Python basics VI: File I/O and modules

Modules

Modules are 'packages' of code that needs to be grouped together for a specific function or to distribute. Modules are can be loaded with the `import` python keyword

```
# import the built-in math module
# math has a lot of common mathematical functions, such as
trig functions
>>>import math
>>>math.sin(0) # to access a function or variable in a
module you do module_name.function, here we calculate the
sin of 0 radians
0.0
>>>math.pi # the value of PI
3.141592653589793
```

Writing your own module

Here is an short module for testing purposes `mod.py` which you can download in this directory.

```
# string
test_str = "I am in a module"

# list
test_list = [100, 200, 300]

# function
def add(x, y):
    return x + y
```

if you download it and use the python interpreter in the same directory you can load it

Writing your own module

```
>>>import mod
>>>mod.test_str
'I am in a module'

>>>mod.add(10, 20)
30
```

Finding modules

If you are not in the correct directory and you try and load mod

```
>>> import mod
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named mod
```

The python interpreter cannot find it, python has a set of directories that it looks for modules to load.

Finding modules

You can see this list using the sys module

```
>>> import sys
>>>sys.path
['', '/Users/jyesselm/Downloads', '/Users/jyesselm/projects/RNAMake',
'/Users/jyesselm/projects/RNAMake.projects/SimulateTectos',
'/Users/jyesselm/projects/RNAMake.New’]
```

Finding modules

There are a few ways you can tell python where other python modules are. First add it directly to the sys.path list

```
>>> import sys
>>>sys.path.append('/Users/jyesselm/projects/Chem991E/04_introduction_to_python_part_iv')
>>>import mod # now I can load the module since python knows where to look
```

Installing new modules

One of the other things that have lead to Python's popularity is relative ease to install new modules. There are multiple package managers for python we are going to cover pip but be aware there are others.

```
# this will install the numpy package we will be using later, notice we
# need to put 'sudo' up front which will prompt you for your system
# password. This is because we are changing your system by adding this
new package.
$ sudo pip install numpy
```

File I/O

Opening files, writing and reading files.

In short, the built-in `open` function creates a Python file handler, which serves as a link to a file residing on your machine. After calling `open`, you can transfer strings of data to and from the associated external file by calling the returned file hander's functions.

Reading text files

We are going to use a dummy file given in this directory `test.txt`

```
# show the file contents of test.txt
$ cat test.txt
I am text in a file
I am on the second line

# in python 2.7
for line in open("test.txt"): # open file test.txt for reading
    print line, # print out one line at a time

# in python 3.x
for line in open("test.txt"):
    print(line, end='')

#output
I am text in a file
I am on the second line
```

Reading text files

We are going to use a dummy file given in this directory `test.txt`

```
f = open("test.txt") # open file test.txt for reading
lines = f.readlines() # read all the lines in the file
f.close()

print(lines) # lines is list of strings where each element is a line in
the file

#output
['I am text in a file\n', 'I am on the second line\n']
```

Writing text files

```
f = open("test_output.txt", "w") # need to add the "w" argument to
write
f.write("I am writing text to a file\n") # "\n" is a newline character
f.close()

# check output of file in terminal
$ cat test_output.txt
I am writing text to a file
```

Examples: writing/reading lists

```
numbers = range(10)

# write each number to the file seperated by a " "
f = open("numbers.txt", "w")
for n in numbers:
    f.write(str(n) + " ")
f.close()

# numbers.txt looks like this
0 1 2 3 4 5 6 7 8 9

numbers_2 = []

# read the first line of number.txt
f = open("numbers.txt")
number_string = f.readline()
f.close()

# split numbers by " " and put them in numbers_2
for n_str in number_string.split():
    numbers_2.append(int(n_str))

print(numbers_2)

#output
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Writing lists and dictionaries directly to files with pickle

Now as you can see from above its a real pain to read/write complex pieces of data. Thankfully python supplies a simpler way of reading and writing a list or dictionary directly to a file.

Writing lists and dictionaries directly to files with pickle

```
import pickle # allows direct reading and writing of lists and dictionaries to
file

dict = { 'a' : 0, 'b' : 1, 'c' : 2}
f = open("test.p", "wb") # notice its "wb" not just "w", this is because we are
writing in binary not plain text
pickle.dump(dict, f)

# test.p looks like this, its in binary so not readable to humans
<80>^C}q^@(X^A^@^@^@cq^AK^BX^A^@^@^@aq^BK^@X^A^@^@^@bq^CK^Au.

f = open("test.p", "rb")
dict_2 = pickle.load(f)

print(dict_2)

#output
{'c': 2, 'a': 0, 'b': 1}
```