# Homework 2

## 1 Without Coding

### Problem 1 (7 points)

Rewrite this following "for" loop as a "while" loop:

```
for i in range(1, X+1):
    if X % i == 0:
        print(i)
```

[1]

### Problem 2 (7 points)

What is the output of the following code? Show your work, including how x, i, and j change as the code runs.

```
i = 0
j = 0
while i<10:
    for x in range(3):
        j+=2
        i+=1
print(i,j)
```

[2]

### Problem 3 (10 points)

Briefly explain the following errors and an example that could cause them:

- ZeroDivisionError
- TypeError
- NameError
- SyntaxError
- IndexError

### Problem 4 (6 points)

What is the value of lines $2, 4, 5, 6, 9$, and $11$? Assume the lines are executed in order and errors may occur (as if they are separate notebook cells).

1. d = {'apples': 15, 'bananas': 35, 'grapes':12}
2. d['banana']
3. d['oranges']=20
4. **len**(d)
5. 'grapes' **in** d
6. d['pears']
7. fruits = **list**(d.keys())
8. **sorted**(fruits)
9. **print**( fruits )
10. **del** d['apples']
11. 'apples' **in** d

## 2 Coding Problems

### Problem 5 (10 points)

Write a method to check if a year is a leap year. To calculate if a year is a leap year, it must be divisible by 4, but NOT divisible by 100, UNLESS it's divisible by 400.

The method signature should be is_leap(year) and should return either True or False.

### Problem 6 (10 points)

Write a method to check if a number is a triangular number.
The method signature should be is_triangle (num) and return either True or False.
It is not trivial to check whether a number is triangular, but we can easily generate them using the formula for the nth triangular number:
$T_n = 1 + 2 + \ldots + n$
Example: is_triangle (1) = True
Example: is_triangle (5) = False

## Problem 7 (10 points)

Write a method to find the sum of all the triangular numbers in an inclusive range.
The method signature should be triangle_sum(lower_bound, upper_bound) and return the integer sum. You **must** also call is_triangle ().

Example: triangle_sum(1,10) = 1+3+6+10 = 20
Example: triangle_sum(10,20) = 10+15 = 25

## Problem 8 (10 points)

Write a method that will return a list of random non-zero digits (1-9) of length n.
The method signature should be random_gen(n) and return a list of random numbers.

You must use a loop, appending to a list, and the random module (specifically use random.randint). Your code may only call randint n times and cannot otherwise use the random module (extra calls will break the autograder).

## Problem 9 (10 points)

Write a method that manually calculates the **bitwise and** of two bit strings, a and b, and returns the resulting base 10 integer. You must solve this problem using loops and cannot use any built in binary functions in Python.

The method signature should be bit_and(a,b)
Example: bit_and("101","110") = 4
Example: bit_and("1011","10") = 2

## Problem 10 (11 points)

Write a method digit_sum that will add the digits of a given input and if the sum is a single digit, it will return that output and if not, it will add the digits of the following sum until a single digit sum is reached. The method signature should be digit_sum(num) and return an integer.

Ex:
digit_sum(129) = 3 because...
1+2+9 = 12 $\times$
1+2 = 3 ✓

digit_sum(825) = 6 because...
8+2+5 = 15 $\times$
1+5 = 6 ✓

digit_sum(11) = 2 because...
1+1 = 2 ✓

## Submission Instructions

Submit 2 files:
hw2.py - For the coding part
hw2.pdf - For the written part

Submit both parts on **Gradescope.**

At the top of both hw2.py and hw2.pdf, include the following information:

- Your name
- The name of any classmates you discussed the assignment with, or the words "no collaborators"
- A list of sources you used (textbooks, wikipedia, research papers, etc.) to solve the assignment, or the words "no sources"
- Whether or not you're using your extension

## Grading Methodology

The **written part** of the assignment will be graded for correctness. In the case that your answer is incorrect, partial credit may be awarded based on the provided work.

The **code part** of the assignment will be graded by an automatic grading program. It will use the method signatures specified in the assignment and use multiple test cases.

**10 points** are allocated for proper comments and styling and edge case detection:

- Descriptive variable names
- Comments for what functions do
- Comments for complex parts of code
- Proper naming conventions for any Variables, Functions, and Classes