22c:005 Spring 2014

Homework 1

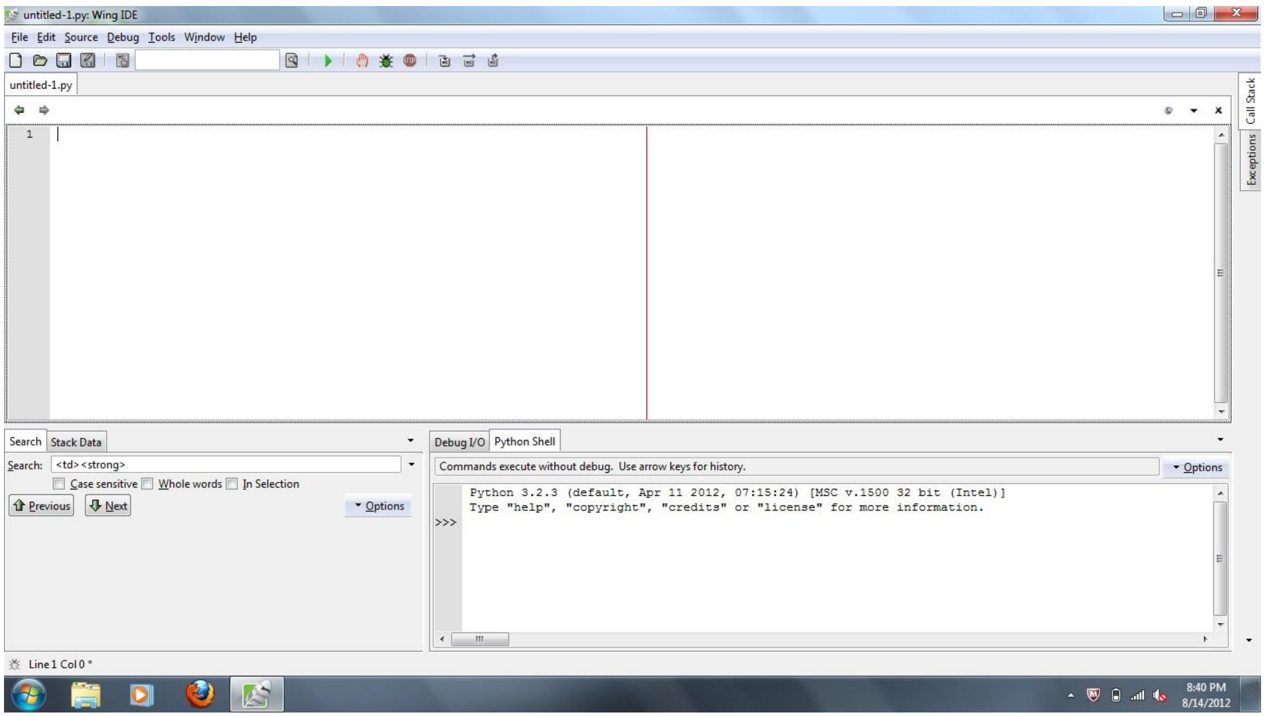
Due Friday, January 31, 2014 by 11:59 PM
(partial credit available until 11:59 PM February 1, 2014)

1. Start the Python IDE

It is usually convenient to write programs in whatever language you are using (Python in this case) using an *integrated development environment*, or an IDE. The IDE provides many nifty features to make developing your program easier. The IDE used in this class is the *Wing IDE 101*. Other IDEs are equally good, but this one is free…

(1) Click on the Windows icon in the lower left corner of the screen.
(2) Start typing "Wing" and it should show up as available. Click on it…do it…
(3) If this is your first time using *Wing IDE 101 4.1.6*, you will have to accept the *End User License Agreement* (EULA) defining the terms of use for the software.

Once it starts, the IDE should resemble the image below. The main window is where you will be writing in Python. The small(er) window on the bottom right is where output will be printed.



