Unit 2 Conditional Execution and Iterations

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.

Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise.

Following is the general form of a typical decision making structure found in most of the programming languages −

Decision making statements in Python Python programming language assumes any non-zero and non-null values as TRUE, and if it is either zero or null, then it is assumed as FALSE value.

Python programming language provides following types of decision making statements. Click the following links to check their detail.

Sr.No. Statement & Description

1. if statements

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

2. if else statements

An if statement can be followed by an optional else statement, which executes when the boolean expression is FALSE.

3. elif statements

In Python, we have an elif keyword to chain multiple conditions one after another. With elif ladder, we can make complex decision-making statements.

The elif statement helps you to check multiple expressions and it executes the code as soon as one of the conditions evaluates to True. 4. nested if statements You can use one if or else if statement inside another if or else if statement(s).

Let us go through each decision making briefly −

Single Statement Suites If the suite of an if clause consists only of a single line, it may go on the same line as the header statement.

Python if Statement:

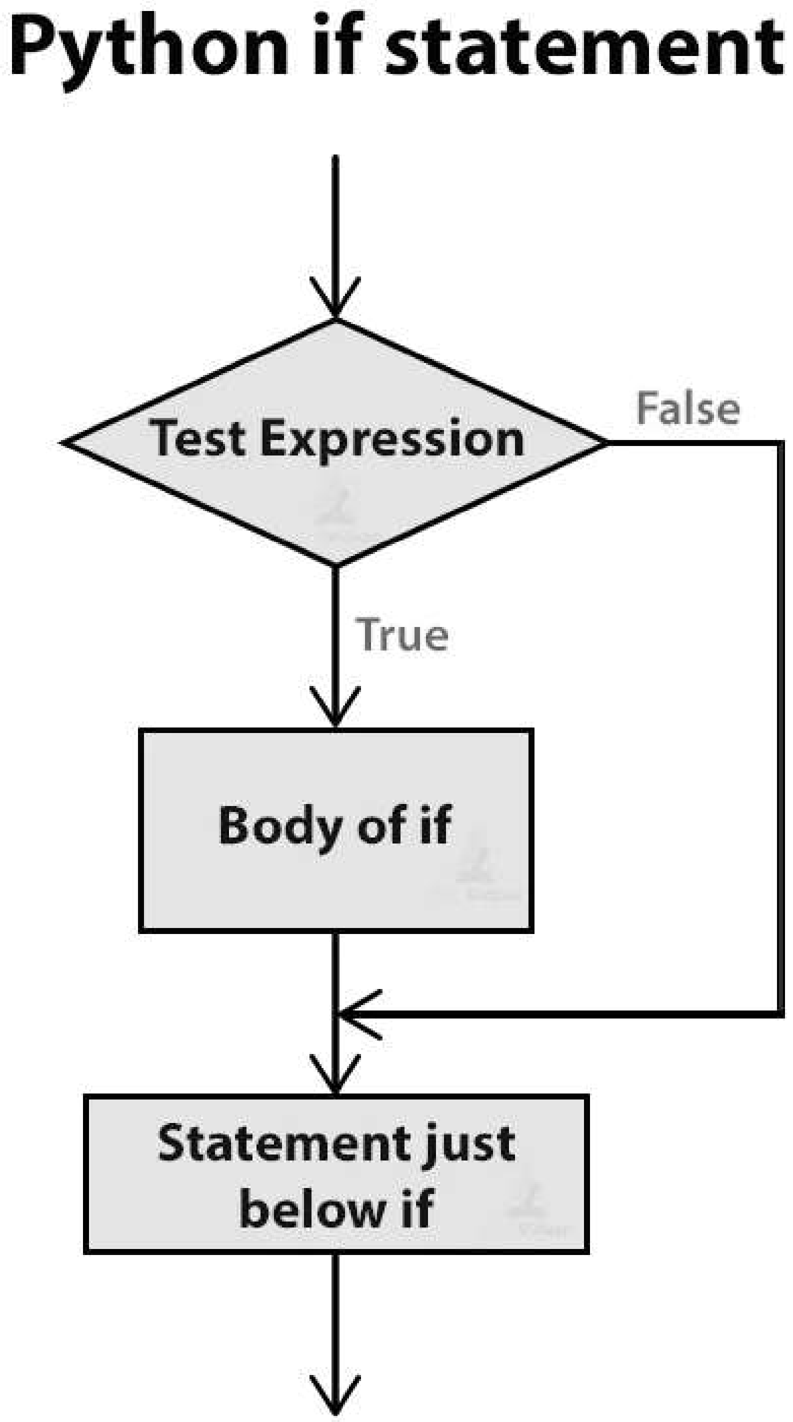
Python if statement if statement is the most simple form of decision-making statement. It takes an expression and checks if the expression evaluates to True then the block of code in if statement will be executed.

