Lesson 6

Introduction to Loops

When we are writing code, sometimes we would like to **repeat certain blocks of code multiple times** until some condition is met (the user presses a key, it has run up to a certain limit, we find a desired substring when looking through a string etc.). Loops are used to save you time so you don't have to write the same piece of code over and over again. The two most commonly used types of loops in coding are **while loops** and **for loops**.

While Loops

The way in which while loops behave are quite similar to that of if statements. They are pieces of programming logic followed by a code block. The while loop will execute the contents of its code block while **a certain condition is true**. Try out the example below!

```
count = 0 # initialise count to 0

while count < 10: # while count is less than 10
    print(count) # print count

    count += 1 # increment count

print("The while loop executed " + count + " times") # prints out "The while loop executed 9 times"
```

- Change the value of count. What happens?
- Change the 10 in the condition. What happens?
- Change count += 1; to: count += 2; What happens?

So we can summarize the while loop below:

```
# initialise the counter

while # condition is true:
    # execute the code block
    # iterate the counter

# do something new, outside loop
```

Tasks:

1. Print out all of the **odd** numbers **from 0 up to 30** using a while loop.
2. Print out all of the **even** numbers **from 25 down to 0** using a while loop.
3. Generate and print a random number **between 0 and 100** while the random number is **not equal** to 50.

Hint: In Python, when comparing values to see if they are not equal we use "!=". This is similar to "==" we use when checking to see if values are equal.

For Loops

for loops are the other type of loops we will use a lot in coding. for loops are very similar to while loops in the way they behave, however, they are structured a bit differently. Unlike while loops, for loops include everything required to count the loops within the loops itself (we don't need external variables).

```
for i in range(0, 10): # "i" increases by one until it reaches 10
    print(i) # prints numbers 0 - 9
```

The code block above functions the same way as the first while loop example we looked at. The variable "i" acts as the counter in our for loop and is incremented on each loop, until it reaches the limit stated in the ***range()*** function (limit is 10 in this case).

Note: You can use whatever variable you want in for loops but usually we like to use "i" as our counter. The reason for this is because "i" is used a lot in maths.

The counter in a for loop is incremented by one by default. If we wish the counter to increase by something else then all we need to do is add the value we wish to increment by to our ***range()*** function.

```
for i in range(0, 10, 2): # will print every 2nd number until it reaches 10
    print(i) # prints numbers 0, 2, 4, 6, 8
```

Tasks:

1. Print out all of the **odd** numbers **from 0 up to 20** using a for loop.
2. Print out all of the **even** numbers **from 18 down to 0** using a for loop.
3. Generate and print a random number **between 0 and 120** twenty times using a for loop.

Iterate Lists with For Loops

for loops can also be useful to go through a list of objects that we have declared.

Without For Loops	**With For Loops**
countyList = ["Laois", "Roscommon", "Dublin", "Cork"] print(countyList[0]) # prints Laois print(countyList[1]) # prints Roscommon print(countyList[2]) # prints Dublin print(countyList[3]) # prints Cork	countyList = ["Laois", "Roscommon", "Dublin", "Cork"] for item in countyList: # for each county in countyList print(item) # prints out each county, one at a time

In the above example, when using a for loop, it initially stores the first element of the list ("Laois") in the item variable (kind of like "i" in the for loops we have seen previously). It then moves onto the other elements in the list as it loops.

Create Your Own List

In a for loop, we can do more than just print out the contents of an existing list. Another thing we can do is create new lists using the counter in a for loop. **Try out the example below and see what happens!**

```
import random

myList = [] # declare empty list

for i in range(0, 20): # for loop will run 20 times
    myList.append(i * 2) # stores double the current count ("i") in list
    print(myList) # prints the current contents of the list
```

Tasks:

1. Declare a list of **six** integer values.
2. Iterate through the list using a for loop.
3. For each value in the list, **square the value** (multiply it by itself) and add the new, squared value to a **new** list.
4. Print the contents of this new list in each iteration of the for loop.

Calculator Project

Write code for a simple calculator that can do the following:

- Addition
- Subtraction
- Multiplication
- Divide
- Modulo
- Anything else you can think of!

Tasks:

1. Print a message at the beginning, welcoming the user to the calculator.
2. User must input the first number and then input the second number.
3. User must input what operation they wish to perform. Number each operation so the user can input the number of the operation.

```
print("Choose one of the following operations:")
print("1 - addition")
print("2 - subtraction")
print("3 - multiplication")
print("4 - division")
print("5 - modulo")

option = int(input("")) # user inputs number of operation
```

4. If the user has chosen an invalid option, print “Invalid Option”.
5. Or else if the user has chosen a valid option, depending on what option was chosen, perform the operation on the two numbers that the user has inputted.

   Hint: Use if/elif/else statements.

6. Print the result of the operation.

7. User must input whether they want to run the calculator again.

```
repeat = input("Would you like to make another calculation? [yes/no]")
```

Use a while loop to make the calculator repeat the steps above while the answer the user has inputted is yes.

```
while (goOn == "yes"):
    # calculator code
```

Note: Ensure that you declare “repeat” at the beginning of the program and set it equal to “yes” so the while loop will execute the first time around.

Turtle Looping Project

We will now practice our looping skills using Tina the Turtle.

```
import turtle

tina = turtle.Turtle()
tina.shape('turtle')

tina.left(90)
tina.forward(20)
tina.write("What colour am I now?")

tina.forward(20)
tina.color("blue")
tina.write("What colour am I now?")

tina.forward(20)
tina.color("purple")
tina.write("What colour am I now?")

tina.forward(20)
tina.color("green")
tina.write("What colour am I now?")
```

The above code will result in the image below:

Tasks:

1. Write the code in the box above.
2. Change the code to use a loop to write the message on the screen.

Hint: Since we know how many times we have to loop, a for loop would be best suited for the task.