Fundamentos da Programação

Ficheiros

Aula 17

José Monteiro

(slides adaptados do Prof. Alberto Abad)

Ficheiros

- Todos os programas que vimos até agora obtêm dados do *standard input* e enviam dados para o *standard output*:
  - Quando um programa termina, todos os dados utilizados pelo mesmo desaparecem e não os podemos recuperar.
- Armazenando dados em ficheiros podemos ter estado/dados **persistentes**.
- **Características dos ficheiros:**
  - tipo estruturado de informação constituído por uma sequência de elementos (leitura/escrita tipicamente sequencial);
  - podem existir independentemente de qualquer programa;
  - durante a execução de um programa, um ficheiro pode estar num de dois estados, em modo de **leitura** ou em modo de **escrita**.

Tipo Ficheiro em Python

Em Python recorremos à função pré-definida `open` para abrir um ficheiro:

```
open(<expressão>, <modo>{, encoding = <tipo>})
```

- `<expressão>` string que representa a localização do ficheiro, incluindo o seu nome;
- `<modo>::= 'r' | 'w' | 'a'` denotando a abertura para leitura (*read*), escrita (*write*) e escrita a partir do final do ficheiro (*append*);
- `<tipo>` indica qual a codificação dos carateres no ficheiro.

O resultado da operação `open` é o valor que corresponde à identidade associada ao ficheiro:

```
<file> = open(<expressão>, <modo>{, encoding = <tipo>})
```

In [85]:
```
f = open('FP17.ipynb', 'r')
```

Tipo Ficheiro em Python

Atributos

As instâncias do tipo ficheiro têm alguns atributos que fornecem informação sobre o ficheiro:

Atributo	Descrição
`file.closed`	Devolve `True` se o ficheiro está fechado, ou `False` se está aberto.
`file.mode`	Devolve o modo de abertura com que foi aberto o ficheiro.
`file.name`	Devolve o nome do ficheiro.

In [87]:
```
f
```

Out[87]:
```
<_io.TextIOWrapper name='FP17.ipynb' mode='r' encoding='UTF-8'>
```

Tipo Ficheiro em Python - Método `close()`

Em Python, podemos fechar um ficheiro previamente aberto com:

```
<file>.close()
```

Deste modo, realizamos a operação de fecho do ficheiro, desfazendo a associação entre o programa e o ficheiro.

In [89]:
```
f.closed
```

Out[89]:
```
True
```

Leitura de Ficheiros em Python

Abertura de ficheiros para leitura

- Quando abrimos um ficheiro para leitura está subentendido que esse ficheiro existe. Caso o ficheiro não exista, é gerado um erro.

```
>>> open('nofile.txt', 'r')
FileNotFoundError: [Errno 2] No such file or directory:
'nofile.txt'
```

- Se o ficheiro `teste.txt` existe na diretoria em que estamos a trabalhar:

```
>>> open('teste.txt', 'r', encoding = 'UTF-16')
<_io.TextIOWrapper name='teste.txt' mode='r' encoding='UTF-16'>
```

- Em geral os ficheiros são lidos de forma sequencial. O Python utiliza um **indicador de leitura** que indica o próximo elemento a ser lido.
  - o indicador é inicializado no primeiro elemento do ficheiro e desloca-se a cada leitura em direção ao final do ficheiro.

Leitura de Ficheiros em Python

Operações de leitura

- Uma vez aberto o ficheiro `<file>` para **leitura**, podemos efectuar algumas **operações**:
  - `<file>.readline()` , lê uma linha do ficheiro e retorna a *string* correspondente, ou a *string* vazia `''` caso tenhamos chegado ao final do ficheiro;
  - `<file>.readlines()` , lê todas as linhas no ficheiro e retorna uma lista com as *strings* correspondentes às linhas lidas, ou a lista vazia `[]` caso tenhamos chegado ao final do ficheiro;
  - `<file>.read()` , lê todos os carateres do ficheiro e retorna uma *string* com todos os carateres lidos, ou a *string* vazia '' se o indicador de leitura estiver já no final do ficheiro.

Leitura de Ficheiros em Python

Exemplos

```
>>> f = open('teste.txt', 'r')
>>> l = f.readline()
>>> ll = f.readlines()
>>> f.read()
>>> f.close()
```

```
>>> f = open('teste.txt', 'r')
>>> f.read()
>>> f.read()
>>> f.close()
```

In [105…

```
f = open('teste.txt', 'r')
```

Leitura de Ficheiros em Python

Movendo o indicador de leitura

- É possível alterar a posição do indicador de leitura:
  - `<file>.tell()` , devolve a posição atual do indicador de leitura;
  - `<file>.seek(offset{, ref})` , move o indicador `offset` número de posições desde a posição indicada pelo `ref` :
    - ref = 0, início do ficheiro;
    - ref = 1, posição atual indicador de leitura; offset pode ser negativo (apenas em binário!);
    - ref = 2, fim do ficheiro.

