WAKE TECH
Biotechnology, Engineering, & Skilled Technologies

Python Lab 2- Loops

BAT-212: BAT Logic and Programming

This material is based upon work supported by the National Science Foundation Advanced Technical Education grant program, A New Technician Training Program for Advanced Building Technologies, DUE-2000190.

The opinions, findings, and conclusions or recommendations expressed are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Python Lab 2 - Loops

OBJECTIVES

Upon completion of this activity the student will be able to:

- Write a program correctly using if-elif-else structure
- Write programs correctly using loop structures.

PARTS AND EQUIPMENT

- Circuit Playground Express board.
- PC Computer with CircuitPython and Mu editor installed.

REFERENCES

The following are immediately relevant to the lab:

- Introduction to Python Programming (first lab in this series)
- https://www.w3schools.com/python/python_for_loops.asp

The following are general references:

- https://greenteapress.com/thinkpython/html/thinkpython002.html#toc5
- https://docs.python.org/3/tutorial/

BACKGROUND

In the introductory lab, you learned about CircuitPython, the Circuit Playground Express board, and the Mu editor. You also wrote 2 programs using the if-elif-else decision structure. We will review this structure, move on to loops, and then discuss functions.

Part 1: if-elif-else Decision Structure

The format of the if decision structure is:

if *condition:*

    executable statement(s)

elif *condition:*

    executable statement(s)

else:

    executable statement(s)

The important elements for the syntax of the structure are:

1. each if, elif, and else statement ends with a colon (:)
2. the executable statements are indented below each decision statement to the same level.

Good programming style requires the if-elif structure be used rather than separate if's when the conditions are testing the same set of variables. For example:

```
if light>70:
    ……
elif light <40
    ……
```

Rather than:

```
if light >70:
    ……..
if light <40
    ……..
```

Part 2: Loops

We have seen the while loop in previous programs.

```
while TRUE:
```

creates a loop that will execute as long as the program is running – "forever." The syntax of while statement in general is

while *condition*:

    executable statements

The indented executable statements will execute repeatedly until the condition is not true. Then execution will go to the next non-indented statement. Since TRUE as a condition is always true, the while TRUE: statement is a way to write a loop that runs forever, in the program's terms. Anything that precedes the while TRUE: statement will not be repeated in the loop; anything after and not indented will not be executed, unless perhaps there is a break statement in the loop. We'll talk about that later.

Another type of loop in Python is the for loop. A for loop is used to repeat the loop a fixed number of times. There are a few different ways to determine how many times the loop is executed. We will focus on the for loop using the range function for this lab.

range() can have 1, 2, or 3 integer parameters. If only one parameter is specified, then a sequence of numbers is generated, starting at 0, and continuing to one less than the given number, in increments of plus one. If the given number is 0 or a negative number, the sequence will be empty.

If two parameters are specified in the *range()* function, the first is taken as the starting point of the sequence, and the values increment by plus one up to one less than the second parameter. If the first parameter is equal to or greater than the second parameter, the sequence is empty.

If three parameters are specified in the *range()* function, the first is taken as the starting point of the sequence, the second is the stopping point, but not included in the sequence, and the third value is the increment. The increment can be any integer value, positive or negative. *range(1,6,2)* would result in the sequence *1,3,5*. *range(6,1,-2)* would result in the sequence *6,4,2*.

PROCEDURES

Decision Statements Program 3

- Go to the directory where you unzipped the library bundle. Refer to the introductory lab if you don't have the libraries. Inside that directory find the *examples* directory. In that directory, find the file *circuitplayground_light* and copy it to the CIRCUITPY drive.
- Use the "Load" button to bring this file up in the Mu editor. Notice that the code includes the line "while True:" which means the code will loop forever.
- Save the code as code.py and see what the code does.
- We will implement a three-state switch using the if-elif-else structure. Choose two values in the range. I used 70 and 40 – just pick 2 levels that you can easily differentiate and get as output. If the light level is greater than your top value, turn the red LED on. If it drops below the low value, turn the LED off. For in-between values, it should blink at a rate of 1 second.
- Test the operation of your code and demonstrate to the instructor.
- **Comment your code well and submit.** Using the commenting guidelines given in the introduction lab.

Loop Program 1

- Go to https://www.w3schools.com/python/python_for_loops.asp . Read and do each "Try it for yourself".
    - On this example,

```
for x in range(2, 30, 3):
  print(x)
```

after you try it, try this

```
for x in range(30, 2, -3):
  print(x)
```

- Go back to the directory where you unzipped the library bundle, into the *examples* directory. In that directory, find the file *circuitplayground_light_neopixel.* and copy it to the CIRCUITPY drive.
- Use the "Load" button to bring this file up in the Mu editor.
    - Notice that the code includes the line *while True:* which means the code below this point will loop forever.
    - Notice the *for* loop within the *while True:* This is the main focus for this program. Pay attention to the syntax.
    - Notice the code following the keyword *def* above the *while True:* This marks the beginning of a function. All of the indented lines below the *def* are part of the function. We will discuss functions further is a later lab.
- Save the code as code.py and see what the code does.
- Change the code so that turning the neopixels off and on no longer depends on the light sensor value but is just the series of light 0 on then off, light 1 on then off, up to light 9 on then off, with a blink rate of 1 second. Let that loop within the *while True:* loop.
- Test the operation of your code and demonstrate to the instructor.
- **Comment your code well and submit.** Using the commenting guidelines given in the introduction.

Loop Program 2:

For this program you should start with the working version of Loop Program 1.

- Change this program (be sure to save as a different filename) so that lights 0 through 9 turn on, with a pause of half a second between each light. Then lights 9 through 0 turn off at the same rate. So the last one on is the first to turn off.
- Test the operation of your code and demonstrate to the instructor.
- **Comment your code well and submit.** Using the commenting guidelines given in the introduction.

Loop Program 3:

For this program you should start with the working version of Loop Program 1 again.

- Change this program (be sure to save as a different filename) so that lights 1, then 3, then 5, then 7 and then 9 turn on then off, at a 1 second rate. Then lights 0, 2, 4, 6, 8 turn off and on at the same rate. Place this code within the *while True:* loop.
- Test the operation of your code and demonstrate to the instructor.
- **Comment your code well and submit.** Using the commenting guidelines given in the first lab.