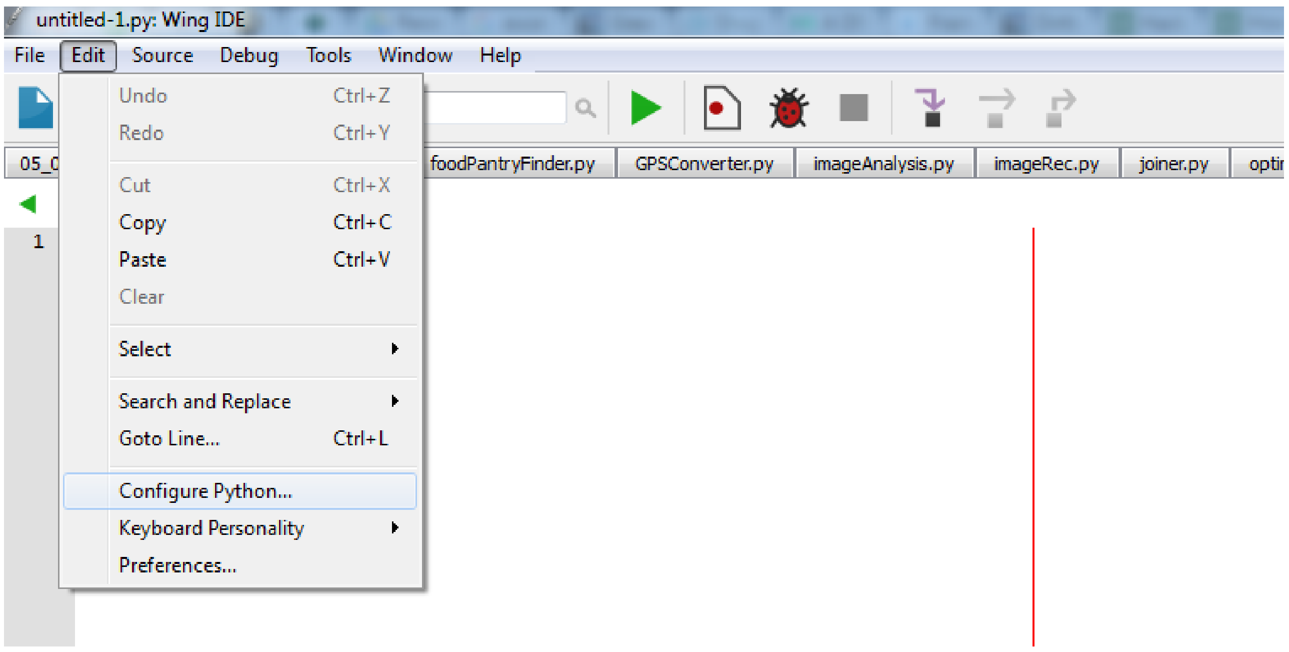
Commands execute without debug. Use arrow keys for history.

```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit (Intel)]
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Before you start running we want to make sure we're using the python version that's compatible with ArcGIS/ArcMap. You'll notice the following in the python window:

```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit (Intel)]
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

To switch Python versions is easy. Go to Edit > Configure Python



Select “Browse” and navigate to: c:\Program Files (x86)\Python\ArcGIS10.3\ and select python.exe. Restart your python shell by selecting “options” in the python Shell and clicking “restart”. You should only have to do this once.

