Linguagem de Programação Python

Arquivos

Professor: Ritomar Torquato

1

Arquivos

Objetivos: Conceitos básicos; Criação e abertura de arquivos; Leitura e escrita; Processamento de dados dos arquivos.

2

Arquivos em disco

Processador — **Memória Principal** — **Memória Auxiliar**

O processador (computador) só trabalha com a memória principal. Para acessar arquivos em disco serão necessárias duas coisas na memória principal:

- Manipulador do Arquivo
- Buffer

3

Manipulador de arquivo (Handler)

Processador — **Memória Principal** — **Memória Auxiliar**

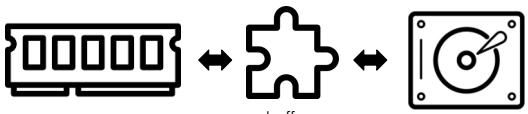
É preciso ter um **manipulador** em memória principal para o arquivo que está na memória auxiliar.

É uma variável que sabe como funciona o arquivo no disco.

4

Buffer

Espaço na memória principal (variável) que irá receber e mandar dados para o disco.



buffer

Uma variável do mesmo tipo do valor armazenado no disco: inteiro, real, string ou qualquer tipo personalizado.

É necessário para ler ou gravar dados em disco.

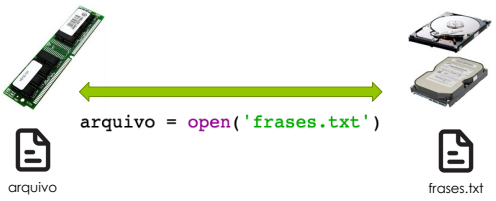
5

Arquivos

ciclo dos arquivos em memória

6

Arquivos em memória



```
arquivo = open('frases.txt')
```

arquivo

frases.txt

Abre "frases.txt" (do disco) e guarda na variável "arquivo" (em memória). Dizemos que "arquivo" é o **manipulador** de "frases.txt"

7

Modos de abertura

Modo	Operações
r	**Leitura (padrão)**
w	escrita, apaga o conteúdo existente
x	escrita, falha se o arquivo existir
a	escrita, mantém o conteúdo (append)
+	Atualização (leitura e escrita)

```
arquivo = open("frases.txt", "w")
arquivo = open("frases.txt")
```

8

Modos de abertura

Um arquivo é um monte de bytes. Quem faz a leitura deve saber tratar o que está armazenado.

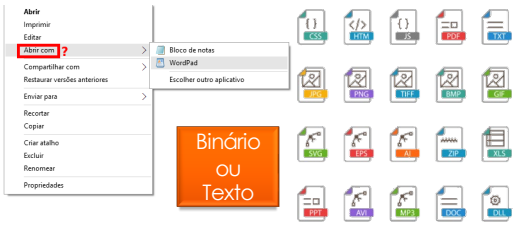
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

</body>
</html>
```

Se os bytes representam texto, ou outro tipo de dado.

9

Modos de abertura




```
arquivo = open('frases.txt')
```

A extensão lhe dá uma dica de como deve ser aberto.

10

Modos de abertura

é possível abrir arquivos como texto ou abrir como binário e dar o tratamento que desejar

Modo	Operações
t	**arquivo modo texto (padrão)**
b	arquivo modo binário

```
arquivo = open("frases.txt", "wb")
arquivo = open("frases.txt")
```

11

Exemplos de uso do **open()**

```
# abre o arquivo em modo texto para leitura
arquivo = open('frases.txt')