If the expression evaluates to False, then the block of code is skipped.

Syntax:

```
if ( expression ):
    Statement 1
    Statement 2
    .
    Statement n
```

python decision making - if statement



In [2]:
```
#Example of if statement
a = 20 ; b = 20
```

```
if ( a == b ):
    print( "a and b are equal")
print("If block ended")
```

```
a and b are equal
If block ended
```

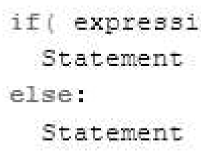
In [3]:
```
#Example 2 of if statement
num = 5
if ( num >= 10):
    print("num is greater than 10")
print("if block ended")
```

```
if block ended
```

Python if-else statement

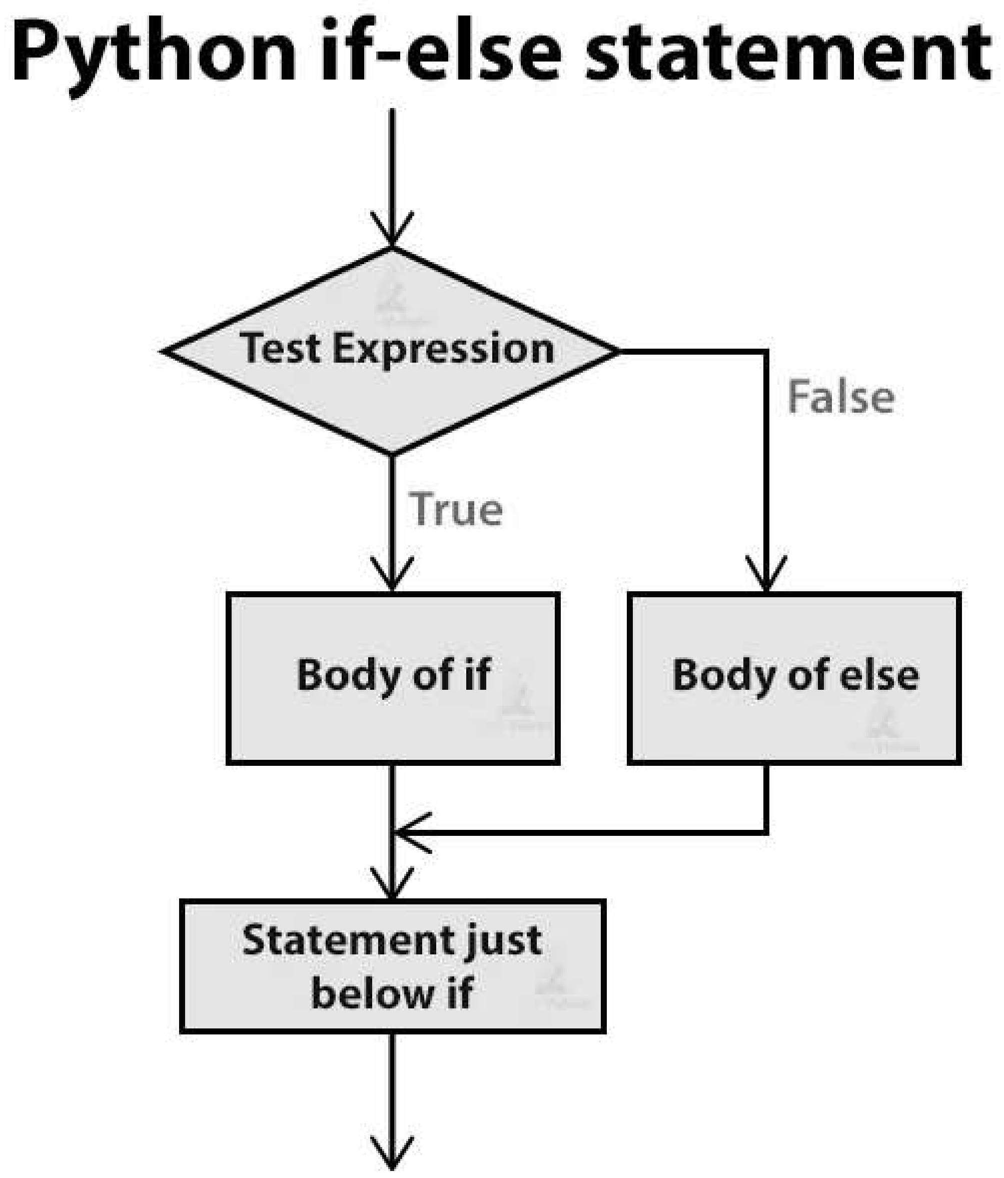
From the name itself, we get the clue that the if-else statement checks the expression and executes the if block when the expression is True otherwise it will execute the else block of code. The else block should be right after if block and it is executed when the expression is False.

