5.3.1 Python Loops: for loop

- A for loop in Python is a programming statement that repeats a block of code a fixed number of times.
- The for-loop is always used in combination with an iterable object[^1], like a list or a range.
- The Python for statement iterates over the members of a sequence in order, executing the block each time.

video: What is it and Why do we Use it? | Python For loop Tutorial

Syntax:

```
for item in iterable:
    # code block
```

`iterable` is a sequence of elements such as a list, tuple, dictionary, set, or string. item is a variable that takes on the value of each element in the sequence, one at a time. The code block is executed once for each element in the sequence.

range() function:

- We can use the `range()` function as an iterable in a for loop in Python. The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.
- We can also specify the starting value and the increment value of the sequence using the range() function. For example, range(2, 10, 2) returns a sequence of numbers starting from 2, incrementing by 2, and ending at 8. read more ...

Here are the common usages:

Example #5.8

1. **Single Argument (`stop`)**:

   ```
   range(stop)
   ```

   - Generates numbers from `0` up to, but not including, `stop`.

   ```
   range(5)  # Output: 0, 1, 2, 3, 4
   ```

2. **Two Arguments (`start`, `stop`)**:

   ```
   range(start, stop)
   ```

   - Generates numbers from `start` up to, but not including, `stop`.

   ```
   range(2, 5)  # Output: 2, 3, 4
   ```

3. **Three Arguments (`start`, `stop`, `step`):**

   ```
   range(start, stop, step)
   ```

   - Generates numbers from `start` up to, but not including, `stop`, incrementing by `step`. The `step` can be positive or negative.

   ```
   range(1, 10, 2)  # Output: 1, 3, 5, 7, 9
   range(10, 1, -2) # Output: 10, 8, 6, 4, 2
   ```

Note: `range()` produces an immutable sequence type, which is often used in loops. To get a list of numbers, you can convert the `range` object to a list:

```
list(range(5))  # Output: [0, 1, 2, 3, 4]
```

- Video: The range() Function
- Video: Use of range() in for loop

Example #5.9: Print Numbers from 1 to 5

video

Question: Write a Python program to print the numbers from 1 to 5, using a for loop.

```
for i in range(6):
    print(i)
```

Example #5.10: Printing "Building the future, one line at a time." 5 Times Using a for Loop

Question: Write a Python program to print the string "Building the future, one line at a time." 5 times, using a for loop.

```
for i in range(5):
    print("Building the future, one line at a time.")
```

Example #5.11: Sum of Numbers from 1 to N

Question: Write a python program to calculate the sum of the first N natural numbers using a for loop.

video: Calculate the sum of the first N natural numbers | Python for loop example

Task #5.8: Sum of Numbers in a Range

Description:

Write a Python program that asks the user to input two numbers, a start and an end value, and calculates the sum of all the numbers in that range (inclusive) using a `for` loop.

Input:

- The program should prompt the user to enter two integers:
    1. The start of the range.
    2. The end of the range.

Output:

- The program should display the sum of all the numbers in the given range.

Example:

Input:

```
Enter the start of the range: 1
Enter the end of the range: 5
```

Output:

```
The sum of numbers from 1 to 5 is: 15
```

Requirements:

- Use a `for` loop to iterate through the range.
- Accumulate the sum of numbers using a variable inside the loop.

Task #5.9: Display Even Numbers in a Range

Description:

Write a Python program that asks the user to input two numbers, a start and an end value, and displays all the even numbers between the two values (inclusive) using a `for` loop.

Input:

- The program should prompt the user to enter two integers:
    1. The start of the range.
    2. The end of the range.

Output:

- The program should display all the even numbers in the range in ascending order, one number per line.

Example:

Input:

```
Enter the start of the range: 2
Enter the end of the range: 10
```

Output:

```
2
4
6
8
10
```

Requirements:

- Use a `for` loop to iterate through the range.
- Check if a number is even using the modulus (`%`) operator.

Example #5.11: Print Even Numbers from 2 to 10 and sum of even numbers

Question: Write a Python program to display the even numbers from 2 to 10 and sum of even numbers, inclusive, using a for loop.

```python
sum = 0
for i in range(2,11):
    if i%2 == 0:
        sum +=i
        print(i)
print(f"Sum of even numbers: {sum}")
```

Task #5.11: Display Multiplication Tables**

Description:

Write a Python program that asks the user to input a number and displays its multiplication table up to 10 using a `for` loop.

Input:

- The program should prompt the user to enter an integer value.

Output:

- The program should display the multiplication table for the entered number in the following format:

```
n x 1 = n
n x 2 = 2n
...
n x 10 = 10n
```

Example:

Input:

```
Enter a number: 5
```

Output:

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
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
```

Requirements:

- Use a `for` loop to generate and display the multiplication table.

Example #5.12: String as an iterable

video: String as an iterable

Example #5.13: Lists as an iterable

```
collection = ['python', 5, 'd']
for x in collection:
    print(x)
```

Example #5.14: Loop over Lists of lists

```
list_of_lists = [ [1, 2, 3], [4, 5, 6], [7, 8, 9]]
for list in list_of_lists:
    for x in list:
        print(x)
```

- A **nested loop** is a loop inside another loop. It is a powerful programming technique that can be used to solve a wide variety of problems.

Example #5.15 Nested Loops - Multiplication Tables

```
for i in range(1, 11):
    print(f"Multiplication table of {i}")
    for j in range(1, 11):
        print('%d * %d = %d' % (i, j, i*j))
```

Why We Use Variables in For Loops

In Python, a variable in a `for` loop is used to iterate over a sequence (like a list, tuple, string, or range) and access each element in that sequence one at a time. This variable is often called the "loop variable" or "iterator variable."

Understanding the Role of the Variable:

1. **Accessing Elements**: The loop variable allows you to access each element in the sequence during each iteration of the loop. This makes it possible to perform operations on each element.
2. **Dynamic Assignment**: The loop variable automatically takes the value of the next element in the sequence during each iteration. This saves you from having to manually update the variable's value.
3. **Readability and Simplicity**: Using a loop variable makes the code more readable and easier to understand. You can name the variable in a way that reflects the data it represents, making the code more self-explanatory.
4. **Control Over Iteration**: The loop variable gives you control over the loop's execution. You can use it to control the flow, such as skipping certain elements, breaking out of the loop early, or performing specific actions based on the variable's value.

Example #5.16: Loop Through a List of Fruits in Python

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

In this example, the loop variable `fruit` takes on the value of each element in the list `fruits` during each iteration. The output will be:

```
apple
banana
cherry
```

The variable `fruit` is used to access and print each element in the `fruits` list.

Use of underscore in 'For Loop'

- In Python, an underscore (_) is often used as a variable name in a for loop (or any other context) when the value of the variable is not needed.
- This is a common convention to indicate that the value is intentionally being ignored or discarded.

For example, if you want to repeat an action a certain number of times but don't need to use the loop variable, you can use `_`:

Example #5.17: Using an Underscore in a Loop to Print "Hello" Multiple Times

```
for _ in range(5):
    print("Hello")
```

Here, `_` is used instead of a variable name like `i` or `j` because the value is not important. The loop will simply print "Hello" five times without using the loop index. This helps to make the code more readable by signaling to other programmers that the loop variable is not used in the loop's body.

- video: Underscore to Ignore Values in for loop