For Loop

Table of contents

01

For loop basics

Overview, Syntax, Flowchart, Example & Loop Controls

For Loop

- A **loop** is a code block that executes specific instructions repeatedly.
- Python provides **two** main types of loops for iterative tasks: ***for loop*** and ***while loop.***
- For loop allows you to iterate over elements in a **sequence** (like a list, tuple, or string), performing an action for each item.
- For loop focuses on **element-wise iteration**, making it intuitive for working with collections of data.

For Loop (Syntax)

Python

```
for item in sequence:
    # Loop Body
```

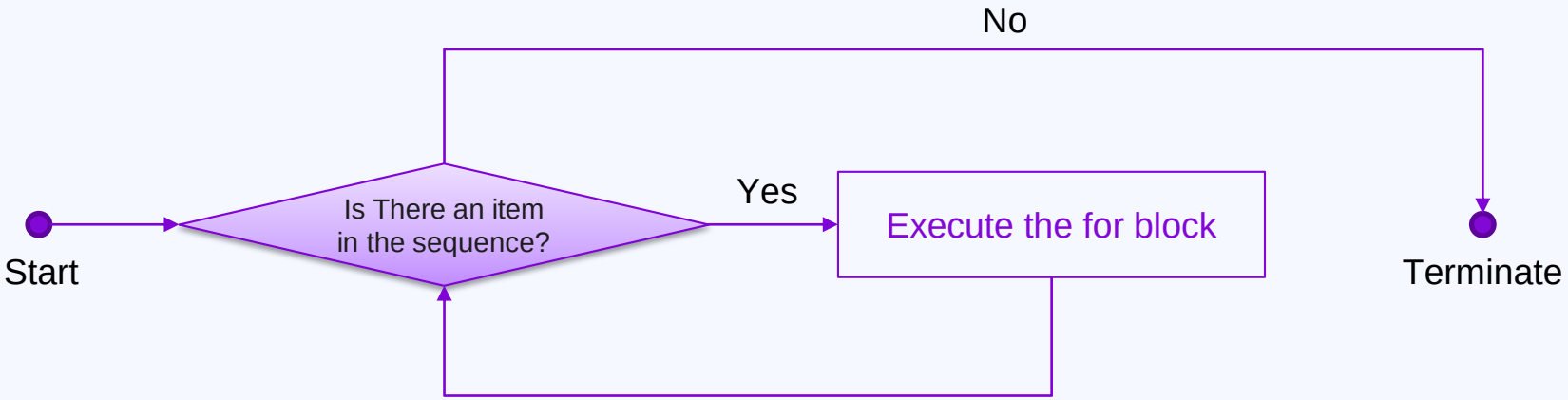
item: This is a variable that takes the value of each element in the sequence one by one during each iteration.

sequence: This can be any iterable like a list, tuple, string, range, etc.

Indented Block: The block of code inside the loop (indented) is executed for each item in the sequence.

Repetition: This loop will run for each items in the sequence and perform some task.

For Loop (flowchart)



For Loop (Example)

Python	Output

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

For Loop (Loop Control)

Loop control is primarily managed through **two** statements:

Break:
Immediately exits the loop.
Usage: Exit the loop early after reaching some value or satisfying some condition.

Continue:
Skips the current iteration and moves directly to the next.
Usage: For ignoring certain condition/ value and move on.

Pass or … :
A placeholder statement that does nothing
Usage: while development to avoid errors temporarily.

