1 Introduction to For Loops

In Python, a **for loop** is a control structure that allows you to repeat a block of code a specific number of times by iterating over a sequence, such as a list, tuple, string, or range of numbers. For loops are ideal for tasks like processing items in a collection or performing repetitive actions a known number of times. They make your code more efficient by avoiding repetitive manual coding and are a fundamental tool for beginners learning Python programming.

The basic syntax of a for loop is:

```
for variable in iterable:
    # Code to repeat
```

Here, `variable` takes on each value in the `iterable` (e.g., a list or `range()`) one at a time, and the indented code block runs for each iteration. The loop stops when all items in the iterable have been processed.

2 Example: Using For Loops

Let's explore two simple examples to understand how for loops work.

Example 1: Printing Numbers 1 to 5
The `range()` function generates a sequence of numbers, commonly used in for loops. Here's a program that prints numbers from 1 to 5:

```
for number in range(1, 6):  # range(1, 6) generates 1, 2, 3, 4, 5
    print(number)
```

Output:

```
1
2
3
4
5
```

In this example, `number` takes each value from `range(1, 6)`, and the loop prints it. The `range(start, stop)` function includes numbers from `start` up to, but not including, `stop`.

Example 2: Summing a List
For loops can also iterate over lists. This example calculates the sum of numbers in a list:

```
numbers = [10, 20, 30, 40]
total = 0
for num in numbers:
    total += num
print("Sum:", total)
```

Output:

```
Sum: 100
```

Here, the loop iterates over each element in `numbers`, adding it to `total`. The final sum is printed after the loop completes.

3 Tips for Using For Loops

- Use meaningful variable names to make your code readable.
- The `range()` function is versatile: `range(start, stop, step)` lets you skip numbers (e.g., `range(0, 10, 2)` for 0, 2, 4, 6, 8).
- Nest for loops for complex tasks, like iterating over a grid, but keep nesting minimal for clarity.

4 Student Task

Write a Python program that uses a `for` loop together with the `datetime` library.

- Starting from today's date, print the next 5 days (including today).
- Use a `for` loop with `range()` to generate the sequence of days.
- Each printed line should show the date in the format `YYYY-MM-DD`.

Hint: You can use `datetime.date.today()` to get today's date and `datetime.timedelta(days=1)` to add days.