4 - Iteration

### General Teacher Notes

#### What is Iteration?

This week is focused on **iteration,** the code that makes the computer repeat certain lines of code. Iteration is often more commonly known as **looping.**

A loop repeats the code inside it. There are several different types of loop, but in this course we are going to learn about **conditional,** or **while loops.**

A while loop repeats the code inside it **while** the condition is true. When the computer gets to the start of the loop it checks the condition. If the condition is **true** the computer enters the loop and runs the code inside it. If the condition is **false,** the computer skips past the loop and carries on with the rest of the program.

When the computer gets to the end of the loop it goes back and checks the condition again. If the condition is still true it goes back into the loop and runs the code inside again. This keeps happening until the condition changes to become false.

See the teacher notes for task 1 for a more concrete example.

To summarise, this code should be written like this:

*while condition:*

*Code to repeat if condition above is true.*

*Code to run once condition is false and the program moves on.*

#### Iteration - Key Concepts/Misconceptions

Not all code has to be inside a loop - only put the steps you want to repeat in there. Students often have trouble identifying these steps. Question them to help them break down the problem into individual instructions and then identify which ones need repeating.

Iteration runs ALL of the code inside the loop - it doesn’t skip some like selection does. Later units will cover how to combine the two techniques.

### 1 - Iteration with a condition

TEACHER NOTES

Task 1

This task example is in the form of a quiz which works in the following way:

* The user is asked a question and their response is stored in a variable.
* A loop is started with a condition that can be explained as ‘while the user gets the answer wrong’ - this is a typical use of the not equal to operator. It often helps to explain to students that, for this sort of problem, the loop should run while the input is NOT what they want/correct.
* Code inside the loop is indented, just like selection.
* Inside the loop, the user is given an error message - not essential but good practice for usability.
* Inside the loop the user is given another chance to input - **this is essential!** Missing this step means that the user never has a chance to change their answer. Therefore the loop condition will never change from true to false. This creates an infinite loop that never ends. Students often make this error.
* Getting the answer right breaks the loop as the condition is no longer true. The computer moves on to the code after the loop, in this case a congratulations message.

Task and instructions - <https://repl.it/@MrAColley/41-Introducing-While>

Example solution - <https://repl.it/@MrAColley/41-Introducing-While-Example-solution>

# Task 1 - Add comments to explain which code is in the loop and will be repeated.

# Explain the circumstances in which the loop will run.

answer = input("What is the capital of France?")

while answer != "Paris":

print("Incorrect! Try again.")

answer = input("What is the capital of France?")

print("Correct!")

# Task 2

# Add a line of code to the loop that outputs an 'incorrect' message to the user and tells them what to do again.

# Add a line of code to the loop that outputs an 'incorrect' message to the user, tells them what to do again and lets them re-input their number

# Add a line of code AFTER the loop that ouputs a thank you message and repeats the number entered back to the user. Example - 'Thank you, you typed 7'

num1 = int(input("Enter a number 10 or less"))

while num1 > 10:

# Task 3

# Write a program that stores a secret number in a variable (you decide the number and the name of the variable)

# The user has to guess the secret number, the program should loop until they get it right.

# Once the user has guessed correctly they get a congratulations message

### 2 - Iteration With A Counter

TEACHER NOTES

Counters are **really useful** when combined with loops. When using a counter with a while loop there are a couple of points to remember:

* Create the counter variable and assign it a start value **outside** the loop.
* Increment (or decrement if you’re counting down) the counter **inside** the loop. Not doing this will create an infinite loop. +=1 increments by 1. -=1 decrements by 1.

I have always called the counter variable ‘counter’ here as I find that students understand this more readily. Traditionally in computing, this variable will be called ‘i’ which stands for index. You will probably see i used for this purpose if you look up other examples online.

Task and instructions - <https://repl.it/@MrAColley/42-While-loops-with-counters>

Example solution - <https://repl.it/@MrAColley/42-While-loops-with-counters-Example-Solution>

### Homework Task - Cubes Cubes Cubes

Example solution - <https://repl.it/@MrAColley/43-Homework-Cubes-Example-solution>

The cubed number sequence starts: 1, 8, 27, 64, 125.

Write a program that:

* Asks the user to input a number.
* Display N numbers in the cubed sequence according to user input.