Loop Control Example - break

Python	Output

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

Loop Control Example - continue

Python	Output

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

Loop Control Example - pass

Python	Output

`fruits = ["apple", "banana", "cherry"]`
 	
`for fruit in fruits:`
`  pass`	

02

for-else loop

Overview, Syntax, Flowchart, Example, Loop Controls

for-else loop

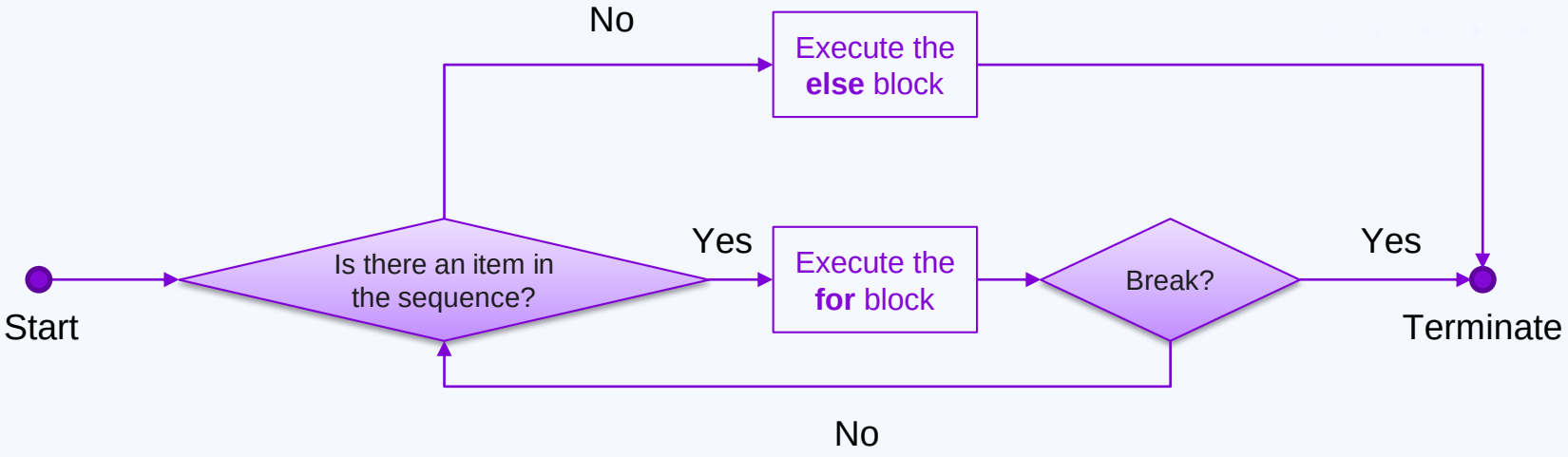
- Python allows an **else** block with a **for** loop.
- **else** block used with a **for** loop will only be executed if the **for** loop terminates normally.
- Normal termination means *completion* of **all iterations** without encountering a **break** statement.
- The **else** doesn't mean "*otherwise*" but rather "*if the loop didn't break*."

for-else loop (Syntax)

Python

```python
for item in sequence:
    # loop body
    if condition:
        break # Exit the loop prematurely
else:
    # Executed only if the loop wasn't broken
```

for-else loop (flowchart)



for-else loop (Example)

Python	Output

`nums = [1, 3, 5]`
`target = 4`
`for num in nums:`
`    if num == target:`
`        print("Found the target!")`
`        break`
`else:`
`    print("Target not found.")`
	Target not found.

Loop Control Example - break

Python	Output

`fruits = ["apple", "banana", "cherry"]`
 	
`for fruit in fruits:`
`  if fruit == "banana":`
`  break`
`  print(fruit)`
`else:`
`  print(“No More Fruits Left”)`
	apple

Loop Control Example - continue

Python	Output

`fruits = ["apple", "banana", "cherry"]`
 	
`for fruit in fruits:`
`  if fruit == "banana":`
`  continue`
`  print(fruit)`
`else:`
`  print(“No More Fruits Left”)`
	apple
	
	Cherry
	
	No More Fruits Left

03

Nested for loop

Overview, Syntax, Example, loop controls

Nested for loop

- A loop inside another loop is called **nested loop**.
- A **nested for loop** is simply a for loop **inside** another for loop.
- Each time the inner loop **completes all its iterations**, the outer loop moves to the **next item**, and the inner loop **starts over**.

Nested for loop (Syntax)

Python

```
for outer_item in outer_sequence:
    for inner_item in inner_sequence:
        # Code to execute with each combination of
        # outer_item and inner_item
```

Nested for loop (Example)

Python	Output

outer_li = [1, 2]
inner_li = [“a”, “b”, “c”]
 	
for outer in outer_li:
  for inner in inner_li:
  print(f"{outer} {inner}")
	1 a
	1 b
	1 c
	2 a
	2 b
	2 c

Nested for loop (Loop Control)

Both **break** and **continue** statements can be used within nested loops. They will only affect the loop in which they are used.

- **break in an Inner Loop**: Only breaks out of the inner loop, not the outer loop.
- **continue in an Inner Loop**: Skips the rest of the current inner loop iteration but does not affect the outer loop.

04

For loop with

string, list, tuple, dictionary, Range(), enumerate() & zip()

For loop with string

Python	Output
`for char in “hello”:` `    print(char)`	h e l l o

For loop with list

Python	Output
`for number in [1, 2, 3, 4]:` `    print(number)`	1 2 3 4

For loop with tuple

Python	Output
`for number in (1, 2, 3, 4):` `    print(number)`	1 2 3 4

For loop with dictionary

Python	Output

`my_dict = {"a": 1, "b": 2, "c": 3}`
 	
`# Looping through keys`
`for key in my_dict:`
`    print(key)`
	a
	b
	c

For loop with dictionary

Python	Output

`my_dict = {"a": 1, "b": 2, "c": 3}`
 	
`# Looping through values`
`for value in my_dict.values():`
`    print(value)`
	1
	2
	3

For loop with dictionary

Python	Output

`my_dict = {"a": 1, "b": 2, "c": 3}`
 	
`# Looping through key-value pairs`
`for key, value in my_dict.items():`
`    print(f"{key}: {value}")`
	a: 1
	b: 2
	c: 3

For loop with range()

For looping over a sequence of numbers, the **range()** function within a for loop is extremely common.

Syntax:

Python
range(start, stop, step)

start: Optional. The starting number (default is 0).

stop: Required. The ending number - 1.

step: Optional. The difference between each number in the sequence (default is 1). Negative step will go in backward (reverse) direction

For loop with range() Example

Python	Output
`for i in range(5):` `    print(i)`	0 1 2 3 4

For loop with range() Example

Python	Output
`for i in range(2,5):` `    print(i)`	2 3 4

For loop with range() Example

Python	Output
`for i in range(2,8,2):` `    print(i)`	2 4 6

For loop with range() Example

Python	Output
`for i in range(4,1,-1):` `    print(i)`	4 3 2

For loop with enumerate()

The **enumerate()** function allows you to loop over a sequence and get both the **index** and the **item** in each iteration, which is helpful when you need to know the position of items.

Python	Output

`fruits = ["apple", "banana", "cherry"]`
 	
`for index, fruit in enumerate(fruits):`
`  print(index, fruit)`
	`0 apple`
	`1 banana`
	`2 cherry`

For loop with zip()

The **zip()** function allows you to loop over **multiple sequences simultaneously**. Each iteration provides a **tuple** containing elements from each sequence.

Python

```
names = ["Amar", "Nitin", "Anvesha"]
ages = [24, 30, 28]
res = zip(names, ages)
print(list(res))
```

Output

```
[('Amar', 24), ('Nitin', 30), ('Anvesha', 28)]
```

For loop with zip() - example

Python

```python
names = ["Amar", "Nitin", "Anvesha"]
ages = [24, 30, 28]
for name, age in zip(names, ages):
    print(f"{name} is {age} years old")
```

Output

```
Amar is 24 years old
Nitin is 30 years old
Anvesha is 28 years old
```

One last thing…

- In Python, the **underscore (_)** is often used as a placeholder for variables that are not actually needed.
- In simple terms, in **for _ in range(...)**, we don't actually care about the variable itself – we just want to repeat something a certain number of times.
- Think of **_** as a way of saying, "**I don't need this value – just repeat!**"

One last thing…

Python	Output
`for _ in range(3):` `    print("Hello!")`	Hello! Hello! Hello!

Knowledge Reinforcement

1

Question

What does the following code print?

Python

```
for i in range(3):
    for j in range(2):
        print(f"{i}, {j}")
```

Answer

A) (0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)

B) (0, 0), (1, 0), (2, 0), (0, 1), (1, 1), (2, 1)

C) (0, 0), (1, 0), (1, 1), (2, 0), (2, 1), (0, 1)

D) (0, 1), (1, 1), (2, 1)

1

Question

What does the following code print?

Python

```
for i in range(3):
    for j in range(2):
        print(f"{i}, {j}")
```

Answer

A) (0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)

B) (0, 0), (1, 0), (2, 0), (0, 1), (1, 1), (2, 1)

C) (0, 0), (1, 0), (1, 1), (2, 0), (2, 1), (0, 1)

D) (0, 1), (1, 1), (2, 1)

2

Question

Which of the following statements about the **for-else** loop are correct?
(Choose all that apply)

Answer

A) The else block executes if the loop completes all iterations without a break.

B) The else block executes if the loop encounters a break.

C) The else block can contain code that runs only when a loop finishes normally.

D) If the loop is interrupted by a continue statement, the else block will not execute.

