

Programação com Interfaces Gráfica

Mario Benevides e Paulo Roma

Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil

Arquivos
Entrada e Saída

Agenda

Nesta Aula: Arquivos (Python)

- Introdução: Arquivos
- Arquivos Default
- Abrindo arquivos
- O Objeto file
- Métodos Read, Write e Close
- Lendo e escrevendo linhas

Introdução: Arquivos

Arquivos

- Entrada e saída são operações de comunicação de um programa com o mundo externo
- Essa comunicação se dá usualmente através de arquivos
- Arquivos estão associados a dispositivos
 - Por exemplo, disco, impressora, teclado
- Em Python, um arquivo pode ser lido/escrito através de um objeto da classe *file*

Arquivos Default

Já usamos, sem saber, três arquivos default

- Sempre que um comando *print* é executado, o resultado vai para um arquivo chamado *sys.stdout*
- Sempre que lemos um dado através do comando *input* ou *raw_input*, na verdade estamos lendo de um arquivo chamado *sys.stdin*
- Mensagens de erro ou de rastreamento de exceções são enviadas para um arquivo chamado *sys.stderr*

Exemplo: Arquivo

```
$python
>>> import sys
>>> sys.stdout.write("Oi!")
Oi!
>>> print ("Oi!")
Oi!
>>> sys.stdin.readline()
fluminense
'fluminense\n'
>>> raw_input()
fluminense
'fluminense'
```

Abrindo arquivos

open (name, mode, buffering)

- name : nome do arquivo a abrir
- mode : (opcional) modo de abertura – string contendo
 - r : leitura (default)
 - w : escrita
 - b : binário
 - a : escrita a partir do final
 - + : (usado com r) indica leitura e escrita
- buffering : (opcional) indica se memória (buffers) é usada para acelerar operações de entrada e saída
 - 0 : buffers não são usados
 - 1 (ou número negativo): um buffer de tamanho padrão (default)
 - 2 ou maior: tamanho do buffer em bytes

O Objeto *file*

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

```
>>> arq = open ("teste", "w")
>>> arq.write("Oi!")
>>> arq.close()
>>> arq = open ("teste")
>>> x = arq.read()
>>> x
"Oi!"
```

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.

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.

- Arquivos de texto são divididos em linhas usando caracteres especiais
 - Linux/Unix: `\n`
 - Windows: `\r\n`
 - Mac: `\r`
- Python usa sempre `\n` para separar linhas
 - Ao se ler/escrever um arquivo aberto em modo texto (não binário) faz traduções de `\n` para se adequar ao sistema operacional;
 - Em modo binário, entretanto, a conversão não é feita

Interação com o Sistema Operacional

- Operações de entrada e saída são na verdade realizadas pelo sistema operacional
- O módulo *OS* possui diversas variáveis e funções que ajudam um programa Python a se adequar ao sistema operacional, por exemplo:
 - *os.getcwd()* retorna o diretório corrente;
 - *os.chdir(dir)* muda o diretório corrente para *dir*;
 - *os.path.exists(path)* diz se *path* se refere ao nome de um arquivo existente.

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.

readlines(*n*)

- Se *n* não é especificado, retorna o restante do conteúdo do arquivo em uma lista de strings ;
- Caso *n* seja especificado, a leitura é limitada a *n* caracteres no máximo.

writelines(*seqüência*)

- Escreve a lista (ou qualquer seqüência) de strings, uma por uma no arquivo;

Acesso direto

- É possível ler e escrever não seqüencialmente em alguns tipos de arquivo;
 - Devem estar associados a dispositivos que permitem acesso direto, como discos, por exemplo.
- `seek(offset, whence)`
 - `offset` indica o número do byte a ser lido e escrito pela próxima operação de entrada e saída;
 - `whence` indica a partir de onde `offset` será contado
 - 0 (default) : do início
 - 1 : do ponto corrente
 - 2 : do final
 - `tell()`
 - Indica a posição corrente (número de bytes a partir do início do arquivo)