Pattern Maker Lab

Lab Goals: In this lab, you will write a Python program to generate different patterns using ASCII characters. In particular, you will get practice with the following:

1. Printing strings to the terminal
2. Manipulating strings with the `+` and `*` operators
3. Writing comments
4. Working with `FOR` loops

Hand-In: Submit the file `Lab2.py` to Moodle for participation credit.

---

1 Preliminaries

1.1 Writing a Python program

We will now move from using the Python interpreter (where we got the *prompt* `>>>`) to writing a *Python program*. Suppose we want to add two numbers; we can do so at the prompt by typing

```
>>> a = 5
>>> b = 10
>>> c = a + b
>>> c
```

Here, the value stored in variable `c` will be printed to the screen. Instead, we can put each line in a plain file (use Sublime, TextWranger, Notepad++, etc.) and save it with a `.py` extension. Here, let's save these commands in a file called `Lab2.py` (note the directory location where you saved it on your machine):

```
a = 5
b = 10
c = a + b
print(c)
```

Save the file. Open Terminal and `cd` to the directory containing `Lab2.py`. Run the command

```
python3 Lab2.py
```

This tells Python to run each line within `Lab2.py`.

Hint: Use the up arrow to re-run the previous line in Terminal.

1.1.1 The `print()` Function

An important difference between the prompt and a program is that **expressions aren't printed to the Terminal screen**. Instead, we need to use a print function to tell Python to print the result to the Terminal. Add these lines to `Lab2.py` and observe the output:

```
1 + 2
print(1 + 2)
'this expression does not print anything!'
print('we have to tell Python to print expressions')
```

Hint: if you want a blank line between parts you can use the `print()` function with no inputs.

1.1.2 Comments

Lines that begin with a pound sign are **comments** and are ignored by Python. Add the two lines to the top of your file (where `<NAME>` is your name):

```
## Lab 2
## <NAME>
```

1.1.3 Code Spacing

Your Python program should be easy to read - you can clarify points with comments and add extra blank lines if necessary. My `Lab2.py` file may now look like this:

```
## Lab 2
## Anna Ritz

## Add two numbers and print the result
a = 5
b = 10
c = a + b
print(c)

## Test evaluating expressions vs. printing expressions
1 + 2
print(1 + 2)
'this expression does not print anything!'
print('we have to tell Python to print expressions')
```

1.2 String Replication

We learned in class that we can use the `+` operator to concatenate strings. We can generate copies of the same string by using the `*` operator. This operator takes a **string** and an **integer**. Try the following lines and observe what happens.

```
print('A')
print('A'+'A')
print('A'*2)
print('AB'*2)
print('A'*4)
```

1.3 The `range()` Function

A **function** is a usable block of code. We have already seen some built-in functions:

`quit()`: Quit the Python interpreter (return to the Terminal)

`type(var)`: Return the type of variable `var` (e.g., `type(4)` or `type('hey')`)

The built-in function we will make use of here is the `range()` function, which takes as input an integer `x` and returns integers ranging from 0 up to **but not including** `x`. We will learn more about this function later in the week; for now we will use it to simply count from 0 to a number.

```
print(list(range(5)))
print(list(range(10)))
```

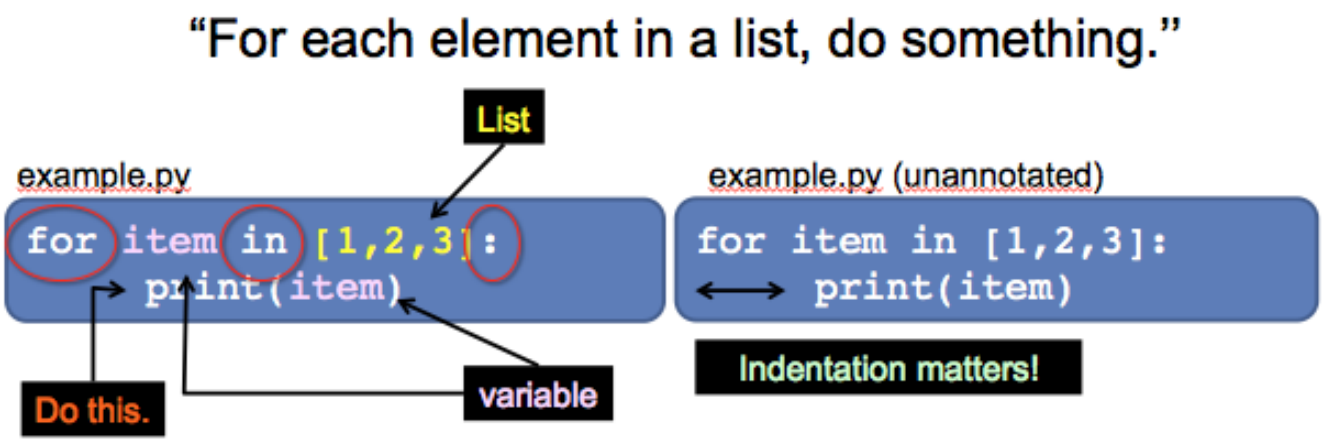
1.4 `FOR` Loops

Let's print the numbers from 1 to 5. Here's one way to do it:

```
print(1)
print(2)
print(3)
...
```

If I asked you to print the numbers from 1 to 10, you could keep writing these lines, but it's going to get long. Python allows us to say **"for each element in a list, *do* something."** This syntax, called a `FOR` loop, are statements that specify repeated execution. Pay attention to the form:

```
for i in [1,2,3,4,5]:
    print(i)
```



Instead of a list, we will use the `range()` function in this lab:

```
for i in range(5):
    print(i)
```

Question: How do we change this code so it prints the same thing as the `FOR` loop above it?

We don't need to use the `FOR` loop to print the variable `i`:

```
for i in range(5):
    print('Lab2')
```

Task: Write your name **eight times** using a `FOR` loop.

Finally, we can execute multiple things *within* the `FOR` loop. Pay attention to the indentation!

```
for i in range(5):
    print('Lab2 iteration...')
    print(i)
    print('!!!')
```

2 Patterns

In this section, you will use a combination of string operations and `FOR` loops to make a series of text patterns. Begin with the asterisk (`'*'`) as your character. Add your code to the `Lab2.py` program; when you run this program you will print each pattern you have made. **Remember: make use of `FOR` loops, the `range()` function, and the `*` and `+` string operators.**

1. Make a 5×5 rectangle:

```
*****
*****
*****
*****
*****
```

2. Make a right triangle:

```
*
**
***
****
*****
```

3. Make a 'play' button:

```
*
**
***
****
*****
*****
****
***
**
*
```

4. Make another triangle:

```
    **
   ****
  ******
 ********
**********
```

5. Make a diamond:

```
    **
   ****
  ******
 ********
**********
**********
 ********
  ******
   ****
    **
```

6. Make a rectangle outline:

```
*****
*   *
*   *
*   *
*****
```

7. Make a diamond outline:

```
    **
   *  *
  *    *
 *      *
*        *
 *      *
  *    *
   *  *
    **
```

If you have time...

1. Use print statements to annotate the output (so you have a header introducing each shape).
2. Make two variables: an integer `size` and a single character `char`. Put these at the top of the file. Modify your patterns so when you update either `size` or `char` you will get different sizes of patterns and characters displayed.
3. Make your own pattern(s), using only the concepts we have discussed in class and lab.

*Remember to submit **`Lab2.py`** to Moodle.*