Syntax:



```
if( expression ):
  Statement
else:
  Statement
```

decision making in python if-else statement



In [5]:
```
#Example of if...else statement
number1 = 20 ; number2 = 30
if(number1 >= number2 ):
    print("number 1 is greater than number 2")
else:
    print("number 2 is greater than number 1")
```

```
number 2 is greater than number 1
```

In [6]:
```
#Example 2 of if...else statement
name=input("Enter Name:")
if name== "Arman" :
    print("Hello Arman Good Morning")
else:
    print("Hello Guest Good Moring")
    print("How are you!!!")
```

```
Enter Name:Arman
Hello Arman Good Morning
```

In [7]:
```
#Example 2 of if...else statement
name=input("Enter Name:")
if name== "Arman" :
    print("Hello Arman Good Morning")
else:
    print("Hello Guest Good Moring")
    print("How are you!!!")
```

```
Enter Name:Govind
Hello Guest Good Moring
How are you!!!
```

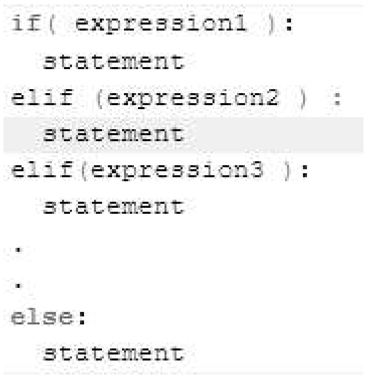
Python if-elif ladder

You might have heard of the else-if statements in other languages like C/C++ or Java.

