# Python Serialization

www.ScottMacri.com

So far in my videos I've discussed serializing and de-serializing data using JSON. In many cases this is the preferred method. However, this isn't the only technique. Python has a built in tool named "pickle" for serializing (pickling) objects.

Pickling is quite a bit different from serializing JSON. For one, it is Python specific, and you cannot trust data from unknown sources. This is because malicious data can be created and pickled (serialized).

Its always best to not use non-trusted data. However, there is one technique that can help you determine your data has not been tampered with. You can use another tool called hmac. This can help prevent use of any unexpected malicious code from being used.
Pickle is very powerful compared to JSON and enables you to use the full capabilities of Python. Pickle allows you to serialize Python objects and write them to a byte stream or a file. In the example below we will focus on pickling to a file.

Let's get started with a simple example. I've pulled our Person class from a previous Bits-N-Bytes video.
Create a new file called person.py (our module). Import the pickle library. Now create a person class within that file.

```
import pickle

class Person:
   def __init__(self, f_name, l_name, street, city, state,
             zip, phone):
     self.f_name = f_name
     self.l_name = l_name
     self.street = street
     self.city = city
     self.state = state
     self.zip = zip
     self.phone = phone

   def __str__(self):
     return f"{self.l_name}, {self.f_name}"
```

```
   def greeting(self, message):
     return message
```

Add the main method to your module as shown below:

```
def main():
  person = Person("Joe", "Blow", "124 A st.", "Hickory",
              "Fl", "44444", "549-987-1245")

if __name__ == "__main__":
  main()
```

Now the fun part. Pickle has two methods: load() and dump(). The dump method gives us the ability to serialize (pickle) an object to a file. We must first open a file with both the read and binary attributes set. Remember from the previous video we learned that the default mode is “text” so we must explicitly set the mode to binary.
Binary is different from text when it is written to disk. A binary file is a non-text file and is not human readable. It can only be read by a computer. Previously we learned about JSON files, these are considered text files since they are human readable text.

The first thing we are going to do is create a function to pickle (serialize) an object to disk in a file. Remember this is a function not a method. It will exist in our module and will be outside the person class.

The function is below:

```
def save_person(person, file_path):
  file = open(file_path, "wb")
  pickle.dump(person, file)
  file.close()
```



Lets break this method down:
The first line:

This is our method definition which allows us to execute the code contained within its block. We have two parameters defined, person and file_path. The argument passed for person parameter will be the actual

person object and the argument passed for the file_path parameter will be the string path to the file where we want it saved (ie - “./person.dat”). Notice I used the term parameter and argument with slightly different meanings. That’s because they do actually mean different things. I mentioned this in one of my video’s but never explained what I meant.

Parameter – Variable defined in a function (ie – file_path)
Argument – Data passed when function is called (ie - “./person.dat”) – value of parameter

However, most people use these terms interchangeably, which is totally fine unless you are writing a scholarly article.

The second line:

```
file = open(file_path, "wb")
```

This opens a new file at the location defined in the file_path argument (“./person.dat”). It enables the write (w) and binary (b) functionality. In other words it allows us to write a binary file to disk as “./person.dat”.
The third line:

```
pickle.dump(person, file)
```

This line is what does the work of pickling the file to disk. We pass the actual person object as the first argument and the file object we just created as the second object. This will all make complete sense in a minute. Keep reading.
The final line:

```
file.close()
```

This closes our file. Remember we always close files after they are done being used because we don’t want to create a memory leak and have bad things occur.
Now we create a second function as shown below (outside the person class, in the module):

```
def read_person(file_path):
  file = open(file_path, 'rb')
  person = pickle.load(file)
  file.close()
  return person
```

This function doesn’t need much explanation because its nearly identical to the one we just explained. The main difference is line 3. Also, the file path is the path to the existing file we just pickled in the steps above.

Line 3 reads the pickled object from the file and stores it in a variable named person. This person object is now usable as we have seen in previous exercises. Basically we can now call its methods and access its attributes.

The final step is to add calls to our functions to our main method so we can execute them. If you have difficulty understanding the code below please watch the modules & classes video.

```
# this creates our person object
person = Person("Joe", "Blow", "124 A st.", "Hickory", "Fl",
                "44444", "549-987-1245")
save_person(person, "./person.dat") # pickles person
saved_person = read_person("./person.dat") # reads person
print("Saved Person: ", saved_person) # calls __str__()
method
```

The complete code is below:

```
import pickle

class Person:

   def __init__(self, f_name, l_name, street, city,
            state, zip, phone):

     self.f_name = f_name
     self.l_name = l_name
     self.street = street
     self.city = city
     self.state = state
     self.zip = zip
     self.phone = phone

   def __str__(self):
     return f"{self.l_name}, {self.f_name}"

   def greeting(self, message):
     return message

def save_person(person, file_path):
  file = open(file_path, "wb")
  pickle.dump(person, file)
  file.close()

def read_person(file_name):
  file = open(file_name, 'rb')
  person = pickle.load(file)
```

```
  file.close()
  return person

def main():
  person = Person("Joe", "Blow", "124 A st.", "Hickory",
              "Fl", "44444", "549-987-1245")
  save_person(person, "./person.dat") # pickles person
  saved_person = read_person("./person.dat")
  print("Saved Person: ", saved_person)

if name == "__main__":
  main()
```