**Carleton University**
**Department of Systems and Computer Engineering**
**ECOR 1051 - Fundamentals of Engineering I**

# Lab 4 - Defining and Testing Functions

## Objectives

To gain experience using *Practical Programming*'s function design recipe (FDR).

*Learning outcomes: 4, 5; Graduate attributes: 1.3, 5.3 (see the course outline)*

## Getting Started

Launch Wing 101. Create a new file called lab4.py. All your solutions to all the exercises in this lab will be stored in this one file, one after the other. By the end of the lab, your file should be organized as shown below. Broadly stating, import statements come first, function definitions come second, the main script (made up of call expressions that test the use of those functions) come last.

```
# Any and all import statements
# Function definition for Exercise 1
# Function definition for Exercise 2
…
# Main Script
# Call(s) of Function in Exercise 1
# Call(s) of Function in Exercise 2
…
```

Tip: The math library has some useful constants and functions for the mathematics required in this lab. There are two ways to import these items:

Alternative 1:

```
      import math       # Perhaps you want to use the constant pi
     # Whenever you use pi, you must give a fully qualified name
      area = math.pi * radius * radius
```

Alternative 2:

```
      from math import pi
      # You can now simply use pi
      area  = pi * radius * radius
```

**Exercise 1**

In the previous lab, you wrote the function definition for `area_of_disk`, which returns the area of a circle with a non-negative radius. Copy-paste this code (or perhaps you could write it again as practice) into this lab's file. Also, copy the corresponding call expressions into the main script.

Write a full and complete **docstring and type annotations** for this function, as described in the lectures and the textbook.

**Exercise 2**

Use the Function Design Recipe (FDR) to design, code and test the definition of a new function named `distance`. This function returns the distance between two points, given by the coordinates ($x_1$, $y_1$) and ($x_2$, $y_2$). The function parameters are the *x* and *y* values. (For those students who know Python, do not use lists or tuples to represent the points.)

If you don't remember the formula to use, check this page:

http://mathworld.wolfram.com/PythagoreanTheorem.html

Your function definition must have type annotations and a complete docstring.

**Exercise 3**

Use the FDR to design, code and test the definition of a function named `area_of_circle`. This function takes two points: the center of a circle, ($x_c$, $y_c$), and a point on the perimeter, ($x_p$, $y_p$), and returns the area of the circle. The function parameters are the *x* and *y* values.

Your function definition:

- must have type annotations and a complete docstring.
- **must call the functions you wrote for Exercises 1 and 2**

**Exercise 4**

You've volunteered to repaint the walls in Leo's Lounge and will need to use a ladder when painting close to the ceiling. You know that a ladder has to be placed against a wall at an angle; otherwise, it will fall over. Use the FDR to design, code and test the definition of a function named `height`. This function takes the length of a ladder, measured in metres, and the angle that it forms with the ground as it leans against the wall, measured in degrees. It returns the height reached by the ladder.

Your function definition:

- must have type annotations and a complete docstring. Think carefully about the range of permitted values for the two parameters. Remember to document these in your docstring.

Hint 1:The most challenging part of this exercise is the math, not the Python code. In Step 1 of the FDR, use trigonometry to express the solution's formula using mathematical notation before you prepare the function call examples. In Step 4 of the FDR, translating the formula to Python should be straightforward.

Hint 2: When you're coding the function body, use the shell to investigate which trigonometric functions are provided by Python's `math` module:

```
>>> import math
>>> help(math)
```

**Wrap Up**

Ensure that your code meets the posted marking rubrics for the labs.

Submit file lab4.py.

You are required to keep a backup copy of (all) your work for the duration of the term.

**Extra Practice (How should I study for an exam?)**

During the midterm and final exams, you will be expected to draw diagrams similar to those created by Python Tutor. Use PyTutor to visualize the execution of your solutions.

Last edited: April 23, 2020