**Recap:**

1. Different Code Sequences

**Learning Outcomes:**

1. Double For Loops
2. Looping through multiple terms

**Explanation Points:**

* Explain why we want to loop through two variables
* Showcase the effect of looping through the two variables
  + Get the student to recognize how the loops flow, especially between the 2 for loops

**Breakdown of Lesson Plan:**

|  |  |
| --- | --- |
| Recap Lesson 8 Quiz | 20 min |
| Lesson 9.1 | 45 min |
| Lesson 9.2 | 20 min |
| Lesson 9.3 | 25 min |
| Lesson 9.4 | 25 min |
| Lesson 9.5 | 20 min |
| Lesson 9 Quiz | 25 min |

*\*Note: There is a high chance of student not being able to complete on time.*

**Lesson 8 Recap:**

Study the 2 different for-loops below and answer the questions.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nested for-loops   |  |  | | --- | --- | | ***1*** | **counter1 = 5** | | ***2*** | **counter2 = 3** | | *3* | for i in range(3): # LOOP 1 | | *4* | counter1 = counter1 + 1 | | *5* | print(counter1, counter2, “LOOP1”) | | *6* | for j in range(3): # LOOP 2 | | *7* | counter2 = counter2 + 1 | | *8* | print(counter1, counter2 , “LOOP2”) | | *9* | print(counter1, counter2) | | 2 for-loop   |  |  | | --- | --- | | ***1*** | **counter1 = 5** | | ***2*** | **counter2 = 3** | | *3* | for i in range(3): # LOOP 1 | | *4* | counter1 = counter1 + 1 | | *5* | print(counter1, counter2, “LOOP1”) | | *6* | for j in range(3): # LOOP 2 | | *7* | counter2 = counter2 + 1 | | *8* | print(counter1, counter2 , “LOOP2”) | | *9* | print(counter1, counter2) | |  |

Question

Nested for-loop, how many times Line 7 is executed? \_\_\_\_\_\_

2 for-loop, how many times Line 7 is executed? \_\_\_\_\_\_

Nested for-loop, how many times Loop1 runs? \_\_\_\_\_\_

2 for-loop, how many times Loop2 runs? \_\_\_\_\_\_

Nested for-loop, what is the value of ***counter1*** at the end? \_\_\_\_\_\_

2 for-loop, what is the value of ***counter2*** at the end? \_\_\_\_\_\_

**Lesson 9.1**

**In the last lesson, we had seen several different code sequences with for loops.**

**In this lesson, we will dive in to one of them called a nested loop. (Seen in Lesson 8.5). In Python, sometimes we need multiple for loops to help us loop through varying values.**

**We utilize nested loops when we need to change 2 or more variables at a time. Here is an example:**

**Recap**

|  |  |
| --- | --- |
| ***1*** | **for num1 in range(3):** |
| ***2*** | **for num2 in range(3):** |
| ***3*** | **print(“num1=”,num1, “, num2=”, num2)** |

**Alternatively**

|  |  |
| --- | --- |
| ***1*** | **print(“num1\tnum2”)** |
| ***2*** | **for num1 in range(3):** |
| ***3*** | **for num2 in range(3):** |
| ***4*** | **print(num1, “\t”, num2)** |

**Write the program above and observe the output.**

**Draw a table to show the values of num1 and num2 in the space below. Header for the table has been given. You may use the space on the right to make remarks on the table.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Num1** | **Num2** | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |

**Notice that the values of num1 and num2 changes with each iteration. There should be 9 entries in the table.**

**Let us explore more features of nested loops.**

**Lesson 9.1**

Let us now explore the differences between a nested for loop and 2 for loops.

|  |  |
| --- | --- |
| ***Nested*** *for loop* | ***2 for loop*** |
| |  |  | | --- | --- | | *1* | for num1 in range(5): | | *2* | print(num1) | | *3* | for num2 in range(5): | | *4* | print(num2) | | |  |  | | --- | --- | | *1* | for num1 in range(5): | | *2* | print(num1) | | *3* | for num2 in range(5): | | *4* | print(num2) | |

Which Lines are different?

|  |
| --- |
|  |

How are these lines different?

|  |
| --- |
|  |

How many Lines of output for the Nested for loop? \_\_\_\_\_

How many Lines of output for the 2 for loop? \_\_\_\_\_

Why are there difference between the 2 codes?

|  |
| --- |
|  |

How do we differentiate between the 2 codes?

|  |
| --- |
| Always look at indentation present |

**Lesson 9.1**

Task 1

|  |  |
| --- | --- |
| *1* | i = 0 # Declaring the initial value of i |
| *2* | j = 0 # Declaring the initial values of j |
| ***3*** | **for num1 in range(5):** |
| ***4*** | **for num2 in range(3):** |
| *5* | i = num1 |
| *6* | j = num2 |
| *7* | print(i, j) |

How many lines of outputs should we expect? \_\_\_\_\_\_\_\_

What is the final value of i? \_\_\_\_\_\_

What is the final value of j? \_\_\_\_\_\_

Task 2

|  |  |
| --- | --- |
| *1* | i = 0 # Declaring the initial value of i |
| *2* | j = 0 # Declaring the initial values of j |
| ***3*** | **for num1 in range(3):** |
| ***4*** | **for num2 in range(5):** |
| *5* | i = num1 |
| *6* | j = num2 |
| *7* | print(i, j) |

How many lines of outputs should we expect? \_\_\_\_\_\_\_\_

What is the final value of i? \_\_\_\_\_\_

What is the final value of j? \_\_\_\_\_\_

**Lesson 9.1**

Task 3

|  |  |
| --- | --- |
| *1* | i = 0 # Declaring the initial value of i |
| *2* | j = 0 # Declaring the initial values of j |
| *3* | for num1 in range(3): |
| *4* | for num2 in range(5): |
| *5* | i = num1 |
| *6* | j = num2 |
| ***7*** | **print(i, j)** |

How many lines of outputs should we expect? \_\_\_\_\_\_\_\_

What is the final value of i? \_\_\_\_\_\_

What is the final value of j? \_\_\_\_\_\_

Task 4

|  |  |
| --- | --- |
| *1* | i = 0 # Declaring the initial value of i |
| *2* | j = 0 # Declaring the initial values of j |
| *3* | for num1 in range(3): |
| ***4*** | **i = num1** |
| *5* | for num2 in range(5) |
| *6* | j = num2 |
| *7* | print(i, j) |

How many lines of outputs should we expect? \_\_\_\_\_\_\_\_

What is the final value of i? \_\_\_\_\_\_

What is the final value of j? \_\_\_\_\_\_

**Lesson 9.2**

Let us now explore the different types of nested for loop. To make it simple, we will identify the first nested for loop as a static nested for loop and the second nested for loop as dynamic nested for loop.

|  |  |
| --- | --- |
| ***Static*** *nested for loop* | ***Dynamic*** *nested for loop* |
| |  |  | | --- | --- | | *1* | for num1 in range(5): | | *2* | print(num1) | | *3* | for num2 in range(5): | | *4* | print(num2) | | |  |  | | --- | --- | | *1* | for num1 in range(5): | | *2* | print(num1) | | *3* | for num2 in range(num1): | | *4* | print(num2) | |

Example

|  |  |
| --- | --- |
| *1* | i = 0 # Declaring the initial value of i |
| *2* | j = 0 # Declaring the initial values of j |
| ***3*** | **for num1 in range(3):** |
| ***4*** | **for num2 in range(num1):** |
| *5* | i = num1 |
| *6* | j = num2 |
| *7* | print(i, j) |

Variable Table

|  |  |
| --- | --- |
| **i** | **j** |
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 2 | 1 |
| 3 | 0 |
| 3 | 1 |
| 3 | 2 |

Are you able to identify the pattern behaviour in the j column? (Optional)

|  |
| --- |
|  |

**Lesson 9.2**

How many lines of outputs should we expect? **6**

Explanation:

How many times Line 3 is executed? **3 times** because **for num1 in range(3) giving num1 as 0, 1, 2**

How many times Line 4 is executed? **6 times** because for every num1 value in range(3),

Line 4 is executed **for num2 in range(num1)**

i.e. for num1 in range(0) 🡪 0 times

for num1 in range(1) 🡪 1 time

for num1 in range(2) 🡪 2 times

Since Line 7 is located inside the 2nd for loop, it will be executed **6 times.**

In order to get the final value of i & j, use the code tracing experience from Module 2 Lesson 8.

