MANIPULAÇÃO DE ARQUIVOS EM PYTHON

Prof. Dieisson Martinelli
dieisson.martinelli@udesc.br

Programação

- Introdução
- Abrindo arquivos
- Lendo arquivos
- Fechando arquivos
- Exercícios

Introdução

- Programas manipulam dados que se encontram na memória do computador
  - Ex.: variáveis declaradas em um programa
- Estes dados são perdidos quando o programa é finalizado
  - **Problema**: como recuperar dados obtidos em uma execução anterior do programa?
  - **Solução**: armazenar esses dados em disco, na forma de arquivos (não volátil)
- Uma das tarefas mais importantes que realizamos no dia a dia é a manipulação de dados gravados em arquivos
  - Ex.: um administrador de sistemas precisa, com frequência, acessar informações de configuração armazenadas em arquivos de texto, e muitas vezes, efetuar alterações nesses arquivos

Introdução

- O fluxo de dados entre a aplicação e o arquivo pode ser de entrada ou de saída
  - **Fluxo de entrada**: a aplicação lê dados do arquivo e armazena em uma variável
  - **Fluxo de saída**: a aplicação escreve o valor de uma variável no arquivo

Introdução

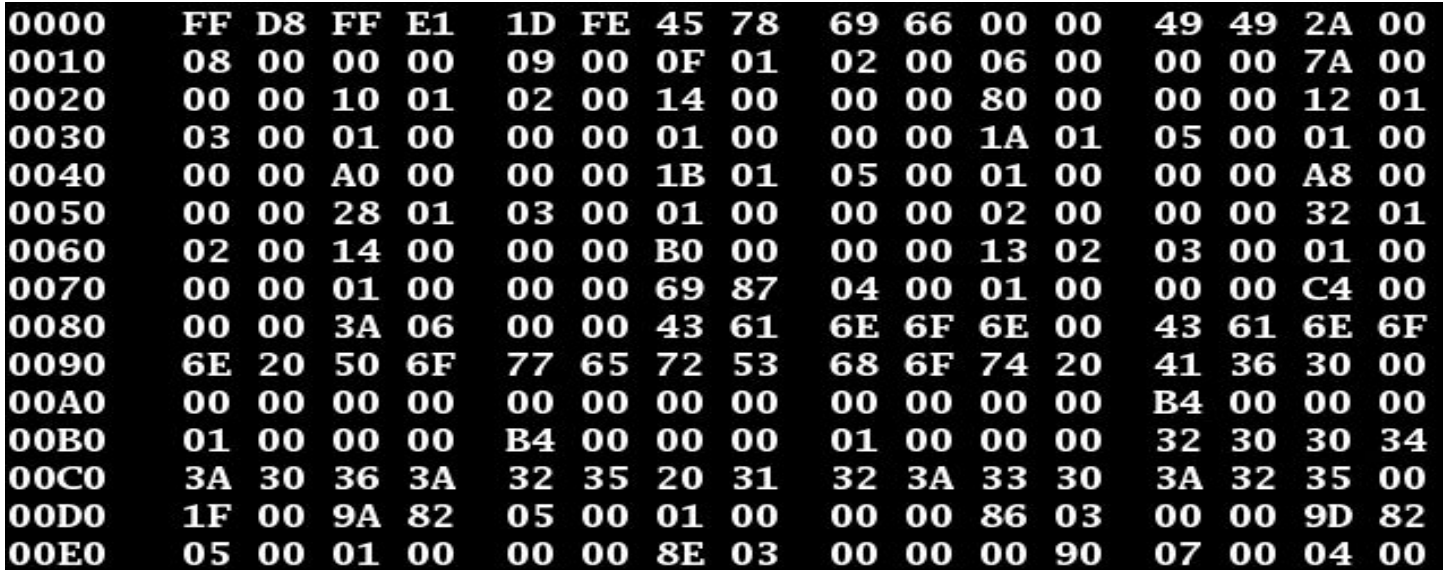
- Os arquivos podem ser classificados em arquivos de texto ou arquivos binários
- **Arquivos de texto**: são compostos por uma série de caracteres ASCII agrupados em uma ou mais linhas. Fácil de serem compreendidos pelos seres humanos

Figura: tabela ASCII

dec.	hex.	octal	ASCII	mnm.	dec.	hex.	octal	ASCII	dec.	hex.	octal	ASCII	dec.	hex.	octal	ASCII
0	00	000	^@	NUL	32	20	040		64	40	100	@	96	60	140	`
1	01	001	^A	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	02	002	^B	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	03	003	^C	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	04	004	^D	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	05	005	^E	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	06	006	^F	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	07	007	^G	BELL	39	27	047	'	71	47	107	G	103	67	147	g
8	08	010	^H	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	09	011	^I	HTAB	41	29	051	)	73	49	111	I	105	69	151	i
10	0A	012	^J	LF	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	0B	013	^K	VTAB	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	0C	014	^L	FF	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	0D	015	^M	CR	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	0E	016	^N	SO	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	0F	017	^O	SI	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	^P	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	^Q	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	^R	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	^S	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	^T	DC4	52	34	064	4	84	54	124	T	116	74	164	t

Introdução

- **Arquivos binários**: compostos por uma série de bytes representados por caracteres não compreendidos por humanos. São menores que os arquivos de texto



0000	FF	D8	FF	E1	1D	FE	45	78	69	66	00	00	49	49	2A	00
0010	08	00	00	00	09	00	0F	01	02	00	06	00	00	00	7A	00
0020	00	00	10	01	02	00	14	00	00	00	80	00	00	00	12	01
0030	03	00	01	00	00	00	01	00	00	00	1A	01	05	00	01	00
0040	00	00	A0	00	00	00	1B	01	05	00	01	00	00	00	A8	00
0050	00	00	28	01	03	00	01	00	00	00	02	00	00	00	32	01
0060	02	00	14	00	00	00	B0	00	00	00	13	02	03	00	01	00
0070	00	00	01	00	00	00	69	87	04	00	01	00	00	00	C4	00
0080	00	00	3A	06	00	00	43	61	6E	6F	6E	00	43	61	6E	6F
0090	6E	20	50	6F	77	65	72	53	68	6F	74	20	41	36	30	00
00A0	00	00	00	00	00	00	00	00	00	00	00	00	B4	00	00	00
00B0	01	00	00	00	B4	00	00	00	01	00	00	00	32	30	30	34
00C0	3A	30	36	3A	32	35	20	31	32	3A	33	30	3A	32	35	00
00D0	1F	00	9A	82	05	00	01	00	00	00	86	03	00	00	9D	82
00E0	05	00	01	00	00	00	8E	03	00	00	00	90	07	00	04	00

Figura: arquivo binário visualizado em um editor hexadecimal

Abrindo arquivos em Python

- Na programação, o primeiro processo que deve ser realizado para fazer qualquer operação sobre algum arquivo é abri-lo
- Para abrir um arquivo em Python, utiliza-se o método ***open()***. Este método retorna um objeto de arquivo e é comumente empregado com **dois argumentos**:
- ***open(filename, mode)***
  - Primeiro argumento (***filename***): é simplesmente o nome do arquivo que se deseja abrir
  - Segundo argumento (***mode***): é uma *string* que indica como o arquivo vai ser aberto

Abrindo arquivos em Python

- Modos disponíveis no Python para abrir um arquivo são mostrados na tabela:

r	Abre o arquivo de texto para leitura
r+	Abre para leitura e escrita
w	Abre apenas para gravação. Se o arquivo não existir, será criado um novo
w+	Abre para leitura e escrita. O arquivo é criado se ele não existir, caso contrário será sobrescrito
a	Abre para escrita. O arquivo é criado caso não exista
a+	Abre para leitura e escrita. O arquivo é criado se ele não existir

Abrindo arquivos em Python

- Exemplo: abrindo um arquivo com o Python
  - Criar em algum editor de texto (ex.: bloco de notas) um arquivo de texto chamado “**teste.txt**” e inserir algum conteúdo no arquivo, como “**Estrutura de Dados II**”
  - Criar um arquivo chamado “**exe_arq.py**” e inserir o seguinte código:
    - **arquivo = open(“teste.txt”, “r”)**
  - Executar o código

Lendo arquivos em Python

- O arquivo é como uma caixa secreta. Abrimos a caixa no passo anterior, e agora pode ser visto o que há dentro
  - Para ler um arquivo simples em Python é utilizado o método ***read()***. O método ***read()*** lê o arquivo todo de uma só vez e pode carregá-lo em uma variável
- Exemplo:

```
arquivo = open("teste.txt", "r")
print(arquivo.read())
```

Lendo arquivos em Python

- Além do método *read()*, visto anteriormente, também podemos ler o conteúdo de um arquivo usando os métodos *readline()* e *readlines()*

- ***readline()***: retorna uma linha do texto a cada chamada, na ordem em que aparecem no arquivo

- ***readlines()***: retorna uma lista do tipo *string*, sendo que cada valor da lista corresponde a uma linha do arquivo

Lendo arquivos em Python

- Exemplo:

```
arquivo = open("teste.txt", "r")
print(arquivo.readline())
print(arquivo.readline())
```

```
arquivo = open("teste.txt", "r")
print(arquivo.readlines()[0])
 print(arquivo.readlines())
```

Escrevendo arquivos em Python

- Talvez seja necessário escrever outra frase ou parágrafo no arquivo já lido. Por exemplo, digamos que é preciso adicionar a seguinte frase:

- **“Escrevendo em arquivos”**

- Isso pode ser feito em Python usando o método ***write()***. Este método utiliza como argumento a *string* que desejamos gravar no arquivo