2

Question

Which of the following statements about the **for-else** loop are correct?
(Choose all that apply)

Answer

A) The else block executes if the loop completes all iterations without a break.

B) The else block executes if the loop encounters a break.

C) The else block can contain code that runs only when a loop finishes normally.

D) If the loop is interrupted by a continue statement, the else block will not execute.

3

Question

In a **nested loop**, a **break** statement in the **inner loop** will also stop the **outer loop**.

Answer

A) True

B) False

3

Question

In a **nested loop**, a **break** statement in the **inner loop** will also stop the **outer loop**.

Answer

A) True

B) False

Explanation:
A break statement in the inner loop only exits the inner loop, not the outer loop.

4

Question

What does the following code print?

Python

```
matrix = [
    [1, 2, 3],
    [4, 5, 6]
]

for row in matrix:
    for element in row:
        print(element, end=" ")
    print()
```

Answer

A) 1 2 3 on the first line, and 4 5 6 on the second line

B) 1 4 2 5 3 6

C) 1 2 3 4 5 6

D) Error

4

Question

What does the following code print?

Python

```
matrix = [
    [1, 2, 3],
    [4, 5, 6]
]

for row in matrix:
    for element in row:
        print(element, end=" ")
    print()
```

Answer

A) 1 2 3 on the first line, and 4 5 6 on the second line

B) 1 4 2 5 3 6

C) 1 2 3 4 5 6

D) Error

5

Question

What does the following code print?

Python

```
for i in range(5):
    if i == 3:
        break
    print(i)
else:
    print("Loop completed")
```

Answer

A) 0 1 2 3

B) 0 1 2

C) 0 1 2 followed by "Loop completed"

D) Only "Loop completed"

5

Question

What does the following code print?

Python

```
for i in range(5):
    if i == 3:
        break
    print(i)
else:
    print("Loop completed")
```

Answer

A) 0 1 2 3

B) 0 1 2

C) 0 1 2 followed by "Loop completed"

D) Only "Loop completed"

6

Question

The **enumerate()** function can be used in a **for** loop to get both the index and the value of each element in the sequence.

Answer

A) True

B) False

6

Question

The **enumerate()** function can be used in a **for** loop to get both the index and the value of each element in the sequence.

Answer

A) True

B) False

7

Question

Which of the following are valid uses of the **range()** function in a **for** loop? *(Choose all that apply)*

Answer

A) range(5)

B) range(1, 10, 2)

C) range(-5, 5)

D) range(10, 1, -1)

7

Question

Which of the following are valid uses of the **range()** function in a **for** loop? *(Choose all that apply)*

Answer

A) range(5)

B) range(1, 10, 2)

C) range(-5, 5)

D) range(10, 1, -1)

8

Question

What does the following code print?

Python

```
for i, value in enumerate(["apple", "banana", "cherry"]):
    print(i, value)
```

Answer

A)
```
0 apple
1 banana
2 cherry
```

B)
```
apple 0
banana 1
cherry 2
```

C)
```
1 apple
2 banana
3 cherry
```

A)
```
apple
banana
cherry
```

8

Question

What does the following code print?

Python

```
for i, value in enumerate(["apple", "banana", "cherry"]):
    print(i, value)
```

Answer

A)
```
0 apple
1 banana
2 cherry
```

B)
```
apple 0
banana 1
cherry 2
```

C)
```
1 apple
2 banana
3 cherry
```

A)
```
apple
banana
cherry
```

9

Question

What does the following code print?

Python

```
colors = ["red", "green", "blue"]
hexcodes = ["#f00", "#0f0", "#00f"]

for color, hexcode in zip(colors, hexcodes):
    print(f"{color} : {hexcode}")
```

9

Question

What does the following code print?

Python

```
colors = ["red", "green", "blue"]
hexcodes = ["#f00", "#0f0", "#00f"]

for color, hexcode in zip(colors, hexcodes):
    print(f"{color} : {hexcode}")
```

Answer

```
red : #f00
green : #0f0
blue : #00f
```

10

Question

What does the following code print?

Python

```
items = ["apple", "banana", "cherry"]

for index, _ in enumerate(items):
    print("Index:", index)
```

10

Question

What does the following code print?

Python

```
items = ["apple", "banana", "cherry"]

for index, _ in enumerate(items):
    print("Index:", index)
```

Answer

```
Index: 0
Index: 1
Index: 2
```

WATCH

Level up your coding with each episode in this focused Python series.

Next Video!

While Loop