Working with Functions in Python

Introduction to Programming - Python
Notes on Exams
Logistics

- **FINAL EXAM SCHEDULE**
  - **Tuesday 5/9 2:00pm-3:50pm**
  - CIWW 517

- Let me know ASAP if you have a conflict with that time.

- If you have not turned in one or more homework assignments, or scored less than an 80 on any of them, I invite you to do or re-do them, and submit them asap.

  - Please let me know about your intention to resubmit.
  - Please meet with me to go over any feedback on your prior submissions, and to resolve any questions you have. If you cannot come to my office hour, make an appointment for another time.
Notes on Exams
Mean = 78.3; Median = 79.0
If you do more than 10 points better on the 2nd midterm, we can discard this grade, and use that one for both exams.

Programming is a skill that takes practice to develop.

Everyone who scored less than 70 should make an appointment to meet with me.

The rest of you are welcome to meet with me as well. If you can't make my office hours, let's make an appointment.
Modules
**Modules**

- All programming languages come pre-packaged with a standard library of functions that are designed to make your job as a programmer easier.

- Some of these functions are built right into the “core” of Python (print, input, range, etc).

- Other more specialized functions are stored in a series of files called “modules” that Python can access upon request by using the “import” statement.
  - import random
  - import time
Modules

- On a Mac you can actually see these files here:
  - /Library/Frameworks/Python.framework/Versions/3.2/lib/python3.2/

- To see information about a module, you can do the following in IDLE:
  - import modulename
  - help(modulename)
The import statement tells Python to load the functions that exist within a specific module into memory and make them available in your code.

We call functions that exist within a module by using “dot notation” to tell Python to run a function that exists in that module.

Example:
- `num = random.randint(1,5)`
You can list the functions that exist in a particular module by using the `help()` function.

The `help()` function takes one argument (a string that represents the name of the module) and returns the user manual for that module.
Creating your own modules

- You can easily create your own modules that you can populate with your own functions. Here’s how:
  - Create a new python script (i.e. “myfunctions.py”)
  - Place your function definitions in this script
  - Create a second python script (i.e. “myprogram.py”)
  - Import your function module using the import statement:

```
import myfunctions
```

- Call your functions using dot notation

```
myfunctions.function1()
myfunctions.dosomethingelse()
```
Programming Challenge

- Create a module called “geometry_helper”

- Write two functions in this module:
  - Area of circle
  - Perimeter of circle

- Each of these functions will accept one argument (a radius) and will print out the result to the user.

Circle

Perimeter: $C = \pi d \quad \text{or} \quad C = 2\pi r$

Area: $A = \pi r^2$

Note: The value of $\pi$ is 3.1415926 (to 7 decimal places)
Some additional functions inside the random module

- Floating point random #’s
  - `num = random.random()`  # generates a float
    # between 0 and 1
  - `num = random.uniform(1,10)`  # generates a float
    # between 1 and 10
Seeding the random number generator

- The computer does not have the ability to generate a truly random number. Instead, it uses a series of complicated mathematical formulas that generate a pseudo-random sequence.

- The formula takes a "seed" value to start. For any given seed, it will always generate the same sequence.

- Usually the system clock (with precision of milliseconds) is used. The seed function in the random module allows you to specify a seed value:

  ```python
  random.seed(x)  # x can be int or str
  ```

- You can seed the random # generator with whatever value you want – and you can even reproduce a random range this way!