CPS 109 - Review

Agenda

1. Reading/Deciphering questions
2. Data types
3. Loops
4. Recursion
5. Testing
6. Classes

Understanding Questions

It came to my attention that everyone is having a bit of trouble understanding what a problem or question is looking for.

Understanding Questions

```
def q3(x, epsilon) :
    '''
    Returns guess, the approximate square root of x, such that
    abs(guess**2 - x) < epsilon using Heron's algorithm, where
    start with guess = x / 2 and improve guess to be (guess + x / guess) / 2
    '''
    pass
```

Understanding Questions

```
class myTests(unittest.TestCase):
    def test0(self):
        self.assertTrue(abs(q3(4, 0.1) - 2) <= 0.1)
    def test1(self):
        self.assertTrue(abs(q3(99, 0.1) - math.sqrt(99)) <= 0.1)
    def test2(self):
        self.assertTrue(abs(q3(999, 0.1) - math.sqrt(999)) <= 0.1)
    def test3(self):
        self.assertTrue(abs(q3(1, 0.1) - 1) <= 0.1)
    def test4(self):
        self.assertTrue(abs(q3(0.25, 0.01) - 0.5) <= 0.01)
```

Data Types

So, let’s talk about data types. However, let’s recall a few handy definitions...

Data Types

Something is **mutable** if its contents can be changed (like a list).

Something is **immutable** if its contents cannot be changed (like a string or a tuple) and instead you have to overwrite it.

Data Types

So what are the data types you should know?

Integers - A whole number. 1, 7, -10, etc.

Boolean - A value of either True or False

Floats - A number decimal number. 3.33, 2.1, -1100.9999

Data Types

String - A string is a series of letters, symbols and other characters in between a set of “” or ‘’. “This is a string”.

List - A list is a mutable collection of any data you like, separated by a comma. [1, 2, 5] or [“a”, “b”, “c”]

Data Types

Tuple - A tuple is an immutable collection of objects. (1, 2, 5) or (“a”, “b”, “c”)

Dictionary - A dictionary is an UNORDERED set of keys and their associated values. {“cat”:[“Larry”, “Jerry”], “dog”: [“Beans”]}

Data Types

For every collection of data (except for a dictionary) you can iterate over it in order!

Recall, tuples, strings and lists all start indexing at 0 and go up to (length - 1) in indices.

You can also index backwards, starting at -1

Loops

Loops are your friends. Loops are how you do 99% of the things you need to do while programming.

Loops are blocks of code which repeat a certain amount of times until a condition is met or it they run out of things to iterate over.

Loops

There are two kinds of loops:

While loops - A loop that will run until the condition in its brackets evaluates to False.

For Loops - A loop that will iterate over a count or collection of things (like a list)

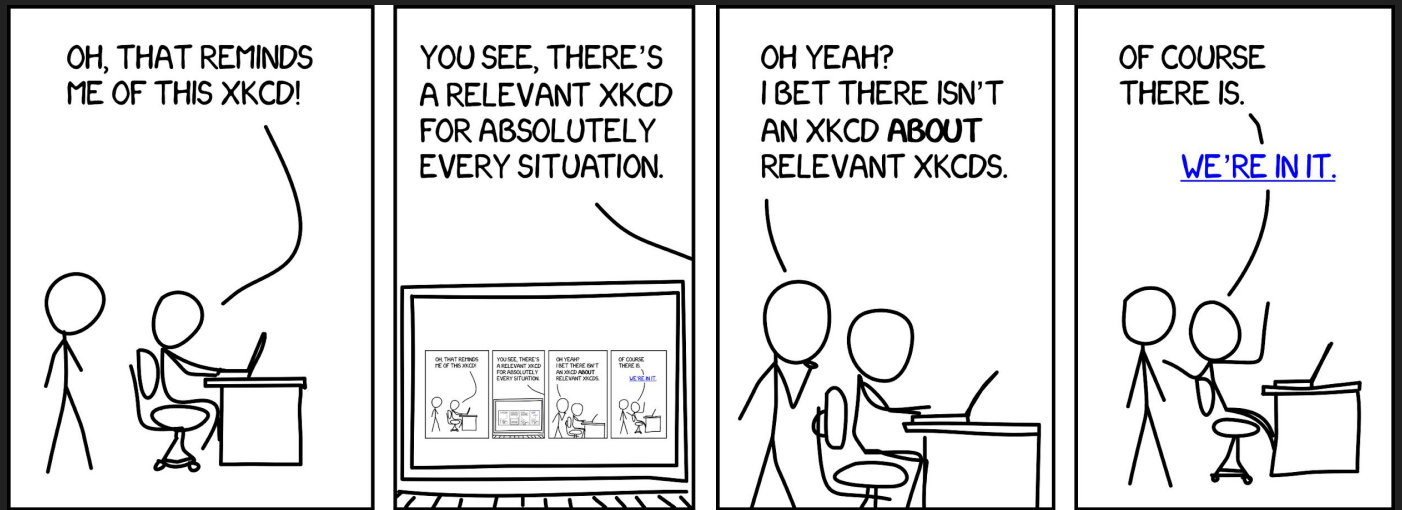
Loops

When should you use a while loop? Whenever you need to loop over some code until some condition comes true (for example, if a user enters a specific input).

When should you use a for loop? Whenever you have a collection of something (a list or a range, etc) that you need to iterate over.

Recursion

Recursion is a special sort of “loop”, so to speak. It refers to a type of function that can call itself in its execution.



Recursion

How do you design a recursive function? Here are my suggested steps:

1. Make sure your function has a clear goal
2. Write out (then type up) your 1 or more base cases.
3. Design your recursive step.

Testing

Testing is important and good for your health. How do you test things? In this course, it’s pretty simple. You declare a test class with member methods and then call it in your __main__

Testing

```
class myTests(unittest.TestCase):
    def test0(self):
        self.assertEqual(outside_of(5, 6, 2), False)
    def test1(self):
        self.assertEqual(outside_of(5, 7, 1.5), True)
    def test2(self):
        self.assertEqual(outside_of(1, 2, 1), False)
    def test3(self):
        self.assertEqual(outside_of(9, 9, 0), False)
    def test4(self):
        self.assertEqual(outside_of(9, 9, 100), False)
    def test5(self):
        self.assertEqual(outside_of(-90, 100, 0), True)
```

Classes

Which brings us to classes. Classes are custom data types that you can make yourself! They can contain any and all member variables so long as you declare the class properly.

Classes

```
class Rectangle(object) :
    '''This class represents a rectangle in the x-y coordincate system
    where the edges of the rectangle are aligned with the x- and y- axes.
    A Rectangle object has data attributes lowerleft and upperright,
    which are tuples representing the (x, y) coordinates of the lower
    left corner and the upper right corner.
    '''
    def __init__(self, x1, y1, x2, y2) :
        '''Assumes the (x1, y1) are the coordinates of the lower left corner
        and (x2, y2) are the coordinates of the upper right corner.
        '''
        self.lowerleft = (x1, y1)
        self.upperright = (x2, y2)
```

In Closing...

That about wraps everything up that we should cover.