()%20function%20opens,our%20chapters%20about%20File%20Handling.)

Manipulação de arquivos

- Abertura e fechamento de arquivos:

```
arquivo = open("qb_data.txt", "r")  
arquivo.close()
```

- Para abrir um arquivo, deve-se utilizar a função open().
- A variável “arquivo” armazena uma referência ao objeto do tipo file retornado por open().
- Após o uso dos dados do arquivo, é necessário fechá-lo por meio da função close().
- Uma vez fechado, qualquer tentativa de acessar “arquivo” resultará em erro.

Manipulação de arquivos

- Arquivo exemplo (quarterback):
 - Formato: First Name, Last Name, Position, Team, Completions, Attempts, Yards, TDs Ints, Comp%, Rating

Colt McCoy QB, CLE	135	222	1576	6	9	60.8%	74.5
Josh Freeman QB, TB	291	474	3451	25	6	61.4%	95.9
Michael Vick QB, PHI	233	372	3018	21	6	62.6%	100.2
Matt Schaub QB, HOU	365	574	4370	24	12	63.6%	92.0
Philip Rivers QB, SD	357	541	4710	30	13	66.0%	101.8
Matt Hasselbeck QB, SEA	266	444	3001	12	17	59.9%	73.2
Jimmy Clausen QB, CAR	157	299	1558	3	9	52.5%	58.4
Joe Flacco QB, BAL	306	489	3622	25	10	62.6%	93.6
Kyle Orton QB, DEN	293	498	3653	20	9	58.8%	87.5
Jason Campbell QB, OAK	194	329	2387	13	8	59.0%	84.5
Peyton Manning QB, IND	450	679	4700	33	17	66.3%	91.9
Drew Brees QB, NO	448	658	4620	33	22	68.1%	90.9
Matt Ryan QB, ATL	357	571	3705	28	9	62.5%	91.0
Matt Cassel QB, KC	262	450	3116	27	7	58.2%	93.0
Mark Sanchez QB, NYJ	278	507	3291	17	13	54.8%	75.3
Brett Favre QB, MIN	217	358	2509	11	19	60.6%	69.9
David Garrard QB, JAC	236	366	2734	23	15	64.5%	90.8
Eli Manning QB, NYG	339	539	4002	31	25	62.9%	85.3
Carson Palmer QB, CIN	362	586	3970	26	20	61.8%	82.4
Alex Smith QB, SF	204	342	2370	14	10	59.6%	82.1
Chad Henne QB, MIA	301	490	3301	15	19	61.4%	75.4
Tony Romo QB, DAL	148	213	1605	11	7	69.5%	94.9
Jay Cutler QB, CHI	261	432	3274	23	16	60.4%	86.3
Jon Kitna QB, DAL	209	318	2365	16	12	65.7%	88.9
Tom Brady QB, NE	324	492	3900	36	4	65.9%	111.0
Ben Roethlisberger QB, PIT	240	389	3200	17	5	61.7%	97.0

Manipulação de arquivos

- Iterando sobre as linhas de um arquivo:
 - Arquivos de texto são compostos por linhas de conteúdo, que podem ser percorridas usando um laço for.
- Extraíndo informações de cada linha:
 - O método `split()` permite dividir cada linha em uma lista de campos, facilitando a manipulação dos dados de interesse, como informações de um quarterback.

```
arquivo = open("qb_data.txt", "r")
for linha in arquivo:
    valores = linha.split()
    print(f"QB {valores[0]} {valores[1]} obteve a avaliação {valores[10]}")
arquivo.close()
```

Manipulação de arquivos

- Gerenciadores de contexto (with):
 - Garantem que recursos, como arquivos, sejam abertos e fechados automaticamente, mesmo em caso de erro.

```
# file handling
```

```
# 1) without using with statement  
file = open('file_path', 'w')  
file.write('hello world !')  
file.close()
```

```
# 2) without using with statement  
file = open('file_path', 'w')  
try:  
    file.write('hello world')  
finally:  
    file.close()
```

```
# using with statement  
with open('file_path', 'w') as file:  
    file.write('hello world !')
```