The three characters ">>>" are the *Python prompt*. Think of the shell as an interactive calculator; you can click in it and type Python expressions at the blinking cursor just after the Python prompt: Python will evaluate the expression and give you the result. Try it now: click on the frame containing the prompt and type:

>>> *2 + 5*

(italicized characters are those you actually typed). Press the *Enter* key, and the shell should respond:

7

followed by a new prompt:

>>>

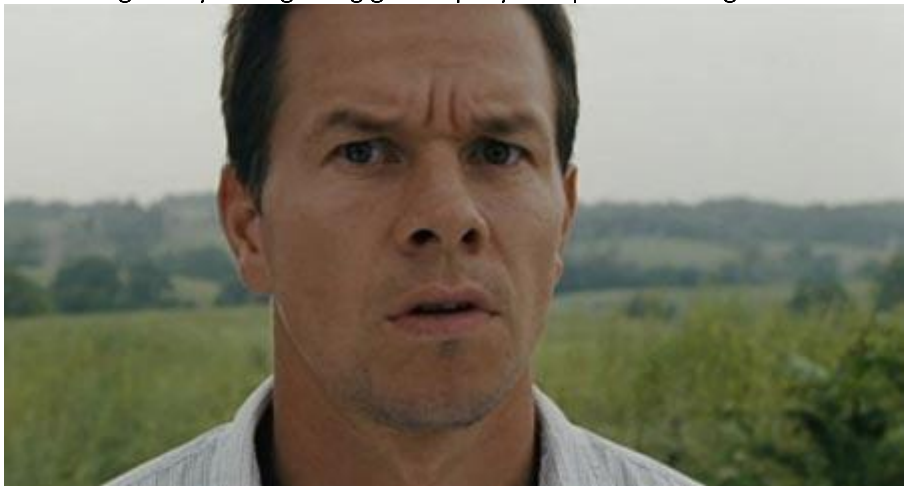
In this window you can quickly test ideas you might be trying to implement.

2. Writing simple programs

The Python shell is easy to use, but not the best for building a set of functions, loops, or conditional statements you want to implement or run repeatedly. A more efficient way to work is try and test ideas and save them as a script. This saved script can be run over and over again by simply reloading the stored file into the Python interpreter.

(1) Create a new Python script by selecting *File* then *New* near the upper left corner of the window, by clicking on the *New* button on the far left side of the toolbar, or pressing “Ctrl” + “n”.
(2) A new editor window should appear, with the name "unititled-1.py" (the ".py" filename extension by convention indicates a Python script).

Next, we will write a function called typer(var) that returns the type of variable entered into the function. For example, typer('this is a string') would return string. Functions take input and execute a series of instructions or processes. You will also want to develop the habit of documenting what your programs/scripts do. When you don't use them for an extended period of time and run into an error, you'll end up looking at your code like Mark Wahlberg in the happening – except you won't be wondering how you're getting grotesquely overpaid for being in a terrible movie….



```
#this function input and return the type of variable the input is

def typer(var):
   return type(var)
```

Now that we have written this function, we can test it. Press the *Run* button near the top of the screen. We can test our function to make sure that it works. In the smaller window in the lower right part of the screen, type the following (take a look at the screen shot to be sure that you understand where to type this text) and hit *Enter*:

```
>>>typer(2.45)
>>> <type 'float'>
```

As you become more comfortable with writing Python you'll develop your own style of writing and documenting code. Create a new window (by pressing the *New* button near the top of the screen), and do the following:

(1) Write a function, called *echo(word, x)*, that takes, as arguments, a word and a value (called x), and returns the word repeated *x times.* For example, echo("hello", 3) should return the string "hellohellohello". Try some attempts at this, then see if you can improve it to add spaces between the words. If you figure that out try to add spaces between the every word, except after the last.
(2) Write a function called *stringer*(word1,word2) "fill in the blanks of the following sentence. "*word1 is the best Nicolas Cage movie, because of Nic Cage's word2".* So, for example, if I put stringer("Con Air", "accent and hair - flawless") the output would be "Con Air is the best Nicolas Cage Movie, because of Nic Cage's accent and hair - flawless"
(3) Write a function called *avgCalc*(list), this function will take a list as input and return the average number of that list. Think about variable types and what you want your average to be. Do you want it to be an integer or a floating point (decimal) number? For example, is the average of a 4 and 5 4.5? or is it 4?
(4) Write a function, called *medCalc*(list), this function takes a list as input and a finds the median.

