Computadores e Programação

2009–2010 1º semestre — aula 5

Helmut Wolters

helmut@coimbra.lip.pt

2009-10-23

Ficheiros

- Os ficheiros permitem guardar objectos de forma permanente.
- O Python tem um conjunto rico de instruções para manipular ficheiros.
- Um objecto do tipo ficheiro é criado com a instrução `open`:

```
f = open(filename,mode)
```

  onde `filename` é o nome do ficheiro (ex. `'report.txt'`), e `mode` uma das alternativas `'r','w','a','r+'` que significam abertura para leitura, escrita, continuação (*append*) e leitura e escrita.
- Um ficheiro é um objecto iterável onde o iterador devolve, em sequência, cada linha do ficheiro.

Ficheiros

- Uma vez aberto um ficheiro pode ser lido da seguinte forma:

```
for line in f:
    print line
```

- Os ficheiros possuem vários métodos (`dir(f)` , onde `f` é um objecto do tipo ficheiro dá a lista de todos os métodos). Os mais importantes são `read, readline, readlines` e `write`.
- `f.read(n)` lê `n` caracteres do ficheiro devolvendo-os numa *string*. Se `n` for omitido, todo o ficheiro será lido. Se o final do ficheiro for encontrado,o número de caracteres devolvido poderá ser, eventualmente, inferior ao pedido.

Ficheiros

- O seguinte programa imprime todo o conteúdo do ficheiro `'report.txt'`.

```
f = open('report.txt','r')
print f.read()
```

- `f.readlines(n)` lê `n` linhas do ficheiro devolvendo-as numa lista de *strings* (uma por linha).

```
lines = f.readlines()
for line in lines:
    print line
```

Ficheiros

- Os métodos `write` e `writelines` permitem escrever uma ou mais linhas num ficheiro. O argumento é, no primeiro caso, uma *string*, no segundo uma lista de *strings*.

```
f.write('Hello world!\n')
messages = ['How are you?,'And your wife?']
f.writelines(messages)
```

- Notar que apenas *strings* podem ser guardadas em ficheiros. Objectos de outro tipo têm primeiro de ser convertidos em *strings* antes de poderem ser guardados num ficheiro.

Ficheiros

- Objectos simples podem ser convertidos com a função `str`, as aspas oblíquas (‘ ‘) que são interpretadas como conversão para `string`, ou utilizando o operador de formatação `(%)`.

```
a = str(123.2)
b = ‘1234‘
f = 123.2; n = 189
s = "%10.2f %d %s %s" % (f,n,a,b)
```

- Objectos mais complexos podem ser *serializados* automaticamente utilizando o módulo `pickle`.

Formatação de *output*

- A conversão de objectos em *strings*, convenientemente formatadas, é facilitada pela utlização do *operador de formatação* %. Uma *string* formatada é produzida com a expressão:

```
format-string % (o1,o2,o3...)
```

- Exemplo:

```
>>> x = "This is %5.3f with %d decimals" %\
        (math.pi,3)
>>> print x
This is 3.142 with 3 decimals
```

Formatação de *output*

- A *format-string* é uma *string* que pode conter códigos de formatação da tabela seguinte:

`%s`	*string*
`%c`	caracter
`%d`	número inteiro decimal
`%o`	numero inteiro octal
`%x`	número inteiro hexadecimal
`%f`	número em vírgula flutuante
`%e`	número em vírgula flutuante, notação científica
`%g`	número em vírgula flutuante, formato alternativo