Name: \_\_\_\_\_\_\_Yang Zhang\_\_\_\_\_\_\_\_\_\_\_ Section: 1 2

**Test 2 – Practice Problems for the Paper-and-Pencil portion**

1. 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.

**Output:**

0 [44,77]

1 [44,77,60]

2 [44,77,60,30]

3 [44,77,60,30,50]

A. 50,30,60,77

B. 44

def **main**():

b = [44]

a = (50, 30, 60, 77)

x = 3

for k in range(len(a)):

b.append(a[x - k])

print(k, b)

print(*'A.'*, a)

print(*'B.'*, b)

1. Consider the following two candidate function definitions:

**def foo(x):**

**print(x)**

**def foo():**

**print('hello')**

* 1. Which is “better”? Circle the better function.
  2. Briefly explain why you circled the one you did.

X can be anything and print without error

1. True or false: ***Variables are REFERENCES to objects***. **True** **False** (circle your choice)
2. True or false: ***Assignment*** (e.g. **x = 100**)  
   causes a variable to refer to an object. **True** **False** (circle your choice)
3. True or false: ***Function calls*** (e.g. **foo(54, x)**)  
   also cause variables to refer to objects. **True** **False** (circle your choice)
4. Give one example of an object that is a ***container*** object:

rg.Circle

1. Give one example of an object that is ***NOT*** a ***container*** object:

8

1. True or false: *W*hen an object is mutated, it no longer refers  
   to the same object to which it referred prior to the mutating. **True** **False**  
    (circle your choice)
2. Short answer:
   1. What is the difference between a ***class*** and an ***instance of a class*** (in other words, the difference between a ***class*** and an ***object***)?

Class indicates the type of the object while the object indicates a thing.

* 1. Write a line or two of code that contains an example of each, clearly identifying the ***class*** and the ***object***.

Class: str(thing)

Object: 56

1. Draw a portion of the UML class diagram  
   for Rosegraphics’ ***Circle*** class. You don’t  
   have to get the details right (in fact, you can  
   invent details as you wish), nor to show the  
   entire UML class diagram – all you have to do  
   is show that you know what it means  
   to draw a UML class diagram.

|  |
| --- |
| **Circle** |
| **Center: point**  **Radius: int** |
| **Circle.attachto()**  **Circle.moveto()** |

1. Consider the following statements:

**c1 = zg.Circle(zg.Point(200, 200), 25)**

**c2 = c1**

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

1. 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)

1. 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)
2. Which of the following two statements mutates an object? (Circle your choice.)

**numbers1 = numbers2**

**numbers1[0] = numbers2[0]**

1. Mutable objects are good because:

It is much more efficient

1. Explain briefly why mutable objects are dangerous.

The coding can keeping running while there is a bug

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

**numbers[3]:index3 of number numbers = [3]:numbers equal to set [3]**

1. Consider the code in the below. To the right of the box of code, draw the ***box-and-pointer diagram*** for what happens when ***main*** runs. In the space below, show what the code would ***print*** when ***main*** runs.

import rosegraphics as rg

def **main**():

point1 = rg.Point(8, 10)

point2 = rg.Point(20, 30)

x = 405

y = [7, 4, 13]

print(*'Before:'*,

point1, point2, x, y)

**z = change(point1, point2, x, y)**

print(*'After:'*,

point1, point2, x, y, z)

def **change**(point1, point2, x, a):

point1.x = point2.y

point2 = rg.Point(5, 6)

point2.x = point1.y

x = 99

a[1] = 888

print(*'Within:'*,

point1, point2, x, a)

return a

**Draw box-and-pointer diagram below here**

**What prints when *main* runs?** (Assume that *points* get printed as per this example: **Point(8, 10)**.)

**Before: \_rg.Point(8,10)\_rg.Point(20,30)\_405\_[7,4,3]\_**

**Within: \_rg.Point(30,10)\_rg.Point(10,6)\_405\_[7,888,13]\_**

**After: \_rg.Point(8,10)\_rg.Point(20,30)\_405\_[7,4,3]\_**