Nesta aula...

Conteúdo

1 Ficheiros

Ficheiros

Ficheiros

- ficheiro: um *conjunto de dados* conexo, e.g.:
  - uma carta... uma faixa de música... o código-fonte de um programa ... um "executável"...
- identificado por um *caminho* (absoluto/relativo)
- ao contrário de valores de variáveis, os ficheiros são *persistentes*
- Para o sistema operativo um ficheiro é sempre uma sequência de *bytes*
- suportes físicos: discos magnéticos, memórias flash, CD-Rs, ...

Manipular ficheiros em Python

Três fases:

1. abrir o ficheiro
2. ler e/ou escrever no ficheiro
3. fechar o ficheiro

Manipular ficheiros em Python

Acesso por meio de objectos de tipo `file`:

```
>>> f = open("test.dat","w")
>>> type(f)
<type 'file'>
>>> print f
<open file 'test.dat', mode 'w' at fe820>
```

1.1 Métodos sobre ficheiros

Abrir um ficheiro

```
f = open(caminho, modo)
```

Modos:

'r' leitura (ficheiro deve já existir)

'w' escrita (se o ficheiro já existir: remove o conteúdo)

'a' escrita (se o ficheiro já existir: acrescenta ao final)

Métodos sobre ficheiros

Mais comuns:

f.write(str) escrever uma cadeia

f.read() lê todo o conteudo (uma "string")

f.read(n) lê apenas n caracteres

f.readline() lê uma linha de texto

f.close() terminar leitura/escrita no ficheiro

Importante: deve *sempre* usar `close` para garantir que o ficheiro é escrito correctamente!

Exemplo de escrita

```
>>> f = open("test.dat", "w")
>>> f.write("Ola mundo!")
>>> f.write("Adeus mundo...")
>>> f.close()
```

test.dat

```
Ola mundo!Adeus mundo...
```

Exemplo de leitura

```
>>> f = open("test.dat", "r")
>>> txt = f.read()
>>> txt
'Ola mundo!Adeus mundo...'
>>> f.close()
```

Se o ficheiro não existe

```
>>> f = open("test.cat", "r")
Traceback (most recent call last):
IOError: [Errno 2] No such file or directory:
  'test.cat'
```

1.2 Ficheiros de texto vs. binários

Ficheiros de texto vs. binários

- Ficheiros de texto: contêm apenas caracteres imprimíveis e espaços, tabulação, etc.
- Ficheiros binários: imagens JPEG, audio MP3, programas, etc.
- Por omissão: `open` assume que o ficheiros é de texto
- Em linux/unix: *não há diferença* essencial entre os ficheiros de texto e binários; ambos podem ser processados – abertos, lidos, escritos... – da mesma forma.

1.3 Procurar ocorrências de uma palavra

Procurar ocorrências de uma palavra

- função para procurar uma palavra num ficheiro
- imprimir a(s) linha(s) em a palavra ocorre
- versão reduzida do comando `grep` de UNIX

Procurar ocorrências de uma palavra

```
>>> procurarPalavra("lusiadas.txt", "armas")
As armas e os barões assinalados,
Na qual vos deu por armas, e deixou
Fizeram, só por armas tão subidos,
Por armas têm adargas o terçados;
Mostra das fortes armas de que usavam,
De mim, da Lei, das armas que trazia.
Se as armas queres ver, como tens dito,
Dá-lhe armas o furor desatinado.
```

Procurar ocorrências de uma palavra

```
def procurarPalavra(fich, palavra):
    "Imprime linhas em que ocorre uma palavra."
    f = open(fich, "r")
    linha = f.readline()
    while linha!="": # fim do ficheiro?
       # palavra ocorre na linha?
       if palavra in linha:
          print linha
       # próxima linha
       linha = f.readline()
    # fim do ciclo
    f.close()
```

2 Excepções

Excepções

2.1 Apanhar excepções

Apanhar excepções

```
try:
   # código que poderá lançar um erro
   ⋮
except Excepção:
   # código em caso de erro
   ⋮
```

Algumas excepções pré-definidas:

IOError leitura/escrita de ficheiros

ValueError argumento fora do domínio e.g. `sqrt(-1)`

IndexError índice fora de limites

TypeError erro de tipos

Exemplo: conteúdo de um ficheiro

```
def conteudo(fich):
   "Lê o conteúdo de um ficheiro, se existir."
   try:
```

```
      # tenta abrir o ficheiro
      f = open(fich, "r")
      # obtém o conteúdo e fecha
      cont = f.read()
      f.close()
   except IOError:
      # erro: o ficheiro não existe
      # conteudo é vazio
      cont = ""
   return cont
```

2.2 Lançar excepções

Lançar excepções

- O programador pode *lançar* excepções explicitamente:

`raise` *excepção*[, *descripção*]

- Útil em situações de erro
- Evita confundir um erro com um valor correcto

Exemplo: factorial

```
def factorial(n):
   "Calcula o factorial de n."
   # lança excepção para n<0
   if n<0:
      raise ValueError, "n é negativo"
   # para n>=0, calcula n!
   fact = 1
   while n>0:
      fact = fact*n
      n = n-1
   return fact
```

Exemplo: factorial

```
>>> factorial(5)
120
>>> factorial(-1)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "<stdin>", line 4, in factorial
ValueError: n é negativo
```

Mais informação...

Mais informação sobre ficheiros, excepções e python em

www.python.org/doc/2.5.2/tut/tut.html