Python Decision Making & Looping constructs

Decision Making In Python

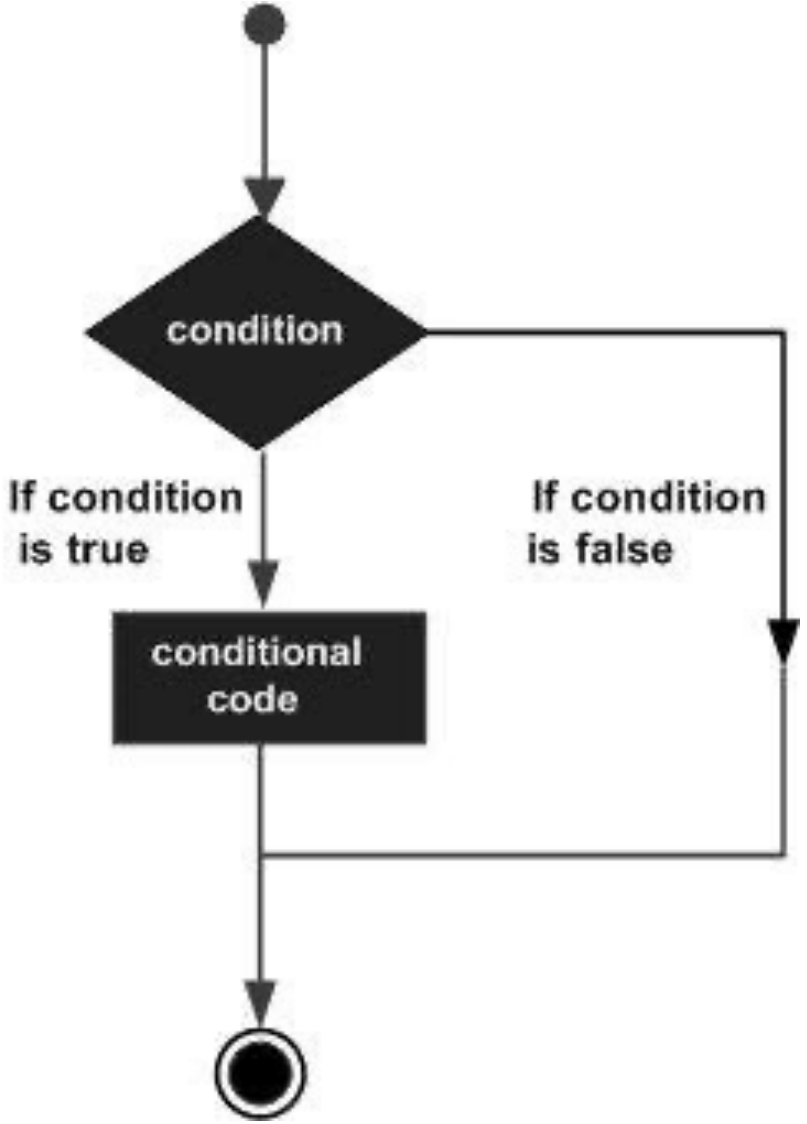
If statements

If-else statements

If elif else statements

Nested if statements

Decision Making In Python



Decision Making In Python

if statements: An **if statement** consists of a boolean expression followed by one or more statements.

if...else statements: An **if statement** can be followed by an optional **else statement**, which executes when the Boolean expression is FALSE.

nested if statements: You can use one **if** or **else if** statement inside another **if** or **else if** statement(s).

Example

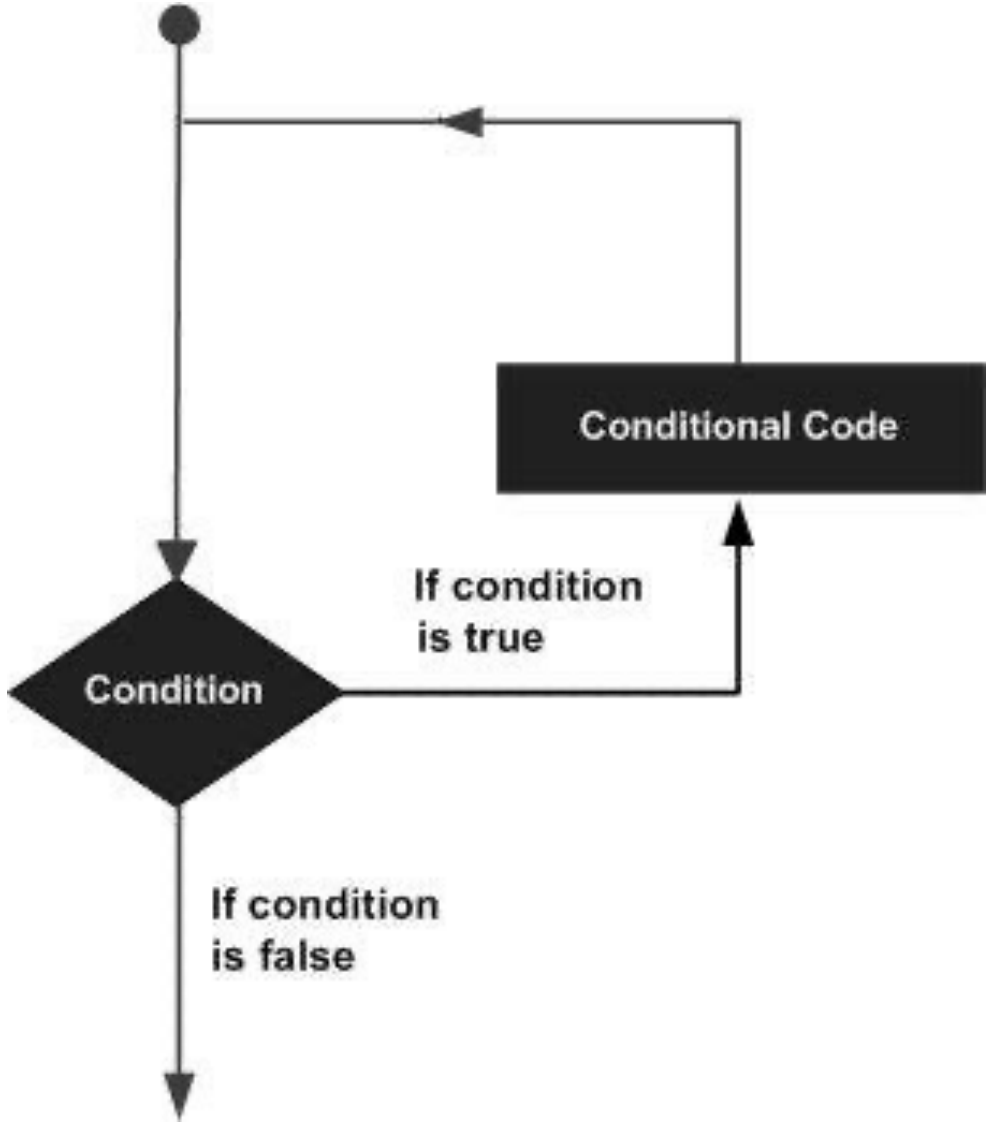
```
x = int(input("Enter an integer value: "))
if x < 0:
   x = 0
   print('Negative changed to zero')
elif x == 0:
   print('Entered value is Zero')
elif x == 1:
   print('Entered value is One')
else:
   print(' Entered value is greater than One')
```

Python Loops

Loops are used in programming to repeat a specific block of code.

Python has two primitive loop commands:

- while loops
- for loops



for loop in Python

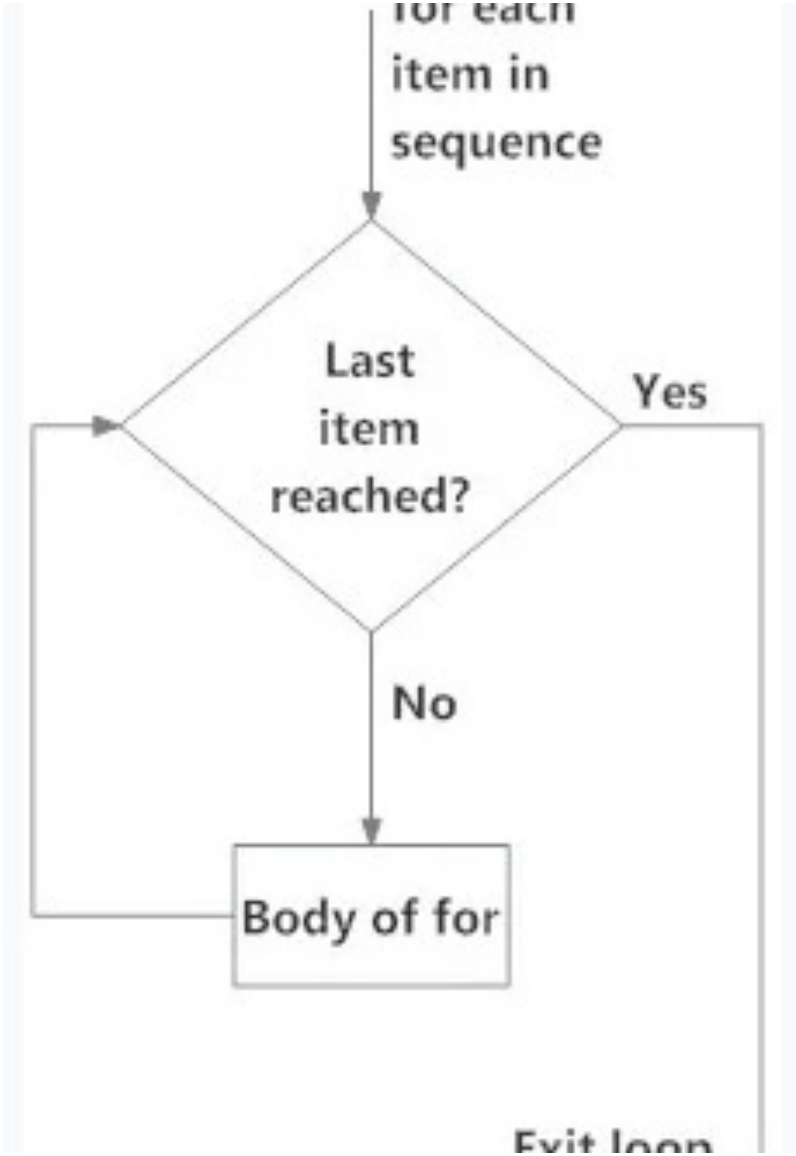
The for loop in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects. Iterating over a sequence is called traversal.

Syntax:

for val in sequence:

Body of for

Loop continues until we reach the last item in the sequence. The body of for loop is separated from the rest of the code using indentation.



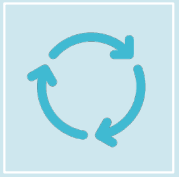
The range() function

- We can generate a sequence of numbers using range() function. range(10) will generate numbers from 0 to 9 (10 numbers).
- We can also define the start, stop and step size as range(start, stop, step_size); where step_size defaults to 1 if not provided.
- We can use the range() function in for loops to iterate through a sequence of numbers. It can be combined with the len() function to iterate through a sequence using indexing.

for loop with else

A for loop can have an optional else block as well. The else part is executed if the items in the sequence used in for loop exhausts.

The break keyword can be used to stop a for loop. In such cases, the else part is ignored.



Hence, a for loop's else part runs if no break occurs.

Nested Loops

- A nested loop is a loop inside a loop.
- The "inner loop" will be executed one time for each iteration of the "outer loop":

```
adj = ["red", "big", "tasty"]

fruits =["apple", "mango", "kiwi"]

for x in adj:
    for y in fruits:
        print(x, y)
```

Output:

red apple
red mango
red kiwi
big apple
big mango
big kiwi
tasty apple
tasty mango
tasty kiwi

While loop in Python

The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.

