Serialización

Antonio Espín Herranz

Serialización de Objetos

- El proceso de serialización consiste el convertir el objeto en un flujo de bytes.
- Un objeto serializado se puede grabar en un fichero para luego restaurar el estado.
- O se puede utilizar para transmitir un objeto por la red.

Serialización de Objetos

- El módulo **marshal:**
  - Es el más **básico** de los tres
  - **No serializa objetos** si no, trabaja con bytecode de python (archivos **.pyc**).
  - No proporciona mecanismos de seguridad en cuanto a datos corruptos.
  - Puede cambiar entre versiones de python.

Serialización de Objetos

- El módulo **pickle:**
  - Permite serializar cualquier tipo de objeto.
  - Cuenta con algunos mecanismos de seguridad básicos.
  - Más complejo que marshal, y escrito en python es lento.
  - Existe una implementación en C que es **cpickle**.
  - **OJO**, necesitamos abrir un fichero para serializar y tiene que ser binario, al abrir “wb” y al leer “rb”

Serialización de Objetos

- **cpickle** si lo intentamos importar y se produce algún error se lanzará la excepción **ImportError**.
- **Ejemplo**:

```
try:
    import cPickle as pickle
except ImportError:
    import pickle
```

Serialización de Objetos

- **as** en un **import** sirve para **importar** el elemento seleccionado **utilizando otro nombre** indicado, en lugar de su nombre.

- **serializar** un objeto usando pickle es mediante una llamada a la función **dump** pasando como argumento el **objeto a serializar** y un **objeto archivo en el que guardarlo.**

Ejemplo

```
try:
    import cPickle as pickle
except ImportError:
    import pickle

fichero = open(“datos.dat”, “wb”)
animales = [“piton”, “mono”, “camello”]
pickle.dump(animales, fichero)
fichero.close()
```

Serialización Objetos

- La función **dump** también tiene un **parámetro opcional protocol** que indica el protocolo a utilizar al guardar.
  - **Posibles valores: 0,1,2**:
    - Por defecto su valor es 0, que utiliza formato texto y es el menos eficiente.
    - El protocolo 1 es más eficiente que el 0, pero menos que el 2.
    - Tanto el protocolo 1 como el 2 utilizan un formato binario para guardar los datos.

Ejemplo

```
try:
    import cPickle as pickle
except ImportError:
    import pickle

fichero = file(“datos.dat”, “wb”)
animales = [“piton”, “mono”, “camello”]
pickle.dump(animales, fichero, 2)
fichero.close()
```

Ejemplo

- Para volver a cargar un objeto serializado se utiliza la función load, a la que se le pasa el archivo en el que se guardó.

```
try:
    import cPickle as pickle
except ImportError:
    import pickle

fichero = open(“datos.dat”, “wb”)
animales = [“piton”, “mono”, “camello”]
pickle.dump(animales, fichero)
fichero.close()

# Cargar el objeto serializado en el fichero:
fichero = open(“datos.dat”,”rb”)
animales2 = pickle.load(fichero)
print (animales2)
```

Ejemplo

- Podemos serializar mas de un objeto en el mismo fichero:

```
fichero = open(“datos.dat”, “wb”)
animales = [“piton”, “mono”, “camello”]
lenguajes = [“python”, “mono”, “perl”]

pickle.dump(animales, fichero)
pickle.dump(lenguajes, fichero)

fichero = open(“datos.dat”,”rb”)
animales2 = pickle.load(fichero)
lenguajes2 = pickle.load(fichero)
print (animales2)
print (lenguajes2)
```

Serialización de Objetos

- **El módulo shelve** extiende pickle / cPickle para proporcionar una forma de realizar la **serialización más clara y sencilla**, en la que podemos acceder a la versión serializada de un objeto mediante una cadena asociada, **a través de una estructura parecida a un diccionario**.

- **Métodos**:
  - **open(filename, [protocolo])**
    - Parámetro **filename** mediante el que indicar la ruta a un archivo en el que guardar los objetos.
    - **protocolo**: opcional, indica el protocolo que utilizará por debajo el módulo pickle.
    - **Devuelve un objeto Shelf que podemos utilizarlo como un diccionario.**

Serialización de Objetos

- **Como un diccionario cualquiera la clase Shelf cuenta con métodos get, has_key, items, keys, values**, …

- Una vez hayamos terminado de trabajar con el objeto Shelf, lo cerraremos utilizando el método close.

- import shelve
- animales = [“piton”, “mono”, “camello”]
- lenguajes = [“python”, “mono”, “perl”]
- shelf = shelve.open(“datos.dat”)
- shelf[“primera”] = animales
- shelf[“segunda”] = lenguajes
- print (shelf[“segunda”])
- shelf.close()