**4.3 Nested loops**

A ***nested loop*** is one loop inside of another. In other words, the action of a loop contains another loop. Frequently, a nested loop is a nested **for** loop, in which one **for** loop is part of the action of another **for** loop.

In the following example, a list of words is created. A **for** loop loops through all of the words, and then for each word, another **for** loop loops to print each character followed by a space.

| *wordlist = ['hello', "hi", 'ciao'] for myword in wordlist:*  *for c in myword:*  *print(c, end = ' ')*  *print()*  *print("That's it!")* |
| --- |

h e l l o

h i

c i a o

That's it!

The first loop is called the ***outer loop***. The action of the outer loop consists of two statements: • a loop to print each character and a space

• a **print()** to move the cursor down for the next word

The second loop, that loops over the characters, is called the ***inner loop***. The action of the inner loop is a single **print** statement. After the nested loop, there is another **print** statement to print “That’s it!”. The outer loop iterates over the words in the list. For each word, the inner loop iterates over the characters in that word.

So, the first time in the outer loop the variable *myword* will store ‘hello’. The inner loop variable *c* will then iterate through all of the characters in the variable *myword*, and for each it will print the character and then a space (all on the same line). After the inner loop has completed, the **print()** statement will print a newline character, and that is it for the action of the outer loop. Once this action has completed, the variable *myword* will store the second string in the list, “hi”. The inner loop will then iterate through all of its characters and print each one followed by a space. After both characters have been printed, the **print()** moves the cursor down to the next line. Finally, *myword* gets the value ‘ciao’, and the inner loop prints all of its characters on one line and then a newline at the end. Once the variable *myword* has iterated through all 3 words in the list, the outer loop is done so the code prints “That’s it!” after the nested loop has completed.

The following is an example of a nested loop, in which the outer loop repeats 3 times. Each time the action is executed, the next integer (0 then 1 then 2) is printed, followed by a colon and a space. The inner loop then prints, 5 times, a single ‘\*’. The second argument, end=*''*, keeps all of the \*’s on the same line and does not print anything between them. The **print()**

after the inner loop prints the newline character by default, so each of the 3 actions from the outer loop is on a separate line.

| *for num in range(3):*  *print(f'{num}:', end=' ') for n in range(5):*  *print('\*', end='') print()* |
| --- |

0: \*\*\*\*\*

1: \*\*\*\*\*

2: \*\*\*\*\*

The outer loop specifies that the action will be repeated 3 times. The action of the outer loop consists of 3 statements:

• A **print** statement to print the value of the outer loop variable

• A **for** loop to repeat the action of printing a single ‘\*’ 5 times on the same line • A **print** statement to move the cursor down for the next value of the outer loop variable

Instead of looping 5 times to print a ‘\*’, the following inner loop repeats *num* times, where *num* is the value of the outer loop variable (plus 1, so it repeats once, then twice, then three times).

| *for num in range(3):*  *print(f'{num}:', end=' ') for n in range(num+1): print('\*', end='') print()* |
| --- |

0: \*

1: \*\*

2: \*\*\*

If instead of printing 0, 1, and 2 in the beginning of the lines, we wanted 1, 2, and 3, the code would instead look like this:

| *for num in range(3):*  *print(f'{num+1}:', end=' ') for n in range(num+1): print('\*', end='') print()* |
| --- |

1: \*

2: \*\*

3: \*\*\*

These were examples of nested **for** loops. Nested loops can also contain **while** loops.

For example, let’s say we want the user to enter 3 positive numbers. That means we will use a **for** loop to repeat the action 3 times. Each time, we will error-check using a **while** loop to make sure that each time a positive number is entered. The algorithm is:

• Loop 3 times

o Prompt the user for a positive number

o While the number is not positive

▪ Print error message

▪ Prompt again

o Print the positive number

Here is an implementation of the algorithm:

| for i in range(3):  number = input('Enter a positive number: ') number = float(number)  while (number <= 0):  print('Please follow directions!') number = input('Enter a positive number: ') number = float(number)  print('Thanks for entering', number)  print() |
| --- |

Enter a positive number: 4

Thanks for entering 4.0

Enter a positive number: -3.3

Please follow directions!

Enter a positive number: 5.1

Thanks for entering 5.1

Enter a positive number: -6

Please follow directions!

Enter a positive number: 11

Thanks for entering 11.0

An extra **print()** was added to the action of the outer **for** loop in order to separate each of the actions of the **for** loop.