We generally use this loop when we don't know the number of times to iterate beforehand.

Syntax:

```
while test_expression:
    Body of while
```

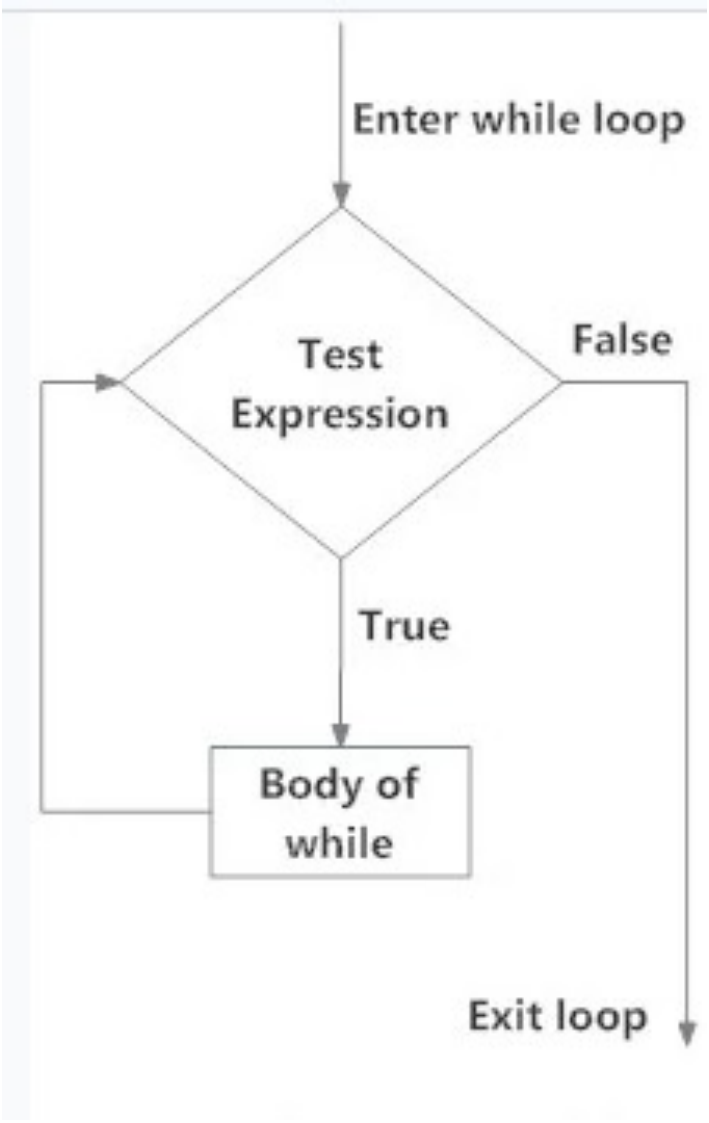
A while loop evaluates the condition

If the condition evaluates to True, the code inside the while loop is executed.

condition is evaluated again.

This process continues until the condition is False.

When condition evaluates to False, the loop stops.



Example

```
i = 1

n=int(input("Enter upper value"))

while i < n:
    print(i)
    i += 1
```

While loop with else

- Same as with for loops, while loops can also have an optional else block.
- The else part is executed if the condition in the while loop evaluates to False.
- The while loop can be terminated with a break statement. In such cases, the else part is ignored. Hence, a while loop's else part runs if no break occurs and the condition is false.

Example

```
counter = 0
while counter < 5:
    print('Inside loop block')
    counter = counter + 1
else:
    print('Inside else block')
```

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed. Python supports the following control statements:

- break statement: Terminates the loop statement and transfers execution to the statement immediately following the loop.
- continue statement: Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
- pass statement: The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute. Or in simple terms, for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.