

Ficheiros

Programação I

2016.2017

Teresa Gonçalves
tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Como programar?



Perceber o problema

Input e Output

Pensar numa solução

Dividir o problema em problemas mais simples

Resolver os problemas mais simples

Resolver o problema mais complexo

Implementar a solução

Utilizar funções para estruturar a resolução dos sub-problemas

Testar a solução

Fazer vários testes

Escolher valores que induzam comportamentos diferentes do programa

Sumário

Persistência

Propriedades

Ficheiro

Permite o armazenamento persistente de dados

Input – leitura do ficheiro

Output – escrita do ficheiro

Exemplo

```
>>> f=open('texto.txt')  
>>> type(f)  
<class 'io.TextIOWrapper'>
```

Persistência

Característica que permite a permanência para além da execução do programa

Variáveis

Apenas existem durante a execução do programa

Funções e métodos

Operações

Abertura

Associar uma variável a um ficheiro

Leitura

Ler uma linha do ficheiro (para string)

Ler todo o ficheiro para uma string

Ler todo o ficheiro para uma lista de linhas

Escrita

Fecho

Terminar a associação da variável ao ficheiro

Posicionamento

Obter posição atual

Alterar posição: avançar, recuar

Função open()

f=open(name, mode)

name: nome do ficheiro a aceder

mode: modo de utilização (opcional)

‘r’ - read: apenas para leitura (valor por omissão)

‘w’ - write: apenas para escrita. Apaga o conteúdo anterior do ficheiro, se existir

‘a’ - append: o que for escrito é adicionado no final do ficheiro

‘r+’ - acesso para leitura e escrita

Devolve erro se name não for encontrado

```
Traceback (most recent call last):
```

```
File "file_exp0.py", line 4, in <module>
```

```
    f= open("texto1.txt")
```

```
IOError: [Errno 2] No such file or directory: 'texto1.txt'
```

Método close()

f.close()

Fecha o ficheiro, libertando os recursos associados ao mesmo

Notas

É possível invocar o fecho mais que uma vez (sem erro)

Depois de fechado, uma tentativa de leitura gera um erro

É importante fechar cada um dos ficheiros abertos pelo programa!

Método read()

str=f.read()

Devolve a string com o texto do ficheiro da posição atual até do final do mesmo

str=f.read(size)

Devolve a string com o texto do ficheiro da posição atual até um comprimento máximo de `size` bytes

Exemplo

Conteúdo do ficheiro f.txt

abcdefghijkl

Exemplo 1

```
>>> f=open('f.txt')
>>> s=f.read()
>>> f.close()
>>> print(s)
abcdefghijkl
```

Exemplo 2

```
>>> f=open('f.txt')
>>> s1=f.read(4)
>>> s2=read(4)
>>> f.close()
>>> print(s1)
abcd
>>> print(s2)
efgh
```

Método readline()

str=f.readline()

Lê uma linha do ficheiro e devolve a string correspondente incluindo o carácter de mudança de linha (\n), se existir (a última pode ter ou não).

str=f.readline(size)

Lê uma linha do ficheiro e devolve a string correspondente incluindo o carácter de mudança de linha (\n), se existir, até ao máximo de size caracteres

Notas

Linha em branco: a string devolvida é '\n'

Final do ficheiro: assinalado com uma string vazia

Exemplo

```
f=open('f.txt')
s= f.readline()      # lê primeira linha
while s!='':         # enquanto não termina (str vazia)
    print( s )
    s= f.readline()  # e lê a próxima linha
    print(s)
# após a leitura de todas as linhas
f.close()
```

Método readlines()

l1st=f.readlines()

Faz a leitura entre a posição atual e o final do ficheiro, devolvendo o conteúdo numa lista de strings. Cada string representa uma linha, terminando com o respetivo '\n' (a última pode ter ou não)

Exemplo

```
f=open( 'f.txt' )  
lineList= f.readlines()  
f.close()  
for str in lineList:  
    print(str)
```

Método write()

f.write(str)

Escreve a string str no ficheiro f

Nota

Dependendo do buffering do sistema operativo o conteúdo escrito poderá demorar algum tempo até surgir no ficheiro...

