Directions:
Please print this lab and fill it in as you complete the exercises below. If any part is confusing, be sure to watch the class video explaining each part.

The topic for today is the idea of repetition or “looping.”

In Scratch there were some obvious ways to introduce looping through having sprites repeat some movement over and over again. In Python some of the more simple and meaningful starting examples involve looping through the characters of a string or the elements in a list. This means that in order to introduce looping we have to simultaneously introduce the very basics of Strings and lists.

We will only scratch the surface of these two data structures. But don’t worry. We will come back later and study them in more detail later in the course. However, for the time being we will look at just enough to get us started with looping.

Before starting this lab, make sure IDLE is open to the interpreter screen. That screen should look similar to this:
Strings

A String is a sequence of characters.

Most of the time we think about the String as a word or sentence, but the sequence can include anything you can type on the keyboard – letters, digits, punctuation, “cartoon swear symbols”, etc. This should be familiar to you since we learned earlier that when you ask a question in Python such as the following that the answer always starts as a String. As programmers, we are required to convert (cast) the String into the number we expect it to be.

```
answer = input("How old are you? ")
age = int(answer)
```

The most common way to make a String in Python code is to declare it using EITHER a pair of single quotes or double quotes.

```python
>>> course = "Fundamentals of Programming"
>>> classroom = 'Room #28, ITTC Building, UNI Campus'
```

Type the previous assignment statements in to the interpreter window in Python.

[Q1] **Predict** what will happen when you type the following expressions. **After** making your prediction, enter the statement at the interactions prompt and check if you were correct. Warning, one of these will produce an error.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Predicted Result</th>
<th>Actual Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>len(course)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>len(classroom)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>course[2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>classroom[5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>course[0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>classroom[len(classroom)]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>course[len(course)-1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>‘d’ in course</td>
<td></td>
<td></td>
</tr>
<tr>
<td>‘D’ in course</td>
<td></td>
<td></td>
</tr>
<tr>
<td>course.find(‘d’)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>course.find(‘D’)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>classroom.find(‘T’)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>classroom.find(“UNI”)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>course[3] == ‘d’</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lists

A list is a sequence of data. This data can be of any type including Strings, ints, strings, Boolean values, and other types we haven’t even studied yet. While it is common for the list to contain data all of the same type, that isn’t required in Python.

Lists are often defined in Python by creating a comma separated sequence of data contained inside of square brackets such as:

```python
>>> teachers = ["Diesburg", "East", "Schafer"]
>>> scores = [10, 8, 7.5, 10, 9.5, "pass"]
```

Type the previous assignment statements in to the interpreter window in Python. There is a lot of detail in these statements so please pay careful attention to what goes where. If you get an error message you typed something incorrectly.

[Q2] Predict what will happen when you type the following expressions. After making your prediction, enter the statement at the interactions prompt and check if you were correct. Warning, one of these will produce an error.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Predicted Result</th>
<th>Actual Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>len(teachers)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>len(scores)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>teachers[2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scores[4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scores[0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scores[len(scores)-1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>‘East’ in teachers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>‘10’ in scores</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scores.find(10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scores[3] == 10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Loops (repetition)

In Scratch, if you wanted to do something five times you said:
In Python, if you want to do something five times you might say:

```python
for index in range(5):
    print(str(index))
```

[Q3] What do you PREDICT will happen if you type this in to a program editor and run these two lines of code? Try it and find out.

Like Scratch (and like the body of conditionals in Python) you can put as much into the “loop” as you want as long as you indent it to the same level.

```python
for value in range(5):
    print("This is a "+str(value)+"."))
    print("A "+str(value)+" this is."))
    print("I can count higher cause I'm a math whiz."))
    print()
print("The End")
```

[Q4] What do you PREDICT will happen if you type this in to a program editor and run these lines of code? Try it and find out.

**Using Loops with Strings and Lists**

In this section we will look at two common uses of loops with Strings and lists.

[Q5] Consider the following code. What do you PREDICT will happen if you type this in to a program editor and run this program? Try is and find out.

```python
language = "Python"

for index in range(len(language)):
    character=language[index]
    print("Letter "+str(index)+" is a "+character)
```
[Q6] Consider the following code. What do you PREDICT will happen if you type this in to a program editor and run this program? Try is and find out.

```python
scores = [10, 8, 7.5, 10, 9.5, "pass"]

for value in range(len(scores)):
    grade=scores[value]
    print("Grade "+str(value)+" was a "+str(grade))
```

[Q7] Consider the following code. What do you PREDICT will happen if you type this in to a program editor and run this program? Try is and find out.

```python
#Create a variable with an "empty" version
numbers = 0
strings = ""
lists = []

#Add to it 10 times
for value in range(10):
    numbers = numbers + value
    strings = strings + str(value)
    lists.append(value)

#Look at the final results
print(str(numbers))
print(strings)
print(str(lists))
```