Escrevendo arquivos em Python

- Exemplo:

```
arquivo = open(“teste.txt”, “w”)
arquivo.write(“\n Escrevendo em arquivos \n”)

arquivo = open(“teste.txt”, “r”)
print(arquivo.read())
```

- Antes o arquivo “**teste.txt**” possuía outro conteúdo, agora o arquivo foi sobrescrito com a frase adicionada pelo método *write()*

Escrevendo arquivos em Python

- Quando é necessário manter o texto original do documento, utiliza-se o modo “a” (de *append*)

```
arquivo = open(“teste.txt”, “a”)
arquivo.write(“\n Escrevendo em arquivos \n”)

arquivo = open(“teste.txt”, “r”)
print(arquivo.read())
```

- Apenas é necessário substituir o modo de “w” para “a” no argumento *mode*: ***open(filename, “a”)***

Fechando os arquivos

- Ter o hábito de fechar um arquivo após a leitura ou escrita permitirá que você libere memória
- Sim, o Python fecha automaticamente os arquivos depois que o *script* termina
  - Mas se você não fizer isso de antemão, todos os arquivos abertos ocuparão espaço que o Python poderia usar, por exemplo, em uma rotina do programa que precise recuperar o dado que ainda não foi salvo...
- O fechamento de um arquivo pode ser facilmente realizado usando o método ***close()***

Fechando os arquivos

- Exemplo:

```
arquivo = open(“teste.txt”, “a”)
arquivo.write(“\n Escrevendo em arquivos \n”)

arquivo = open(“teste.txt”, “r”)
print(arquivo.read())

arquivo.close()
```

Fechando os arquivos

- Outra forma de lembrarmos de fechar o arquivo é utilizando a instrução ***with***
- A instrução *with* fecha o arquivo automaticamente assim que saímos de sua *suite*. Ele nos permite abrir o arquivo, processá-lo e certificar-se de que está fechado
- Exemplo:

```
with open("teste.txt", "a") as arquivo:
    arquivo.write("Escrevendo em arquivos")
    arquivo = open("teste.txt", "r")
    print(arquivo.read())
```

Fechando os arquivos

- A instrução *with* é usada no tratamento de exceções para tornar o código mais limpo e legível

```
# 1) sem a instrucao with
arquivo = open('teste.txt', 'w')
arquivo.write('hello world !')
arquivo.close()

# 2) com tratamento de exceções
arquivo = open('teste.txt', 'w')
try:
    arquivo.write('hello world')
finally:
    arquivo.close()
```

```
# com a instrução with
with open('teste.txt', 'w') as arquivo:
    arquivo.write('hello world !')
```

Navegando pelo arquivo

```
for linhas in arquivo:
```

```
arquivo.seek(bytes)

arquivo.tell()
```

- Cuidado com o posicionamento do cursor

Vamos programar!

- Crie um programa que peça e continue pedindo para que o usuário informe palavras e vá armazenando tais palavras em um arquivo. Quando o usuário digitar “/exit”, você deve interromper as leituras (a palavra “/exit” não deve ser armazenada no arquivo).
  Por fim, imprima na tela, o conteúdo do arquivo.

Exercícios

1. Crie um arquivo chamado “**dados.txt**” e, como conteúdo, salve o seu nome no arquivo; depois faça a leitura do arquivo para uma variável e também na tela; utilize ***read()*** e ***readline()***

- Agora acrescente a data de nascimento, cidade que nasceu e sua idade (em linhas separadas no arquivo); depois faça a leitura do arquivo novamente com ***readlines()***
- Ainda utilizando o arquivo “dados.txt”, leia o que está dentro do arquivo, mas agora utilizando um laço de repetição e ***readlines()***, apresentando o número das linhas a cada iteração

Exercícios

2 Criar um novo arquivo chamado “**frases.txt**” com a seguintes frases: “**Estudando Python.**”, “**Programando em Python.**” e “**Manipulação de arquivos.**”

- Criar um programa que retorne somente as linhas que possuem a palavra Python
- Desenvolver um método que retorne somente as linhas que contém uma palavra específica a critério do usuário

Exercícios

3 Criar um programa que carregue um arquivo chamado “**contatos.txt**”, contendo nomes e números de contatos tal como os dados a seguir:

Nome:	Celular:
Guilherme	(41) 9 9123-4567
Rafaela	(47) 9 2002-2222
Eduardo	(48) 9 9876-5432
Mercado	(47) 3003-1010

- Inclua um laço de repetição para buscar por um determinado contato, exibindo na tela apenas o número deste contato. Para sair do laço utilize “sair”

Exercícios

4 Considere um arquivo chamado dados.txt que possui o seguinte conteúdo:

```
Python é uma linguagem poderosa.
Manipular arquivos é fácil em Python.
Podemos ler, escrever e mover o cursor facilmente.
```

- Abra o arquivo dados.txt no modo leitura.
- Utilize o método seek() para posicionar o cursor exatamente no início da segunda linha.
- Leia e imprima na tela apenas a segunda linha do arquivo.