Test 3 – Practice Problems for the Paper-and-Pencil portion

1. Consider the two functions below and the user input shown below that.

```
def sum_numbers1():
    total = 0
    while True:
        number = float(input('Enter a number (0 to quit): '))
        if number == 0:
            break
        total = total + number

    return total


def sum_numbers2():
    total = 0
    while True:
        try:
            number = float(input('Enter a number (0 to quit): '))
            if number == 0:
                break
            total = total + number
        except ValueError:
            print('  You entered a string that is NOT a number.')
            print('  Try again.')

    return total
```

User enters:

30

5

zero

4

0

Questions are on the next page.

a. What happens if a program calls **`sum_numbers1`** and the user attempts to enter the user-input shown on the previous page?

b. What happens if a program calls **`sum_numbers2`** and the user attempts to enter the user-input shown on the previous page?

c. Why might it be better for the program to call **`sum_numbers2`** instead of **`sum_numbers1`**?

2. Consider the following statements:

```
c1 = rg.Circle(zg.Point(200, 200), 25)
c2 = c1
```

   At this point, how many ***rg.Circle*** objects have been constructed? **1** **2**
   (circle your choice)

3. Continuing the previous problem, consider an additional statement that follows the preceding two statements:

```
c1.radius = 77
```

   After the above statement executes, the variable ***c1*** refers to the same object to which it referred prior to this statement. **True** **False**
   (circle your choice)

4. Continuing the previous problems:

   - What is the value of ***c1***'s radius after the statement in the previous problem executes? **25** **77** (circle your choice)
   - What is the value of ***c2***'s radius after the statement in the previous problem executes? **25** **77** (circle your choice)

5. Which of the following two statements mutates an object? (Circle your choice.)

```
numbers1 = numbers2
```

```
numbers1[0] = numbers2[0]
```

6. Mutable objects are good because:

7. Explain briefly why mutable objects are dangerous.

8. What is the difference between the following two expressions?

`numbers[3]` **`numbers = [3]`**

9. Consider the two functions below.

```
def f1(list of numbers):
    """
    RETURNs a new list that is the same as the given list of numbers
    except that the last item in the list is doubled.  For example, if
    the given list is [4, 2, 8, 5], this function returns [4, 2, 8, 10].
    """

def f2(list of numbers):
    """
    MUTATEs the given list of numbers by doubling the last item
    in the list. For example, if the given list is [4, 2, 8, 5],
    then this function mutates the list to [4, 2, 8, 10].
    """
```

Suppose that the main program has a list named my_list that contains 10,000 numbers.

Suppose further that the program makes two function calls:

- **`f1(my_list)`**
- **`f2(my_list)`**

Which function call will take longer to execute? (Circle your choice.) Why?

10. In the space below, write an implementation for the function whose specification is shown in the following box. Do NOT use your computer for this (or for any other of these paper-and-pencil problems).

```
def shape(r):
    """
    Prints shapes per the following examples:
    When r = 5:                          When r = 3
        *****5                           ***3
        ****54                           **32
        ***543                           *321
        **5432
        *54321
    Precondition:  r is a non-negative integer.
    For purposes of "lining up", assume r is a single digit.
    """
```

11. Consider the code snippet below. It is a contrived example with poor style, but it will run without errors. What does it print when ***main*** runs?

    Write your answer in the box to the right.

```
def main():
    for j in range(5):
        for k in range(j):
            print(j, k)
```

Output:

Output:

12. Consider the code snippet below. It is a contrived example with poor style, but it will run without errors. What does it print when ***main*** runs?

    Write your answer in the box to the left.

```
def main():
    for j in range(5):
        print('here')
        for k in range(1, j - 1):
            print(j, k)

        print('there')
        for k in range(2, j + 1):
            print(j, k)
```

13. Consider the code snippet in the box below. It is a contrived example with poor style, but it will run without errors. What does it print when ***main*** runs?

    Write your answer in the box shown to the right of the code.

```
def main():
    seq = [('one', 'two', 'three', 'four'),
           ('five', 'six', 'seven'),
           ('eight', 'nine', 'ten'),
           ['is this ok?'],
           (),
           ('123456', '1234')]

    for k in range(len(seq)):
        for j in range(len(seq[k])):
            print(j, k)
            if len(seq[k][j]) > 3:
                print(seq[k][j], len(seq[k][j]))
```

Output: