

Arquivos

Introdução à Programação
SI2

Arquivos

- **Entrada** e **saída** são operações de **comunicação** de um programa com o **mundo externo**
- Essa comunicação se dá usualmente através de *arquivos*
- Em Python, um arquivo pode ser lido/escrito através de um objeto da classe **file**

Abrindo Arquivos

- `open(name, mode)`
 - `name`: nome do arquivo a abrir
 - `mode`: (opcional) modo de abertura – string contendo
 - `r` : leitura (default)
 - `w` : escrita (se o arquivo existir terá o conteúdo apagado)
 - `a` : escrita a partir do final (não apaga conteúdo)
 - `r+` : indica leitura e escrita
 - `b` : binário (`rb`, `wb`, `ab`)

Exemplo

- O comando `open` retorna um objeto do tipo *file* (arquivo)
- O objeto *file* é usado para operações de entrada e saída:

```
>>> arq = open ("teste.txt", "w")
```

```
>>> arq.write ("Oi")
```

```
>>> arq.close ()
```

```
>>> arq = open ("teste.txt")
```

```
>>> x = arq.read()
```

```
>>> x
```

```
'Oi'
```

Open

```
>>> f = open('texto.txt','w')
```

```
>>> f.write('Ola Mundo!')
```

- Não devemos esquecer de fechar o arquivo:

```
>>> f.close()
```

Métodos **Read**, **Write** e **Close**

- **read(num)**

- Lê **num** bytes do arquivo e os retorna numa string
- Se **num** não é especificado, todos os bytes desde o ponto atual até o fim do arquivo são retornados

Métodos **Read**, **Write** e **Close**

- **write(string)**

- Escreve **string** no arquivo
- Devido ao uso de buffers, a escrita pode não ser feita imediatamente
 - Use o método **flush()** ou **close()** para assegurar a escrita física

- **close()**

- Termina o uso do arquivo para operações de leitura e escrita

ExemploRead

```
>>> f = open('texto.txt','r')
```

```
>>> ler = f.read()
```

```
>>> f.close()
```

```
>>> print ler  
Ola Mundo!
```

- Metodo `read()` usado sem nenhum argumento, mostrará tudo que esta no aquivo

ExemploRead

- Se quisermos ler apenas os três primeiros caracteres (bytes), podemos usar da seguinte maneira:

```
>>> f = open('texto.txt') # r é default
```

```
>>> ler = f.read(3)
```

```
>>> f.close()
```

```
>>> print ler
```

Ola

ExemploRead

- e caso queira ler o restante, o read() começará de onde parou:

```
>>> restante = f.read()
```

```
>>> print restante  
Mundo!
```

Fim de Linha

- Arquivos de texto são divididos em linhas usando caracteres especiais
- Python usa sempre **\n** para separar linhas
 - Leitura ou escrita de arquivo aberto em modo **texto**
 - Em modo **binário**, entretanto, a conversão **não é feita**

Lendo e Escrevendo Linhas

- `readline(n)`
 - Se `n` não é especificado, retorna exatamente uma linha lida do arquivo
 - Caso contrário, lê uma linha, mas busca no máximo `n` caracteres pelo final de linha
 - Se o tamanho da linha é zero significa que o **final do arquivo** foi atingido

Lendo e Escrevendo Linhas

- `readlines()`
 - Retorna o restante do conteúdo do arquivo em uma lista de strings

Lendo e Escrevendo Linhas

- `writelines(lista)`
 - Escreve a `lista` (ou qualquer seqüência) de strings, uma por uma no arquivo
 - Caracteres de final de de linha *não são* acrescentados

Exemplos

- adicionar mais alguma coisa no arquivo do exemplo:

```
>>> f = open('texto.txt', 'a')
```

```
>>> f.write('\nOla Python')
```

```
>>> f.close()
```

- o 'a' (append) é usado para adicionar texto sem apagar o que já havia no arquivo
- '\n' é usado pra pular uma linha, lembrando que tudo é string (este caractere fica visível) 16

Exemplos

```
>>> f = open('texto.txt','r')
```

```
>>> linha1 = f.readline()
```

```
>>> linha2 = f.readline()
```

```
>>> f.close()
```

```
>>> print linha1
```

```
Ola Mundo!\n
```

```
>>> print linha2
```

```
Ola Python
```


Exemplos

```
>>> lista = ['Ola mundo\n', 'Ola Python\n', 'Ola UFRPE']
>>> f = open('texto.txt', 'w')
>>> f.writelines(lista)
>>> f = open('texto.txt', 'r')
>>> cont = f.readlines()
>>> print cont
['Ola mundo\n', 'Ola Python\n', 'Ola UFRPE']
```

