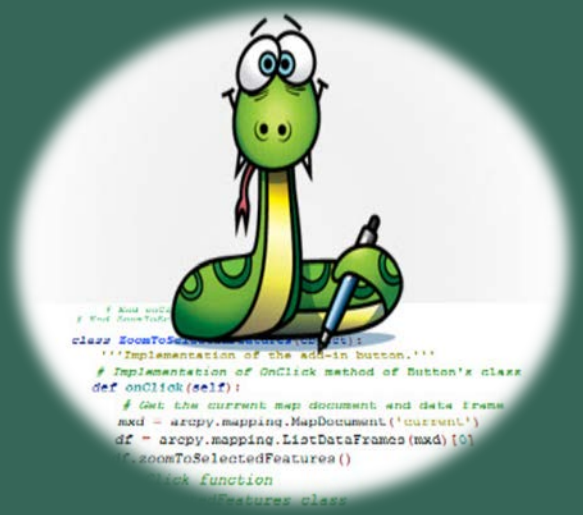
ARXIUS

OPERACIONS AMB ARXIUS



Mª Belén Tortosa Pedrón

PROPIETATS DE *file*

- Es poden accedir a les següents propietats de l'objecte ***file***:
  - **closed**: retorna ***True*** si l'arxiu s'ha tancat. En cas contrari, ***False***.
  - **mode**: retorna la manera d'obertura.
  - **name**: retorna el nom de l'arxiu.
  - **encoding**: retorna la codificació de caràcters d'un arxiu de text.

```
f = open("arxiu.txt", "r+")
contingut = f.read()
nom = f.name
modo = f.mode
encoding = f.encoding
f.close()

if f.closed:
    print ("L'arxiu s'ha tancat correctament")
else:
    print ("L'arxiu està obert")

print(nom)
print(modo)
print(encoding)
```

DIRECTORI ACTUAL: **getcwd()**

- En Python per saber la ruta actual des d'on s'executa el nostre script fem servir el mètode **getcwd()**.

```
import os
print(os.getcwd()) #Current working directory
```

LLISTAT D'ARXIUS I CARPETES: **listdir()**

- La funció **listdir(directory)** retorna una llista que conté arxius i carpetes en un determinada ubicació.
- Pot utilitzar-se **'.'** per indicar el directori actual.

```
import os
print(os.listdir("/usr/games"))
```

CREACIÓ I ELIMINACIÓ DE DIRECTORIS: MAKEDIRS**mkdir ()**

- Les funcions **mkdir(directory)** i **makedirs (directory)** permeten crear carpetes indicant la seva ubicació i, en alguns sistemes, els permisos. (0 ó 777 per defecte, es a dir, totes les operacions).

```
import os
os.mkdir("Libros")
os.makedirs("Musica/Pop/2014")
```

REANOMENAR UN FITXER: **rename()**

- En Python per reanomenar un fitxer fem servir el mètode **rename()**.

```
import os

# Renombrem arxiu.txt per arxiu2.txt
os.rename( "arxiu.txt", "arxiu_v1.txt" )
```

ELIMINAR UN ARXIU: **remove()**

- Per a eliminar un arxiu, utilitzem la funció: **remove()**

```
import os

# Eliminar l'arxiu arxiu_v1.txt
os.remove("arxiu_v1.txt")
```

COPIAR FITXERS

- ***shutil*** (Shell Utilities) és el nom del mòdul que utilitzarem per portar a terme diferents operacions amb arxius i directoris:
- Exemple 1:

```
import shutil
shutil.copy('hola.bin', 'data')
```

- Exemple 2:

```
shutil.copy('/Users/Abder/Desktop/sample.pdf', '/Users/Abder/Desktop/Temp')
```

- El nom i extensió dels arxius, ha de ser diferent.

COPIAR DIRECTORIS

- copytree: Copiar un directori complert en lloc de només un arxiu. Si tenim:

```
Original
  |- Original-1
    |- Original-2
      |- Original-3
        |- sample.pdf
```

```
import shutil
shutil.copytree('Original', 'Original-Copy')
```

MOVER FITXERS I DIRECTORIS

- Per a moure (tallar) un arxiu a un nou destí:

```
import shutil
shutil.move('Sample.pdf', 'Temp')
```

  - El fitxer ***sample.pdf*** ja no existeix en el directori original, ara està en ***Temp***.

```
import shutil
shutil.move('Sample.pdf', 'New-Sample.pdf')
```

  - En aquest cas, només tenim ***New-Sample.pdf*** amb el mateix contingut que ***sample.pdf***, però ***sample.pdf*** ja no existeix.

```
import shutil
shutil.move('Original', 'Original-Copy')
```

  - En aquest cas, només tenim el directori ***Original-Copy*** amb el mateix contingut que ***Original***, però ***Original*** ja no existeix.

ELIMINAR UN DIRECTORI

- Per eliminar el directori ***Original_Copy***:

```
import shutil
shutil.rmtree('Original-Copy')
```

ESCRIURE DADES EN BINARI: DUMP

- **pickle** utilitza un format binari propi per serialitzar (emmagatzemar la informació d'alguna manera amb la finalitat de trametre-la a través d'una connexió de xarxa) les dades. Per escriure dades en format binari, utilitzarem la funció: **dump**. El mode d'obertura de la funció open ha de ser 'wb'.

```
import pickle

dades = {1:2, 2: [3, 4]}
f = open('exemple.dat', 'wb')
pickle.dump(dades, f)
f.close()
```

- Cada vegada que s'obre en mode ***wb***, es crea de mou, es a dir, la informació es perd.

LLEGIR DADES EN BINARI: LOAD

- Per llegir la informació que ha estat emmagatzemada en format binari, s'utilitza la funció **load**:

```
import pickle


f = open('exemple.dat', 'rb')
dades = pickle.load(f)
f.close()
print(dades)
```

FUNCIÓ *lambda*

- Funció ***lambda***: Ens permet crear funcions anònimes. ***lambda*** és una expressió i per això pot aparèixer en llocs a on no es permet que s'escrigui una ***def***. Com a expressió, lambda retorna un valor que opcionalment se li pot assignar un nom. El cos de la lambda és similar al que havia posat en la instrucció de retorn d'un cos def.
- Exemple:

```
valor = lambda parametro1,parametro2: parametro1+parametro2

print(valor(2,3))
```

- Això equival a

```
def funcio(parametro1,parametro2):
    return parametro1+parametro2

print(funcio(2,3))
```

ORDENAR UNA LLISTA DE LLISTES: sort, sorted

- **Exemple 1:**

```
lista=[['Coritel','Carrer Balmes, 25',600],['Accenture','Carrer Guitar,30',890],['everis','Avinguda diagonal, 65',989]]
lista.sort(key=lambda lista1: lista1[1])

print(lista)
```

- **Exemple 2:**

```
lista=[['Coritel','Carrer Balmes, 25',600],['Accenture','Carrer Guitar,30',890],['everis','Avinguda diagonal, 65',989]]
lista1=sorted(lista, key=lambda element: element[0])
print(lista1)
```