Manipulação de arquivos

- Gerenciadores de contexto (with):
 - Garantem que recursos, como arquivos, sejam abertos e fechados automaticamente, mesmo em caso de erro.

```
# file handling
```

```
# 1) without using with statement  
file = open('file_path', 'w')  
file.write('hello world !')  
file.close()
```

```
# 2) without using with statement  
file = open('file_path', 'w')  
try:  
    file.write('hello world')  
finally:  
    file.close()
```

Melhor forma



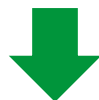
```
# using with statement  
with open('file_path', 'w') as file:  
    file.write('hello world !')
```

Saiba mais sobre gerenciadores de contexto:
<https://medium.com/@sasidharan01/understanding-and-implementing-python-context-managers-8e45884dfe14>

Manipulação de arquivos

- Iterando sobre as linhas de um arquivo:

```
with open("qb_data.txt", "r") as arquivo:
    for linha in arquivo:
        valores = linha.split()
        print(f"QB {valores[0]} {valores[1]} obteve a avaliação {valores[10]}")
```



```
QB Colt McCoy obteve a avaliação 74.5
QB Josh Freeman obteve a avaliação 95.9
QB Michael Vick obteve a avaliação 100.2
QB Matt Schaub obteve a avaliação 92.0
QB Philip Rivers obteve a avaliação 101.8
QB Matt Hasselbeck obteve a avaliação 73.2
QB Jimmy Clausen obteve a avaliação 58.4
QB Joe Flacco obteve a avaliação 93.6
```

Manipulação de arquivos

- Iterando sobre as linhas de um arquivo:

```
with open("qb_data.txt", "r") as arquivo:  
    linha = arquivo.readline()  
    while linha:  
        valores = linha.split()  
        print(f"QB {valores[0]} {valores[1]} obteve a avaliação {valores[10]}")  
        linha = arquivo.readline()
```



```
QB Colt McCoy obteve a avaliação 74.5  
QB Josh Freeman obteve a avaliação 95.9  
QB Michael Vick obteve a avaliação 100.2  
QB Matt Schaub obteve a avaliação 92.0  
QB Philip Rivers obteve a avaliação 101.8  
QB Matt Hasselbeck obteve a avaliação 73.2
```

Manipulação de arquivos

- Alguns métodos para iteração em arquivos:
 - read() lê todo o conteúdo do arquivo de uma vez como uma única string.

```
with open("qb_data.txt", "r") as arquivo:  
    conteudo = arquivo.read()  
    print(conteudo)
```

- read(n) lê apenas os primeiros n caracteres, permitindo processar partes do arquivo de forma controlada.

```
with open("qb_data.txt", "r") as arquivo:  
    conteudo = arquivo.read(10)  
    print(conteudo)
```

Manipulação de arquivos


- Alguns métodos para iteração em arquivos:
 - `readlines()` retorna uma lista de strings, em que cada elemento representa uma linha do arquivo. É útil para processar todas as linhas de uma vez sem precisar iterar com `for` sobre o arquivo.

```
with open("qb_data.txt", "r") as arquivo:  
    conteudo = arquivo.readlines()  
    print(conteudo[2])  
    print(conteudo)
```


Manipulação de arquivos

- Alguns métodos para iteração em arquivos:
 - `readlines()` retorna uma lista de strings, em que cada elemento representa uma linha do arquivo. É útil para processar todas as linhas de uma vez sem precisar iterar com `for` sobre o arquivo.

```
with open("qb_data.txt", "r") as arquivo:  
    conteudo = arquivo.readlines()  
    print(conteudo[2])  
    print(conteudo)
```



A função `readlines()` em Python também pode receber um parâmetro opcional que especifica o tamanho máximo de bytes a serem lidos de cada linha. Esse parâmetro é chamado de `sizehint`.

