Chapter No 4. Control Flow

Conditional Statements:

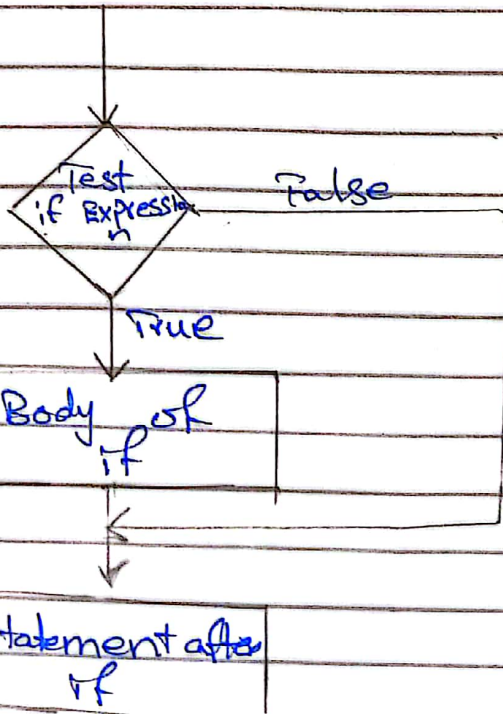
IF Statement:

If statement is the most simple decision-making statement, for deciding whether a certain statement should be executed or not.

Syntax:

```
if condition :
Indent    # Statements to execut
          # condition is true.
        #Statement after if block
```

Flowchart:



Example:

See 01_if Statement.ipy

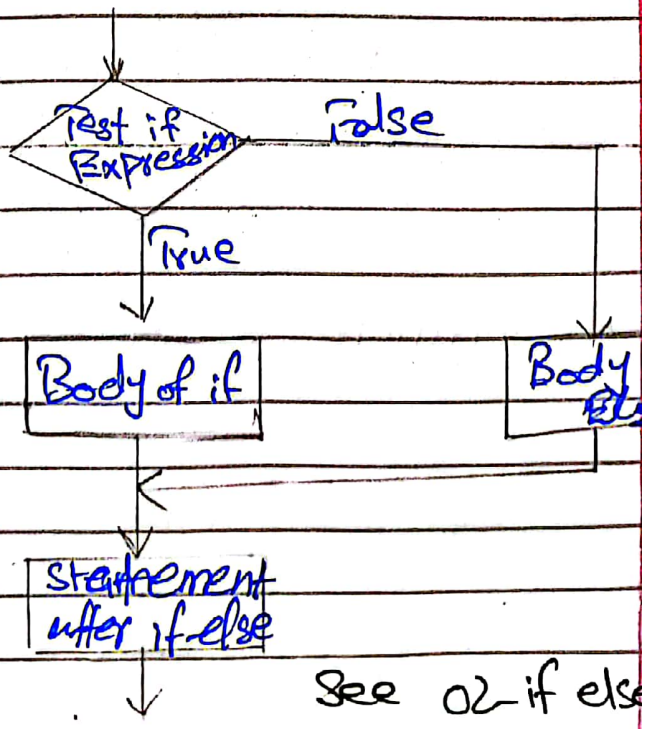
IF - Else Statement:

If we want to execute one statement for if block is True and another for false, then we use if-else statements.

Syntax:

```
if (condition):
Indent    # Executes this block
          # condition is True
else:
Indent    # Execute this block
          # condition is False
```

Flowchart:



See 02_if else

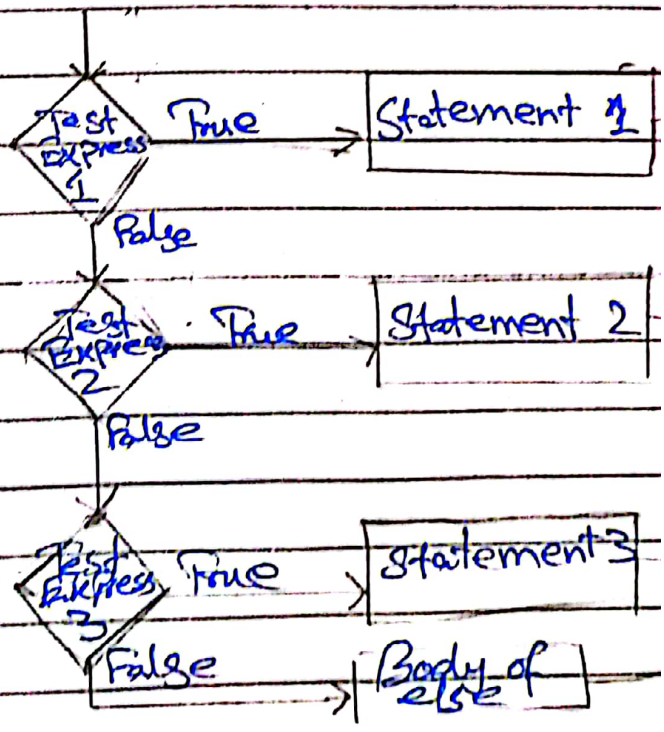
11- Elif Statements

Statements In the if-elif else ~~todoter~~, multiple conditions are given. As soon as the one of the if condition is evaluated is True, all other conditions are bypassed.

Syntax:

```
if (condition):
        statement
elif (condition):
        statement
        :
        :
else:
        statement
```

Flowchart:



Statemen
if-elif

Example: See 03_if elif and else.ipynb

Nested if Statement:

A nested if is an if statement inside the body of another if statement. We can do this with if-else statement and if-elif as well

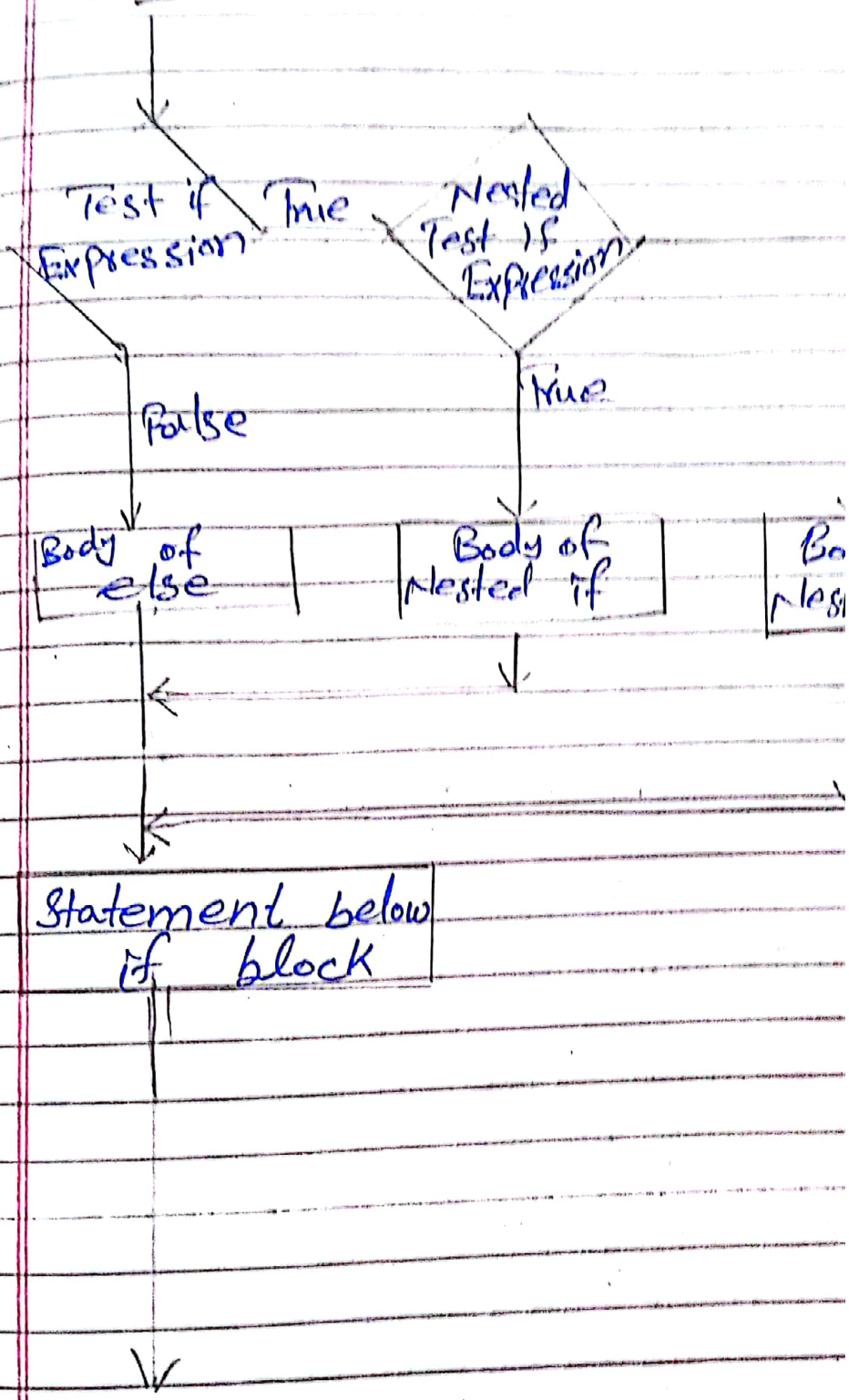
Syntax:

```
if (condition 1):
Indent  # Executes when conditio
        #1 is True
Indent  if (condition 2):
        Indent # Executes when
        Indent # condition 2 is True
        #if block end
    # if block end
```

Example:

See 04_Nested if.ipynb

Flowchart:



Shorthand if and if-else :

Short hand if and if-else are simple if else statements written in a single line

Example:

See 05-shorthand if else.ipynb

: Loops :

1 For Loop:

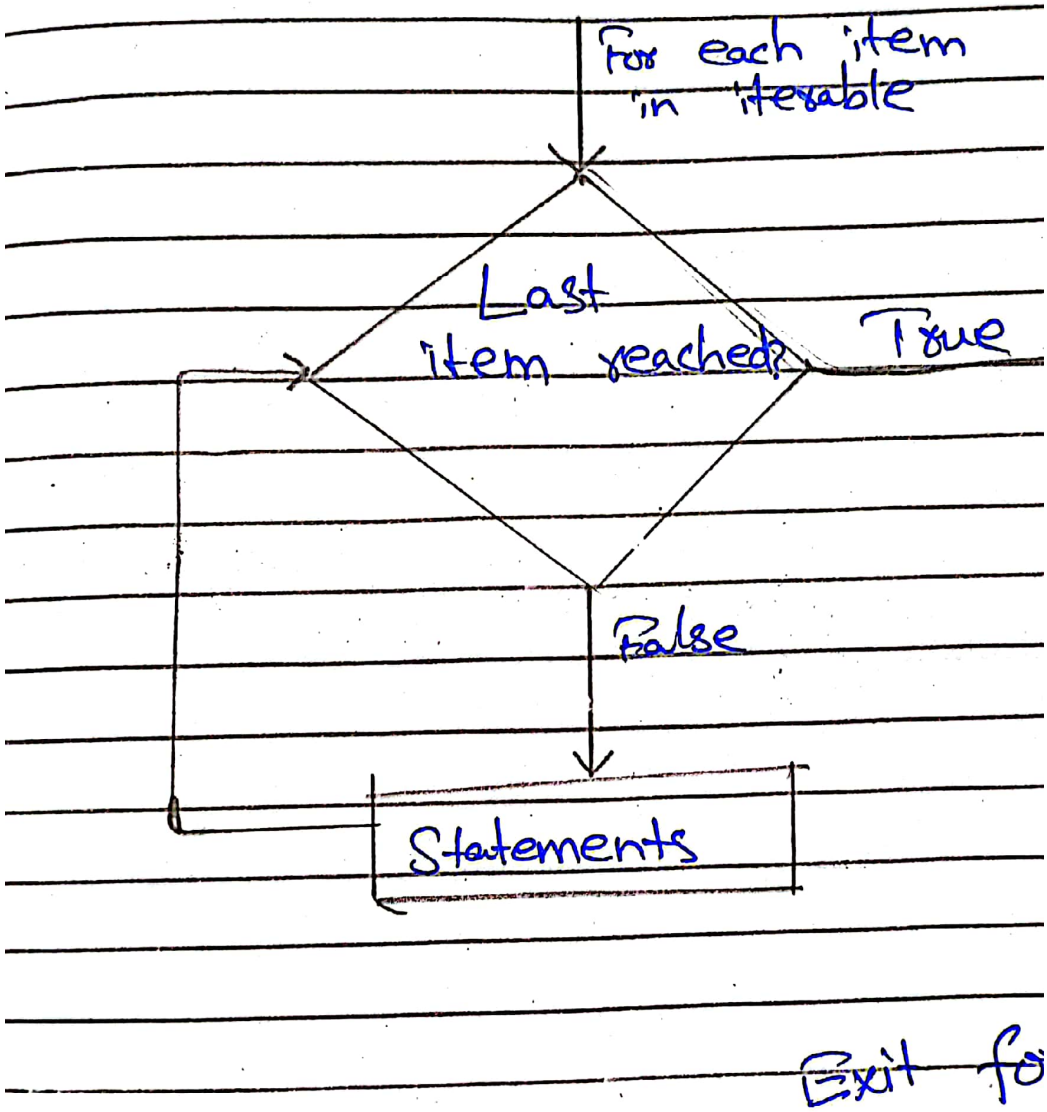
Python for loop is used for sequential traversal. It is used to iterate over iterables like lists, strings, Tuple, Dictionary, or set.

Note: In Python for loop only can be implemented on collection-based iterables

Syntax:

```
for var in iterable:
    #statements
```

Flowchart:



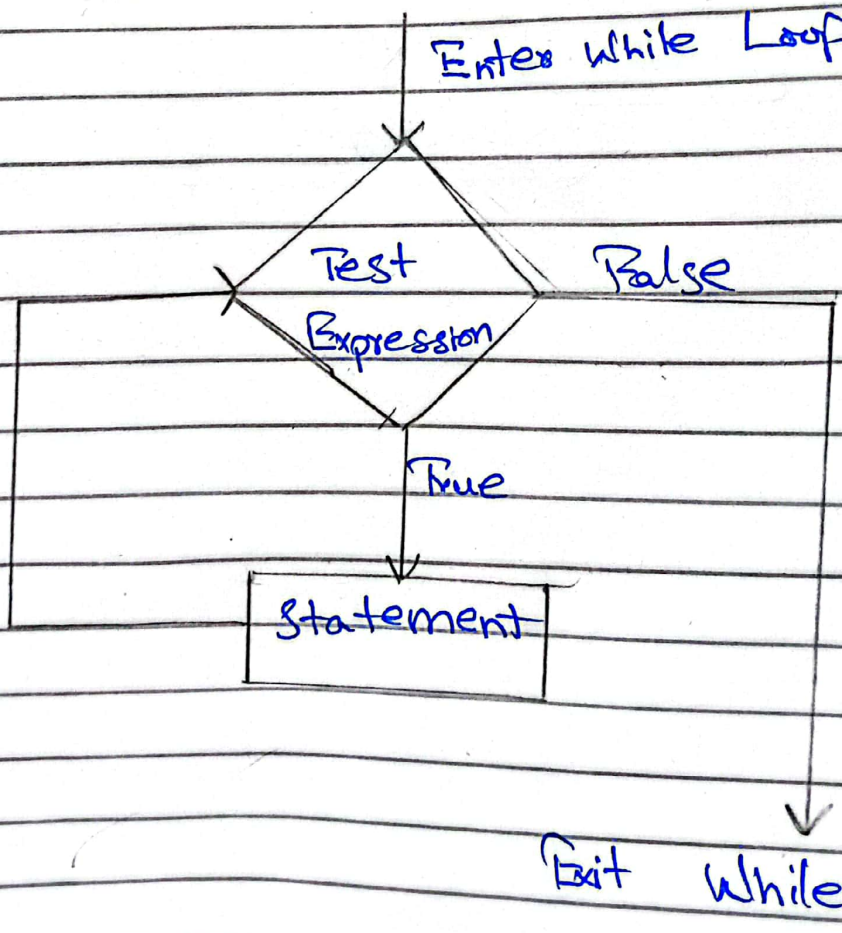
2 While Loop:

To execute a block of statement repeatedly until a given condition is satisfied, we use while loop. If condition is evaluated as false, the statement after loop is executed.

Syntax:

```
while (condition):
    #Indent statement(s)
```

Flowchart:



Example: See 02_While Loops.ipynb in Loops folder.

Python Break Statement:

Python break statement is used to terminate the execution of the loop. Break statement bring the control out of loop when a condition is triggered. It is usually inside an if statement.

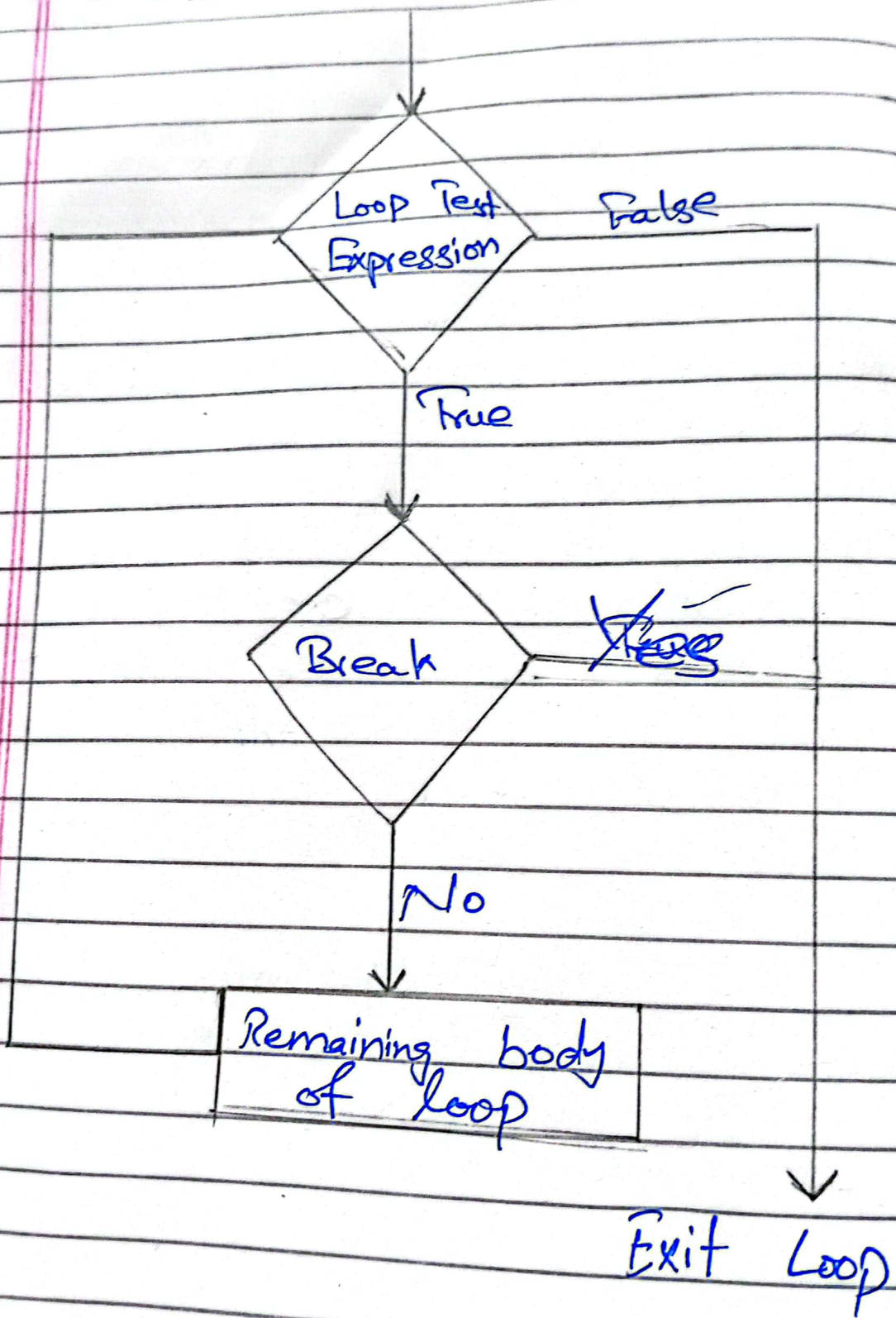
Syntax:

```
Loop {
    Condition:
        break
}
```

Example:

See Break Statement.ipynb

Flowchart:



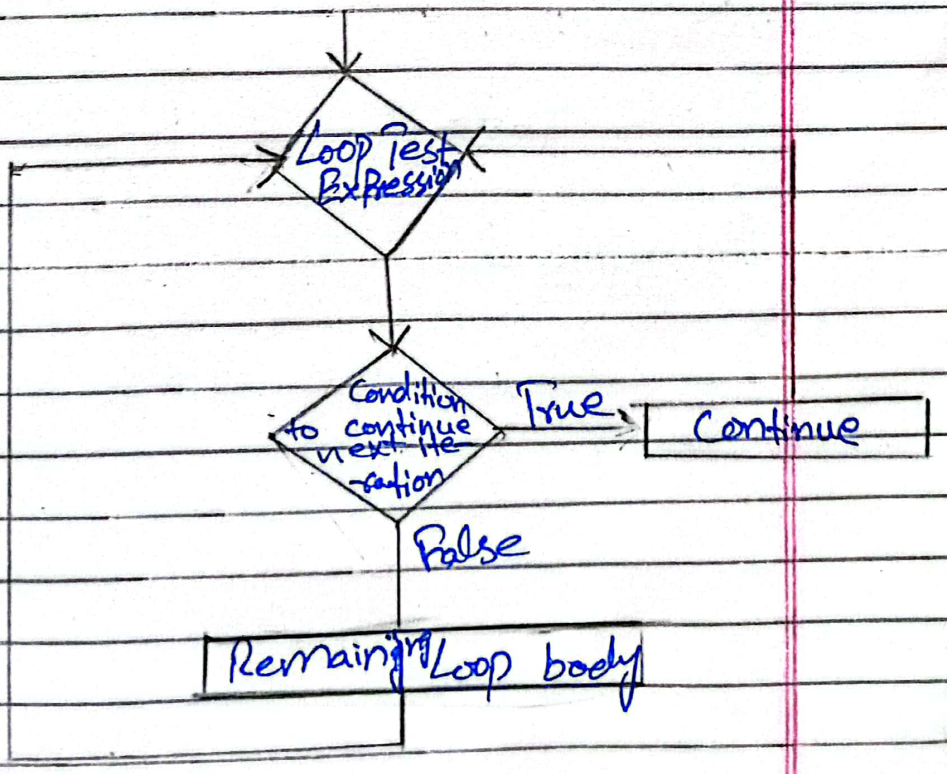
Python Continue Statement:

To skip the execution of the program block if a certain condition is true and making the control to start next iteration, continue statement is used in python.

Syntax:

```
while True:
    .
    .
    if x==10:
        continue
    print x
```

Flowchart:



Example:

See Continue Statement.ipynb

Python Pass Statement:

Pass statement in python is like a null statement but unlike comment ;it is not ignored by interpreter.

We use pass statement when the user or in this case we don't know what code to write.

Syntax:

```
pass
```

Example:

See Pass Statement.ipynb