

Operaciones sobre ficheros (IV)

- Con ficheros de texto:

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

```
for line in f :
```

```
    print (line)
```

```
f.close()
```

- Con ficheros binarios:

```
with open("fichero", "rb") as f:
```

```
    byte = f.read(1)
```

```
    while byte:
```

```
        byte = f.read(1)
```

© Prof.Miguel García Silvente

146

Operaciones sobre ficheros (V)

Consultas sobre el tipo fichero:

name Devuelve el nombre del archivo

closed Devuelve si el fichero está cerrado

mode Devuelve el método de apertura

Ejemplo lectura de datos (I)

Leer datos y ponerlos en dos listas de números reales:

```
x = []; y = []
lineas = open("datos.txt")
for linea in lineas:
    xval, yval = linea.split()
    x.append(float(xval))
    y.append(float(yval))
lineas.close()
```

© Prof.Miguel García Silvente

148

Ejemplo lectura de datos (II)

Leer datos en un diccionario:

```
códigos = {}
f = open("códigos.txt", "r")

for line in f :
    [código, lugar] = line.rstrip().split(' ')
    códigos[código] = lugar
f.close()
```

© Prof.Miguel García Silvente

149

Guardar datos

- Escribir cadenas

`write(<cadena>)`

cada cadena se guarda a continuación de la anterior (sin separadores)

`f.write('Hola')`

guardaría `Hola`

`f.write ('Adiós')`

- Varías líneas al mismo tiempo

`f.writelines(<secuencia>)`

Ejemplo escritura (I)

Escribir cadenas

```
with open("salida.txt", "w") as f :
```

```
    f.write("Esto")
```

```
    f.write(" es ")
```

```
    f.write("una prueba")
```

```
L = ['esto', ' es ', 'una prueba']
```

```
with open("salida2.txt", "w") as f :
```

```
    f.writelines(L)
```

Ejemplo escritura (II)

Escribir datos formateados:

```
with open("datos.txt", "w") as f :  
    for i in range(100, 200) :  
        f.write('{:3d} {:7d}'.format(i, i**2))  
        f.write('\n')
```

Entrada y salida estándar (I)

Módulo sys

```
import sys
```

sys.stdin Entrada estándar, abierto para lectura

sys.stdout Salida estándar, abierta para escritura

sys.stderr Salida de error, abierta para escritura

```
import sys
```

```
for line in sys.stdin :
```

```
    sys.stdout.write(line)
```

Entrada y salida estándar (II)

Import sys

```
linea = sys.stdin.readline()
```

```
while linea :
```

```
    sys.stdout.write(line)
```

```
    linea = sys.stdin.readline()
```

Información sobre el sistema

sys.path Devuelve una lista de strings con el path del sistema.

sys.argv Lista de argumentos del script.

sys.maxint El entero más grande que procesa la CPU

sys.platform Devuelve una cadena que identifica la plataforma.

sys.version Versión de python junto con otros datos.

Ejemplo ficheros csv

Leer datos separados por comas

```
with open("datos.csv", "r") as f :  
    datos = f.readline().split(',')  
  
import csv
```

```
with open('datos.csv', 'Ur') as f:  
    data = list(tuple(rec) for rec in csv.reader(f,  
delimiter=','))
```

Generar datos en la web

Hay que generar las cabeceras adecuadas.
Por ejemplo:

```
print('Content-type: text-plain\n\nDatos:', data)
```

Leer datos de la web

Leer datos

```
import urllib.request  
f = urllib.request.urlopen('http://www.python.org/')  
print(f.read(100).decode('utf-8'))  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML  
1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml
```

Leer página de la web

```
import urllib.request  
f = urllib.request.urlopen('http://www.ugr.es/')  
print(f.read(100).decode('utf-8'))  
linea = ""  
while (linea != "") :  
    linea = f.readline().decode('utf-8')  
    print (linea)
```

Enviar datos en la web

```
import urllib.request  
  
req =  
urllib.request.Request(url='http://arrecife.ugr.es/cgi-bin/escribir.py',  
data=b'data=2')  
  
f = urllib.request.urlopen(req)  
  
print(f.read().decode('utf-8'))
```

_____ En el servidor:

```
import cgitb;  
  
import cgi  
form = cgi.FieldStorage()  
form_data = form.getFirst('data', '1')  
data = [int(x) for x in form_data.split(',')]  
print('Content-type: text-plain\n\nDatos:', data)
```

Autenticación HTTP

```
import urllib.request  
  
# OpenerDirector con autenticación HTTP  
auth_handler = urllib.request.HTTPBasicAuthHandler()  
auth_handler.add_password(realm='PDQ Application',  
    uri='https://mahler:8092/site-updates.py',  
    user='klem', passwd='kadidd!ehopper')  
  
opener = urllib.request.build_opener(auth_handler)  
# instalado para poder usarse con urlopen  
urllib.request.install_opener(opener)  
urllib.request.urlopen('http://www.example.com/login.html')
```

módulo requests

```
import requests
page = requests.get("https://www.ugr.es")
page
<Response [200]>
page.status_code
200
page.content
b'<!DOCTYPE html>\n<html lang="es" dir="ltr"
prefix="content: http://purl.org/rss/1.0/modules/content/ dc:
http://purl.org/dc/terms/ foaf: http://xmlns.com/foaf/0.1/ og:
http://ogp.me/ns# rdfs: http://www.w3.org/2000/01/rdf-sch
```

© Prof.Miguel García Silvente

162

Guardar estructuras de datos

Guardándolos como cadenas de caracteres:

```
lista = ['text1', 'text2']
a = [[1.3, lista], 'texto']
f = open('tmp.dat', 'w')
```

```
# convertimos los datos a string
f.write(str(a)) # o f.write(repr(a))
f.close()
```

© Prof.Miguel García Silvente

163

Leer estructuras de datos

```
f = open('tmp.dat', 'r')
```

```
datos = eval(f.readline())
# [[1.3, ['text1', 'text2']], 'texto']
# a = eval(repr(a))
```

Guardar datos en binario

- Usando cadenas

```
myFile = open('datos.bin', 'wb')
myFile.write("\x5F\x9D\x3E");
myFile.close()
```

- Usando el módulo struct

```
st = struct.pack(<formato>, <dato1>, <dato2>, ...)
```

Ejemplo:

```
with open('prueba.bin', 'wb') as f :
```

```
    st= struct.pack('2d', 1.345, 3.456)
```

```
    f.write(st)
```

Recuperar datos binarios

- Usando `ord()`

```
ch1 = myFile.read(1) # read 1 byte  
d1 = ord(ch1)
```

- Usando el módulo `struct`

```
lista = struct.unpack(<formato>, <datos>)
```

Ejemplo (dependiente del sistema):

```
with open('prueba.bin', 'rb') as f :  
    datos = f.read()  
    valores = struct.unpack('2d', datos[0:8*2])  
    print (valores[0], valores[1])
```

© Prof.Miguel García Silvente

166

Guardar datos completos (I)

- Se debe usar el módulo `pickle`:

```
pickle.dump(<variable>, <fichero>)  
>>> pickle.dump(L, f)
```

- Se puede recuperar:

```
<variable> = pickle.load(<fichero>)  
>>> L = pickle.load(f)
```

- Se recupera el tipo y contenido de la variable/objeto

Guardar datos completos (II)

```
L = {23, "aa", (3,5)}
```

```
pickle.dump(L, open("ll.txt", "wb"))
```

```
L2 = pickle.load(open("ll.txt", "rb"))
```

```
print (L2)
```

Trabajar con datos en disco

Permite manejar los datos sin cargarlos.

```
import shelve
```

```
database =
```

```
shelve.open('datos_shelve.bin')
```

```
database['a1'] = 1234
```

```
database['a2'] = 543
```

```
database['a3'] = 234
```

```
database['a123'] = (1234, 543, 234)
```

```
if 'a1' in database:
```

```
    a1 = database['a1']
```

```
del database['a2']
```

```
database.close()
```

Programación Funcional

© Prof.Miguel García Silvente

170

Programación funcional

- Python usa funciones de orden superior (similar a Scheme). Son funciones cuyos parámetros son, a su vez, funciones:
 - `map`
 - `reduce`
 - `filter`
- Las funciones lambda se suelen utilizar como parámetros.

Programación funcional: map

- `map(<función>, <objeto iterable>, ...)`
- Se aplica la función a cada uno de los elementos del objeto y crea una secuencia con los resultados.
- Hay que hacer un cambio de tipo a `list()` (porque devuelve un iterador)

```
nums = [0, 4, 7, 2, 1, 0 , 9 , 3, 5, 6, 8, 0, 3]
nums = list( map(lambda x : x % 5, nums) )
print(nums)
[0, 4, 2, 2, 1, 0, 4, 3, 0, 1, 3, 0, 3]
```

© Prof.Miguel García Silvente

172

Programación funcional (I)

```
def cuadrado(x):
    return x*x
>>> list( map(cuadrado, range(10,20)) )
[100, 121, 144, 169, 196, 225, 256, 289, 324, 361]
```

Muchos programadores de python prefieren usar comprensión de listas.

- Usando comprensión de listas:

```
[ 2*x+1 for x in [10, 20, 30] ]
```

- Usando programación funcional:

```
list( map( lambda x: 2*x+1, [10, 20, 30] ) )
```

Ejemplo map (I)

```
a = [1,2,3,4]
```

```
b = [17,12,11,10]
```

```
c = [-1,-4,5,9]
```

```
map(lambda x,y:x+y, a,b)
```

```
[18, 14, 14, 14]
```

```
map(lambda x,y,z:x+y+z, a,b,c)
```

```
[17, 10, 19, 23]
```

```
map(lambda x,y,z:x+y-z, a,b,c)
```

```
[19, 18, 9, 5]
```

Ejemplo map (II)

¿Qué hace el siguiente código?

```
sec = range(8)
```

```
list(map(lambda x,y: (x,y), sec,list(map(lambda x: x*x, sec))))
```

```
[ (0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49)]
```

en python2

```
sec = range(8)
```

```
map(None, sec, map(lambda x: x*x, sec))
```

© Prof.Miguel García Silvente

176

Programación funcional: problema (I)

Dada una lista de puntos 3D en forma de tuplas, crea una lista con la distancia de cada uno de ellos al origen de coordenadas

Usando bucles

- distancia(x, y, z) = $\sqrt{x^{**}2 + y^{**}2 + z^{**}2}$)
- iterar sobre la lista y añadir los resultados a la nueva lista

Programación funcional: problema (II)

```
from math import sqrt  
  
puntos = [(2, 1, 3), (5, 7, -3), (2, 4, 0), (9, 6, 8)]  
  
def distancia(punto) :  
    x, y, z = punto  
    return sqrt(x**2 + y**2 + z**2)  
  
distancias = list(map(distancia, puntos))
```

Programación funcional: filter (I)

filter(<función>, <objeto iterable>)

- Procesa cada elemento del objeto iterable aplicándole la función.
- Si la función devuelve True para ese elemento, entonces **filter** lo devuelve.
- En versiones anteriores a la 3 se devuelve una lista
- En la versión 3 es necesario hacer un cambio de tipo a list

Programación funcional: filter (II)

```
nums = [0, 4, 7, 2, 1, 0 , 9 , 3, 5, 6, 8, 0, 3]
nums = list(filter(lambda x : x != 0, nums))
```

```
print(nums)
[4, 7, 2, 1, 9, 3, 5, 6, 8, 3]
```

```
def par(x):
    return x % 2 == 0
cuadrado = lambda x : x*x
list( map(cuadrado, filter(par, range(10,20)))) )
[100, 144, 196, 256, 324]
```

Ejemplo filter (I)

```
Nan = float("nan")
scores = [[NaN, 12, .5, 78, math.pi],
          [2, 13, .5, .7, math.pi / 2],
          [2, NaN, .5, 78, math.pi],
          [2, 14, .5, 39, 1 - math.pi]]
```

Dada una lista de listas que contiene respuestas a un examen, filtra aquellas preguntas para las que no hay respuesta (indicada por NaN)

Ejemplo filter (II)

```
NaN = float("nan")
puntuaciones = [[NaN, 12, .5, 78, pi],[2, 13, .5, .7, pi / 2],
                 [2,NaN, .5, 78, pi],[2, 14, .5, 39, 1 - pi]]
#solución 1
def es_NaN(respuestas) :
    for num in respuestas :
        if isnan(float(num)) :
            return False
    return True
valid = list(filter(es_NaN, puntuaciones))
print(valid)
#solución 2
valid = list(filter(lambda x : NaN not in x, puntuaciones))
print(valid)
```

© Prof.Miguel García Silvente

182

Programación funcional: reduce (I)

reduce(<función>, <objeto iterable>[,inicializador])

- Se aplica a cada elemento del objeto iterable junto con el resultado hasta ese momento, acumulando los resultados.
- **función** debe tener dos argumentos.
- Si existe el inicializador, se usará como primer argumento para aplicar la función
- En python 3 reduce() hay que importar un módulo:
`from functools import reduce`

© Prof.Miguel García Silvente

183

Programación funcional: reduce (II)

```
def mas(x,y):  
    return (x + y)  
from functools import reduce  
lista = [1,2,3,4,5,6]  
reduce(mas, lista)  
21
```

```
def mas(x,y):  
    return (x + y)  
from functools import reduce  
lista = ['h','o','l','a']  
reduce(mas, lista)  
'hola'
```

© Prof.Miguel García Silvente

184

Programación funcional: reduce (III)

```
nums = [1, 2, 3, 4, 5, 6, 7, 8]
```

```
from functools import reduce
```

```
nums = list(reduce(lambda x, y : (x, y), nums))
```

```
print(nums)
```

```
(((((1, 2), 3), 4), 5), 6), 7), 8)
```

© Prof.Miguel García Silvente

185

Programación funcional: media (I)

Dada una lista de números, calcula su media usando
`reduce()`

Usando bucles:

- sumar todos los elementos de la lista.
- dividir la suma por la longitud de la lista.

Programación funcional: media (II)

```
nums = [92, 27, 63, 43, 88, 8, 38, 91, 47, 74, 18, 16,  
       29, 21, 60, 27, 62, 59, 86, 56]
```

```
media = reduce(lambda x, y : x + y, nums) / len(nums)
```

Ejemplos de programación funcional

```
def pot(x, y):  
    return x*y**2
```

```
print (reduce(pot, [1,2,3,4,5,6]))
```

```
map(lambda x: x*x*x, range(1, 11))
```

¿Qué hace el siguiente código?

```
from functools import reduce  
f = lambda a,b: a if (a > b) else b  
reduce(f, [47,11,42,102,13])
```

map reduce

Problema: Dado un email determina si es spam

Solución: Cuenta las ocurrencias de ciertas palabras y si aparecen con demasiada frecuencia se considera spam.

<https://es.mailjet.com/blog/news/palabras-alarma-spam/>

map reduce

```
email = ['tarjeta', 'reducimos deuda', 'sorteo', 'ventas',  
gratis]  
  
def inEmail (x):  
    if (x in palabras_spam):  
        return 1;  
    else:  
        return 0;  
  
map(inEmail, email)          #[1, 0, 0, 0, 1, 1, 0]  
  
reduce((lambda x, xs: x + xs), map(inEmail, email)) #3
```