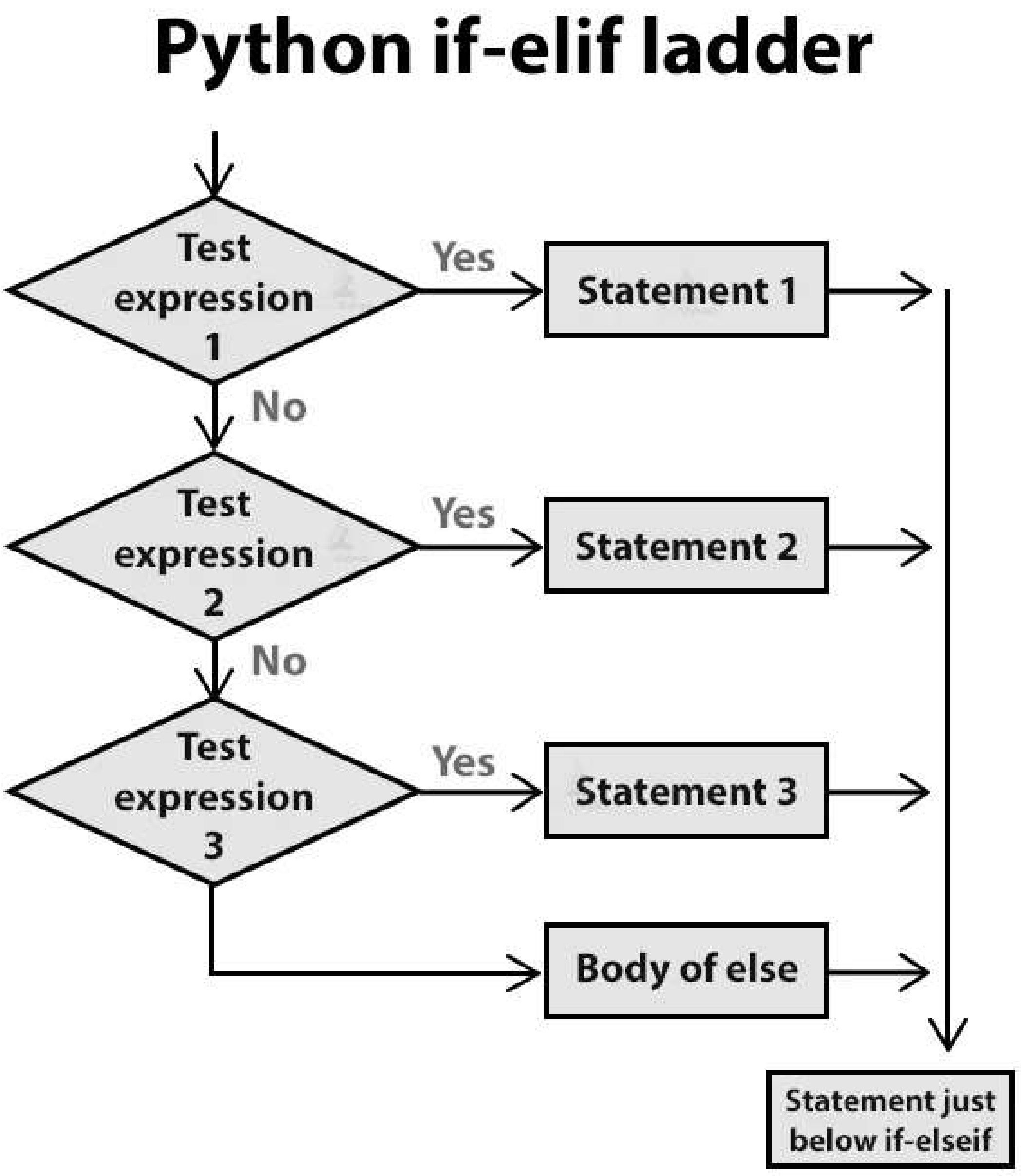
In Python, we have an elif keyword to chain multiple conditions one after another. With elif ladder, we can make complex decision-making statements.

The elif statement helps you to check multiple expressions and it executes the code as soon as one of the conditions evaluates to True.

Syntax for elif ladder:



```
if( expression1 ):
  statement
elif (expression2 ) :
  statement
elif(expression3 ):
  statement
.
.
else:
  statement
```



In [10]:
```
#Example of elif ladder
print("Select your ride:")
print("1. Bike")
print("2. Car")
print("3. SUV")
choice = int(input())
if(choice == 1):
    print("You have selected Bike")
elif( choice == 2 ):
    print("You have selected Car")
elif( choice == 3 ):
    print("You have selected SUV")
else:
    print("Wrong choice!")
```

```
Select your ride:
1. Bike
2. Car
3. SUV
2
You have selected Car
```

In [11]:

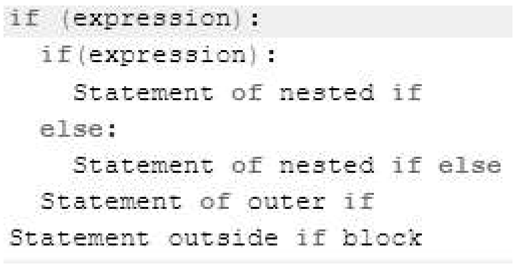
```python
#Example of elif ladder
print("Select your ride:")
print("1. Bike")
print("2. Car")
print("3. SUV")
choice = int(input())
if(choice == 1):
    print("You have selected Bike")
elif( choice == 2 ):
    print("You have selected Car")
elif( choice == 3 ):
    print("You have selected SUV")
else:
    print("Wrong choice!")
```

```
Select your ride:
1. Bike
2. Car
3. SUV
4
Wrong choice!
```

