1. Consider the following code, which is implemented as part of a traffic control program:

```
1  class Stoplight:
2      def __init__(self, currentState):
3          # Current state is a number between 0 and 2 (inclusive)
4          # representing the state of the stoplight
5          self.states = ["Green", "Yellow", "Red"]
6          self.currentState = currentState
7
8      def updateState(self):
9          self.currentState = (self.currentState + 1) % len(self.states)
10
11     def getCurrentState(self):
12         return self.states[self.currentState]
13
14 if __name__ == "__main__":
15     stoplight1 = Stoplight(0)
16     stoplight2 = Stoplight(0)
17
18     print(stoplight1.getCurrentState())
19     print(stoplight2.getCurrentState())
20     print()
21
22     stoplight1.updateState()
23     print(stoplight1.getCurrentState())
24     print(stoplight2.getCurrentState())
```

(a) Observe the function on line 2. What is its purpose? When is it called?

it's the Constructor method
it initializes Stoplight

(b) What is the self keyword, and why is it necessary for this code to work properly?

it refers to the Class
it allows it to refrence itself

(c) After the code is run, what is the output?

Green
Green
Yellow
Green

2. Consider the following code:

```
1  class PetriDish:
2      def __init__(initialCells):
3          self.cells = initialCells
4
5      def updatePetriDish(numSeconds):
6          # Assumes the number of cells in the petri dish double every second
7          for i in range(numSeconds):
8              self.cells = self.cells * 2
9
10     def getNumCells():
11         return self.cells
12
13 if __name__ == "__main__":
14     p = PetriDish(3)
15     p.updatePetriDish(6)
16     print(p.getNumCells())
```

There is an error in this code. What is it? And why is it an issue?

Line 2, it should have self as the first Parameter

It needs to access it's own attributes

3. A key aspect to understanding the usefulness of classes is to recognize the value of objects. We haven't used the word much in this class, but virtually everything in Python is an object. All objects contain attributes (a.k.a variables, fields, data) and behavior (a.k.a functions, methods, code). Choose either the String or List object and answer the following questions.

   (a) What data does your chosen object contain?

   (b) What are examples of behavior or methods your attribute contains? (These are typically identified with the dot notation. E.g. [3, 2, 1].sort())

   (c) What is another example of an object we've used in class besides the String and List objects?

4. Describe, in your own words, the difference between a class and an object.

A class is a blueprint that defines the structure.

An object is an instance of a class.

5. Imagine you are trying to create your own class, to model a car. What are some attributes you think would be good to include? What are some methods?

Attributes [Make, Model
Color, Speed

Methods [refuel, getSpeed
start
stop

6. Why are classes and objects useful? How might you start using them in your own code?