20.10. Class Variables and Instance Variables

You have already seen that each instance of a class has its own namespace with its own instance variables. Two instances of the Point class each have their own instance variable x. Setting x in one instance doesn't affect the other instance.

A class can also have class variables. A class variable is set as part of the class definition.

For example, consider the following version of the Point class. Here we have added a graph method that generates a string representing a little text-based graph with the Point plotted on the graph. It's not a very pretty graph, in part because the y-axis is stretched like a rubber band, but you can get the idea from this.

Note that there is an assignment to the variable printed_rep on line 4. It is not inside any method. That makes it a class variable. It is accessed in the same way as instance variables. For example, on line 16, there is a reference to self.printed_rep. If you change line 4, you have it print a different character at the x,y coordinates of the Point in the graph.

Save & Run Original - 1 of 1 Show in CodeLens

```
1  class Point:
2      """ Point class for representing and manipulating x,y coordinates. """
3
4      printed_rep = "*"
5
6      def __init__(self, initX, initY):
7
8          self.x = initX
9          self.y = initY
10
11     def graph(self):
12         rows = []
13         size = max(int(self.x), int(self.y)) + 2
14         for j in range(size-1) :
15             if (j+1) == int(self.y):
```

```
4
3 *
2
1
01234

3
2  *
1
0
9
8
7
6
5
4
3
2
1
01234567890123
```

Activity: 1 -- ActiveCode (classvars_1)

To be able to reason about class variables and instance variables, it is helpful to know the rules that the python interpreter uses. That way, you can mentally simulate what the interpreter does.

When the interpreter sees an expression of the form <obj>.<varname>, it:

1. Checks if the object has an instance variable set. If so, it uses that value.
2. If it doesn't find an instance variable, it checks whether the class has a class variable. If so it uses that value.
3. If it doesn't find an instance or a class variable, it creates a runtime error (actually, it does one other check first, which you will learn about in the next chapter).

When the interpreter sees an assignment statement of the form <obj>.<varname> = <expr>, it:

1. Evaluates the expression on the right-hand side to yield some python object;
2. Sets the instance variable <varname> of <obj> to be bound to that python object. Note that an assignment statement of this form never sets the class variable; it only sets the instance variable.

In order to set the class variable, you use an assignment statement of the form <varname> = <expr> at the top-level in a class definition, like on line 4 in the code above to set the class variable printed_rep.

In case you are curious, method definitions also create class variables. Thus, in the code above, graph becomes a class variable that is bound to a function/method object. p1.graph() is evaluated by:

- looking up p1 and finding that it's an instance of Point
- looking for an instance variable called graph in p1, but not finding one
- looking for a class variable called graph in p1's class, the Point class; it finds a function/method object
- Because of the () after the word graph, it invokes the function/method object, with the parameter self bound to the object p1 points to.

Try running it in codelens and see if you can follow how it all works.

You have attempted 2 of 1 activities on this page

| Back to top