Data Serialization All Versions

Examples

Serialization using JSON

JSON is a cross language, widely used method to serialize data

Supported data types : *int* , *float* , *boolean* , *string* , *list* and *dict* . See -> JSON Wiki for more

Here is an example demonstrating the **basic** usage of **JSON** :-

```
import json

families = (['John'], ['Mark', 'David', {'name': 'Avraham'}])

# Dumping it into string
json_families = json.dumps(families)
# [["John"], ["Mark", "David", {"name": "Avraham"}]]

# Dumping it to file
with open('families.json', 'w') as json_file:
    json.dump(families, json_file)

# Loading it from string
json_families = json.loads(json_families)

# Loading it from file
with open('families.json', 'r') as json_file:
    json_families = json.load(json_file)
```

See JSON-Module for detailed information about JSON.

Serialization using Pickle

Here is an example demonstrating the **basic** usage of **pickle** :-

```
# Importing pickle
try:
    import cPickle as pickle  # Python 2
except ImportError:
    import pickle  # Python 3

# Creating Pythonic object:
class Family(object):
    def __init__(self, names):
        self.sons = names

    def __str__(self):
        return ' '.join(self.sons)

my_family = Family(['John', 'David'])

# Dumping to string
pickle_data = pickle.dumps(my_family, pickle.HIGHEST_PROTOCOL)

# Dumping to file
with open('family.p', 'w') as pickle_file:
    pickle.dump(families, pickle_file, pickle.HIGHEST_PROTOCOL)

# Loading from string
my_family = pickle.loads(pickle_data)

# Loading from file
with open('family.p', 'r') as pickle_file:
    my_family = pickle.load(pickle_file)
```

See Pickle for detailed information about Pickle.

WARNING : The official documentation for pickle makes it clear that there are no security guarantees. Don't load any data you don't trust its origin.

Syntax

```
unpickled_string = pickle.loads(string)
```

```
unpickled_string = pickle.load(file_object)
```

```
pickled_string = pickle.dumps([('', 'cmplx'), {('object',): None}],
pickle.HIGHEST_PROTOCOL)
```

```
pickle.dump(('', 'cmplx'), {('object',): None}], file_object, pickle.HIGHEST_PROTOCOL)
```

```
unjsoned_string = json.loads(string)
```

```
unjsoned_string = json.load(file_object)
```

```
jsoned_string = json.dumps(('a', 'b', 'c', [1, 2, 3]))
```

```
json.dump(('a', 'b', 'c', [1, 2, 3]), file_object)
```

Parameters

Parameter	Details
protocol	Using `pickle` or `cPickle`, it is the method that objects are being Serialized/Unserialized. You probably want to use `pickle.HIGHEST_PROTOCOL` here, which means the newest method.

Remarks

Why using JSON?

- Cross language support
- Human readable
- Unlike pickle, it doesn't have the danger of running arbitrary code

Why not using JSON?

- Doesn't support Pythonic data types
- Keys in dictionaries must not be other than string data types.

Why Pickle?

- Great way for serializing Pythonic (tuples, functions, classes)
- Keys in dictionaries can be of any data type.

Why not Pickle?

- Cross language support is missing
- It is not safe for loading arbitrary data