Fundamentos da Programação
LEIC/LETI

Ficheiros

O tipo ficheiro. Leitura de ficheiros. Escrita de ficheiros.

Aula 18

Alberto Abad, Tagus Park, IST, 2021-22

¶

Ficheiros

Introdução 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**.

O 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 o tipo de codificação dos caracteres no ficheiro.

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

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

In [ ]:

Ficheiros

O tipo ficheiro em Python - Modos de abertura adicionais

Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
r+	Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
rb+	Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
ab	ab Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

O tipo ficheiro em Python - Atributos

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

Atributo	Descripção
file.closed	Devolve True se o ficheiro esta 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 [8]: f.mode
```

```
Out[8]: 'w'
```

Ficheiros

O 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 [ ]: f.closed
```

Leitura de ficheiros em Python - Abrir 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 directoria 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 direcção ao final do ficheiro.

In [ ]:

Ficheiros

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 caracteres do ficheiro e retorna uma string com todos os caracteres 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 [117]:
```
ll = f.readlines()
ll
```

```
0
205
0
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi qui
s
```


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!);
    - red = 2, fim do ficheiro.

In [198]:
```
help(f.seek)
```

```
Help on built-in function seek:

seek(cookie, whence=0, /) method of _io.TextIOWrapper instance
    Change stream position.

    Change the stream position to the given byte offset. The offse
t is
    interpreted relative to the position indicated by whence.  Val
ues
    for whence are:

    * 0 -- start of stream (the default); offset should be zero or
positive
    * 1 -- current stream position; offset may be negative
    * 2 -- end of stream; offset is usually negative

    Return the new absolute position.
```

Ficheiros

Escrita de ficheiros em Python - Abrir 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 [68]:
```
```

Out[68]:
```
'a'
```

Escrita de ficheiros em Python - Operações de escrita

- Uma vez aberto o ficheiro `<file>` para **escrita**, podemos efectuar algumas **operações**:
- `<file>.write(<cadeia de caracteres>)` , que escreve uma linha no ficheiro a seguir ao indicador de escrita e retorna o número de caracteres 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;

Ficheiros

Escrita de ficheiros em Python - Operações de escrita (I)

```
>>> 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 [120]:
```
f = open('saida.txt', 'w')
lst = ('ola', 'mundo', 'adeus')
f.writelines(lst)
f.close()
!cat saida.txt
```

```
ola
mundo
adeus
```

Escrita de ficheiros em Python - Operações de escrita (II)

```
>>> f = open('saida.txt', 'a')
>>> f.write('\n2020/21\n1o semestre\n')
???
>>> f.close()
>>> f = open('saida.txt', 'r')
>>> f.read()
???
```

In [125]:

```
2020/21
1o semestre
```

Ficheiros

Ficheiros em Python - Tópicos adicionais/avançados (I)

- 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 = <nome de ficheiro>)
```

- Instrução **with**: (https://www.geeksforgeeks.org/with-statement-in-python/ (https://www.geeksforgeeks.org/with-statement-in-python/))

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

In [6]:
```
file=open('ficheiro.txt', 'w')
print('Ola mundo', file=file)
file.close()

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

data
```

Out[6]:
```
'Ola mundo\n'
```


Ficheiros em Python - Tópicos adicionais/avançados (II)

- **Iterar** sobre um ficheiro:

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

lst = [char for line in open('teste.txt', 'r') for char in line]
print(lst)
```

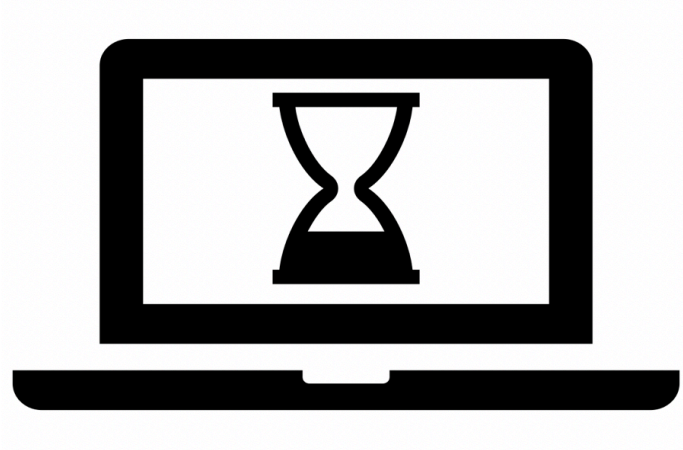
In [ ]:
```
[char for line in open('teste.txt', 'r') for char in line]
```

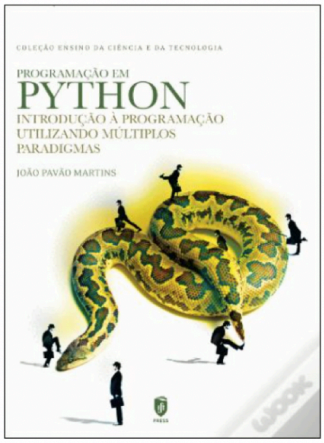

Ficheiros - Comentários finais

- É importante ter em conta as **várias codificações** possíveis para os caracteres e as diferentes formas de terminar linhas e que podem variar de sistema operativo para sistema operativo
- Também as diferentes formas de **especificar as localizações** dos ficheiros dependendo também 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 (https://www.devdungeon.com/content/working-binary-data-python)

Tarefas próximas aulas

- Estudar matéria Ficheiros
- Nas aulas teóricas amanhã --> Dúvidas projeto + Apresentação do enunciado 2 + Exemplos?
- Na aula práticas (hoje e amanhã) --> TADs





In [ ]: