Objective 6:

Understand counter controlled iterations

In this objective you learn how to get code to repeat itself a given number of times without having to enter lots of repeating statements.

Tasks

1. Try entering the following commands and see what happens:

```
for counter in range(5):
TAB ----> print("This is how you get code to repeat")
```

2. Try changing the number 5 in the first line as shown below and see what happens.

3. Modify the code as shown below. Run the program and see what happens.

```
for counter in range(7,12):
TAB ----> print("The counter is",counter)
```

4. Enter the following commands and see what happens:

```
word="Hello"
for counter in range(0,len(word)):
      print("Letter",counter,"is",word[counter])
```

Note how we can now loop through all the letters in a word by combining an iteration with string manipulation commands. Also note how commands inside an iteration must be indented, but also make the block of code easier to read.

Objective 6: Key learning points

Counter controlled iterations

- An **iteration** is a section of code that is repeated.
- Iterations are also known as **loops**.
- Counter controlled iterations are used when you want to iterate a known number of times.
- Counter controlled iterations use a variable which can be used within the code to know how many times the code has been run.
- Code must be **indented** within an iteration.
- It would be good practice to comment code before an iteration to explain what is happening.
- **Incrementing** a variable increases its value by one.
- **Decrementing** a variable decreases its value by one.
- One 'for' loop can be put inside another 'for' loop. This is known as **nesting**.

Objective 6: Key words

For…

Example code:
```
For counter in range(0,5):
    print(counter)
```

Executes code within the 'For' structure a known number of times. Counter is the variable that is counting how many times the code has iterated. 0 is the starting value (default). 5 is the stopping value. The starting value does not need to be specified if zero. E.g. `For counter in range(5)`

A third parameter can be used to specify the step. For example to count from 0 to 10 in twos:

```
for counter in range(0,10,2):
    print(counter)
```

By default this is 1 and therefore step 1 is not necessary. To count backwards, a negative value can be used for step. E.g.

```
for counter in range(10,0,-1):
    print(counter)
```