Tema 1 – Serialització d'objectes
Programació orientada a objecte

Serialització d'objectes

Suposem que tenim la classe `Poligon` que hem utilitzat en exemples anteriors i volem afegir mètodes per guardar tota la informació del polígon en un fitxer i per poder tornar a inicialitzar un polígon a partir de la informació guardada al fitxer.

```
class Poligon:
   maxim = 1000
   def __init__(self):
     self._vertexs = []
     self._topLeft = punt.Point(Poligon.maxim, Poligon.maxim)
     self._bottomRight = punt.Point()

   @property
   def topLeft(self):
     return self._topLeft

   @property
   def bottomRight(self):
     return self._bottomRight

   def afegeixVertex(self, pt):
     ...

   def calculaPerimetre(self):
     ...
```

Serialització d'objectes

Suposem que tenim la classe Poligon que hem utilitzat en exemples anteriors i volem afegir mètodes per guardar tota la informació del polígon en un fitxer i per poder tornar a inicialitzar un polígon a partir de la informació guardada al fitxer.

```
class Poligon:
   maxim = 1000
   def __init__(self):
     self._vertexs = []
     self._topLeft = punt.Point(Poligon.maxim, Poligon.maxim)
     self._bottomRight = punt.Point()

   ...

   def escriu(self, nomFitxer):
      with open(nomFitxer, 'wt') as fitxer:
         for v in self._vertexs:
            fitxer.write(str(v.x)+ ' ' + str(v.y) + '\n')
            fitxer.write(str(self.topLeft.x)+ ' ' + str(self.topLeft.y) + '\n')
            fitxer.write(str(self.bottomRight.x)+ ' ' +str(self.bottomRight.y) + '\n')

   def llegeix(self, nomFitxer):
      with open(nomFitxer, 'rt') as fitxer:
         linies = fitxer.readlines()
         for l in linies[:len(linies) - 2]:
            valors = l.split()
            self.afegeixVertex(Point(int(valors[0]), int(valors[1])))
```

Serialització d'objectes

Mòdul `pickle`:

- Permet "*serialitzar*" objectes: convertir un objecte en una seqüència de bytes que es pot guardar en un fitxer i que conté tota la informació de l'objecte (el valor de tots els seus atributs incloent els objectes que guardi com a atributs)
- `pickle.dump(objecte, fitxer)`

  Guarda tota la informació de l'objecte al fitxer
- `objecte = pickle.load(fitxer)`

  Retorna un objecte que es correspon a una còpia de l'objecte que s'ha guardat prèviament al fitxer amb la instrucció dump.

```
p = Poligon()
p.afegeixVertex(Point(0,0))
p.afegeixVertex(Point(0,1))
p.afegeixVertex(Point(1,1))
p.afegeixVertex(Point(1,0))
with open("poligon.dat", 'wb') as fitxer:
    pickle.dump(p, fitxer)
with open("poligon.dat", 'rb') as fitxer:
    pCopia = pickle.load(fitxer)
print (pCopia.calculaPerimetre())
```

El fitxer ha d'estar prèviament obert en mode binari: `'rb'` o `'wb'`

Serialització d'objectes

```
class Poligon:
   maxim = 1000
   def __init__(self):
     self._vertexs = []
     self._topLeft = punt.Point(Poligon.maxim, Poligon.maxim)
     self._bottomRight = punt.Point()
   ...
   def escriu(self, nomFitxer):
      with open(nomFitxer, 'wb') as fitxer:
         pickle.dump(self, fitxer)

   def llegeix(self, nomFitxer):
      with open(nomFitxer, 'rb') as fitxer:
         self = pickle.load(fitxer)
      return self
```

```
p = Poligon()
p.afegeixVertex(Point(0,0))
p.afegeixVertex(Point(0,1))
p.afegeixVertex(Point(1,1))
p.afegeixVertex(Point(1,0))
p.escriu("poligon.dat")
pCopia = Poligon()
pCopia = pCopia.llegeix("poligon.dat")
print (pCopia.calculaPerimetre())
```