Manipulação de arquivos

- O método `readline()` lê apenas uma linha do arquivo por vez e a retorna como uma string.
- A string retornada mantém o caractere de nova linha (`\n`) no final, indicando a quebra de linha original.

```
with open("qb_data.txt", "r") as arquivo:
    linha_1 = arquivo.readline()
    print(linha_1)
    linha_2 = arquivo.readline().strip()
    print(linha_2)
    arquivo.seek(0)
    linha_1 = arquivo.readline().strip()
    print(linha_1)
```

Manipulação de arquivos

- O método `readline()` lê apenas uma linha do arquivo por vez e a retorna como uma string.
- A string retornada mantém o caractere de nova linha (`\n`) no final, indicando a quebra de linha original.

```
with open("qb_data.txt", "r") as arquivo:
    linha_1 = arquivo.readline()
    print(linha_1)
    linha_2 = arquivo.readline().strip()
    print(linha_2)
    arquivo.seek(0)
    linha_1 = arquivo.readline().strip()
    print(linha_1)
```

O método `seek(pos)` move o cursor de leitura/escrita para uma posição específica no arquivo.

`seek(0)` retorna o cursor para o início do arquivo, permitindo reler seu conteúdo.

Manipulação de arquivos

- Escrevendo em arquivos de texto:

```
with open("arquivo.txt", "w") as arquivo:  
    arquivo.write("IF Baiano")  
    arquivo.write(" - ")  
    arquivo.write("Senhor do Bonfim")
```


Manipulação de arquivos

- “a” = append mode. Possibilita reabertura do arquivo sem apagar o conteúdo existente.

```
with open("arquivo.txt", "w") as arquivo:  
    arquivo.write("IF Baiano")  
    arquivo.write(" - ")  
    arquivo.write("Senhor do Bonfim")  
  
with open("arquivo.txt", "a+") as arquivo:  
    arquivo.write(" - Programação I")  
    arquivo.seek(0)  
    print(arquivo.read())
```

Manipulação de arquivos

- O + é usado para combinar modos de leitura e escrita.
- Ele permite ler e escrever no mesmo arquivo, dependendo do modo base (r, w, ou a).
 - 'r+': Abre o arquivo para leitura e escrita (o arquivo deve existir).
 - 'w+': Abre para leitura e escrita, mas apaga o conteúdo existente.
 - 'a+': Abre para acrescentar (append) e ler sem apagar o conteúdo existente.

Manipulação de arquivos

- Escrevendo em arquivos de texto com writelines()
 - O método writelines() recebe um objeto iterável (como lista, tupla ou outro iterável de strings).
 - Permite escrever várias linhas de uma só vez no arquivo – cada elemento do iterável é gravado na sequência.
 - Importante: as quebras de linha (\n) devem estar incluídas nas strings, pois o método não as adiciona automaticamente.

```
with open("arquivo.txt", "w") as arquivo:  
    lista = ["banana\n", "laranja\n", "maçã\n"]  
    arquivo.writelines(lista)
```

Manipulação de arquivos

- Apagando um arquivo:

```
import os  
os.remove("arquivo.txt")
```


Exercícios

1. Crie um arquivo notas_estudantes.txt com as seguintes notas dos alunos de uma turma:

```
jose 10 15 20 30 40
pedro 23 16 19 22
suzana 8 22 17 14 32 17 24 21 2 9 11 17
gisela 12 28 21 45 26 10
joao 14 32 25 16 89
```

- a) Usando o arquivo texto notas_estudantes.txt escreva um programa que imprime o nome dos alunos que têm mais de seis notas.
- b) Usando o arquivo texto notas_estudantes.txt, escreva um programa que calcula a média das notas de cada estudante e imprime o nome e a média de cada estudante.
- c) Usando o arquivo texto notas_estudantes.txt, escreva um programa que calcula a nota mínima e máxima de cada estudante e imprima o nome de cada aluno junto com a sua nota máxima e mínima.

Exercícios

2. Escreva um programa que abra (modo a) um arquivo chamado alunos.txt e permita ao usuário digitar nomes de alunos. Cada nome deve ser adicionado em uma nova linha, sem apagar os nomes já existentes.
3. Crie um programa que grave em um arquivo chamado notas.txt as notas de 5 alunos digitadas pelo usuário, uma em cada linha. Utilize o método `writelines()` para inserir todas as linhas de uma vez.

Dúvidas



PROGRAMAÇÃO I

Curso Técnico Subsequente em Informática
Lucas Sampaio Leite

