Name:_____ **SOLUTION**

Range expressions:

1. What is the output of the following code snippet?

```
for k in range(200, 215):
    print(k)
```

200 201 202 ... 214 on separate lines

2. Modify the code above so that it also prints the 215 as part of the output.
   Change the 215 to 216.

3. Joe wants his **for** loop to output the numbers counting DOWN from 100 to ***n***, inclusive, for some number ***n*** smaller than 100. He writes:

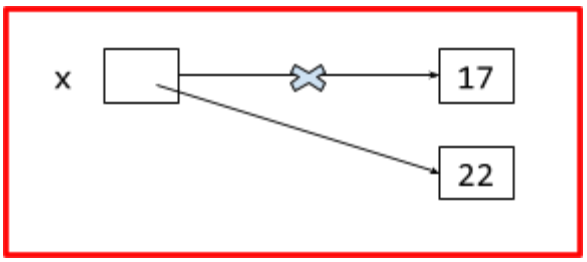
```
for k in range(100, n - 1, -1):
    print(k)
```

He correctly remembered the step of -1, but made another small bug. Find and fix it.
Change the ***n*** to ***n - 1***.

Box and pointer diagrams:

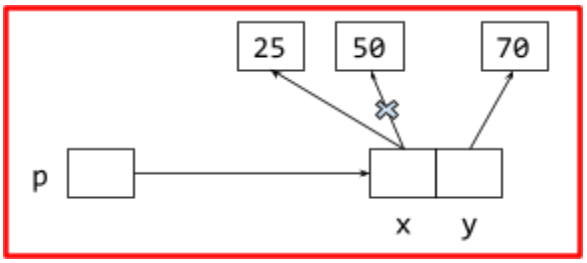
4. Draw a box-and-pointer diagram for the following statements. Recall that you should cross out the arrows rather than erase them:

```
x = 17
x = x + 5
```



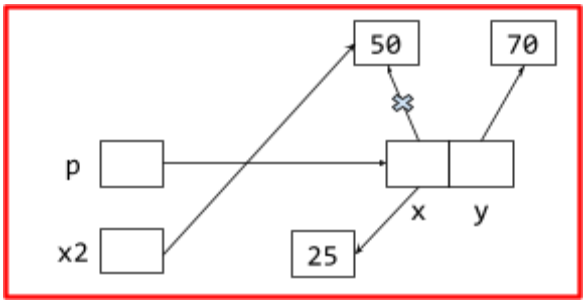
5. Draw a box-and-pointer diagram for the following statements.

```
p = rg.Point(50, 70)
p.x = 25
```



6. Draw a box-and-pointer diagram for the following statements.

```
p = rg.Point(50, 70)
x2 = p.x
p.x = 25
```



What is the value of **x2** after this code runs? ___ **50** _____
Use your box and pointer diagram to help. (Suggestion: ask for the answer to the above and use it to check your diagram.)

Implementing Classes:

7. What gets printed when the code to the right runs?

   999 2

```
class Point(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y

def main():
    point = Point(1, 2)
    blah(point)
    print(point.x, point.y)

def blah(point):
    point.x = 999
    point = Point(33, 44)

main()
```

8. Every object in Python has **two** things: what are they? (Put a mark by TWO of the following items.)

   _____ ***A type***

   _____ ***A value***

   _____ An accumulator

9. In object-oriented programming, you can create custom classes. What is a ***class***?

   _____ A collection of students _____ ***A custom type*** _____ A socioeconomic group

10. What is the name of the ***constructor method*** in Python? (don't forget the underscores) **__init__**

11. Recall that classes have a *name*, *instance variables*, and *methods*. Here (below and to the right) is the definition of part of a simple class that you saw in the video:

    a. Give an example from the code of an ***instance variable***:
       x when used as **self.x**
       y when used as **self.y**
    b. Give an example from the code of a ***method***:
       move_by **__init__**
    c. What is the ***name*** of the class? _____ **Point**
    d. What ***keyword*** was used to define the class? _ **class**

```
class Point(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def move_by(self, dx, dy):
        # Location 1
        self.x = self.x + dx
        self.y = self.y + dy
```

12. Continuing the previous problem (with its ***Point*** class), consider the two lines of code shown to the right. When those two lines of code run, the execution of the second line brings us to Location 1 (see the *Point* class above to find Location 1). **At Location 1, what are the values of:**

```
p = Point(40, 50)
p.move_by(1, 2)
```

    self ___ The object constructed by **Point(40, 50)**;
    the object to which **move_by** is being applied; the object in front of the DOT;
    the object referred to in the 2nd set of code as **p**. (Grader: Any of these, or just **p** is OK.)

    dy _ **2** **self.y** __ **50** (will be set to **52** by **move_by**)