What is the final value of i? **3**

What is the final value of j? **2**

**Lesson 9.2**

Task 1

|  |  |
| --- | --- |
| *1* | i = 0 # Declaring the initial value of i |
| *2* | j = 0 # Declaring the initial values of j |
| ***3*** | **for num1 in range(5):** |
| ***4*** | **for num2 in range(num1):** |
| *5* | i = num1 |
| *6* | j = num2 |
| *7* | print(i, j) |

How many lines of outputs should we expect? \_\_\_\_\_\_\_\_

What is the final value of i? \_\_\_\_\_\_

What is the final value of j? \_\_\_\_\_\_

Task 2

|  |  |
| --- | --- |
| *1* | i = 0 # Declaring the initial value of i |
| *2* | j = 0 # Declaring the initial values of j |
| ***3*** | **for num1 in range(1,5):** |
| ***4*** | **for num2 in range(num1):** |
| *5* | i = num1 |
| *6* | j = num2 |
| *7* | print(i, j) |

How many lines of outputs should we expect? \_\_\_\_\_\_\_\_

What is the final value of i? \_\_\_\_\_\_

What is the final value of j? \_\_\_\_\_\_

**Lesson 9.2**

Task 3 – If Line 3 is replaced with **for num1 in range(1,5)**

|  |  |
| --- | --- |
| *1* | i = 0 # Declaring the initial value of i |
| *2* | j = 0 # Declaring the initial values of j |
| ***3*** | **for num1 in range(1,5):** |
| *4* | for num2 in range(num1): |
| *5* | i = num1 |
| *6* | j = num2 |
| *7* | print(i, j) |

How many lines of outputs should we expect? \_\_\_\_\_\_\_\_

What is the final value of i? \_\_\_\_\_\_

What is the final value of j? \_\_\_\_\_\_

Task 4 – Can you guess?

|  |  |
| --- | --- |
| *1* | i = 0 # Declaring the initial value of i |
| *2* | j = 0 # Declaring the initial values of j |
| ***3*** | **for num1 in range(1,5):** |
| *4* | for num2 in range(0): |
| *5* | i = num1 |
| *6* | j = num2 |
| *7* | print(i, j) |

How many lines of outputs should we expect? \_\_\_\_\_\_\_\_

What is the final value of i? \_\_\_\_\_\_

What is the final value of j? \_\_\_\_\_\_

**Lesson 9.3**

In this section, we will use for-loops with strings together with the concept of string slicing. In the example below, the variable *str1* contains the value “ABC0”. The last character will be “0” to denote the end of string.

Example

|  |  |
| --- | --- |
| *1* | str1 = “ABC0” |
| *2* |  |
| *3* | for num1 in range(len(str1)): |
| *4* | for num2 in range(num1): |
| *5* | print(str1[num2:num1]) |

What is the value of *len(str1)*? **4**

*for num1 in range(len(str1))* what values will be assign to num1? **0,1,2,3**

How many lines of output do we expect? **6**

Will num1 be smaller than num2? **Yes**

If Line 5 is changed to print(str1[num1:num2]), will we see any output? **No**

Why do we not see the last character ‘0’ ?

|  |
| --- |
|  |

**Lesson 9.3**

Task 1:

What about for this case?

|  |  |
| --- | --- |
| *1* | str2 = “logic!” |
| *2* |  |
| *3* | for num1 in range(len(str2)): |
| *4* | for num2 in range(num1): |
| ***5*** | **print(string[num2:num1])** |

What is the value of *len(str2)*? \_\_\_\_\_\_

*for num1 in range(len(str2))* what values will be assign to num1? \_\_\_\_\_\_\_\_

How many lines of output do we expect? \_\_\_\_\_

Will num1 be smaller than num2? \_\_\_\_\_\_\_\_

Task 2:

What about for this case?

|  |  |
| --- | --- |
| *1* | str3 = “logic!” |
| *2* |  |
| *3* | for num1 in range(len(str3)): |
| *4* | for num2 in range(num1): |
| ***5*** | **print(string[num1:num2])** |

What is the value of *len(str3)*? \_\_\_\_\_\_

*for num1 in range(len(str3))* what values will be assign to num1? \_\_\_\_\_\_\_\_

