Loops in python

By Badra

Loops and their importance

- In a programming language, a loop is a statement that contains instructions that continually repeats until a certain condition is reached.
- Loops help us remove the redundancy of code when a task has to be repeated several times.
- With the use of loops, we can cut short those hundred lines of code to a few.

Loops types in python

1) For loop
2) While loop
3) Nested loop

1) for loop

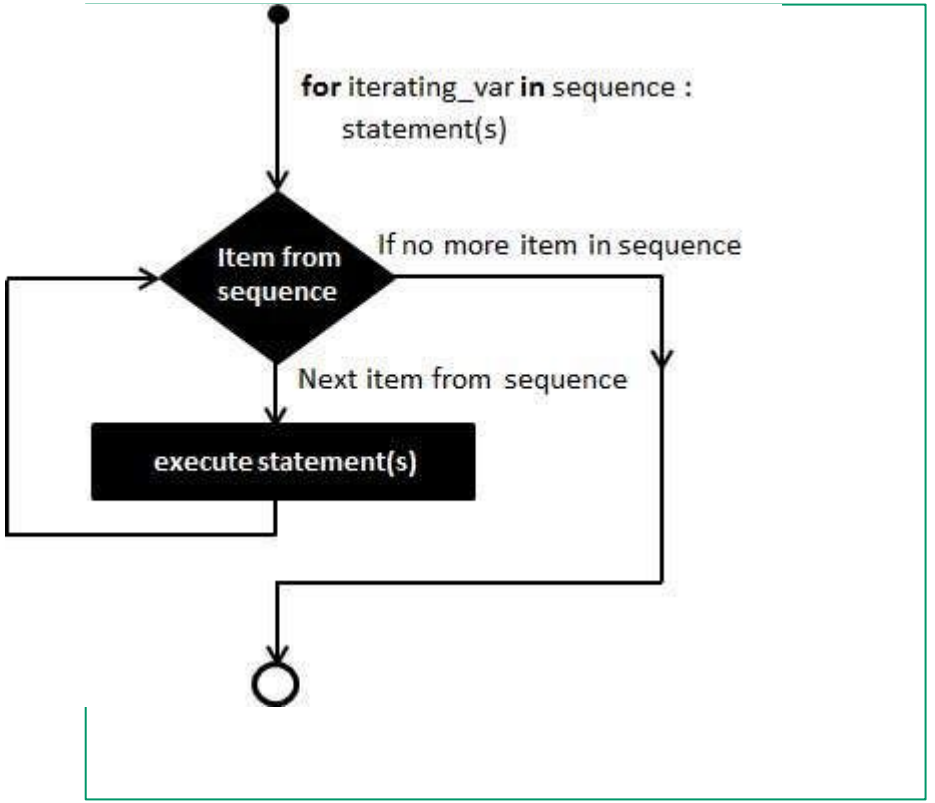
A for loop is used to iterate over a sequence like lists, type, dictionaries, sets, or even strings.

Syntax:

```
for item in iterator:

    statements(code)
```

Takes the first item of the iterable, executes the statement, and moves the pointer to the next item until it reaches the last item of the sequence.



Examples:

```
#  Example 1: Print the text “Hello, World!” 5 times.
list = [1, 2, 3, 4, 5]
for num in list:
    print("Hello, World!")

# Example 2: Create a list of all the even numbers between 1 and 10
# (using the range function to return a sequence of numbers from 1 to 10)
even_nums = []
for i in range(1, 11):
    if i % 2 == 0:
        even_nums.append(i)
print("Even Numbers: ", even_nums)
```

```
# Example 3: Creating an infinite loop.
# An infinite loop can be created using
# a loop by appending a new element to the list after every iteration.
num = [0]
for i in num:
    print(i)
    num.append(i+1)

# Example 4: use else with a for loop (for-else pattern)
iterator = (1, 2, 3, 4)
for item in iterator:
    print(item)
else:
    print("No more items in the iterator")
```

```python
# Example 5: Display the items of a dictionary
example = {
    'iterator': 'dictionary',
    'loop': 'for',
    'operation': 'display dictionary elements'
}
for key in example:
    print(f"{key}: {example[key]}")

# or
# The key, value of a dictionary can be directly accessed using .items()
for key, value in example.items():
    print(f"{key}: {value}")
```

2) while loop

→ It continually executes the statements(code) as long as the given condition is TRUE. It first

```
Syntax:
-------
while condition:
    statements(code)

notes:
--> Inside the while loop, we can have any number of statements
--> The condition may be anything as per our requirement.
--> The loop stops running when the condition fails (become false),
    and the execution will move to the next line of code.
```

