Control Flow

We have so far dealt with a number of data structures in Python. We will now look at techniques for controlling the order of execution of a program.

The Range function

Sometimes we will need to iterate over a sequence of numbers. The built-in function range() comes in handy. It generates a sequence of numbers.
Type this code and click run

```
2   x = range(10)
3   print(x)
```

We can iterate through the range of numbers contained in x as we would in a normal list. Type this and check the output.

```
5 ▾ for y in x:
6       print(y)
```

From the output you will see the range has 10 numbers from 0 to 9.
We can also give the range a starting point like this.

```
8   print("----------")
9   a = range(2, 10)
10 ▾ for x in a:
11      print(x)
```

If statement

The if statement checks if an operation is True or False. If you remember in the early classes we learnt about comparison operations. They return True or False. Let's say we want to print numbers between 0 and 10 that are divisible by 3. We can do it as follows:

```
13  print("-------")
14  x = range(10)
15  for y in x:
16      if y%3==0:
17          print(y)
```

The comparison operation here is y%3==0. It will be True if a number is divisible by 3. The code below an if statement is executed if the if statement returns true.

Do the above and confirm the result.

As an exercise I would like you to print all the numbers between 0 and 10 that are not divisible by 3.

Else Statement

We can combine the if statement with an optional else statement. The code inside else gets executed if the preceding if statement returns False.
As an example let's say we want to print the numbers numbers between 0 and 10 that are divisible by 2 and a statement if they are not. We can do it as follows. Type this in the interpreter and check the output

```
20  print("------")
21  n = range(10)
22 ▾ for x in n:
23 ▾     if x%2==0:
24           print("{} is divisible by 2".format(x))
25 ▾     else:
26           print("{} is not divisible by 2".format(x))
```

As an exercise, using the same format as above, use a range of numbers between 0 and 20. The if statement should check if a number is divisible by 5.

Elif statement

The elif statement allows us to do more than one comparison.
In this example, we first check if a number is divisible by 2 then check if it is divisible by 3. The else part is executed if both previous statements return False. You can have as many elif statements as you want. Type this in the interpreter and check the output.

```
print("------")
n = range(10)
for x in n:
    if x%2==0:
        print("{} is divisible by 2".format(x))
    elif x%3==0:
        print("{} is divisible by 3".format(x))
    else:
        print("{} is not divisible by 2 or 3".format(x))
```

As an exercise, using the same format as above, use a range of numbers between 0 and 30. The if statement should check if a number is divisible by 5, then use elif to check if it's divisible by 7. Share the output.

While, continue and break control flow

While loops

The while loop continues to iterate as long as the set condition remains true. Here is an example. Try it yourself on the interpreter.

```
2   x = 1
3 ▼ while x < 10:
4       print(x)
5       x += 1
```

In this example we initialize x with a value of one and in the for loop we print x and increment it with 1 for each iteration. The loop will stop when the value of x is 9.

Try this other example.

```
2   x = 100
3 ▼ while x >= 30:
4       print(x)
5       x -= 10
```

In this example we keep printing x and subtracting 10 as long as x is greater than or equal to 30.

Break Statement

The break statement allows us to stop the while loop even if the set condition is true. Let us modify our previous example and add a break statement. Type this in the interpreter and share the output.

```
#python 3.7.1
x = 100
while x >= 30:
    print(x)
    if x < 50:
        break
    x -= 10
```

You will see from the output that the moment x becomes 40 the iteration stops and we break out of the loop even though the condition x >= 30 is still true.

I would like you to edit this while loop using a break statement to stop when x is divisible by 17.

```
x = 1
while x < 50:
    print(x)
    x += 1
```

Continue Statement

The continue statement skips the current operation and goes to the next iteration in the loop. It's easy to understand with an example;
Try this in the interpreter

```
#python 3.7.1
x = 0
while x < 10:
    x += 1
    if x%2 == 0:
        continue
    print(x)
```

In this example, everytime the value of x is divisible by 2 we skip the remainder of the current loop which is the print statement in this case. So we will only print the odd numbers.

Try this other example;

```
#python 3.7.1
x = 1
while x < 50:
    x += 1
    if x%7 != 0:
        continue
    print(x)
```

Here the print statement will only be executed when the value of x is divisible by seven. Any other time the print statement is not reached.

Exercise: Using while, if and continue statements, print all the even numbers between 0 and 50.