How many lines of output do we expect? \_\_\_\_\_

*# An empty line printed is considered a line of output*

Will num1 be smaller than num2? \_\_\_\_\_\_\_\_

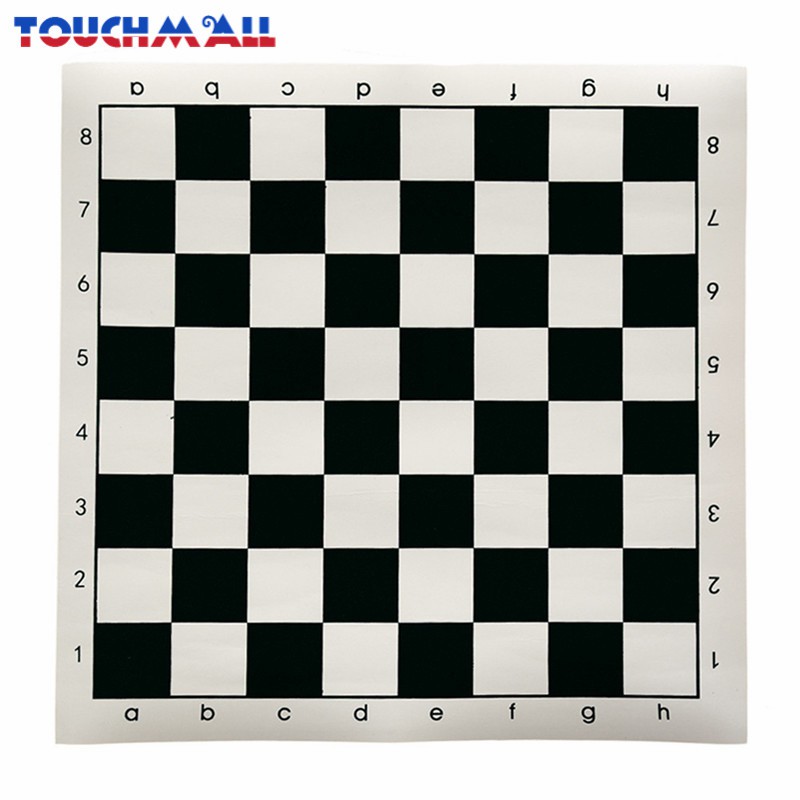
What are the differences between Tasks 1 and 2?

|  |
| --- |
|  |

**Lesson 9.3**

Task 3:

In a chessboard, there are 2 main axes, one that is represented by numbers, and another that is represented by alphabets. In order to identify a certain position, we can call out the alphabet followed by the number as follows “A1”, “B2” and so on..



Using the knowledge of nested loops, try writing all the possible positions on the chessboard as shown in the image above. Complete Line 3, 4, 5.

“A 1”, “B 1”, …

|  |  |
| --- | --- |
| *1* | chess = “ABCDEFGH” |
| *2* |  |
| ***3*** | **for letter1 in** |
| ***4*** | **for num1 in range(** |
| ***5*** | **print(** |

How many positions are there in total? \_\_\_\_\_\_\_

Try adding in additional variables to help you automatically count all the positions.

**Lesson 9.4**

Example:

We can apply for-loops in Mathematics to create good visualization tables. For instance, we can create an additional table of **ones** to have the following output.

Output:

|  |  |
| --- | --- |
| *1* | 1 + 1 = 2 |
| *2* | 2 + 1 = 3 |
| *3* | 3 + 1 = 4 |
| *4* | 4 + 1 = 5 |
| *5* | 5 + 1 = 6 |

In order to create the above output, we need to see the pattern above and convert it into variables - x + y = z

How do we get x? **x** can be obtained using a for loop – **for x in range(1,6)**

How do we get y? Since it is an additional table of **ones** – we can just let **y = 1**

How do we get z? **z** can be calculated from x and y – **z = x + y**

If so, we can write the code as follows.

|  |  |
| --- | --- |
| *1* | y = 1 |
| *2* | for x in range(1,6): |
| *3* | z = x + y |
| *4* | print(x, “+”, y , “=”, z) |

**Lesson 9.4**

Task 1:

Create a multiplication table. In school, we all learn what a multiplication table is. Using your understanding of **for-loops**, write a code to print out the table of a multiplication table of 5 as shown below. You need not use the input() function.

Output:

|  |  |
| --- | --- |
| *1* | 1 x 5 = 5 |
| *2* | 2 x 5 = 10 |
| *3* | 3 x 5 = 15 |
| *4* | 4 x 5 = 20 |
| *5* | 5 x 5 = 25 |

Task 2:

What is factorial? In Mathematics, factorial is the product of all positive integers less than or equal to a given positive integer and denoted by that integer and an exclamation point ‘!’.

i.e.