A escrita é garantidamente processada com `file.flush()` ou `file.close()`

Método writelines()

f.writelines(seq)

Escreve uma sequência de strings no ficheiro f

seq pode ser uma lista ou um tuplo de strings

Nota

Não é adicionado nenhum separador entre as strings...

Exemplo

```
f.writelines(['um', 'dois', 'tres'])
```

```
f.writelines(('quatro', 'cinco'))
```

Conteúdo escrito: umdoistresquatrocinco

Método tell()

f.tell()

Devolve a posição atual de acesso ao ficheiro (inteiro)

Exemplo

```
>>> f=open( 'f.txt' )
>>> print( f.tell() )
0
>>> s1= f.read(4); print( f.tell() )
4
>>> s2= f.read(4); print( f.tell() )
8
>>> f.close()
```

Método seek()

f.seek(pos)

Altera para pos a posição corrente no ficheiro

Exemplo

```
pos=f.tell()      # obter a posição atual
s=f.readline()    # lê uma string
print(s)
f.seek(pos)       # voltar à posição anterior
s= f.readline()   # lê a mesma string
print(s)
```

Método flush()

f.flush()

Pedido ao sistema operativo para esvaziar o buffer (flush) de saída associado ao ficheiro

Obriga que escritas pendentes se concretizem imediatamente

Nota

Operação de baixo nível; raramente necessária

Método truncate()

f.truncate(size)

Altera o tamanho do ficheiro (descartando o restante conteúdo)

size: tamanho máximo do ficheiro (opcional)

Por omissão, o ficheiro é truncado na posição atual

Exemplo

Conteúdo de f: 123456789

```
>>> f= open( 'f.txt' , 'r+' )
```

```
>>> f.truncate(4)
```

```
>>> f.close()
```

Conteúdo de f:1234

Iteração

Iteração

Outra forma de ler o conteúdo linha a linha

Ciclo for

```
>>> inputFile= open(name)
>>> for linha in inputFile:
    print( linha )          # string, inclui eventual \n
>>> inputFile.close()
```

Módulo pickle

Módulo pickle

Facilita a manipulação de objetos que não sejam strings

Automatiza a sua transformação para uma representação serializada (em string)

Exemplo

```
import pickle
d= [1,2,'tres']
f= open("f2.txt", 'w')
pickle.dump(d,f)
f.close()
```

```
f= open("f2.txt")
d2= pickle.load(f)
f.close()
print(d==d2)    # True
print(d2)      # [1,2,'tres']
```

Aplicações

Copiar um ficheiro

```
in_name= raw_input('nome do ficheiro existente')
out_name= raw_input('nome do ficheiro copia')

# iniciar ficheiros, um para leitura outro para escrita
inputFile= open(in_name,'r')
outputFile= open(out_name,'w')

s= inputFile.readline()
while s!='':
    outputFile.write(s)      # enquanto há mais conteúdo
    s= inputFile.readline()  # escreve na cópia
                             # e lê a próxima linha

inputFile.close()
outputFile.close()
```

Copiar um ficheiro, ciclo for

```
in_name= raw_input('nome do ficheiro existente')
out_name= raw_input('nome do ficheiro copia')

# iniciar ficheiros, um para leitura outro para
escrita
inputFile= open(in_name, 'r')
outputFile= open(out_name, 'w')

for linha in inputFile:
    outputFile.write(linha)

inputFile.close()
outputFile.close()
```

Formatação de texto

Imaginemos que um ficheiro tem o seguinte conteúdo

```
manuel rosado: 19
maria ABRANTES:      31
Carlos gomes:159
MANUELA moTA: 0
```

E queremos normalizá-lo para

```
NOME: manuel rosado,      VALOR: 19
NOME: maria abrantess,    VALOR: 31
NOME: carlos gomes,       VALOR: 159
NOME: manuela mota,       VALOR: 0
```

Formatação de texto

```
in_name= raw_input('nome do ficheiro original')
out_name= raw_input('nome do ficheiro normalizado')

infile= open(in_name)
outfile= open(out_name, 'w')

for linha in infile:
    l= linha.split(':')    # separar nome do valor
    nome= l[0].strip().lower()
    num= l[1].strip()
    outfile.write('NOME: %s,\tVALOR: %4s\n' % (nome, num))

infile.close()
outfile.close()
```