Purpose

The main purpose of this lab is to give you some experience writing and testing functions in Python. These functions might not be very exciting, but knowing how to write and test functions is a skill you'll use over and over again during your Python programming career.

Preparation

One of the things that we're going to introduce in this lab is an automatic grading system that we've developed. Download this file:me499.py and put it in the top-level directory for this class (the one in which your lab0 directory lives). If you're in ME 599, I know renaming it as `me599.py` is tempting, but please refrain (otherwise things break :( ). Change the permissions on this file to make it executable. Take a look at the file. It's written in Python!

You're going to use this to do three things. One is to set up new lab assignments. One is to run a subset of our grading scripts over your code, so that you have an idea of the grade that you'll end up with. One is to package up your code for handing it in.

You run the script with two arguments. The first is one of create, grade, or submit. The second is the lab or homework number. For now, run the code like this:

```
./me499.py create lab1
```

This will create an empty directory for lab 1. If this doesn't work, check in with a TA. Once you've written some code for each of the three python files, you can do this:

```
./me499.py grade lab1
```

to see what sort of grade you're likely to earn. You may get an error until you define all of your functions. Again, if this doesn't work, check in with a TA. For these scripts to work, you need to follow the instructions on file and function names exactly. The scripts are expecting things to be in a particular place, and to be named in a particular way. If they're not, then the scripts will fail, and you will lose points.

All of the code you write for this lab should be in the lab1 directory (and not in a subdirectory of it). Make sure you pay attention to file and function names.

Assignment

1. We're going to start by taking the code you wrote for calculating the volume of a cylinder and putting it into a function, so that you can reuse it. Create a file called `volumes.py`, and write a function called `cylinder_volume` that takes a radius and a height (in that order) and returns the volume of the corresponding cylinder. In addition to the function, you should write some code to test that the function is returning the correct values. This can be done by printing the value returned by your function for arguments that you define. This code should go after the `if __name__ == '__main__':` line (which the TAs will explain at the start of the lab).

2. Modify your code so that it deals correctly with arguments that are incorrect. For volume calculations, this basically means negative values. If you get bad function arguments, then the function should return `None`.

3. Do the same for the torus code, putting it in the same file, in a function called torus_volume, which takes the major radius and minor radius (in that order) as arguments. The volume to use for the torus is the simple one (whatever pops up on google first).

4. Write a function called `close` in a file `close.py`, that takes three numbers as arguments. It returns `True` if the absolute difference between the first two numbers is less than the third number. For example `close(1, 2, 0.5)` would return `False`, but `close(1, 2, 3)` would return `True`.

5. Write a function called `letter_count` in a file `words.py` that takes two string arguments. The first string is a bunch of text, and the second is a letter. The function returns the number of occurrences of the letter in the text. Your function should be case insensitive, that is: halLway should return 2 for ('halLway','l') and ('halLway','L').

Grading

You will be graded for showing the TA the following things. Each of the following will get equal weight in your grade:

1. A cylinder volume function that works on valid inputs.
2. A cylinder volume function that deals with invalid inputs appropriately.
3. A torus volume function that works on valid inputs.
4. A torus volume function that deals with invalid inputs appropriately.
5. A working `close` function.
6. A working letter-counting function.

What to Hand In

Make a tarball of your code with ./me499.py submit lab1. This will leave a tarball in the directory that he script is in. Submit this through Canvas.

Thoughts

1. You should the Python value of pi in this lab. Use either `import math` or `from math import pi`.

2. We're going to be sticklers with grading for this lab. If the grading script doesn't work on your code, you'll end up getting zero points for it. This might seem harsh, but we're trying to reinforce the importance of writing to a particular specification. We'd suggest writing the code for the first part of the assignment, verifying that it works with the grading script, and then checking in with a TA in the lab if it doesn't. It's not hard to get it to work, but it does require attention to detail.

3. We haven't really talked about strings or for loops yet. However, you should be able to work out the syntax from the following example function, which prints out the characters in a string, one per line. You'll probably want to use a for loop in your answer to the last question.

   ```
   def foo(t):
       for c in t:
           print c
   ```

4. For all of the code you write for this lab, you should write your own testing code to try to make sure that the functions do the right thing. We're going to talk more about writing testing code during lecture, but you should try to think of some test cases for each of your functions, and write code to make sure that they do the right thing. This is what we're doing with the grading code, but you should get into the habit of writing your own tests (and leaving them in your code, for when you make changes).

Everything you do for this lab should be your own work. Don't look up the answers on the web, or copy them from any other source. You can look up general information about Python on the web, but no copying code you find there. Read the code, close the browser, then write your own code.