

Cuestiones sobre listas y referencias

- ¿qué ocurre en este caso?

`L = [[0, 0]] * 5`

`L[2][0] = 7`

- ¿y en este otro?

`L = ["abc"] * 3`

`L = [L] * 5`

`L[2][0] = 'cde'`

Conversión de str a list

Hay que tener cuidado porque

`l = []`

`s = "Hola"`

`l += s`

- No da como resultado una lista con un único elemento “Hola”

Resumen de Operaciones sobre listas (I)

- `a = []` crea una lista vacía
- `a = [1, 4.4, 'run.py']` asigna un valor
- `a.append(elem)` añade un elemento al final
- `a + [1,3]` concatena dos listas
- `a[3]` accede a un elemento
- `a[-1]` accede al último elemento
- `a[1:3]` devuelve una sublista
- `del a[3]` elimina un elemento

Prof.Miguel García Silvente

97

Resumen de Operaciones sobre listas (II)

- `a.remove(4.4)` elimina un elemento con ese valor
- `a.index('run.py')` devuelve la posición donde se encuentra un elemento (**ValueError** si no está)
- `a.count(v)` cuenta el número de veces que aparece un elemento (en este caso no hay error)
- `len(a)` devuelve el número de elementos
- `min(a)` devuelve el elemento más pequeño
- `max(a)` devuelve el elemento mayor
- `sum(a)` suma todos los elementos

Prof.Miguel García Silvente

98

Resumen de Operaciones sobre listas (III)

- `a.sort()` ordena una lista
- `as = sorted(a)` devuelve una versión ordenada
- `a.reverse()` invierte el orden de los valores de una lista
- `b[3][0][2]` indexado anidado
- `isinstance(a, list)` devuelve True si `a` es una lista
- `l1.copy()` devuelve una copia de la lista

Prof.Miguel García Silvente

99

Ejemplo string + list (I)

- `readline()` Lee una línea completa
`linea = sys.stdin.readline()`
- `readlines()` Lee una serie de líneas, cada línea va en una posición de la lista
`texto = sys.stdin.readlines()`
- `split()` divide una cadena y devuelve una lista de subcadenas
`palabras = cad.split()`
`datos = cad.split("es")`
- `strip()` elimina separadores de ambos extremos

Prof.Miguel García Silvente

100

Ejemplo string + list (II)

```
'el','gato','sentado','sobre','el','coche'].count('el')  
'el gato sentado sobre el coche'.split().count('el')  
palabras = 'el gato sentado sobre el coche'.split()  
' el gato '.strip()  
' el gato '.lstrip()
```

Prof.Miguel García Silvente

101

Ejemplos

```
primos = [2, 3, 5, 7, 11, 13, 17, 19]
```

```
[8 / p for p in primos]
```

```
[p for p in primos if p % 3 > 0]
```

```
[ [0,0,0] for x in range(2,5)]
```

```
[93 % p for p in primos if 93 % p != 0]
```

```
[5 ** p for p in primos if p % 4 == 0]
```

```
[ [fil[i] for fil in matriz] for i in range(len(matriz))]
```

```
L = [ [ 0 for i in range(5)] for j in range(6)]
```

Prof.Miguel García Silvente

102

Tipo de dato set

- Es una colección no ordenada de datos (pueden ser de distinto tipo).
`s1 = {1,3,5,7,5,4,3,2,1}`
`s2 = {1, 'a' }`
- Debe ser de tipo “hashable” (que se pueda “aplicar una función hash sobre él”).

```
>>> s1 = {(1, 'a'), (2, 'b')}
```

```
>>> s2 = {[1,2,3], [1,3,4]}
```

...

```
TypeError: unhashable type: 'list'
```

Prof.Miguel García Silvente

103

Tipos de datos conjunto

- Existe `set` como versión mutable y `frozenset` e `immutableSet` como inmutables
- Ejemplo:

```
dias_semana = {'lun', 'mar', 'mié', 'jue', 'vie'}
```

```
dias_semana = set('lun', 'mar', 'miércoles', 'jueves', 'viernes')
```

```
dias_semana.add('sáb')
```

'mié' in `dias_semana` devuelve `True`

`dias_semana.remove('mié')` elimina elemento

Prof.Miguel García Silvente

104

Operaciones sobre conjuntos (I)

