EXAM 2 REVIEW 2

Question 1

- What are the five steps in the Software Development Life Cycle?

Question 1

- What are the five steps in the Software Development Life Cycle?

1. Understand the problem – specification / analysis
2. Work out the logic – design
3. Write code – implementation
4. Make sure it works – test/debug
5. Maintenance

Question 2

- Given the following accessor, is this valid? Why or why not?

```
class Balloon:
    def __init__(self, color, radius):
        # color must be a valid RGB color
        self.color = color
        # radius must be between 0 and 12 inclusive
        self.radius = radius

    def getRadius(self):
        if self.radius <= 12 and self.radius >= 0:
            return self.radius
        return 0
```

Question 2

- Given the following accessor, is this valid? Why or why not?

```
def getRadius(self):
    if self.radius <= 12 and self.radius >= 0:
        return self.radius
    return 0
```

Yes, it's valid. If the "if" condition tests true, the "return 0" is never reached. If it is not true, the accessor will return the value 0.

It probably should return the max value (12) if self.radius contains a number greater than 12.

Question 3

- What prints out when this code is executed?:

```
def mystery(x):
    if x % 2 == 1:
        if x**3 != 27:
            x = x + 4
        else:
            x = x / 1.5
    else:
        if x <= 10:
            x = x * 2
        else:
            x = x - 2
    return x

x = 8
print(mystery(x))
print(x)
x = 5
print(mystery(x))
print(x)
```

Question 3

- What prints out when this code is executed?:

```
def mystery(x):
    if x % 2 == 1:
        if x**3 != 27:
            x = x + 4
        else:
            x = x / 1.5
    else:
        if x <= 10:
            x = x * 2
        else:
            x = x - 2
    return x

x = 8
print(mystery(x))
print(x)
x = 5
print(mystery(x))
print(x)
```

```
16
8
9
5
```

Question 4

- Consider the following two implementations of this function:

```
def audioReverse(audio):
    if len(audio) <= 0:
        return []
    return audio.reverse()

def audioReverse(audio):
    try:
       list(reversed(audio))
    except:
        return []
```

- Which one is better coding style? Why?

Question 4

- Consider the following two implementations of this function:

```
def audioReverse(audio):
    if len(audio) <= 0:
        return []
    return audio.reverse()

def audioReverse(audio):
    try:
       list(reversed(audio))
    except:
        return []
```

- Which one is better coding style? Why?
- If you anticipate that unknown data types may be sent to this function, you may want to use the second one, since a simple if statement may not catch that. If you are only concerned with the length of the list passed in being greater than 0, it is better to use the first approach. (Principle is, if you can catch an error using logic, do that, rather than using exceptions.)

Question 5

- What are five types of methods you should seriously consider including when you create a class?

Question 5

- What are five types of methods you should seriously consider including when you create a class?

- Constructor
- Accessors
- Mutators
- equals()
- toString()
- (and then any other behaviors your class needs to have)

Question 6

- What is the point of having an API (Application Programming Interface)?

Question 6

- What is the point of having an API (Application Programming Interface)?

  Encapsulation! You want to separate the code using your objects from the code implementing object behavior as much as possible. You also want to protect any data associated with your objects from being directly accessed and possibly improperly used or changed, so you want to provide methods that can check that data is valid before changing attribute values.

Question 7

- What does the “self” keyword refer to in a class?

- What does “super()” refer to?

Question 7

- What does the “self” keyword refer to in a class?
  - The particular instance that is using that data and code. “Self” means the current instance, not the general class, or blueprint.

- What does “super()” refer to?
  - My parent. If I invoke super().<something> I am asking to use the data or method of my parent.