Python for Scientific Data Analysis

Basic Python

Section 2: Prompting, Type Conversions, Argument Passing, and Reading/Writing

Prompting

Sometimes you want to Python to ask something and proceed with a piece of code based on your answer. You are ***IMPORTANT***, after all. That is, you want Python to *prompt* you for something. So how do you do that? Easy:

```
answer= input([the prompt ... usually a string enclosed by " " ])

#an example
unlikely_answer = input("Bro, do you even lift?\n")
```

Try it from interactive mode

```
unlikely_answer=input("Bro, do you even lift?\n")
```

```
Bro, do you even lift?
yes
```

You can also ask for input and do stuff with it

```
very_unlikely_answer=input("Dude, how much can you bench press?\n")
```

```
Dude, how much can you bench press?
330.5
```

```
print("you say you can bench press %s lbs?   Ya right!" % very_unlikely_a
nswer)
```

```
you say you can bench press 330.5 lbs?   Ya right!
```

Notice, in the above example, the answer (330.5 lbs) is saved as a string, not a number. Suppose you want to use the answer to do some kind of operation (e.g. do some math). Then you have to do a *type conversion* . Type conversions mean you convert from one type (say a string) to another type (e.g. an integer, floating point, etc). To figure out the type of a variable enter `type([name of your variable])` .

```
#e.g.
type(very_unlikely_answer)
```

```
str
```

Type Conversions

Doing type conversions with Python are relatively straightforward compared to, say, C. To convert from a string to a float you type `float([string name]` ; to convert from a float to a string, you type `str([float name]` . A float to an integer? `int([float name])` , and so on.

For example, take a look at ex12_a.py and execute it:

```
number='8675309'
print(type(number), number)
#answer will be <class 'str'> 8675309

number=int(number)
print(type(number),number)
#answer will be <class 'int'> 8675309

number=float(number)
print(type(number),number)
#answer will be <class 'float'> 8675309.0
```

Now, we can combine both prompting and type conversion together in ex12_b.py:

```
unlikely_answer=input("Bro, do you even lift?\n")

very_unlikely_answer=float(input("Um, okay then: how much can you bench p
ress?\n"))

print("you say you can bench press %d lbs?   Ya right!" % very_unlikely_a
nswer)

sanity_check=input("Okay, how much do you weigh?")
total_weight=float(sanity_check)

#total_weight=151 #aspirational

print("I don't believe you...")
print("Because that means you can lift %f times your total body weight" %
 (very_unlikely_answer/total_weight))
```

Here's what this looks like when printed out:

```
Bro, do you even lift?
yes
Dude, how much can you bench press?
331.5
you say you can bench press 331 lbs?   Ya right!
Because that means you can lift 2.195364 times your total body weight
```

Argument Passing (at command line)

Okay, so we covered how to a Python program can ask for your input through the "input" function. Now, we can also tell Python to use different variables when we run the program.

Here's an example, ex13.py:

```
from sys import argv

script, first, second = argv

print "The script is called:", script
print "Your first variable is:", first
print "Your second variable is:", second
```

Here, *script* is going to be your program name; the other things are the names of the different variables. Here's an example of execution `python ex13.py do re me`, which will look like:

```
the script is called: ex13.py
your first variable is:  do
your second variable is: re
your third variable is: mi
```

Reading/Writing

You know how to get input from a user with raw_input or argv. Now you will learn about reading from a file. This exercise involves writing two files. One is the usual ex15.py file that you will run, but the other is named ex15_sample.txt. This second file isn't a script but a plain text file we'll be reading in our script.

Here are the contents of that file:

```
This is stuff I typed into a file.
It is really cool stuff.
Lots and lots of fun to have in here.
```

The simplest way to open a file with Python is the *open* command, like this: `open('ex15_sample.txt')`. Great, we have opened a file, to *read* the file, you need to do something more. Thankfully, things that are opened and saved as variables in Python have an attribute called *read* which allows you to read the file:

```
txt=open('ex15_sample.txt')
a=txt.read()
print(a) #this will print out the text of the file
```

Run the file ex15.py with argument ex15_sample.txt: `python ./code/ex15.py ./code/ex15_sample.txt`, it should print out the following:

```
Here is your file 'ex15_sample.txt'

This is stuff I typed into a file.
It is really cool stuff.
Lots and lots of fun to have in here.

Type the filename again
>./code/ex15_sample.txt
This is stuff I typed into a file.
It is really cool stuff.
Lots and lots of fun to have in here.
```

Reading and Writing: Doing Stuff

Now we are going to take these files, read them, and then write new output. Just like an opened text file has an attribute called .read, it also has one called .write. Additionally, there's an attribute called .truncate which -- as you guess -- truncates the file. Now, you can set truncate to be some number of bytes. Or you call it without argument, in which case the entire file is truncated.

ex16.py is an example:

```
from sys import argv

#a new script

script,filename,filename_new=argv

print("We are going to erase file %r" % filename)
input("?")

print("Opening the file ...")
target=open(filename,'w')

print("Truncating the file")
target.truncate()
outfile=target
target.close()

outfile=open(filename_new,'w')

print("Now will ask for three lines")

line1 = input("Give me Line 1 - ")
line2 = input("Line 2 - ")
line3 = input("Line 3 - ")

print("write these to file")

outfile.write(line1)
outfile.write("\n")
outfile.write(line2)
outfile.write("\n")
outfile.write(line3)
outfile.write("\n")
outfile.write(line4)

outfile.close()
```

The program accepts two arguments -- an input file name and an output file name. It takes the input file name, truncates the file (of ...everything), and then populates the file with new text chosen at command line.

Here's an example running the program with `python ex16.py ex15_sample.txt ex15_sample3.txt`:

```
We are going to erase file 'infile'
?
Opening the file ...
Truncating the file
Now will ask for four lines
Give me Line 1 - We don't need no education
Line 2 - We don't need no thought control
Line 3 - No dark sarcasm in the classroom
Line 4 - Hey, teacher leave them kids alone
write these to file
```

And, as expected, the output file (outfile) prints out the three lines we give it (the first three lines to a song).

More Reading and Writing: the Exists function

Now are going to do the same thing, except utilize a new function `exists` . Here's the code for `ex17.py` :

```
from sys import argv

from os.path import exists

script,from_file,to_file = argv

print("Copying %s to %s" % (from_file,to_file))

in_file=open(from_file)
indata=in_file.read()
#indata=(open(from_file)).read()

print("The input file is %d bytes long" % len(indata))
print("Does the output file exist? %r" % exists(to_file))
print("Ready, hit return (Cntr-C to abort).")
input()

out_file=open(to_file,'w')
out_file.write(indata)

print("Alright, done.")

in_file.close()
out_file.close()
```

The program asks about the length of the file (er, the string contained in the file) and whether it exists. If it doesn't, the program throws an error. If it does, then the program proceeds to write out the string to a new file.

To run, let's consider a file -- test.txt -- with the following string: `"It's the end of the world as we know it. And I feel fine."` And ask the program to save to a new file called duh.txt

To run, type: `python ex17.py test.txt duh.txt` in the terminal window. The following is what you should see:

```
Copying test.txt to duh.txt
The input file is 59 bytes long
Does the output file exist? True
Ready, hit return (Cntr-C to abort).

Alright, done.
```

What about if you try to open a file that doesn't exist? Well, here: `python ex17.py test2.txt duh2.txt` results in:

```
Copying test2.txt to duh2.txt
Traceback (most recent call last):
  File "ex17.py", line 9, in <module>
    in_file=open(from_file)
FileNotFoundError: [Errno 2] No such file or directory: 'test2.txt'
```