Python for Loop with Lists

Purpose, Syntax, Working, Indentation and Simple Practice Programs

1. What is a for Loop?

A for loop in Python is used to repeat a block of statements for each item in a sequence, such as a list.

For example, consider the following list of student names:

```
students = ["Amit", "Ravi", "Neha", "Priya"]
```

Instead of writing a separate print() statement for every student, a for loop can process the names one by one.

```
students = ["Amit", "Ravi", "Neha", "Priya"]

for student in students:
    print(student)
```

Output:

```
Amit
Ravi
Neha
Priya
```

2. Purpose of the for Loop

The main purpose of a for loop is to repeat a set of statements for every item in a list.

A for loop is useful when we want to:

- access every item in a list.
- perform the same operation on all list elements.
- display all elements of a list.
- calculate or process information from each element.

Example: multiply every number in a list by 2.

```
numbers = [10, 20, 30, 40]

for number in numbers:
    print(number * 2)
```

Output:

```
20
40
60
80
```

Understanding the Example

numbers is the name of the list. The values 10, 20, 30 and 40 are the elements (objects) stored in the list.

number is the loop variable. It receives one item from the list at a time.

in is a Python keyword that tells Python to take items from the list.

The statement print(number * 2) is executed once for each item.

So, the important idea is:

for number in numbers means: take each item from the numbers list, one at a time, and store it in number.

3. Syntax of for Loop

The basic syntax of a Python for loop with a list is:

```
for variable in list:
    statement
```

Example:

```
fruits = ["Apple", "Mango", "Orange"]

for fruit in fruits:
    print(fruit)
```

Parts of the Syntax

- for – Python keyword used to start the loop.
- variable – loop variable that stores one item from the list at a time.
- in – Python keyword used to take items from the list.
- list – the list whose items are processed one by one.
- : – indicates the beginning of the loop body.
- statement – the indented statement executed for each item.

The loop variable does not have to have the same name as the list. A meaningful name such as fruit makes the program easier to understand.

```
fruits = ["Apple", "Mango", "Orange"]

for x in fruits:
    print(x)
```

4. How Does the for Loop Work?

Consider the following program:

```
fruits = ["Apple", "Mango", "Orange"]

for fruit in fruits:
    print(fruit)
```

The for loop takes the items from the fruits list one by one and stores each item in the loop variable fruit.

First iteration

```
fruit = "Apple"
print(fruit)
```

Output: Apple

Second iteration

```
fruit = "Mango"
print(fruit)
```

Output: Mango

Third iteration

```
fruit = "Orange"
print(fruit)
```

Output: Orange

The loop continues until all items in the list have been processed. After the last item is processed, the for loop ends.

5. Indentation in a for Loop

In Python, indentation is used to identify the statements that belong to the for loop.

```
students = ["Amit", "Ravi", "Neha"]

for student in students:
    print(student)
    print("Student name displayed")
```

Both print() statements are indented, so both statements are executed for every item in the list.

Output:

```
Amit
Student name displayed
Ravi
Student name displayed
Neha
Student name displayed
```

A statement without indentation is outside the loop.

```
students = ["Amit", "Ravi", "Neha"]

for student in students:
    print(student)

print("Loop completed")
```

Here, the last print() statement is outside the loop, so it executes only once after the loop has finished.

Output:

```
Amit
Ravi
Neha
Loop completed
```

Key point: In Python, indentation is not just for appearance. It tells Python which statements belong to the loop.

6. Different Types of Objects in a List

A Python list can contain different types of objects, such as Integer, Float, String and Boolean objects. A for loop can access these objects one by one.

- Integer objects – examples: 10, 20, 30
- Float objects – examples: 1.5, 2.5, 3.5
- String objects – examples: "Amit", "Ravi", "Neha"
- Boolean objects – True and False

7. Simple Practice Programs

The following programs provide simple practice with different types of list objects.

Practice 1 – Integer List

Print each integer.

```
numbers = [10, 20, 30]

for number in numbers:
    print(number)
```

Output:

```
10
20
30
```

Practice 2 – Integer Calculation

Find the square of each number.

```
numbers = [2, 4, 6]
```

```
for number in numbers:
    print(number * number)
```

Output:

```
4
16
36
```

Practice 3 – Float List

Print each float.

```
values = [1.5, 2.5, 3.5]

for value in values:
    print(value)
```

Output:

```
1.5
2.5
3.5
```

Practice 4 – Float Calculation

Add 1 to each float.

```
values = [1.5, 2.5, 3.5]

for value in values:
    print(value + 1)
```

Output:

```
2.5
3.5
4.5
```

Practice 5 – String List

Print each string.

```
names = ["Amit", "Ravi", "Neha"]

for name in names:
    print(name)
```

Output:

```
Amit
Ravi
Neha
```

Practice 6 – String Operation

Print each name in uppercase.

```
names = ["amit", "ravi", "neha"]

for name in names:
    print(name.upper())
```

Output:

```
AMIT
RAVI
NEHA
```

Practice 7 – Boolean List

Print each Boolean value.

```
values = [True, False, True]

for value in values:
    print(value)
```

Output:

```
True
False
True
```

Practice 8 – Mixed List

Print different types of objects.

```
items = [10, 2.5, "Python", True]

for item in items:
    print(item)
```

Output:

```
10
2.5
Python
True
```

8. Key Points to Remember

- A for loop processes the items of a list one by one.
- The loop variable temporarily stores the current item.
- The keyword in connects the loop variable with the list.
- The colon (:) marks the beginning of the loop body.
- Indented statements belong to the for loop.
- Statements outside the indentation execute outside the loop.
- A list can contain Integer, Float, String, Boolean and other types of objects.
- The for loop can process each object in a list one at a time.

Note: These notes introduce the for loop using lists only. The range() function is not covered here and can be studied separately.