

Chapter 4 - Loops

Loops

Lab - Loops

Lab - For Loop

Tutorial Lab 1: Using the for loop

Use the text editor open in the left pane, and enter the following code:

```
for x in range(0, 10):  
    if x % 2 == 0:  
        print("Even")  
    else:  
        print("Odd")
```

Code Visualizer

1. The for loop runs through all the values from 0 to 10, specified in the range command.
2. Then a comparison is made using an if statement.
3. If x modulo 2 results in 0, then print Even.
4. If x modulo 2 is any other number that is not 0, then print Odd.

Lab - Range

Tutorial Lab 2: for loop for calculating powers (exponents)

Use the text editor open in the left pane, and enter the following code:

```
exp = 4
base = 2
result = 1
for x in range(exp):
    result = base * result

print(result)
```

Click [here](#) to see the loop run in the code visualizer.

This is a crude way of calculating powers (in the example above it would be 2 to the power of 4), but it demonstrates the use of a for loop in an interesting way.

- 1) First off, we create `exp` (represents the exponent) and assign it the value of 4.
 - 2) Then we create `base` and assign it the value of 2.
 - 3) `result` (the final value of our program) is assigned the value of 1.
 - 4) The for loop will run `exp` number of times. In this case, it will run 4 times
 - 5) The calculation takes `result` and multiplies it by `base` and assigns it back to itself. This is essentially how the power function works, you are multiplying a value by itself a specified number of times.
- * When `x` is 0, `result` is assigned $2 * 1$, which is 2.
 - * When `x` is 1, `result` is assigned $2 * 2$, which is 4.
 - * When `x` is 2, `result` is assigned $2 * 4$, which is 8.
 - * When `x` is 3, `result` is assigned $2 * 8$, which is 16.
 - * Finally, `result` is printed to the screen.

Lab - While Loop

Tutorial Lab 3: The while loop

Use the text editor open in the left pane, and enter the following code:

```
counter = 0
while(counter < 10):
    print(counter)
    counter = counter + 1
print('while loop ended')
```

Code Visualizer

1. This loop will run as long as `counter` is less than 10.
2. Each time the loop runs, the value of `counter` is printed to the screen.
3. The value of `counter` is also incremented by 1.
4. At the end, a statement is printed to the screen, indicating the while loop has ended.
5. Recall that the while loop must have an exit condition. By incrementing the `counter` variable, we ensure that the loop will eventually end. If you do not increment `counter` in this loop, you will create an endless loop because `counter` will never reach 10 or greater.

Lab - Break Statement

Tutorial Lab 4: Breaking from the while loop

Use the text editor open in the left pane, and enter the following code:

▼ What does `inp = input('>')` mean?

The `input` command will wait for the user to type some information into the terminal and press `return`. `input` takes a string argument which will be displayed for the user. The information entered by the user is stored in the variable `inp`. All information entered for the `input` command will be stored as a string (even if you type a number).

```
result = 0

while True:
    print('Enter numbers to sum, enter q to quit')
    inp = input('> ')
    if inp != 'q':
        inp = int(inp)
        result = result + inp
    else:
        print(result)
        break
```

Open the [visualizer](#) if you want to see how the loop works.

1. Create the variable `result` and set its value to `0`. `result` will hold the total of the summation.
2. Next we set up a while loop with `True`. We do this because we want the loop to run until the user enters `q`. We don't know how long this will take, so we can limit the loop to a certain number of iterations.
3. We prompt the user to enter a number and use the built-in Python function `input`. `>` will appear, indicating that the user is to enter a value on the keyboard.
4. Next we assign the value the user enters to the `inp` variable.
5. Check to see if the value the user entered is `q`, for quit
6. If not, we convert the value to an integer with the `int()` command. This is required because the `input()` function returns a string value (you

cannot add a string and an integer).

7. The `result` variable is updated to contain its current value plus the value entered by the user.
8. The loop continues accepting values and summing until the user enters the letter `q`. At that point, we step into the `else:` clause. We print the value of `result`, and then exit the loop with the `break` command.

Lab Challenge - Chessboard

Loops Challenge

For this challenge, you will output a pattern that resembles a chessboard by using the letter x and o. The pattern needs to be an 8 x 8 matrix with alternating x and o in each row. Here is the catch, you must also ensure that alternate rows start with a different letter than the previous row.

Here is the required output.

```
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
XOXOXOXO
OXOXOXOX
```

You should make use of a looping structure in combination with a decision structure to achieve this result. Also, by default, Python's `print()` function will add the newline character at the end of each `print()` function so to ensure that this newline does not create a long, single column output of the letters, use this syntax for your `print()` functions, `print('X', end='')`. Note the use of `, end=''` after the letter to output.