Exemplos

```
>>> for linha in f:  
      print linha
```

Resumo

operação	Interpretação
<code>output = open("teste.txt","w")</code>	Cria arquivo de saída ("w" significa gravação)
<code>input = open("teste.txt","r")</code>	Cria arquivo de entrada ("r" significa leitura)
<code>S = input.read()</code>	Lê o arquivo inteiro em uma única string
<code>S = input.read(N)</code>	Lê N bytes (1 ou mais)
<code>S = input.readline()</code>	Lê a próxima linha
<code>L = input.readlines()</code>	Cria uma lista onde cada elemento é uma linha do arquivo
<code>output.write(S)</code>	Grava a string S no arquivo
<code>output.writelines(L)</code>	Grava no arquivo todas as strings da lista L
<code>output.close()</code>	Fechamento manual do arquivo

Método `seek()`

- Muda a posição do cursor no arquivo
- `f.seek(deslocamento, ref)`
- A nova posição é computada adicionando o `deslocamento` ao ponto de referência
- O ponto de referência é determinado pelo parâmetro `ref` que pode ser:
 - 0 (default): deslocar a partir do início do arquivo
 - 1 deslocar a partir da posição atual e
 - 2 usar o final do arquivo como ponto de referência

Método **seek()**

- Exemplos

```
>>> f = open('/tmp/workfile', 'r+')
>>> f.write('0123456789abcdef')
>>> f.seek(5)          # Go to the 6th byte in the file
>>> f.read(1)
'5'
>>> f.seek(-3, 2)      # Go to the 3rd byte before the end
>>> f.read(1)
'd'
```

Interação com o Sistema Operacional

- Operações de **entrada** e **saída** são na verdade realizadas pelo **sistema operacional**
- O módulo **os** possui variáveis e funções que ajudam um programa Python a se adequar ao sistema operacional
 - **import os**

Interação com o SO

```
>>>os.getcwd()
```

retorna o diretório corrente

```
>>>os.chdir(dir)
```

muda o diretório corrente para *dir*

```
>>>os.sep
```

string com o caractere que separa componentes de um caminho ('/' para *Unix*, '\\' para *Windows*)

```
>>>os.path.exists(path)
```

diz se *path* se refere ao nome de um arquivo existente

Exemplos

```
>>> import os
>>> os.getcwd()
'C:\\Python26'
>>> os.chdir('C:\\Python31')
>>>
>>> os.getcwd()
'C:\\Python31'
>>> os.sep
'\\'
>>> os.path.exists('C:\\Python31')
True
>>> os.path.exists('C:\\Python99')
False
```


EXERCÍCIOS

Exercícios

- 1. Faça um programa que escreve uma frase digitada pelo usuário em um arquivo. Em seguida o programa deve ler e imprimir o conteúdo desse arquivo

Exercícios

2. Escreva um programa que lê um arquivo contendo a identidade e o nome de várias pessoas, no seguinte formato

```
5384423  Manoel
4345566  Alberto
3235574  Mariana
...
```

o programa deve gerar um dicionário onde as chaves são as identidades e os valores os nomes. Ao final o programa deve exibir o dicionário.

Exercícios

3. Escreva um programa que lê um arquivo contendo endereços IPs, da seguinte forma:

```
200.135.80.9  
192.168.1.1  
8.35.67.74  
257.32.4.5  
85.345.1.2  
1.2.3.4  
9.8.234.5  
192.168.0.256
```

- O programa deve mostrar os IPS indicando os que são validos e inválidos (um endereço ip válido não pode ter uma de suas partes maior que 254).

Exercícios

- 4. Escreva um programa que leia um arquivo com um conjunto de nomes (1 por linha). O programa deve ordenar os nomes e gerar um novo arquivo com os nomes ordenados.

Exercícios

- 5. Faça um programa que leia as linhas de 3 a 5 de um arquivo de texto (considere que tem mais do que 5 linhas).
 - Copie as linhas selecionadas em um novo arquivo.

Exercícios

- 6. Escreva um programa que leia um arquivo em python (nome fornecido pelo usuário).
 - O programa deverá informar:
 - Quantas linhas o arquivo tem
 - A quantidade de “print” que o código possui

Bibliografia

- Livro “Como pensar como um Cientista de Computação usando Python” – Capítulo 11
 - <http://pensarpython.incubadora.fapesp.br/portal>
- Python Tutorial
 - <http://www.python.org/doc/current/tut/tut.html>
- Dive into Python
 - <http://www.diveintopython.org/>
- Python Brasil
 - <http://www.pythonbrasil.com.br/moin.cgi/DocumentacaoPython#head5a7ba2746c5191e7703830e02d0f5328346bcaac>