Day 4

Lists – For Loops

ORBTRONICS

Objectives

Students should be able to:

- Understand and create nested loops.
- Create and use lists and list operations.
- Use for-each loops.

Nested Loops

Nested Loops

- Nested loops are when you have a loop inside of another loop.
- This is best demonstrated with an example.

Example 1 – Times Table

Source Code

```
a = 1
while a <= 10:
    b = 1
    while b <= 10:
        print(a, 'x', b, '=', a*b)
        b = b + 1
    a = a + 1
```

Sample of Output

```
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
```

Example 1 – Times Table Explanation

- The output shown above is just a sample of the full output.
- The full output will contain all the multiplication answers from 1x1=1 up to 10x10=100.
- The program starts off with a variable “a” which is set to 1 and a while loop which checks if “a” is less than or equal to 10.
- Inside that while loop the variable “b” is set to 1 and another while loop checks if “b” is less than or equal to 10.
- Inside of this second while loop, we display the answer of multiplying “a” by “b” (which is 1x1=1), then we increase “b” by 1.
- The inside while loop then repeats, this time showing (1x2=2).

Example 1 – Times Table Explanation Continued.

- The inside while loop keeps repeating until “b” gets to 10, the calculation 1x10=10 is displayed and “b” increases to 11, ending the inside while loop.
- Since the inside while loop is finished, we reach line 7 of the code where variable “a” is increased from 1 to 2.
- The outside while loop then repeats, setting “b” back to 1.
- The inside while loop causes the calculations 2x1=2, 2x2=4, etc., to be displayed up to 2x10=20.
- This pattern continues until 10x10=100 is displayed.

– Times Table Without Repetition.

– Prime Numbers

See Day 4 Problem Set For Details

Lists

Lists

- So far, all variables and expressions we've seen only represent a single value.
- Be it a single integer, a single float, or a single string.
- Lists allow us to have a variable which contains several values rather than just one.
- We create a list in python using the square brackets and listing out the values in those brackets, separating them by commas.
- We can then do a lot of computations with these lists.

Example 1

Source Code

```
numbers = [7, 5, -1]
print(numbers)
messages = ["Hello", "Goodbye"]
print(messages)
scores = [-3.4, 2.5, 0.1]
print(scores)
```

Output

```
[7, 5, -1]
['Hello', 'Goodbye']
[-3.4, 2.5, 0.1]
```

Explanation

- Here we create 3 lists. The first one is called numbers and contains 3 numbers.
- We then print out the list and as seen in the output, the entire list is printed out.
- The second list is called messages and contains two strings.
- The third list is called scores and contains 3 floats.

Example 2

Source Code

```
list1 = [4]
list2 = []
list3 = [3, -1.2, "Hello"]
x = 5
list4 = [x, x/3, str(x)]
print(list1)
print(list2)
print(list3)
print(list4)
```

Output

```
[4]
[]
[3, -1.2, 'Hello']
[5, 1.6666666666666667, '5']
```

Explanation

- This example shows that a list can contain just a single element, or even no elements at all (called an empty list).
- You can also put different type of values in the same list as seen for list3. This is usually not recommended.
- You can also put variables and expressions as in the list as seen for list4.
- You can also put lists inside of lists (not shown in this example).

Example 3

Source Code

```
list1 = [5, 8, 2]
list2 = [4, 1]
list3 = list1 + list2
list4 = [1, 7] * 3
print(len(list1))
print(list3)
print(list4)
```

Output

```
3
[5, 8, 2, 4, 1]
[1, 7, 1, 7, 1, 7]
```

Explanation

- This example shows that we can concatenate lists together using the plus sign.
- It also shows that you can repeat a smaller list multiple times into a larger list using the asterisk.
- It also shows that you can get the length of the list (number of items) using the len function.

Example 4

Source Code

```
list1 = [1, 4, "Hi", -3, 2.0]
print(list1[0], list1[-1])
print(list1[2:4])
print(list1[::2])
```

Output

```
1 2.0
['Hi', -3]
[1, 'Hi', 2.0]
```

Explanation

- This example shows that lists can be indexed and sliced just like strings.

Example 5

Source Code

```
list1 = [0, 7, -4]
x = list1.pop()
print(x)
print(list1)
list1.append(3)
print(list1)
```

Output

```
-4
[0, 7]
[0, 7, 3]
```

Explanation

- We can also add elements to the list and remove elements from the list.
- The .pop method gives the last element of the list and removes it.
- So in line 2, the -4 is removed from the list and is stored into the variable x.
- The .append method adds an element to the end of the list.
- There are ways to add and remove elements in other parts of the list other than the end.

Example 6

Source Code

```
list1 = [1, 4, "Hi", -3, 2.0]
i = 0
while i < len(list1):
    print(list1[i])
    i = i + 1
```

Output

```
1
4
Hi
-3
2.0
```

Explanation

- This code shows how we can loop through all the elements of a list.
- We have a variable for the index, usually this variable is called “i”.
- While this index is less than the list’s length, we do something to the value at that index. In this case we just print it out.
- We then increase the index by 1 to go to the next element.

– Reverse Repeat Challenge

– Delete All Challenge

See Day 4 Problem Set For Details

For Each Loops

For-Each Loops

- For-each loops are used as shortcuts to going through each element in a list.
- They remove the need of constantly using while loops and indices.
- A for-each loop works by specifying some code that you want to run for each element in the list.

Example 1

Source Code

```
nums = [4, 7, 8, -1]
for num in nums:
    print(num)
```

Output

```
4
7
8
-1
```

Explanation

- The for each loop in Python has the following syntax: the word “for”, an element variable, the word “in”, and a list.
- Inside the for each loop, we place code that we want to run on each element.
- In this example, num is the element variable. Inside the loop we print out num, therefore each element of the list will get printed.

– Sum Challenge

– Count Challenge

See Day 4 Problem Set For Details

Range Function

Range Function

- The range function is used in python to create a list of numbers over a certain range.
- For example, range(7) will produce a list of numbers from 0 to 6. Notice that 7 itself is not included in the list, as the range function stops at the number before the argument. Notice that the list it creates starts at 0 by default.
- If you do not want to start at 0, you can specify what number to start at. For example, range(3, 8) will produce a list of numbers from 3 to 7.
- You can also specify a step size: range(6, 20, 3) will create the following list of numbers: 6, 9, 12, 15, 18.
- Technically, the range function does not create a real list, it creates something called an iterator, which behaves like a list when it is used in a for each loop.

Example 1

Source Code

```
for i in range(6, 20, 2):
    print(i)
```

Explanation

- This code prints out all the even numbers from 6 to 18.
- Do you see why? Do you see why only the even numbers are printed? Do you see why the number 20 is not printed?
- The use of range(6, 20, 2) creates a list of numbers starting at 6, stopping before 20, and increasing by 2. Therefore, the list [6, 8, 10, 12, 14, 16, 18] is produced.
- The for loop prints out each of them.

– Authentication Challenge

See Day 4 Problem Set For Details