# Computer Science I Conditionals

# CSCI141 Homework

08/22/2020

## Overview

You will write one program that performs two tasks. The program prompts for arguments and calls functions to do the work. The program file should be `conditionals.py`.

## 1 Task 1

Write a function that determines if the larger of two integers is evenly divisible by the smaller integer.

### 1.1 Requirements

1. Name your function `divisible`. Your function must take two parameters. You may assume that the arguments provided to your function are integer values. That is, you do not need to do any error-checking to confirm that they are integer values. Note that the integers are not necessarily in any particular order. The larger integer may be either the first or the second value provided to your function.

2. Your function should return the values described below based on processing the arguments.
   - If either or both of the integers is not positive (i.e. less than 1), return `None`.
   - If the integers are both positive, and are equal, return `True`.
   - If the integers are both positive and are not equal, then check whether the larger integer is evenly divisible by the smaller integer. If the divisibility check is true, then return `True`.
   - If the integers are both positive and are not equal, and the larger integer is not evenly divisible by the smaller integer, then return `False`.

3. To run task 1, create a `run_divisible` function to prompt for inputs, call `divisible` and print the messages described below. Your messages need not exactly match the messages below, but they must provide the same information.

- When `divisible` returns `None`, the function should output the message:
  `Inputs must be positive integers!`

- In the case of equal, positive integers, the function should output the message:
  `a equals b!`
  where `a` and `b` are the integers.

- In the case of divisible integers, the function should output the message:
  `a is evenly divisible by b`
  In this output, `a` is the larger integer, and `b` is the smaller integer.

- Otherwise, in the non-divisible case, the function should output the message:
  `a is not evenly divisible by b`
  In this output, `a` is the larger integer, and `b` is the smaller integer.

### 1.2 Examples

If you were to call the `divisible` function from the interactive console, here are some example results:

```
>>> divisible(6,4)
False
>>> divisible(-4, 9)
None
>>> divisible(2,4)
True
>>> divisible(0,4)
None
>>> run_divisible()
Getting input for divisible:
enter an integer: 3
enter another integer: 28
28 is not evenly divisible by 3
>>> run_divisible()
Getting input for divisible:
enter an integer: 27
enter another integer: 3
27 is evenly divisible by 3
```

## 2 Task 2

Write a function that determines if three integer arguments could represent the lengths of the sides of a triangle. (Look up the triangle inequality if necessary.)

### 2.1 Requirements

- Name your function `is_triangle`. Your function must take three parameters. You may assume that the arguments provided to your function are integer values. That is, you do not need to do any error-checking to confirm that they are integer values.
- Your function should return the values described below based on processing the arguments.
  - If any of the integers is not positive, then return `None`.
  - If the integers are all positive, and can represent the lengths of the sides of a triangle, then return `True`.
  - If the integers are all positive, and can not represent the lengths of the sides of a triangle, then return `False`.
- To run task 2, create a `run_is_triangle` function to prompt for inputs, call `is_triangle` and print the messages described below. Your messages need not exactly match the messages below, but they must provide the same information.
  Your `run_is_triangle` should print one of the following messages based on the results of the call:
  - When `is_triangle` returns `None`, the function should output the message: `Triangles require sides of positive length!`.
  - If the integers are all positive, and can represent the lengths of the sides of a triangle, output the message: `a, b and c can form a triangle`, where `a`, `b` and `c` are the integer arguments of the function.
  - If the integers are all positive, and can not represent the lengths of the sides of a triangle, output the message: `a, b and c can not form a triangle`, where `a`, `b` and `c` are the integer arguments of the function.
- There is a degenerate case when the integers are all positive and the sum of two of the integers is exactly equal to the third integer; that counts as a case of arguments that can legally form a triangle.

### 2.2 Examples

If you were to call the `is_triangle` function from the interactive console:

```
>>> is_triangle(-3, 6, 5)
None
>>> is_triangle(4, 6, 5)
True
>>> is_triangle(8, 11, 32)
False
```

## 3 `main` and the Last Lines

The `main` function will simply call `run_divisible` and then call `run_is_triangle`.

At the bottom of your program, put the call to `main` inside an `if __name__ == "__main__"` clause block so that `main` executes only if the file is the main module.

## 4 Programming Tips

### 4.1 Modulo

Python provides an operator that computes the remainder when one integer is divided by another. It is very useful, and you will see it again in this course! It is called the modulo operator. From the interactive console:

```
>>> 8 % 3
2
>>> 4 % 11
4
>>> 60 % 5
0
```

### 4.2 Simple `print` Formatting

Normally, if you execute a print statement like

```
print( 5, "x", 2 )
```

your output looks like

```
5 x 2
```

If you want there to be no spaces between the values printed, add a value for a special, named parameter `sep`:

```
print( 5, "x", 2, sep="" )
```

### 4.3 Grading

- 35%: Correct functionality of the `divisible` function.
- 35%: Correct functionality of the `is_triangle` function.
- 10%: Correct functionality provided in `run_divisible`.
- 10%: Correct functionality provided in `run_is_triangle`.
- 5%: Each function has a *docstring* containing a sentence describing its purpose. This documentation helps others understand how they may reuse the function. An example is provided on the Course Resources webpage:
  `http://www.cs.rit.edu/~csci141/Docs/style-example-py.txt`
- 5% The program is in the correct, standard style, starting with a *docstring* for the whole file. This program file docstring must contain your *full* name as the author.

### 4.4 Submission

Be sure to document your code and test it with a variety of argument values to check its behavior. Grading will use an external, automated test program which checks that you named and implemented the functions correctly.

Zip your file called `conditionals.py` into a zip file called `hw02.zip` and submit that zip file to the MyCourses dropbox for this assignment.