CS150 - Assignment 1
Due: Wednesday Sept. 18, at the beginning of class

This assignment will walk you through the basics of using WingIDE and you'll write your first functions in Python. For each section

1 Some Housekeeping

If you haven't done so yet, please bookmark the course web page in your browser at:

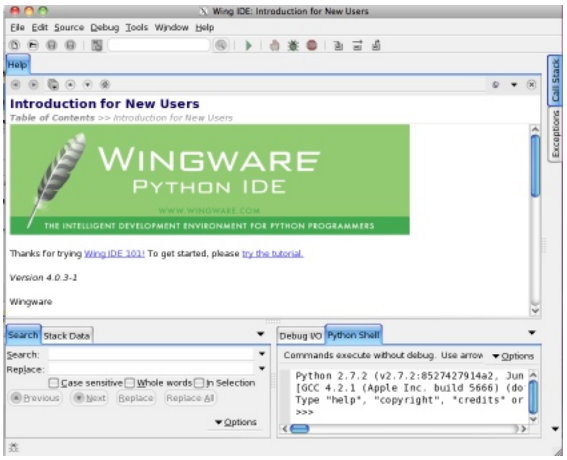
`http://www.cs.middlebury.edu/~dkauchak/classes/cs150/`

2 WingIDE [3 points]

Start up WingIDE. If you're in the lab, you can do this by clicking on the feather icon in your dock at the bottom of the screen:

If you're on your own laptop, do what you have to do to get it started :)

When you start WindIDE is should look something like:



The IDE has three different working sections. We'll mostly use the top one, which is where we will edit files and the bottom right corner, which gives us an interactive shell. You can adjust these internal window sizes to make them bigger or smaller, move where they are, etc.

Try a few commands in the interactive shell.

`>>>`

is the prompt, meaning Python is ready for a command,

`...`

means it's a line continuation (i.e. Python is waiting for you to finish the statement) and a line without anything in front of it is generally the response from the interpreter.

- Try a few simple mathematical equations (e.g. "1+1", "2**3", "(100/20)+45*7"). Notice that Python makes for a pretty easy to use calculator.
- 22/7 is an approximation for Pi. Type this in. Python gives the wrong answer. There are many ways of fixing this statement. Try two different ways.
- Remember that we can use variables to store intermediary values. Assign 22/7 (the correct version) to a variable called `pi`. Use that variable to calculate the area of a circle with radius 15 (remember, the area of a circle is π times the radius squared).
- The Middlebury field house *used to be* a giant bubble[1]. Technically, it's not a round bubble, but we're going to approximate it here as one.

In the shell you can use the up and down arrow keys to revisit commands you typed previously. Use the up arrow key to get your previous statement and then edit it to get the area of the floor of the field house bubble if we assume that the radius is 100 feet.

Calculate how many students could fit standing up in the field house. Discuss with your neighbor what assumptions you made to get this calculation.

[1] They're finally renovating it this year, but it's still a fun problem to think about :)

When you're ready to move on, copy and paste your interactive session into a text editor (e.g. TextWrangler or TextEdit on Mac, which you can execute/find using the *search* functionality like you did for the lab preparation, or Notepad on Windows). Don't copy more than a page. Put your name at the top. Save this page in your assignment 1 folder since you'll be handing it in. If you're on the lab machines you can print it out right now if you'd like.

3 Your first Python program [3 points]

So far, we've only been interacting with the Python shell. This is good for some situations, but eventually, you're going to want to write longer programs that you can edit easily and persist when you quit Wing.

- Create a new file in WingIDE

  Click `File->New` (or the new file button)

- Save the file in the folder you created during the lab prep as `[last_name]-assignment1.py`

  Click `File->Save` (or the save button or type `command + s`)

  This will bring up a prompt for the location of the file.

  If you're in the lab: Your desktop can be found in `/home/yourusername/Desktop/`.

  If you're on your own laptop: Navigate to wherever you created your assignment 1 folder.

  Underneath the buttons in the "save" window, you'll see displayed the current directory. Double-clicking on things in the left side of the window (e.g. "Desktop") will move into that folder. Navigate until you're in your `cs150` folder. We will use the file extension ".py" for all of our Python programs, which is the standard way to indicate that the file contains Python code.

- Put a few statements from above in the file, save it and then run your program

  Each line you enter in the file will be executed in the shell as if you typed it. For example, enter "22/7" in the file. Save the file again (this time you shouldn't have to enter the file location).

  You can run your file by clicking the green arrow  .

  When you do this, you'll notice that the Python shell in the bottom right of Wing restarts and then your program will execute. Most likely, your program won't show anything in the shell window. *Make sure you understand why your program displays what it does!* Discuss the output with your neighbor.

- Put some `print` statements in your program

  When you are running a program (vs. interacting with the shell) the intermediary results are not shown. Instead, if you want things to be displayed you need to use the `print` statement. You can print anything that represents a value, for example:

```
print 10  # printing a number
print 22/7 # printing another number
pi = 22/7  # assign 22/7 to pi
print pi # print out what is stored in pi
print "Hello computer user" # printing a string
```

Add a few print statements to your program and run it again. You should now see some results printed out.

- Define a function called `circle_area` that takes the radius as a parameter and outputs the area of a circle with that radius.

  Remember the basic structure for defining a function is:

  `def` *function_name* (*parameters*) :
  statement1
  statement2
  ...
  `return` *something* `# not all functions will have return statements`

  Remember that the way that Python can tell what is part of the function is based on the indenting.

- Put in a `print` statement that prints out the area of a circle of radius 25 at the end of your program and then run your program. You should NOT have any `print` statements inside your `circle_area` function. Instead, you should call your function and print out the value returned.

  Remember, we can print anything that represents a value:

  ```
  print circle_area(25)
  ```

  This statement has multiple parts. `circle_area(25)` calls our defined `circle_area`, which will in turn, execute the statements inside your defined function. When it returns, that value will then be printed by the `print` statement.

- After running your program, play with the function in the interpreter.

  After you've run your program, you can still interact with the interpreter (bottom right). For example, you can type:

  ```
  >>> circle_area(12.4)
  483.2457142857143
  ```

What's in your file so far:

At this point, you program/file should have some misc. statements at the top, followed by some print statements and then your definition for `circle_area`. Put comments above each of these sections so they are clearly delimited and then move on to the next section.

4 A semester abroad in Europe [11 points]

http://www.bildergeschichten.eu/euro_cartoon_witz.htm

For the last part of this assignment, you will write some functions of your own in the *same* file created for the previous section.

You're going to do a semester abroad in Europe and have decided to write a few functions that will help you out with some common questions you might find yourself asking while you're there.

1. **[2 points]** Write a function called `euros_to_dollars`, with a single parameter, the price in euros, and the function will give you the price in dollars. Lookup the current price exchange from euros to dollars online. For example, after running your program you could type:

   ```
   >>>  euros_to_dollars(13.5)
   18.97695
   ```

   (Note: depending on the exchange rate you use, your value will be slightly different)

   Make sure that you are using the `return` statement and not printing the answer in your function. In particular, try running the following and make sure you get something similar:

   ```
   >>> dollars =  euros_to_dollars(13.5)
   >>> print dollars
   18.97695
   ```

2. [**2 points**] Write a function called `welcome` that doesn't take any parameters and prints out "welcome" in some european language. For example:

```
>>> welcome()
te lutem
```

   Remember that you can create functions with zero parameters. To call a function without any parameters, you still need to put the parenthesis at the end.

3. [**2 points**] Write a function called `kilometers_to_miles`, with a single parameter, the number of kilometers, and the function will give you the distance in miles. For example, after running your program you could type:

```
>>> kilometers_to_miles(100)
62.137
```

4. [**3 points**] Write a function called `mpg_from_metric` that takes *two* parameters: first the number of kilometers and second the number of liters. The function returns the miles per gallon (i.e. miles divided by gallons) by converting the kilometers and liters appropriately. Remember, to have multiple parameters for functions, you separate them with commas.

```
>>> mpg_from_metric(400, 30)
31.358472666666668
```

5. [**2 points**] Write a function of your own that takes one or more parameters and does something interesting/useful. Brownie points for creative functions :)

6. **Extra credit (1 point)**: The right way to calculate the `mpg_from_metric` function is to utilize other functions you write that do some of the work for you. To calculate the mpg, write an additional function `liters_to_gallons` and then use this function and your `kilometers_to_miles` function inside the `mpg_from_metric` function.

7. **Extra credit (1 point)**: Write another function that will be useful on your European vacation. The amount of extra credit given will be determined based on the usefulness of your function as well as the difficulty to implement.

5 When you're done

For this assignment you'll submit your work on paper, however, after this, it will all be digital submissions.

What to hand in:

- A print out your work from section 2

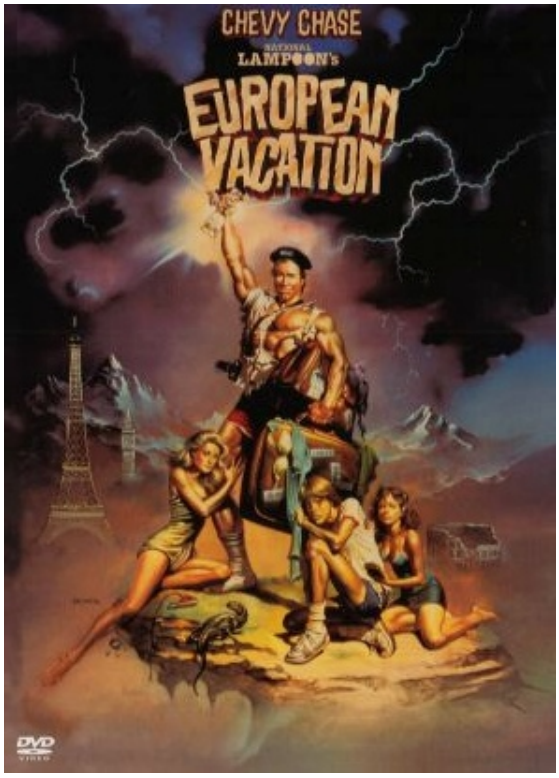
- A print out of your work from section 3 (specifically, look at the end of that section to see of list of things that should be there)
- Your functions from section 4
- Nothing else :)

Your work from sections 3 and 4 should be in a single file. Make sure that this file is properly commented:

- You should have comments at the very beginning of the file stating your name, course (including section number), assignment number and the date.
- Each function should have a short comment above it describing what it does
- Other miscellaneous comments to make things clear

6 Grading

	points
section 2	3
section 3	3
section 4	11
comments and code style	3
extra credit	2
total	20 (+2)


CHEVY CHASE
NATIONAL LAMPOON'S
EUROPEAN VACATION
DVD