- Añadir un elemento `add(x)`
- Eliminar un elemento `remove(x)` `discard(x)`
con diferente comportamiento:
`s1 ={1,3,5,7,5,4,3,2,1}`
`s1.discard(8) # no ocurre nada`
`s1.remove(8) # genera excepción KeyError: 8`
- Devuelve y elimina un elemento aleatorio `pop()`
- Copiar todos los elementos `copy()`
- Eliminar todos los elementos `clear()`

Prof.Miguel García Silvente

105

Operaciones sobre conjuntos (II)

Incluye `union`, `intersection`, `difference`,
`symmetric_difference`

`s1 ={1,3,5,7,5,4,3,2,1}`

`s2 ={2, 4, 6, 8, 6, 4, 3, 2, 1}`

`union_s1_s2 = s1.union(s2)`

`intersect_s1_s2 = s1.intersection(s2)`

`diff = s1.difference(s2) #set((5,7))`

`s_diff = s1.symmetric_difference(s2) # set((5,6,7,8))`

Prof.Miguel García Silvente

106

Comprensión de conjuntos

De forma similar a las listas

```
>>> a = {x for x in 'abracadabra' if x not in 'abc'}  
set(['r', 'd'])
```

```
>>> vocalesPalabra = {x for x in 'esto es una prueba'  
if x in 'aeiou'}  
set(['a', 'u', 'e', 'o'])  
>>> L = [34, 45, 87, 90, 32, 4]  
>>> mayores50 = {x for x in L if x>50}  
set([90, 87])
```

Prof.Miguel García Silvente

107

Tipo de dato dict (I)

Tipo *diccionario*: Permite guardar pares (clave, valor) y poder acceder a ellos de forma eficiente a través de la clave

Las claves deben ser **inmutables**. Los valores no tienen restricciones.

Prof.Miguel García Silvente

108

Tipo de dato dict (II)

Se pueden guardar claves y valores de distintos tipos

Operaciones: buscar, borrar, modificar o definir pares.

También se les conoce como **tablas hash** o **arrays asociativos**

Asignación y acceso (I)

- Se usa {}, separando clave y valor con : y cada par con ,
`datos = {'nombre' : 'Pepe', 'edad': 40}`
`# datos = dict(nombre='Pepe', edad=40)`
- Se accede con [] usando la clave
`datos['nombre']`
`datos['edad']`
`datos['Pepe'] ?????`
- Internamente usan hashing

Asignación y acceso (II)

- Ejemplo: temperaturas de ciudades

```
temps = {'Madrid' : 29, 'Granada': 30, 'Valencia' :  
28}
```

```
#o temps = dict(Madrid=29, Granada=30,  
Valencia=28)
```

- Modificar datos (si ya existe)

```
temps['Granada'] = 29
```

- Insertar datos (si no existía)

```
temps['Málaga'] = 30
```

Prof.Miguel García Silvente

111

Actualización

- Se actualiza usando la clave con []

```
datos['nombre'] = 'José'
```

```
datos['domicilio'] = 'Casa'
```

```
datos['id'] = 4532
```

- Los diccionarios no tienen ningún orden preestablecido.

```
>>> datos
```

```
{'nombre': 'José', 'id': 4532, 'edad': 40,  
'domicilio': 'Casa'}
```

Prof.Miguel García Silvente

112

Asignación directa

- Ejemplo: temperaturas de ciudades

```
ciudades = ['Madrid', 'Granada', 'Valencia']
```

```
temps = [29, 30, 28]
```

```
d = dict(zip(ciudades, temps))
```

```
print (d)
```

Borrar datos

- Para borrar un elemento se usa **del**
del datos['nombre']
del temps['Valencia']
- También se puede usar pop
temps.pop('Málaga')
- Se pueden eliminar sólo los valores
temps['Granada'] = None
- Para borrar todos se usa **clear**
datos.clear()

Iterar y buscar

- Se puede iterar sobre las claves:

```
for i in datos :
```

```
    print (i, datos[i])
```

- Se puede usar **in** para comprobar si una clave está en el diccionario

```
if 'Granada' in temps :
```

```
    print ('Temperatura Granada: ', temps['Granada'])
```

```
else :
```

```
    print ('No hay temperatura para Granada')
```

Obtener claves y valores (I)

Los datos y los valores se pueden consultar como listas:

```
>>> temps.keys()
```

```
['Valencia', 'Granada', 'Madrid']
```

```
>>> temps.values()
```

```
[28, 30, 29]
```

Los elementos no están ordenados, solución: **sorted**

```
for i in sorted(temps.keys()) :
```

```
    temp = temps[i]
```

```
    print (temp)
```

Obtener claves y valores (II)

