**PICKLING**

You just ran through a time-consuming process to load a bunch of data into a python object. Maybe you scraped data from thousands of websites. Maybe you computed a zillion digits of pi. If your laptop battery dies or if python crashes, your information will be lost.

Pickling allows you to save a python object as a binary file on your hard drive. After you pickle your object, you can kill your python session, reboot your computer if you want, and later load your object into python again.

You could back up your pickle file to Google Drive or DropBox or a plain old USB stick if you wanted. You could email it to a friend.

A word of warning: don’t load pickles that you don’t trust. Malicious people can make malicious pickles that may execute unexpected code on your computer (SQL injection, password brute forcing, etc). Stay away from bad pickles.

**import** pickle

*# make an example object to pickle*

some\_obj **=** {'x':[4,2,1.5,1], 'y':[32,[101],17], 'foo':True, 'spam':False}

To save a pickle, use pickle.dump.

A convention is to name pickle files \*.pickle, but you can name it whatever you want.

Make sure to open the file in 'wb' mode (write binary). This is more cross-platform friendly than 'w' mode (write text) which might not work on Windows, etc.

**with** open('mypickle.pickle', 'wb') **as** f:

pickle**.**dump(some\_obj, f)

*# note that this will overwrite any existing file*

*# in the current working directory called 'mypickle.pickle'*

For the purposes of demonstration, I’ll delete the original object from memory to show you that it’s really gone.

**del** some\_obj

**print** some\_obj

**---------------------------------------------------------------------------**

NameError Traceback (most recent call last)

**<**ipython**-**input**-**3**-**3080f97d7e85**>** **in** **<**module**>**()

1 **del** some\_obj

2

**---->** 3 **print** some\_obj

NameError: name 'some\_obj' **is** **not** defined

Loading the pickled file from your hard drive is as simple as pickle.load and specifying the file path:

**with** open('mypickle.pickle') **as** f:

loaded\_obj **=** pickle**.**load(f)

**print** 'loaded\_obj is', loaded\_obj

loaded\_obj **is** {'y': [32, [101], 17], 'x': [4, 2, 1.5, 1], 'foo': True, 'spam': False}

**Pickling pandas DataFrames**

Pandas has a very easy to use pickling functions. First we’ll make an example dataframe:

**import** pandas **as** pd

df **=** pd**.**DataFrame([range(11), range(100,110)], columns**=**list('abcdefghijk'))

df

|  | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **1** | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | NaN |

**pandas.DataFrame.to\_pickle**

Save the dataframe to a pickle file called my\_df.pickle in the current working directory.

Then for the purposes of demonstration again, I’ll delete the original DataFrame

df**.**to\_pickle('my\_df.pickle')

**del** df

**pandas.DataFrame.read\_pickle**

To load the pickled dataframe, simply do:

df2 **=** pd**.**read\_pickle('my\_df.pickle')

df2

|  | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **1** | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | NaN |

And that’s all you need to know for simple pickling in python.

It’s an easy way to back up important objects, pass objects between scripts, or even email a python object to another pythonista (but they shouldn’t open it unless they’re sure you haven’t given them a malicious pickle…)