UNIT III - Conditional Statements & Loop Statements

3.1 If Statement

If- else Statement

If-elif-else Statement

Nested if –else Statement

3.2 Loop Statement

For Loop

While Loop

Nested loops

Break, Continue, Pass statements

The primary flow control constructs in Python include:

Conditional Statements (if, if else, if elif else , nested if else Statements).

Iterative(looping) Statements (for loop, while loop).

Transfer Statements (break, continue, pass).

Conditional Statements: -

- ✓ In Python, conditional statements act depending on whether a given condition is true or false.
- ✓ You can execute different blocks of codes depending on the outcome of a condition.

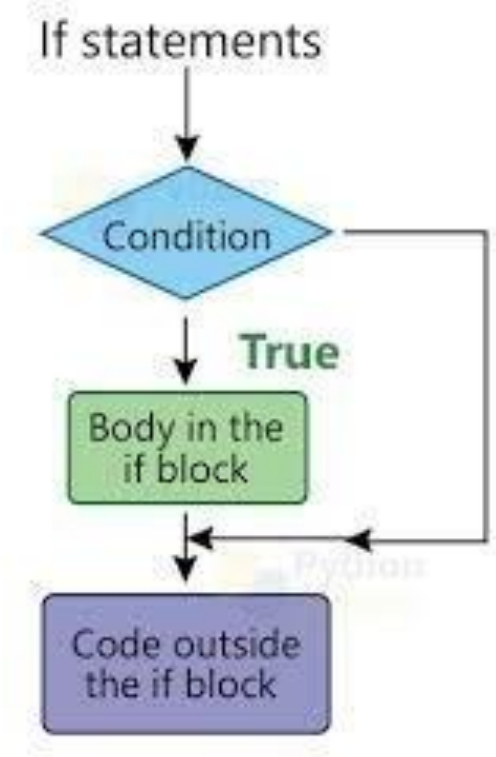
1. if statement
2. if-else

3. if-elif-else
4. nested if-else

1…if Statement: -

- ✓ If statement is the simplest form of a conditional statement.
- ✓ Executes a block of code if its condition is true.

Flowchart of Python if statement: -



Syntax:

if condition:

statement 1

statement n

Example:

age=20

```
if age>=15:
    print("Eligible to vote")
```

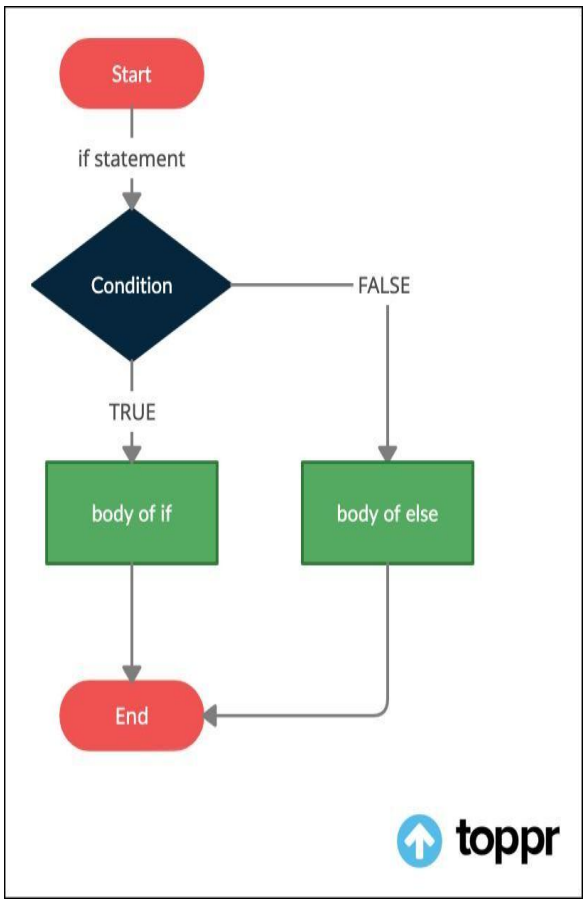
Output:

Eligible to vote

2…if else Statement: -

When condition is True Then Print if block otherwise print else block.

Flowchart of if else:



- **Syntax:**

if condition:

Statement 1

Statement n

else:

Statement 1

Statement n

Example:

```
x = 5
if x > 1:
   print("x is greater than 5")
else:
   print("x is not greater than 5")
```

3…if elif else Statement: -

Evaluates multiple conditions.

Executes the block of the first true condition, or the else block if none are true.

Syntax:

if condition:

Statement

elif condition:

Statement

.

.

else:

Statement

Example:

```
x = 5
if x > 1:
   print("x is greater than 5")
elif x == 6:
   print("x is equal to 5")
else:
   print("x is less than 5")
```

4…Nested if..else

Nested if..else means an if-else statement inside another if statement.

We can use nested if statements to check conditions within conditions.

Syntax:

```
if condition1:
    # Code to execute if condition1 is true
if condition2:
    # Code to execute if both condition1 and condition2 are true
```

Example:

```
i = 5;
if i != 0:

   if i > 0:
      print("Positive")

   if i < 6:
      print("Negative")
else:
   print("Zero")
```

Looping Statements (Iterative statements):

If we want to execute Group of statements Multiple times, then we use Iterative statements.

Python programming language provides two types of loops to handle looping requirements.

while loop

for loop

For loop:-

- A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- This is less like the **for keyword** in other programming languages.
- With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.
- ***Syntax:***

for iterator in sequence:

statement (body)

-

statement (body)

Explanation of Syntax:iterator:

In each iteration, this variable takes the value of the next item in the sequence.

Example:

```
numbers = [1, 2, 3, 4, 5]      # List of numbers
for x in numbers:
 print(x , end="….")         # Output:1,2,3,4,5….


fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

While Loop:-

With the while loop we can execute a set of statements as long as a condition is true.

Syntax:

while condition:

Statement 1

-

Statement n

Example:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Output:

1

2

3

4

5

Transfer Statements: -

In Python, transfer statements are used to alter the flow of control in a program.

There are three main types of transfer statements:

1. break
2. continue
3. pass

break: -

we can use break statement inside loops to break loop execution based on some condition.

Example:

```
for i in range(10):
    print(i)
    if i==5:
      print("processing is enough...please break the loop")
      break
```

continue: -

we can use continue statement to skip current(present) iteration and continue next iteration if the condition is satisfied.

If the condition is not satisfied, then it allows the implementation of the current iteration.

Example:

```
for i in range (10):
    print(i)
    if i%1==0:
      continue
```

pass: -

pass is a keyword in python. pass is used to define an empty block of code or a class.

Example:

```
x=10
if x>5:
    pass
else:
    print("X is 5 or less")
```