In [ ]:

```

```

Escrita de Ficheiros em Python

Abertura de ficheiros para escrita

- Quando abrimos um ficheiro para escrita (`w` ou `a`), se o ficheiro não existir, então é criado.

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

- Neste caso temos um **indicador de escrita**:
  - Quando abrimos o ficheiro em modo `w` o indicador é colocado no início (conteúdo é eliminado/perdido).
  - Quando abrimos o ficheiro em modo `a` o indicador é colocado no final do ficheiro.

In [ ]:

Escrita de Ficheiros em Python

Operações de escrita

- Uma vez aberto o ficheiro `<file>` para **escrita**, podemos efetuar algumas **operações**:
  - `<file>.write(<cadeia de carateres>)` , que escreve uma linha no ficheiro a seguir ao indicador de escrita e retorna o número de carateres escritos com sucesso;
  - `<file>.writelines(<sequência>)` , que escreve todas as linhas na sequência (uma lista ou um tuplo, por exemplo) no ficheiro a partir do indicador de escrita e não devolve qualquer valor

Escrita de Ficheiros em Python

```
>>> f = open('saida.txt', 'w')
>>> f.close()
>>> f = open('saida.txt', 'r')
>>> f.read()
???
>>> f.close()
>>> f = open('saida.txt', 'w')
>>> f.write('fundamentos de programacao')
???
>>> f.close()
>>> f = open('saida.txt', 'r')
>>> f.read()
???
>>> f.close()
```

In [111…

Escrita de Ficheiros em Python

```
>>> f = open('saida.txt', 'a')
>>> f.write('\n2021/22\n1o período\n')
???
>>> f.close()
>>> f = open('saida.txt', 'r')
>>> f.read()
???
```

In [ ]:

Escrita de Ficheiros em Python

Utilização da função `print`

- Depois de aberto um ficheiro para escrita, podemos também utilizar a função **print**:

```
<escrita de dados> ::=
    print() |
    print(<expressões>) |
    print(file = <nome de ficheiro>) |
    print(<expressões>, file = <file>)
```

In [42]:
```
f = open('saida.txt', 'w')
print('ola', file=f)
f.close()
```

Ficheiros em Python

Alternativa à abertura de ficheiros

- Instrução **with**:

```
with open('teste.txt', 'r') as file:
  data = file.read()
```

- **Iterar** sobre um ficheiro:

```
with open('teste.txt', 'r') as file:
    for line in file:
        print(line, end='')
```

In [ ]:
```

```

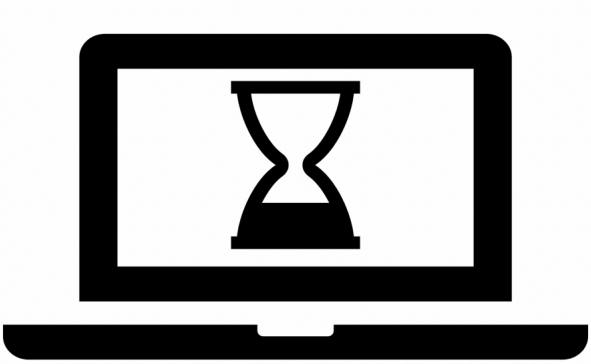
Ficheiros - Comentários Finais

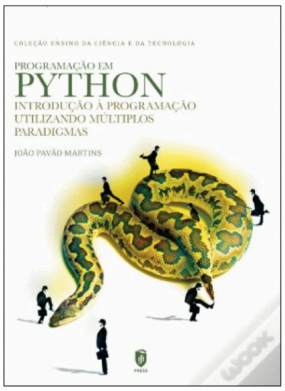
- É importante ter em conta as **várias codificações** possíveis para os carateres e as diferentes formas de terminar linhas, que podem variar de sistema operativo para sistema operativo.
- Também as diferentes formas de **especificar as localizações** dos ficheiros dependem do sistema operativo e se são locais ou remotos.
- Finalmente, nesta aula considerámos apenas **ficheiros de texto**, mas é possível trabalhar com ficheiros binários: https://www.devdungeon.com/content/working-binary-data-python

Ficheiros

Tarefas próximas aulas

- Estudar matéria Abstração e Tipos Abstratos de Dados e Ficheiros
    - Exercícios!
- Nas aulas teóricas --> Exercícios TADs e Ficheiros
- Na aula práticas --> Ficheiros





In [ ]: