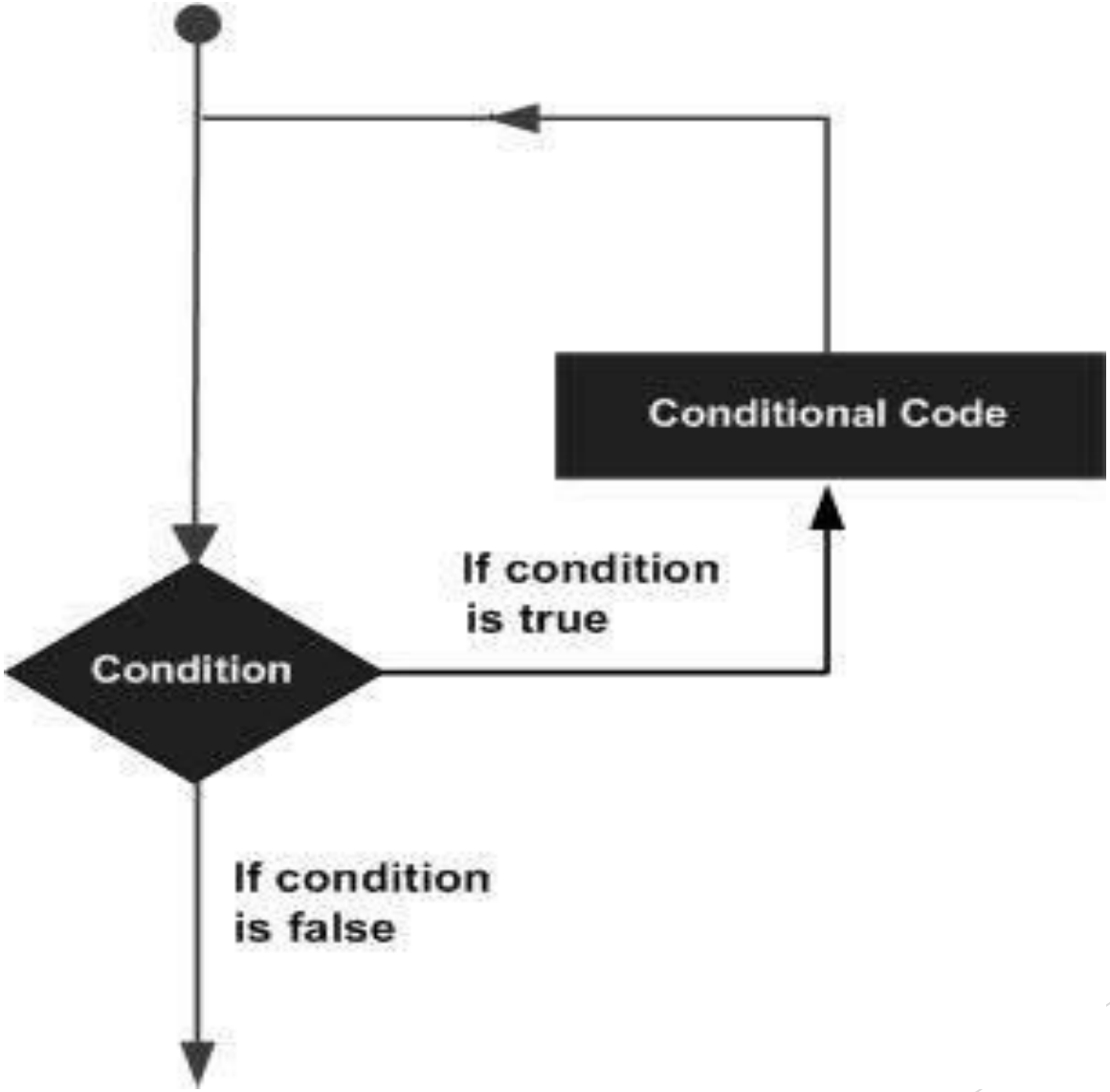
Loop Control in Python

Loops

- Executes a block of statements repetitively
- Tests and executes loop statements repetitively if the condition returns true value
- Stops executing loop statements if the condition returns false value
- Types of loop statements are:
  - `while` loop
  - `for` loop

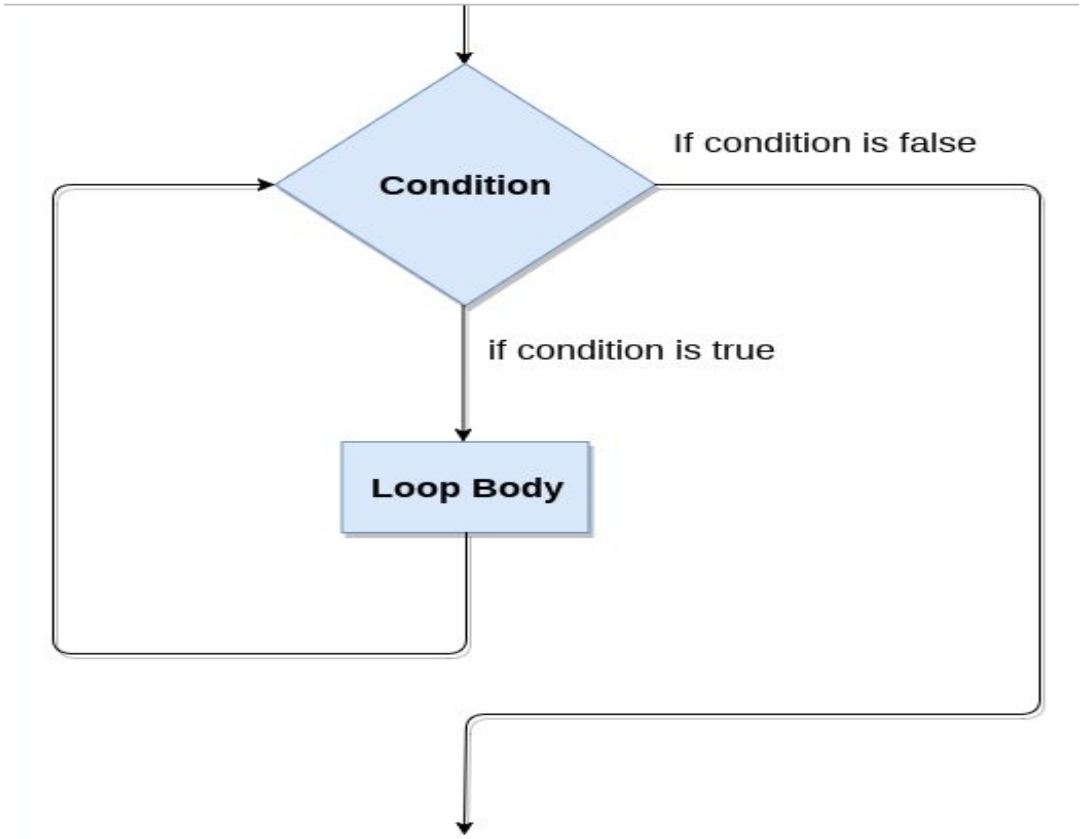

Conditional Code
If condition
is true
Condition
If condition
is false

While Loop - I

- Executes the loop executing statements depending on the return result of the testing condition
- Syntax for while loop is:

```
while (condition):
    Code executes if the condition is true;

Code executes if the condition is false;
```


Condition
If condition is false
if condition is true
Loop Body

While Loop - II

- The `while` keyword is followed by the condition in parentheses
- The `condition` is the test expression consisting of variables and operators. If the condition is satisfied, the code in the loop body is executed. If the condition is false, the loop stops and the control move to the statement following the loop
- The value of the executing statements depends on the condition, therefore it checks the condition after each iteration of the loop

Example for While Loop - I

- For example, to display first 10 odd numbers

```
 number=1
Print("The odd numbers are  ")
while number <= 10:
   print( " number ")
      number=number+2
```

Example for While Loop - II

- Here, the first 10 odd numbers are displayed using the `while` loop
- The value keeps displaying until the counter reaches to the value 10
- The loop stops once the condition gets satisfied

For Loop

- Enables us to execute a block of code repetitively for a fixed number of times
- Executes the loop executing statements till the testing condition gets satisfied
- Stops executing once the condition gets satisfied

Using For Loop - I

- Syntax for the `for` loop is:

for iterating_var in sequence:

Statement

Using For Loop - II

- The `condition` specifies the testing condition
- The `initialization` initializes the value for the counter
- The `re-initialization` specifies the increment and decrement of the counter
- The codes inside the loop gets executed only when the `condition` returns true value
- The `re-initialization` statement helps in the iteration of the loop

Example for the for loop - I

```
for i in range(0,10):
    print(i,end = ' ')
```

Output:

0 1 2 3 4 5 6 7 8 9

The range() Function

To loop through a set of code a specified number of times, we can use the range() function,

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Example

Using the range() function:

```
for x in range(6):
  print(x)
```

Jump Statements

- Controls the execution of the loop statements
- Different jump statements are:
  - `break` statement
  - `continue` statement

Break Statement

- Stops the execution of the loop
- Passes the control either to the beginning of the next loop or to the statement following the loop
- Uses with the `if` statement, `switch` statement, `for` loop, `while` loop, and `do-while` loop

Example for break statement - I

- For example,

```
For i in ‘Computer':
        print (i)
        If i==‘u': break;
```

Example for break statement - II

- Here, the `break` statement is used with the `for` loop
- The condition is specified in the `if` statement
- The `break` statement controls the loop. If the `break` statement is not used, then the loop will go in continuous state

Continue Statement

- Uses with the loop statements
- Stops executing the current loop and continues with the next execution of the program
- Uses with the `if` statement, `for` loop, `while` loop

Example for continue statement - I

- Fruits = ["apple", "banana", "cherry“,”kiwi”]
  for x in fruits:
    if x == "banana":
      continue
    print(x)

Pass Statement

- The pass statement is a null operation; nothing happens when it executes.
- The pass keyword is used as a placeholder.
- The pass keyword can also be used in empty class definitions

Example for pass statement

```
word=‘python’
for alpha in word:
  If alpha == 'h':
      Pass
      Print (‘this is pass block')
  Print ('Current word:’ word)

Print ("Good bye!")
```