Unit-3

Control Statements:

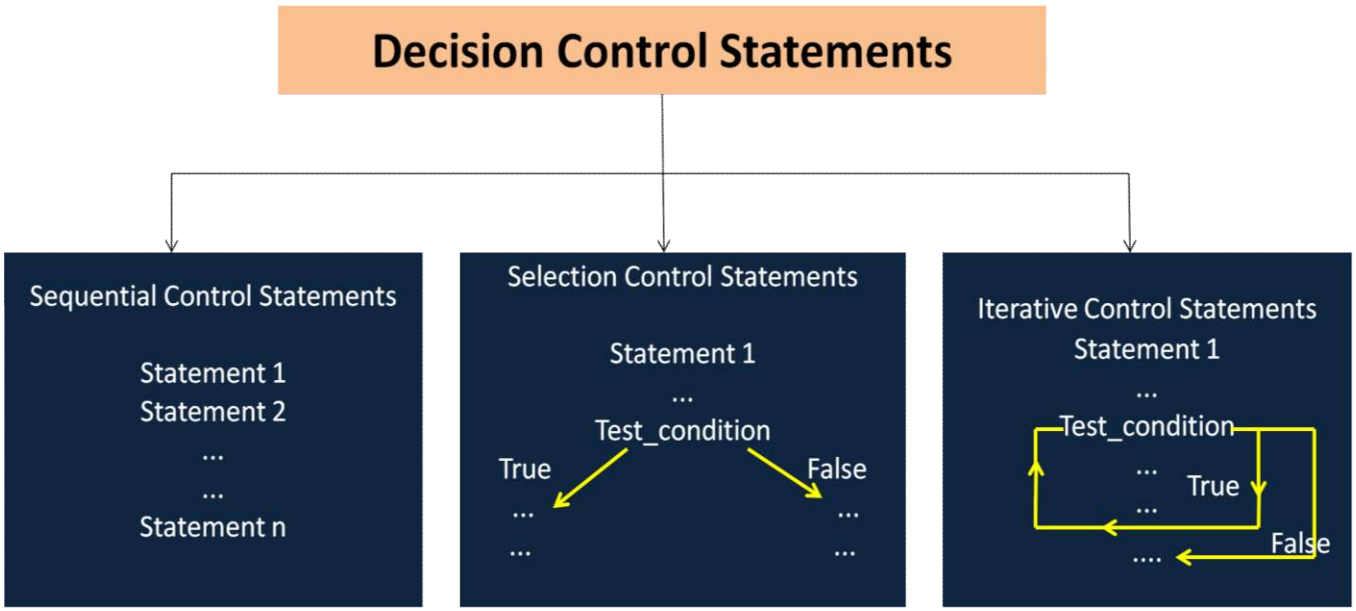
A control statement is a statement that determines the control flow of a set of instructions. It decides the sequence in which the instructions in a program are to be executed. A control statement can either comprise of one or more instructions.

The three fundamental methods of control flow in a programming language are
i) Sequential Control Statements
ii) Selection Control Statements
iii) Iterative Control Statements

i) Sequential Control Statements: The code of Python program is executed sequentially from the first line of the program to its last line. That is, the second statement is executed after the first, the third statement is executed after the second, so on and so forth. This method is known as sequential Control flow and these statements called as sequential control statements.

ii) Selection Control Statements: Some cases, execute only a set of statements called as selection control statements. This method is known as selection control flow.

iii) Iterative Control Statements: execute a set of statements repeatedly called as Iterative control statements. This method is known as Iterative control flow.



SELECTION/CONDITIONAL CONTROL STATEMENTS

The decision control statements usually jumps from one part of the code to another depending on whether a particular condition is satisfied or not. That is, they allow you to execute statements selectively on certain decisions. Such type of decision control statements are known as selection control statements or conditional branching statements. Python supports different types of conditional branching statements which are as follows:

- If statement
- If –else statement
- Nested if statement
- If-elif-else statement

If Statement: An if statement is a selection control statement based on the value of a given Boolean expression. The if structure may contain 1 statement or n statements enclosed within the if block. First, the test expression(Boolean expression) is evaluated. If the test expression is True, the statement of if block are executed, otherwise these statements will be skipped and jump to next statement which is outside of the if block.

