SelfAssessment71

October 18, 2015

1 Exercise 7.1

a) Create a string variable named x consisting of the digits 0 through 9 in ascending order (no need to be fancy here).
b) Create a string variable named y consisting of the digits 0 through 9 in descending order (as above, don't get fancy).
c) For what value of j is x[j] == y[j+1]?

Answer appears after one blank page (so you don't peek).

Are you sure you're ready to peek?

2 Possible Solutions

a) Create a string variable named x consisting of the digits 0 through 9 in ascending order. I said don't get fancy, so the first option is what I'd expect. The second option is here to show you how you might make a more general solution. It also does some extra stuff.

```
In [6]: x = '0123456789'
        # or, if you were to get fancy. . .
        x = ''.join(str(i) for i in range(0,10,1))
        print x
```

```
0123456789
```

So, we see here that we start with an empty string `' '`*. Then we use the string method* ***`join`****, this appends extra strings to the end of the existing string. There's a funny use of the* ***`for`*** *loop here as well. We're taking an index* ***`i`*** *that is cycling through the sequence generated by* ***`range(0,10,1)`*** *and passing it into a function called* ***`str`****, that turns an integer into a character.*

b) Create a string variable named y consisting of the digits 0 through 9 in descending order (as above, don't get fancy).

```
In [8]: y = '9876543210'
        # or, if you were to get fancy. . .
        y = ''.join(str(i) for i in range(9,-1,-1))
        print y
```

```
9876543210
```

c) For what value of j is x[j] == y[j+1]?

```
In [24]: # Simple solution:
         print 'The simple solution is: x[4] == y[5] == \'4\' ==', x[4] == y[5] == '4'

         # More general solution:
         equality = [] # Create an empty tuple

         for i in range(0, len(x)-1, 1):
             # to index the entire length of 'x' we'd use len(x), but the y index is always 1 higher,
             # so we can't go any further than len(x) - 1 in the range
                 if x[i] == y[i+1]:
                     equality.append([i,i+1])

         print 'We can do it this more complex way and find that the solution is also:', equality
```

```
The simple solution is: x[4] == y[5] == '4' == True
We can do it this more complex way and find that the solution is: [[4, 5]]
```

In the more complex solution we're doing a couple things. There is first an empty list, defined using `equality = []`. Then we cycle through a `for` loop that goes through from 1 to 9. We stop at 9 because we're looking for the pair `[i]` and `[i + 1]` which would leave us at 9 and 10 for the last exectution of the loop.

If we ever find a pair that is equivalent we then go into an `if` statement and use the `append` method of the `list` class (that's what `equality` is) to stick in the pair.

One of the reasons I wanted to show you this is that this is a bit of a recipe for going through long lists, looking for particular values or matches, and then outputting some sort of 'map' as a variable. If we have two big shapefiles with a lot of features and want to find pairs of features that are equivalent we could use the same approach.