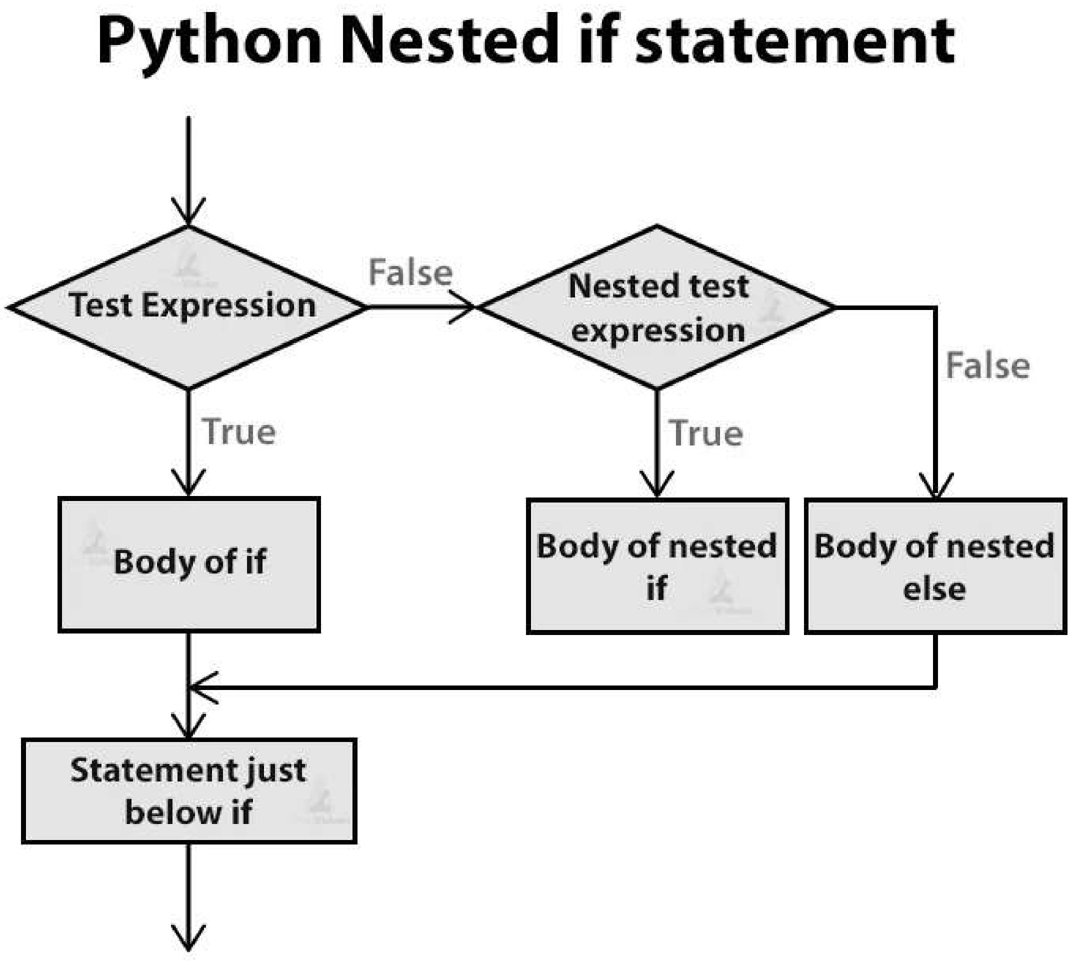
Python Nested if statement

In very simple words, Nested if statements is an if statement inside another if statement. Python allows us to stack any number of if statements inside the block of another if statements. They are useful when we need to make a series of decisions.