# equivalente ao anterior
arquivo = open('frases.txt', 'r')

# abre em modo texto para escrita, apaga o conteúdo existente
arquivo = open('frases.txt', 'w')

# abre em modo texto para escrita, falha se o arquivo já existir
arquivo = open('frases.txt', 'x')

# abre em modo texto para escrita, mantém o conteúdo existente
arquivo = open('frases.txt', 'a')

# abre o arquivo em modo binário para leitura
arquivo = open('frases.txt', 'rb')

# abre em modo binário para escrita, mantém o conteúdo existente
arquivo = open('frases.txt', 'ab')

# abre em modo texto para leitura e escrita, mantém o conteúdo existente
arquivo = open('frases.txt', 'r+')
```

12

encoding

A linguagem natural do computador é a binária (0 e 1), tudo mais é uma representação ou codificação. Assim, caracteres são codificações de números binários em forma de texto.

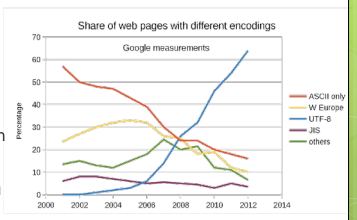
Em objetos que são armazenados de forma nativa em bytes (como arquivos), a codificação e a decodificação é feita de forma transparente, bem como a tradução de caracteres de quebra linha específicos da plataforma.
(Windows CRLF, Mac CR, Linux LF).

13

encoding

Duas codificações comuns para texto são ASCII e UTF-8.

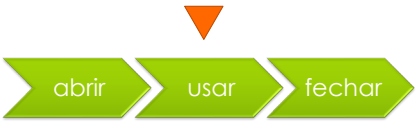
Se tiver problemas de codificação ao abrir um arquivo, é possível informar uma codificação específica ao abrir.




```
f = open("myfile.txt", "r", encoding="utf-8")
```

14

Arquivos



ciclo dos arquivos em memória

15

Geração de arquivo

```
impares = open("impares.txt", "w")
pares = open("pares.txt", "w")

for numero in range(100):
    if numero % 2 == 0:
        pares.write(f'{numero}\n')
    else:
        impares.writelines(f'{numero}')

impares.close()
pares.close()
```

Cria e deixa aberto no modo de escrita os arquivos "impares.txt"e "pares.txt" na mesma pasta do arquivo de script. Se os arquivos já existirem eles serão apagados.

As variáveis **impares** e **pares** são os manipuladores dos respectivos arquivos em disco.

16

Geração de arquivo

```
impares = open("impares.txt", "w")
pares = open("pares.txt", "w")

for numero in range(100):
    if numero % 2 == 0:
        pares.write(f'{numero}\n')
    else:
        impares.writelines(f'{numero}')

impares.close()
pares.close()
```

Usa os métodos **write()** e **writelines()** do manipulador para gravar uma informação no arquivo em disco.

Perceba que a informação é enviada como texto.

O método **write()** grava todos os números **na mesma linha** do arquivo.

A variável **numero** é o buffer. Ela que serve para intermediar a gravação dos dados.

17

Leitura e escrita

O método **readlines()** faz a leitura do arquivo, linha por linha na forma de texto, e colocar na variável número (o buffer)

```
multiplos_de_4 = open("multiplos_de_4.txt", "w")
pares = open("pares.txt")

for numero in pares.readlines():
    if int(numero) % 4 == 0:
        multiplos_de_4.write(numero)

pares.close()
multiplos_de_4.close()
```

O método **write()** escreve o buffer no arquivo de destino.

18

Processamento

entrada.txt

```
;Esta linha não deve ser impressa
>Esta linha deve ser impressa alinhada à direita
^Esta linha deve ser centralizada
Uma linha normal
>Uma linha à direta
Outra linha normal
^Uma linha centralizada
;Fim do arquivo de entrada
```

19

Processamento

```
def imprime_arquivo(nome_arquivo, largura=80):
    arquivo = open(nome_arquivo)
    for linha in arquivo.readlines():
        controle = linha[0]
        if controle == ";":
            continue
        elif controle == ">":
            print(linha[1:].rjust(largura))
        elif controle == "^":
            print(linha[1:].center(largura))
        else:
            print(linha)
    arquivo.close()
```

Neste caso, o buffer é a variável **linha**. Ela é analisada antes para ser impressa de acordo com um caractere de controle.

Se o primeiro caractere da linha (controle) for ";", ">" ou "^" é um significado especial

20

Arquivos

ciclo dos arquivos em memória

21

Fechar arquivos

Após finalizar o uso de um arquivo em disco é importante fazer seu fechamento para assegurar que os recursos alocados sejam novamente disponibilizados para o Sistema Operacional.

O método .close() do manipulador do arquivo assegura o fechamento e liberação dos recursos.

```
f = open("dados.txt", "w")
# codigo usando o arquivo f
f.close()
```

Embora correta, a forma acima não é recomendada. Erros podem acontecer durante o programa, assim, e preciso alguns cuidados ao usar o método.

22

Fechar arquivos

É preciso assegurar que o arquivo será fechado mesmo que ocorra um erro ao usar. Uma forma segura é:

```
try:
    f = open("dados.txt", "w")
    # codigo usando o arquivo f
finally:
    f.close()
```

O comando **with** ajuda ao usar recursos que precisam ser "liberados" (como arquivos) sem fazer explicitamente.

```
with open("dados.txt", "w") as f:
    # codigo usando o arquivo f
```

23