Los datos y los valores se pueden consultar como una lista de pares:

```
for ciudad, temp in temps.items() :
```

```
    print (ciudad, ':', temp)
```

O de forma ordenada:

```
for ciudad, temp in sorted(temps.items()) :
```

```
    print (ciudad, ':', temp)
```

```
ciudades = temps.keys()
```

Prof.Miguel García Silvente

117

Ejemplo de uso de items()

```
estaciones = {'primavera': {'marzo', 'abril','mayo'},  
'verano' : {'junio', 'julio', 'agosto'} }
```

```
for i, j in estaciones.items() :
```

```
    for k in j :
```

```
        print (k)
```

Prof.Miguel García Silvente

118

Número de parámetros variable en una función

- Se indica con * delante del parámetro

```
def func(*datos) :
```

```
    for i in datos :
```

```
        print (i)
```

```
func(5)
```

```
func(5,6,7)
```

```
func([1,'a', 'c'])
```

Parámetros con nombre en una función

- Se indica con ** delante del parámetro

```
def func(**datos) :
```

```
    for i, j in datos.items() :
```

```
        print (i, j)
```

```
func(pepe=1234,juan=34545)
```

Tipo de dato fichero

- Un fichero es una secuencia de bytes almacenada en memoria externa.
- Python maneja los archivos como secuencias de caracteres. Sólo se leen y escriben cadenas.
- Es necesario abrir el fichero para poder acceder y posteriormente cerrarlo.

Operaciones sobre ficheros (I)

- Abrir un fichero (debe poder abrirse):
`open(<nombre>, <modo>)`
donde `<modo>` puede ser r, w, a
`f = open("datos.txt")`
debe poderse acceder a los datos
- Cerrar un fichero.
`f.close()`
- Comprobar si existe un fichero.
`os.path.isfile(<nombre>)`

Operaciones sobre ficheros (II)

- Comprobar permisos sobre el fichero:
`os.stat(<nombre>)`
- Leer un número de bytes de un archivo:
`f.read(<num>)`
si no se indica un número, se lee hasta el final.
- Leer una línea completa:
`f.readline()`
- Leer todo el archivo por líneas (en una lista):
`f.readlines()`

Prof.Miguel García Silvente

123

Operaciones sobre ficheros (III)

- Para posicionarse sobre un byte concreto:
`seek(<num>, <donde>)`
donde puede ser 0:inicio, 1:actual, 2:final
- Conocer la posición actual en el fichero:
`tell()`
- Iterar sobre las líneas del fichero como una lista
`for linea in fich :`
- Como un bloque:
`with open("fichero.txt", "rb") as f:`

Prof.Miguel García Silvente

124

Operaciones sobre ficheros (IV)

- Para ficheros de texto:

```
f = open("datos.txt")  
for line in f :  
    print (line)  
f.close()
```

- Para ficheros binarios:

```
with open("fichero", "rb") as f:  
    byte = f.read(1)  
    while byte:  
        byte = f.read(1)
```

Prof.Miguel García Silvente

125

Operaciones sobre ficheros (V)

Consultas:

name Devuelve el nombre del archivo

closed Devuelve si el fichero está cerrado

mode Devuelve el método de apertura

Ejemplo lectura (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

127

Ejemplo lectura (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

128

Guardar datos

- Escribir cadenas

`write(<cadena>)`

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

`f.write('Hola')`

`f.write ('Adiós')` guardaría **HolaAdiós**

- Varías líneas al mismo tiempo

`f.writelines(<secuencia>)`

Prof.Miguel García Silvente

129

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)`

Prof.Miguel García Silvente

130

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

sys.stdin Entrada estándar abierto para lectura

sys.stdout Salida estándar, abierta para escritura

sys.stderr Salida errores, abierta para escritura

```
import sys
```

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

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

Entrada y salida estándar (II)

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

```
while line :  
    sys.stdout.write(line)  
    line = sys.stdin.readline()
```

Prof.Miguel García Silvente

133

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.

Prof.Miguel García Silvente

134

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

- Sólo hay que tener cuidado de generar las cabeceras adecuadas. Ejemplo:

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

Leer datos de la web (I)

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
```

Prof.Miguel García Silvente

137

Leer página de la web (II)

```
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)
```

Prof.Miguel García Silvente

138

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)
```

Prof.Miguel García Silvente

139

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

140

Leer estructuras de datos

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

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

Prof.Miguel García Silvente

141

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)
```

Prof.Miguel García Silvente

142

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

143

Guardar datos completos

- 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

Prof.Miguel García Silvente

144

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()
```