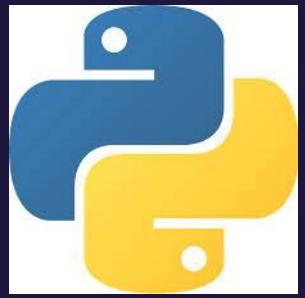
LENGUAJES y HERRAMIENTA PARA CIENCIAS DE DATOS I

Ficheros III
(Ficheros binarios)



UCO ONLINE

Trabajar con ficheros binarios

- Similar a los ficheros de texto, pero:
  - Añadir al final del modo de apertura una b
    - rb, wb, ab, xb, rb+
  - Los datos se escriben en binario → transformarlos
    - encode / bytearray
  - Los datos se leen en binario → transformarlos
    - decode / int / float

Leer y escribir enteros

- Escribir entero a entero
  - bytes([entero])) → para convetir a byte
- Leer el fichero completo → read

```
L = [2, 56, 78]

with open('datos.bin','wb') as f :
    for ele in L:
        f.write(bytes([ele]))

with open('datos.bin','rb') as f:
    t = list(f.read())
    print(t)
```

Leer y escribir enteros

- Escribir un conjunto entero
  - bytes(lista) → para convertir a byte
- Leer el fichero completo → read

```
L = [2, 56, 78]

with open('datos.bin','wb') as f :
    f.write(bytes(L))

with open('datos.bin','rb') as f:
    t = list(f.read())
    print(t)
```

Leer y escribir cadenas

- Utilizar encode y decode para convertir

```
L = ['uno', 'dos', 'tres']

with open('datos.bin','wb') as f :
    for ele in L:
        f.write(ele.encode())

with open('datos.bin','rb') as f:
    t = f.read()
    print(t.decode())
```

Módulo struct

- Crear objetos binarios en formato específico
  - objeto = struct.pack(formato, *valores)
- Extraer datos de un objeto binario creado con un formato específico
  - datos = struct.unpack(formato, objeto)
- Tamaño de un formato
  - struct.calcsize(formato)

Módulo struct

- Formato
  - Orden procesamiento de los bits
    - > big-endian (recomendado)
    - < little-endian
    - = native
  - Tipo y cuántos elementos
    - Enteros → i
    - Booleanos → ?
    - Reales → f
    - Caracteres → s

```
>2if → 2 enteros y 1 float
>i8s → 1 entero y cadena de 8 caracteres
```

Módulo struct

- Almacenar listas de 3 números [int, float, float]

```
import struct
L = [[1, 3.5, 6.7],[2, 5.6, 8.9]]

with open('datos.bin', 'wb') as f:
    for l in L:
        d = struct.pack('>iff',*l)
        f.write(d)
```

Módulo struct

- Leer listas de 3 números (int, float, float)

```
size = struct.calcsize('>iff')
print(size)
with open('datos.bin', 'rb') as f:
    d = f.read(size)
    print(len(d))
    while len(d) == size :
        n = struct.unpack('>iff',d)
        print(n)
        d = f.read(size)
```

Módulo pickle

- Almacenar casi cualquier objeto sin conversion
- Métodos
  - pickle.dump(objeto, fichero_binario)
    - Almacenar un objeto en un fichero
  - objeto = pickle.load(fichero)
    - Leer un objeto desde un fichero

Módulo pickle

- Guardar listas de 3 números

```
import pickle

L = [[1, 3.5, 6.7], [2, 5.6, 8.9]]

with open('datos2.bin', 'wb') as f:
    pickle.dump(2,f)
    for i in range(2):
        pickle.dump(L[i],f)
```

Módulo pickle

- Leer un fichero binario de listas de 3 números

```
10
11 with open('datos2.bin', 'rb') as f:
12     n = pickle.load(f)
13     for i in range(n):
14         a = pickle.load(f)
15         print(a)
```