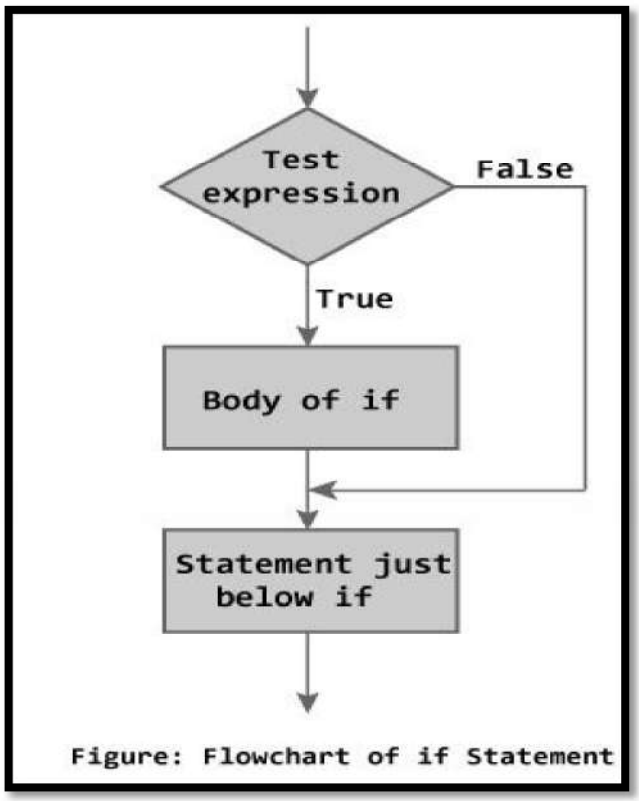


Figure: Flowchart of if Statement

Syntax of if statement

```
if test_expression:
        Statement1
        Statement2
        …….
        Statement n
Statement x
```

Example 1: Program to increment a number if it is positive.

```
x=10
if (x>0):
        x=x+1
print (x)
```

Output:

x=11

Example 2: Write a program to determine whether a person is eligible to vote.

```
age=int(input("Enter the age: "))
```

```
if (age>18):
        print ("You are eligible to vote")
```

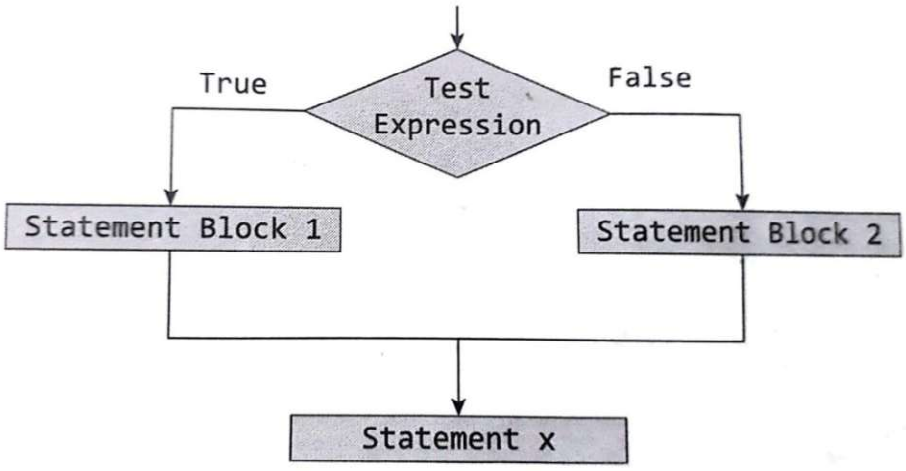
Output:

Enter the age: 35
You are eligible to vote

if –else Statement: An if-else construct, first the test expression (Boolean expression) is evaluated. If the expression is True, statement block 1 is executed and statement block 2 is skipped. Otherwise, if the expression is False, statement block 2 is executed and statement block 1 is ignored.

