Serialització(Deflating)

La serialització és el procés de desar un objecte a un medi d'emmagatzematge (com pot ser un fitxer, o un buffer de memòria), Unes de les seves finalitats és per transmetre'l a través d'una connexió en xarxa com una sèrie de bytes o en un format humanament més llegible com XML o JSON. També es fan servir per persistir objectes dintre d’un fitxer o una Base de Dades(p.e. BD no relacionals)

Aquest procés de serialitzar un objecte també s'anomena desinflar (deflating en anglès) un objecte o ordenar (marshalling en anglès) un objecte.

L'operació oposada, que extreu una estructura de dades d'una sèrie de bytes, és la deserialització (que s'anomena també inflating o unmarshalling).

# Fitxers Binaris(pickle)

Com passar contingut de variables a fitxers i viceversa.

Pickling

**import** **pickle**

class alumne:

def \_\_init\_\_(self,nom,cognom):

self.Nom = nom

self.Cognom = cognom

dades = {'a': [1, 2.0, 3, 4+6j],

'b': ('string', u'Unicode string'),

'c': **None**}

a1 = alumne("Joan","Garcia")

output = open('data.pkl', 'wb')

*# Pickle dictionary using protocol 0.*

pickle.dump(data1, output)

*# Pickle the list using the highest protocol available.*

pickle.dump(a1, output, -1)

output.close()

Unpickling

import pickle

class alumne:

def \_\_init\_\_(self,nom,cognom):

self.Nom = nom

self.Cognom = cognom

pkl\_file = open('data.pkl', 'rb')

data1 = pickle.load(pkl\_file)

print(data1)

data2 = pickle.load(pkl\_file)

print(data2.Nom)

pkl\_file.close()

# Json(dump)

<https://realpython.com/lessons/serializing-json-data/>

Amb el dump pots convertir molts tipus de dades de Python a json i viceversa

| **Python** | **JSON** |
| --- | --- |
| dict | object |
| list, tuple | array |
| str | string |
| int, long, float | number |
| True | true |
| False | false |
| None | null |

Exemple

<https://stackabuse.com/reading-and-writing-json-to-a-file-in-python/>

Exemple amb un objecte. Hem de passar-lo abans a dict per exemple

import json

fitxer = open("pepe.json","w")

class alumne:

def \_\_init\_\_(self,nom,cognom):

self.nom = nom

self.cognom =cognom

def convert\_to\_dict(obj):

obj\_dict = {

"\_\_class\_\_": obj.\_\_class\_\_.\_\_name\_\_,

"\_\_module\_\_": obj.\_\_module\_\_

}

obj\_dict.update(obj.\_\_dict\_\_)

return obj\_dict

def dict\_to\_obj(our\_dict):

if "\_\_class\_\_" in our\_dict:

class\_name = our\_dict.pop("\_\_class\_\_")

module\_name = our\_dict.pop("\_\_module\_\_")

module = \_\_import\_\_(module\_name)

class\_ = getattr(module,class\_name)

obj = class\_(\*\*our\_dict)

else:

obj = our\_dict

return obj

pepe = alumne("Pepe","González")

pepeDict = convert\_to\_dict(pepe)

json.dump(pepeDict, fitxer,sort\_keys=True, indent=4)

fitxer.close()

fitxer = open("pepe.json","r")

nouPepeJSON = json.load(fitxer)

fitxer.close()

nouPepe = dict\_to\_obj(nouPepeJSON)

print(nouPepe.nom)

# Pràctica. 25%

## Part 1. 8 punts

Es tracta de fer un programa, el que vulguis, pot ser qualsevol dels fets a classe, o un de nou, i que et doni la opció de desar l’estat del programa al disc, et demanarà si vols desar-lo en format binari o en format json.

Quan comenci el programa et donarà la opció de carregar l’estat d’un estat desat anteriorment a partir d’un binari o un json

Si no vols fer-lo tot en el mateix programa, fes la versió binaria i la versió json per separats

Pot ser tant fàcil com desar les vides del joc(5 punts), o complicar-lo una mica més i desar les dades de l’exercici de l’Institut(8 punts). Es valorarà la complexitat del que deseu.

## Part 2. 2 punts

Busca com hauries de fer-lo si volguessis desar-lo en xml. Molt important, **res de Parser(massa complicat )**, heu de buscar alguna llibreria per poder **serialitzar un objecte i/o diccionaris a XML i viceversa**.