# Python - serialização

O termo "serialização de objetos" refere-se ao processo de conversão do estado de um objeto em fluxo de bytes. Uma vez criado, esse fluxo de bytes pode ainda ser armazenado em um arquivo ou transmitido via soquetes, etc. Por outro lado, reconstruir o objeto a partir do fluxo de bytes é chamado de desserialização.

A terminologia do Python para serialização e desserialização é decapagem e remoção de decapagem, respectivamente. O módulo pickle disponível na biblioteca padrão do Python fornece funções para serialização (dump() e dumps()) e desserialização (load() eloads()).

O módulo pickle usa um formato de dados muito específico do Python. Conseqüentemente, programas não escritos em Python podem não ser capazes de desserializar os dados codificados (em conserva) adequadamente. Além disso, não é considerado seguro extrair dados de fontes não autenticadas.

## Protocolos de picles

Protocolos são as convenções usadas na construção e desconstrução de objetos Python de/para dados binários. Atualmente, o módulo pickle define 5 protocolos diferentes, conforme listado abaixo -

| Sr. Não. | Protocolo e Descrição |
|---|---|
| 1 | **Versão do protocolo 0**<br>Protocolo original "legível por humanos" compatível com versões anteriores. |
| 2 | **Versão do protocolo 1**<br>Formato binário antigo também compatível com versões anteriores do Python. |
| 3 | **Versão do protocolo 2**<br>Introduzido no Python 2.3 fornece seleção eficiente de classes de novo estilo. |
| 4 | **Versão do protocolo 3**<br>Adicionado no Python 3.0. recomendado quando a compatibilidade com outras versões do Python 3 é necessária. |
| 5 | **Versão do protocolo 4**<br>Foi adicionado no Python 3.4. Adiciona suporte para objetos muito grandes. |

Para saber a versão mais alta e padrão do protocolo de sua instalação Python, use as seguintes constantes definidas no módulo **pickle** -

```
>>> import pickle
>>> pickle.HIGHEST_PROTOCOL
4
>>> pickle.DEFAULT_PROTOCOL
3
```

As funções dump() e load() do módulo **pickle** realizam decapagem e remoção de dados Python. A função dump() grava o objeto em conserva em um arquivo e a função load() descompacta os dados do arquivo para o objeto Python.

## despejar() e carregar()

O programa a seguir seleciona um objeto de dicionário em um arquivo binário.

```
import pickle
f=open("data.txt","wb")
dct={"name":"Ravi", "age":23, "Gender":"M","marks":75}
pickle.dump(dct,f)
f.close()
```

Quando o código acima for executado, a representação de bytes do objeto de dicionário será armazenada no arquivo data.txt.

Para descompactar ou desserializar dados de um arquivo binário de volta ao dicionário, execute o seguinte programa.

```
import pickle
f=open("data.txt","rb")
d=pickle.load(f)
print (d)
f.close()
```

O console Python mostra o objeto de dicionário lido do arquivo.

```
{'age': 23, 'Gender': 'M', 'name': 'Ravi', 'marks': 75}
```

## dumps() e cargas()

O módulo pickle também consiste na função dumps() que retorna uma representação de string de dados em pickle.

```
>>> from pickle import dump
>>> dct={"name":"Ravi", "age":23, "Gender":"M","marks":75}
>>> dctstring=dumps(dct)
```

```
>>> dctstring
b'\x80\x03}q\x00(X\x04\x00\x00\x00nameq\x01X\x04\x00\x00\x00Raviq\x02X\x03\x00\x00\x0
```

Use a função load() para descompactar a string e obter o objeto do dicionário original.

```
from pickle import load
dct=loads(dctstring)
print (dct)
```

Ele produzirá a seguinte **saída** -

```
{'name': 'Ravi', 'age': 23, 'Gender': 'M', 'marks': 75}
```

## Aula de Pickler

O módulo pickle também define as classes Pickler e Unpickler. A classe Pickler grava dados pickle em um arquivo. A classe Unpickler lê dados binários do arquivo e constrói o objeto Python.

Para escrever dados em conserva do objeto Python -

```
from pickle import pickler
f=open("data.txt","wb")
dct={'name': 'Ravi', 'age': 23, 'Gender': 'M', 'marks': 75}
Pickler(f).dump(dct)
f.close()
```



## Classe Unpickler

Para ler os dados removendo a seleção do arquivo binário -

```
from pickle import Unpickler
f=open("data.txt","rb")
dct=Unpickler(f).load()
print (dct)
f.close()
```

Objetos de todos os tipos de dados padrão do Python podem ser selecionados. Além disso, os objetos da classe personalizada também podem ser conservados e não conservados.

```
from pickle import *
class person:
   def __init__(self):
      self.name="XYZ"
      self.age=22
   def show(self):
      print ("name:", self.name, "age:", self.age)
p1=person()
f=open("data.txt","wb")
dump(p1,f)
f.close()
print ("unpickled")
f=open("data.txt","rb")
p1=load(f)
p1.show()
```

A biblioteca Python também possui um módulo marshal que é usado para serialização interna de objetos Python.