1! = 1

2! = 1 \* 2 = 2

3! = 1 \* 2 \* 3 = 6

4! = 1 \* 2 \* 3 \* 4 = 24

5! = 1 \* 2 \* 3 \* 4 \* 5 = 120

Create a **factorial table**. Using your understanding of **nested for-loops**, write a code to print out the table of a factorial table of 5 as shown below. You need not use the input() function.

Output:

|  |  |
| --- | --- |
| *1* | 1! = 1 |
| *2* | 2! = 2 |
| *3* | 3! = 6 |
| *4* | 4! = 24 |
| *5* | 5! = 120 |

**Lesson 9.5**

**In this section, we will be practising more on using the for-loop to getting a certain number pattern before generating the patterns.**

Example:

Using the concepts of for-loop, generate the number pattern.

|  |  |
| --- | --- |
| *1* | for i in range(1,6,2): |
| *2* | for j in range(1,i+1,2): |
| *3* | print(j\*str(j)) |

Output:

|  |  |
| --- | --- |
| *1* | 1 |
| *2* | 1 |
| *3* | 333 |
| *4* | 1 |
| *5* | 333 |
| *6* | 55555 |

Using the concepts of for-loop, generate the outputs.

|  |  |
| --- | --- |
| *1* | for i in range(1,6,2): |
| *2* | for j in range(1,i+1,2): |
| *3* | print(j \* “ox”) |

|  |  |
| --- | --- |
| 1 | ox |
| 2 | ox |
| 3 | oxoxox |
| 4 | ox |
| 5 | oxoxox |
| 6 | oxoxoxoxox |

**Lesson 9.5**

Task 1:

Using the concepts of for-loop, **complete the code** to give the generated number pattern.

|  |  |
| --- | --- |
| *1* |  |
| *2* |  |
| *3* |  |
| *4* |  |
| *5* |  |
| *6* |  |

Output:

|  |  |
| --- | --- |
| *1* | 1 |
| *2* | 2 |
| *3* | 22 |
| *4* | 3 |
| *5* | 33 |
| *6* | 333 |

Task 2:

Using the concepts of for-loop, **complete the code** to give the generated number pattern.

|  |  |
| --- | --- |
| *1* |  |
| *2* |  |
| *3* |  |
| *4* |  |
| *5* |  |
| *6* |  |

Output:

|  |  |
| --- | --- |
| *1* | 1 |
| *2* | 22 |
| *3* | 4444 |
| *4* | 88888888 |

**Lesson 9.5**

Task 3:

Using the concepts of for-loop, **write the code** to give the generated output.

|  |  |
| --- | --- |
| *1* |  |
| *2* |  |
| *3* |  |
| *4* |  |
| *5* |  |
| *6* |  |

Output:

|  |  |
| --- | --- |
| *1* | ox |
| *2* | ox |
| *3* | oxoxox |
| *4* | ox |
| *5* | oxoxox |
| *6* | oxoxoxoxox |

Task 4:

Using the concepts of for-loop, **write the code** to give the generated output.

|  |  |
| --- | --- |
| *1* |  |
| *2* |  |
| *3* |  |
| *4* |  |
| *5* |  |
| *6* |  |

Output:

|  |  |
| --- | --- |
| *1* | ooo |
| *2* | oo |
| *3* | o |
| *4* | oo |
| *5* | o |
| *6* | o |

**End of Lesson 9 Quiz:**

Question 1a

Write a code using **nested for loop** that outputs the following numbers: 0, 1, 0, 1, 2, 0, 1, 2, 3

Output

|  |  |
| --- | --- |
| *1* | 0 |
| *2* | 1 |
| *3* | 0 |
| *4* | 1 |
| *5* | 2 |
| *6* | 0 |
| *7* | 1 |
| *8* | 2 |
| *9* | 3 |

Question 1b

Edit this program to print the following design.

Output

|  |  |
| --- | --- |
| *1* |  |
| *2* | O |
| *3* |  |
| *4* | O |
| *5* | OO |
| *6* |  |
| *7* | O |
| *8* | OO |
| *9* | OOO |

**End of Lesson 9 Quiz:**

Question 2:

In a warehouse, there are 4 main storage locations, called A, B, C and D. In each location, there are a total of 10 possible positions for storage, named accordingly as 0, 1, 2, …, 9. Write a program that prints all the possible storage position for this facility. Your answer should be in the format:

“A0”, “A1”, …

Question 3:

Without changing your program in Question 2, find the total number of storage positions in the warehouse. Print your answer in the format:

“There are \_\_\_ storage position in the warehouse.”

Use a variable to count the number of storage positions.