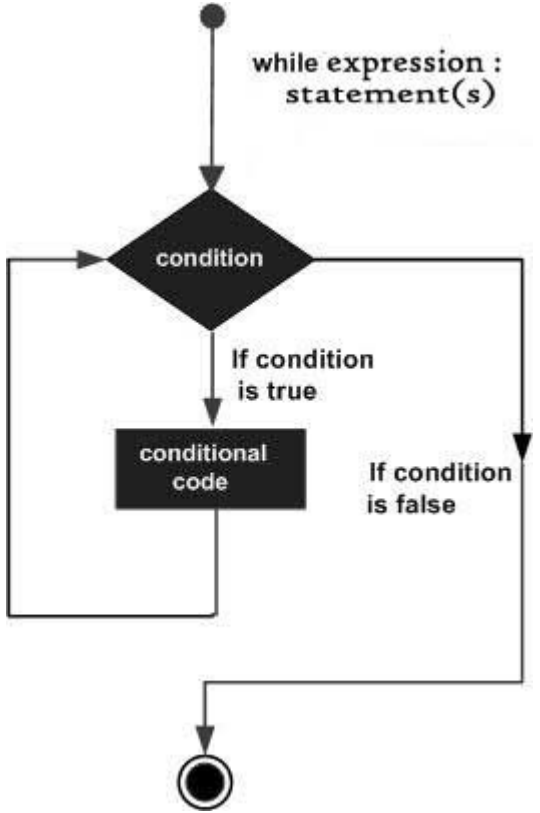
While flow diagram

- It first checks the condition,

  executes the conditional code if the condition is TRUE,

  and checks the condition again.

  Program control exits the loop if the condition is FALSE.



Example while loop

```
# Example 1: Print the text “Hello, World!” 5 times.
num_of_times = 1

while num_of_times <= 5:
    print("Hello, World!")
    num_of_times += 1

# Example 2: Create a list of all the even numbers between 1 and 1
num = 1
even_numbers = []
while num <= 10:
    if num % 2 == 0:
        even_numbers .append(num)

    num += 1
```

```python
# Example 3: Creating an infinite loop
# A loop runs infinite times when the condition never fails.
i = True
while i:
    print("Condition satisfied")

# Example 4: use else with a while loop(while-else pattern)
# When an else statement is used along with a while loop,
# the control goes to the else statement when the condition is False.
var = 1
while var <= 4:
    print(f'Condition is TRUE: {var} <= 4')
    var += 1
else:
    print(f'Condition is FALSE: {var} > 4')
```

3) Nested loops in python

Nested loops

```
while condition:
    statements

    for item in iterator:
        statements
```

```
while condition:
    statements

    while condition:
        statements
```

```
for item in iterator:
    statements

    while condition:
        statements
```

```
for item in iterator:
    statements

    for item in iterator:
        statements
```

Example nested loop

```
# Example: Create a list of even numbers between 1 and 10
even_list = []
for item in range(1, 11):
    while item % 2 == 0:
        even_list.append(item)
        break
print("Even Numbers: ", even_list)
```

Loop control statements in python

purpose:

- Loop control statements are used to change the flow of execution. These can be used if you wish to skip an iteration or stop the execution.
- The three types of loop control statements are:

  1. break statement

  2. continue statement

  3. pass statement

1) break statement

- based on the given condition,

  the break statement stops the execution and brings the control out of the loop.

- break is an built-in keyword in python

```python
# Example : break
# Example #1: Create a list of the odd numbers between 1 and 20 (use while, break)
num = 1
odd_nums = []
while num:
    if num % 2 != 0:
        odd_nums.append(num)
    if num >=20:
        break
    num += 1
print("Odd numbers: ", odd_nums)
# Example #2: Stop the execution if the current number is 5 (use for, break)
for num in range(1, 11):
    if num == 5:
        break
    else:
        print(num)
```

2) continue in python

- **Continue statement is used to skip the current iteration**

  when the condition is met and allows the loop to continue with the next iteration.

- **It does not bring the control out of the loop unlike the break statement.**

```python
# Example : continue
# Example: Skip the iteration if the current number is 6 (use while, continue)
num = 0
while num < 10:
    num += 1
    if num == 6:
        continue
    print(num)

# Example: Skip the iteration if the current number is 6 (use for, continue)
for num in range(1, 11):
    if num == 6:
        continue
    print(num)
```

3) pass statement in python

- **Pass statement is used when we want to do nothing when the condition is met.**

  It doesn’t skip or stop the execution, it just passes to the next iteration.

  Sometimes we use comment which is ignored by the interpreter.

  Pass is not ignored and can be used with loops, functions, classes, etc.

- **It is useful when we do not want to write functionality at present**

  but want to implement it in the future.

```
# Example : pass
# Example: while, pass statement
num = 1
while num <= 10:
    if num == 6:
        pass
    print(num)
    num += 1

# Example: for, pass statement
for num in range(1, 11):
    if num == 6:
        pass
    print(num)
```