Syntax for Nested if statement:



```
if (expression):
  if(expression):
    Statement of nested if
  else:
    Statement of nested if else
  Statement of outer if
Statement outside if block
```



```
In [12]: #Example of Nested if statement
num1 = int( input())
num2 = int( input())
if( num1>= num2):
    if(num1 == num2):
        print(f'{num1} and {num2} are equal')
    else:
        print(f'{num1} is greater than {num2}')
else:
    print(f'{num1} is smaller than {num2}')
```

```
2
3
2 is smaller than 3
```

```
In [13]: #Example of Nested if statement
num1 = int( input())
num2 = int( input())
if( num1>= num2):
    if(num1 == num2):
        print(f'{num1} and {num2} are equal')
    else:
        print(f'{num1} is greater than {num2}')
else:
    print(f'{num1} is smaller than {num2}')
```

```
10
10
10 and 10 are equal
```

Iterators in Python

Python programming language provides the following types of loops to handle looping requirements. Python provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

While Loop in Python In python, a while loop is used to execute a block of statements repeatedly until a given condition is satisfied. And when the condition becomes false, the line immediately after the loop in the program is executed.

Syntax :

```
while expression:
    statement(s)
```

All the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code. Python uses indentation as its method of grouping statements.

In [14]:

```
#Example of while loop in Python

count = 0
while (count < 3):
    count = count + 1
    print("Hello All of you")
```

```
Hello All of you
Hello All of you
Hello All of you
```

In [15]:

```
#Using else statement with while loops
#As discussed above, while loop executes the block until a condition is satisfied.

#The else clause is only executed when your while condition becomes false. If you b

#If else like this:
# Python program to illustrate
# combining else with while
count = 0
while (count < 3):
    count = count + 1
    print("Hello Geek")
else:
    print("In Else Block")
```

```
Hello Geek
Hello Geek
Hello Geek
In Else Block
```

For Loop in Python

For loops are used for sequential traversal. For example: traversing a list or string or array etc. In Python, there is no C style for loop, i.e., for (i=0; i<n; i++). There is “for in” loop which

is similar to for each loop in other languages. Let us learn how to use for in loop for sequential traversals.

Syntax for for loop:

```
for iterator_var in sequence:
    statements(s)
```

In [16]:
```
#Example for for loop:
# Python program to illustrate
# Iterating over range 0 to n-1

n = 4
for i in range(0, n):
    print(i)
```

```
0
1
2
3
```

Nested Loop

Python programming language allows to use one loop inside another loop.

Syntax for nested for loop:

```
for iterator_var in sequence:
   for iterator_var in sequence:
       statements(s)
       statements(s)
```

In [18]:
```
#Example for nested loop
# Python program to illustrate
# nested for loops in Python

for i in range(1, 5):
    for j in range(i):
        print(i, end=' ')
    print()
```

```
1
2 2
3 3 3
4 4 4 4
```

In [22]:
```
#Example for loop with else loop:
#Write a Python program to print all numbers from 0 to 6 except 3 and 6.
for i in range(1,7):
    if (i==3 or i==6):
        continue
    else:
        print(i)
```

```
1
2
4
5
```

Loop Control Statements

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

Continue Statement: It returns the control to the beginning of the loop.

In [24]:
```
#Example of continue statement:
# Prints all letters except 'e' and 's'
for letter in 'PythonPythonPython':
    if letter == 'P' or letter == 'y':
         continue
    print ('Current Letter :', letter)
    var = 10
```

```
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
```

In [25]:
```
#Example 2 of continue statement:
for i in range(10):
    if i%2==0:
        continue
    print(i)
```

```
1
3
5
7
9
```

Break Statement:

It brings control out of the loop

In [26]:
```
#Example of break statement:
for i in range(10):
    if i==7:
        print("processing is enough..plz break")
        break
    print(i)
```

```
0
1
2
3
4
5
6
processing is enough..plz break
```

In [28]:
```python
#Example 2 for break statements
for letter in 'teekforteek':

    # break the loop as soon it sees 'e'
    # or 's'
    if letter == 'e' or letter == 'k':
         break
print('Current Letter :', letter)
```

```
Current Letter : e
```

Pass Statement

Pass Statement: ❖ pass is a keyword in Python. ❖ In our programming syntactically if block is required which won't do anything then we can define that empty block with pass keyword. ❖ Pass is an empty statement. ❖ It is null statement. ❖ It won't do anything.

In [31]:
```python
#Example of pass statement:
for i in range(20):
    if i%9==0:
        print(i)
    else:
        pass
```

```
0
9
18
```

In [32]:
```python
# An empty loop
for letter in 'teeksforteeks':
    pass
print('Last Letter :', letter)
```

```
Last Letter : s
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Prepared by Abhi Shah