For a list length that’s odd, this is easy, but what happens if the list contains an even set of numbers? Here’s where you’ll want to use an “if” statement to help you determine how you should treat the list entered into the function. Also, the “sort” function will help you out. List.sort() will sort a list numerically in ascending order. Have your function return the string “the median of the sorted list: *list* with a length of *n* is: *median*”. You wouldn’t normally output this string if you were using the

```
#test the three functions
print(echo(“DanielBurnham”, 5))
print(stringer(“Red Rock West”,”attitude and wearing of denim”))
print(avgCalc([1,1,2,3,4,4,5,6,7,7]))
print(medCalc([7,8,9,6,5,4,2,1,3])
print(medCalc([1,4,7,2,5,8])
```

You will upload these four functions (in their file) to the digital drop box. The first step is to save your functions (i.e. the contents of the large window). To save the contents of the window to a file:

(1) Select *File* then *Save* (or just press the *Save* button near the top of the screen).
(3) The IDE will ask which files should be saved; select the name that corresponds to the name of window you were working in (the name should be on the tab above the window) and choose *Save Selected Files*.
(4) A file chooser window is presented, asking you where you should save the file. You could choose the default location, or you could start the semester with some organization.
(5) Select *Libraries* along the left side, then double click on *Documents*.
(6) Still in the file chooser window, click on the folder icon with the little star pattern accent on the corner to create a new folder within your *Documents* folder: name this folder *22c005*.
(7) Double click on the new folder *22c005* to open it.
(8) Still in the file chooser window, click on the folder icon with the little star pattern accent on the corner to create a new folder within your *22c005* folder: name this folder *HW1*.
(9) Double click on the new folder *HW1* to open it.
(10) Now type the name you want to give to your new file (following a simple naming structure is preferred. The name YOURLASTNAMEHW1.py works well. For example, I would name my homework file SzecseiHW1.py) where it says *File name:* and select save. The file chooser window should disappear. If you are using a mac, be sure to add the .py extension explicitly. Make sure that your file follows the correct naming structure.

By organizing your folders this way, each new assignment lives in its own folder, and your files will be
easy to find within your computer's file system.

3. Test your functions: run your script

Near the top of the IDE window, there is a green button shaped like a right-facing arrow or triangle. Click this button to run or execute your script in the interactive shell. Each time you click on the green *run* button, your program will be reloaded and run anew. Test your functions, to make sure that they work. Your program should not have any errors. This first homework submission should consist of three functions:

```
#test the functions
print(echo("Burnham", 5))
print(getAvg([1,1,2,3,4,4,5,6,7,7]))
print(introduction("Nathan", "Grinnel"))
```

---

4. Python on your own computer

The Python interpreter is free (as in speech, not as in beer) software, and can be downloaded at no cost from the Internet and installed on your own computer. Visit:

http://python.org/download

and choose the version best suited to your machine (if you have a Mac, or a Linux machine, check first to be sure Python is not already installed, as it often is…for a Mac, open a terminal window and type "python"…if python is installed, you will see the familiar >>> prompt). Note that we are using version 3.2 in class (Macs seem to have version 2.7.2…in that case, I'm ok with keeping that version, but you should realize that there are a few differences between the two versions, which I will discuss in class as they come up). You can test your programs on the machines in the lab to ensure that they run on the correct version of Python. Python comes with its own IDE called *IDLE*. It's a bit less fancy than *Wing IDE 101*, but it can do in a pinch. If you would instead like to have the same IDE as in class, visit:

http://wingware.com/downloads/wingide-101

for the most recent version (again, select the version most appropriate for your machine…I installed 4.1.6 on my machine but the most recent version is 5.0.2. Be sure that you are on the wing 101 tab (this is the free version…the other versions cost money once the trial period is over). The machines in the 301 Lab have both the Pro and the free version. We will be using the free version consistently in lecture and in the lab sections.