



universidade de aveiro
escola superior de tecnologia e
gestão de água

Introdução à Programação

Ficheiros

Leitura e escrita

Para gravar um ficheiro, devemos abri-lo com o modo 'w' como segundo parâmetro:

```
>>> fout = open('output.txt', 'w')
```

Experimental abrir e fechar ficheiro para verificar se o conteúdo é apagado.

- ▶ Se o ficheiro já existir, ao abrir para gravar, o ficheiro fica vazio.
- ▶ Se o ficheiro não existir, um novo será criado. *open* retorna um objeto de ficheiro que fornece métodos para trabalhar com o ficheiro. O método *write* coloca dados no ficheiro.

```
>>> line1 = "This here's the wattle,\n">>> fout.write(line1)
24
```

O valor de retorno é o número de caracteres que foram escritos.

Quando terminar de escrever, devemos fechar o ficheiro.

```
>>> fout.close()
```

Se o programador não fechar o ficheiro, este será fechado quando o programa terminar.

Leitura e escrita

Para ler um ficheiro, devemos abri-lo com o modo 'r' como segundo parâmetro:

```
>>> fin = open('input.txt', 'r')
```

O método *read* lê todo o conteúdo do ficheiro.

```
>>> print(fin.read())
```

O método *readline* lê uma linha do ficheiro.

```
>>> myline = myfile.readline()
>>> while myline:
>>>     print(myline)
>>>     myline = myfile.readline()
>>> myfile.close()
```

Operador de formato

O argumento de *write* tem que ser uma string. O que significa que se quisermos colocar outros valores num ficheiro, temos que convertê-los em strings. A maneira mais fácil de fazer isso é com `str`:

```
>>> x = 52  
>>> fout.write(str(x))
```

Uma alternativa é usar o **operador de formato %**.

- ▶ Quando aplicado a números inteiros, % é o operador de módulo.
- ▶ Quando aplicado a strings, % é o operador de formato.

O primeiro operando é a string de formato, que contém uma ou mais sequências de formato, especificam como o segundo operando é formatado.

O resultado é uma string.

Por exemplo, a sequência de formato '%d' significa que o segundo operando deve ser formatado como um inteiro decimal:

```
>>> camels = 42  
>>> '%d' % camels  
'42'
```

Operador de formato

Uma sequência de formato pode aparecer em qualquer lugar na string, para que possamos incorporar um valor numa frase:

```
>>> 'I have spotted %d camels.' % camels  
'I have spotted 42 camels.'
```

- ▶ Se houver mais do que uma sequência de formato na string, o segundo argumento deverá ser um tuplo.
- ▶ Cada sequência de formato é combinada com um elemento do tuplo, por ordem.

Exemplo:

- ▶ '%d' para formatar um inteiro,
- ▶ '%g' para formatar um número de ponto flutuante
- ▶ '%s' para formatar uma string:

```
>>> 'In %d years I have spotted %g %s.' % (3, 0.1,  
'camels')  
'In 3 years I have spotted 0.1 camels.'
```

Operador de formato

- ▶ O número de elementos do tuplo deve corresponder ao número de sequências de formato na string.

Experimental e verificar o comportamento em situações não previstas

- ▶ Os tipos dos elementos devem corresponder às sequências de formato:

```
>>> '%d %d %d' % (1, 2)
TypeError: not enough arguments for format string
```

```
>>> '%d' % 'dollars'
TypeError: %d format: a number is required, not str
```

Nomes de ficheiros e caminhos

Os ficheiros são organizados em diretorias (também chamados de “pastas”).

Todos os programas em execução possuem uma “diretoria atual”, que é a diretoria padrão para a maioria das operações.

Quando abrimos um ficheiro para leitura, o Python procura-o na diretoria atual.

O módulo *os* fornece funções para trabalhar com ficheiros e diretorias (“os” significa “sistema operativo”). *os.getcwd* retorna o nome da diretória atual:

```
>>> import os
>>> cwd = os.getcwd()
>>> cwd
'/home/tengstedt'
```

Nomes de ficheiros e caminhos

cwd

- ▶ **Current Working Directory**
- ▶ significa “**diretoria de trabalho atual**”.

O resultado neste exemplo é `/home/tengstedt`, que é a diretoria inicial de um utilizador chamado `tengstedt`.

path

- ▶ Uma string como `'/home/tengstedt'` que identifica um ficheiro ou diretoria é chamada de caminho (**path**).
- ▶ Um nome de ficheiro simples, como `memo.txt`, também é considerado um caminho, mas é um caminho relativo porque está relacionado à diretoria atual.
- ▶ Se a diretoria atual for `/home/tengstedt`, o nome do ficheiro `memo.txt` refere-se a `/home/tengstedt/memo.txt`.
- ▶ Um caminho que começa com `/` não depende do diretoria atual, é chamado de caminho absoluto. Para encontrar o caminho absoluto para um ficheiro, podemos usar `os.path.abspath`:

```
>>> os.path.abspath('memo.txt')  
'/home/tengstedt/memo.txt'
```

Nomes de ficheiros e caminhos

os.path fornece outras funções para trabalhar com nomes de ficheiros e caminhos.

Por exemplo, os.path.exists verifica se um ficheiro ou diretória existe:

```
>>> os.path.exists('memo.txt')  
True
```

Se existir, os.path.isdir verifica se é um diretoria:

```
>>> os.path.isdir('memo.txt')  
False  
>>> os.path.isdir('/home/dinsdale')  
True
```

Da mesma forma, os.path.isfile verifica se é um ficheiro.

os.listdir retorna uma lista dos ficheiros (e outras diretorias) na diretoria fornecido:

```
>>> os.listdir(cwd)  
['music', 'photos', 'memo.txt']
```

Catching exceptions

Muitas coisas podem correr mal quando tentamos ler e gravar ficheiros.

Se tentarmos abrir um ficheiro que não existe, receberemos um `FileNotFoundError`:

```
>>> fin = open('bad_file')  
FileNotFoundError: [Errno 2] No such file or directory: 'bad_file'
```

Se não temos permissão para aceder um ficheiro:

```
>>> fout = open('/etc/passwd', 'w')  
PermissionError: [Errno 13] Permission denied: '/etc/passwd'
```

E se tentar abrir uma diretoria para leitura, obtemos

```
>>> fin = open('/home')  
IsADirectoryError: [Errno 21] Is a directory: '/home'
```

Catching exceptions

Para evitar esses erros, podemos usar as funções como `os.path.exists` e `os.path.isfile`.

- ▶ Mais código e tempo para verificar todas as possibilidades
- ▶ Alternativa: **try/except**.

```
try:  
    fin = open('bad_file')  
except:  
    print('Something went wrong.')
```