Lab 4: Functions

CSE/IT 107

NMT Computer Science

"If you don't think carefully, you might believe that programming is just typing statements in a programming language."

— W. Cunningham

"Only ugly languages become popular. Python is the exception."

— Donald Knuth

---

1 Introduction

This lab will mainly not contain a lot of text, but instead point you to places in your text book to read. While we think that repetition is useful, the book often does a better job of explaining concepts and certainly does a better job with graphics than we do in the labs. We will call *The Practice of Computing Using Python* the *Computing Python* book from now on.

2 Turtle

Please see pages 711-719 in *The Practice of Computing Using Python* for more turtle commands that you will need in this lab; for example, those involving color.

Warning

The book lists a lot of turtle commands, but does so without prepending the method name with `turtle.`. This is because instead of using `import turtle`, they write `from turtle import *`.

We will not be using this method; please continue to use `import turtle` and `turtle.method()`. For example, use `turtle.begin_fill()` instead of `begin_fill()`.

3 Functions

Functions are covered by *Chapter 6* in *The Practice of Computing Using Python*.

4 Exercises

letters.py *Computing Python* 6.9

fibonacci.py *Computing Python* 6.10

leap_year.py *Computing Python* 6.12

football.py *Computing Python* 6.17

flag.py *Computing Python* Project 6.1 (a), (b)

Write two functions that draw the United States flag using Turtle. These functions must use at least 2 other functions that you write to help draw the repetitive parts of the flag.

One of the functions should draw the flag with the 13 stars arranged in rows as shown on page 276 in your book.

One of the functions should draw the flag with the 13 stars in a circle. (Hint: Is it really a circle or is it some other *regular* figure?)

Hint: Take a look at polygons.py and star.py from previous labs.

find.py *Computing Python* Project 6.2 (a)

hailstone.py The Collatz Conjecture states that, for any given n, you will always reach 1 if you repeat the following steps for long enough:

- If n is odd, multiply it by 3 and add 1.
- If n is even, divide it by 2.

The sequence of values you have along the way is referred to as the hailstone sequence for the starting number. For example, the process performed on an initial n of 3 is shown in Table 1.

n	Rule performed
3	$3n + 1$
10	$\frac{n}{2}$
5	$3n + 1$
16	$\frac{n}{2}$
8	$\frac{n}{2}$
4	$\frac{n}{2}$
2	$\frac{n}{2}$
1	Done

Table 1: Hailstone Sequence for $n = 3$

The number of steps that must be performed before reaching 1 is called the stopping distance. The stopping time of 3 in the above example is 7. The stopping time of $n = 1$ would be 0.

Write a function that takes in a starting n for the hailstone sequence. The function should then calculate the stopping time for that number as well as the largest value passed on the way to 1. The above example in Table 1 would report a stopping time of 7 and a maximum value of 16.

5 Submitting

Files to submit:

- letters.py
- fibonacci.py
- leap_year.py
- football.py
- flag.py
- find.py
- hailstone.py

You may submit your code as either a tarball (instructions below) or as a .zip file. Either one should contain all files used in the exercises for this lab. The submitted file should be named either `cse107_firstname_lastname_lab4.zip` or `cse107_firstname_lastname_lab4.tar.gz` depending on which method you used.

For Windows, use a tool you like to create a `.zip` file. The TCC computers should have `7z` installed. For Linux, look at lab 1 for instructions on how to create a tarball or use the "Archive Manager" graphical tool.

Upload your tarball or .zip file to Canvas.