Syntax of if statement

```
if test_expression:
        Statement block 1
else:
        Statement block 2
Statement x
```



Example 3: Write a program to determine whether a person is eligible to vote or not. If he is not eligible, display how many years are left to be eligible.

```
age=int(input("Enter the age:"))
if (age>=18):
        print ("You are eligible to vote")
else:
        yrs=18-age
        print ("You have to wait for another"+ str(yrs)+ "years to cast your vote")
```

Output:

Enter the age: 10
You have to wait for another 8 years to cast your vote.

Example 4: Write a program to find whether the given number is even or odd.

```
num=int(input("Enter any number :"))
if (num%2==0):
        print (num, "is even")
else:
        print (num, "is odd")
```

Output:
Enter any number: 125
125 is odd

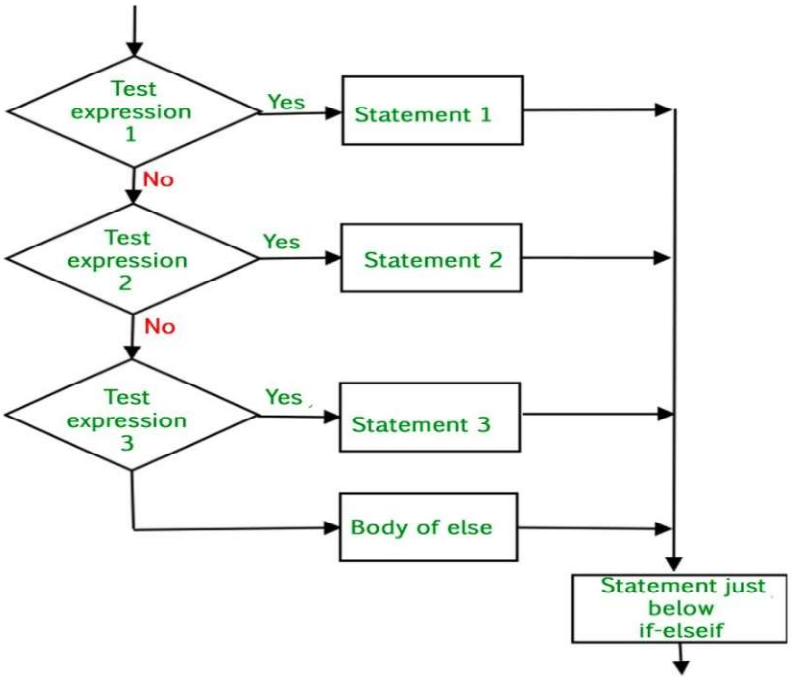
Nested if statements: To perform more complex check, if statement can be nested, that is can be placed one inside the other. In such a case, the inner if statement is the statement part of the outer one. Nested if statements are used to check if more than one condition is satisfied.

Example 5: Write a python program to find the given number is positive or negative number, if positive number then compare with 100, if number is greater than 100, output display as "high" otherwise display as "low".

```
num = float(input("Enter a number: "))
if num >0:
   if num >100:
      print("High")
   else:
      print("Low")
else:
   print("Negative number")
```

Output:
Enter a number: 5
low

if-elif-else Statement: The elif statement allows to check multiple expressions and executes block of statements wherever the condition returns as True. From there, it exit from the entire *if-elif-else* block. If any of the expression not returns as true, then it executes the *else* block code.




```
Syntax of if-elif-else
Statement

if ( test expression 1)
   statement block 1
elif ( test expression 2)
   statement block 2
.............................
elif (test expression N)
   statement block N
else
   Statement Block X
Statement Y
```

Example 6: Program to test whether a number entered by the user is negative, positive or equal to Zero?

```
num=int(input("Enter any number"))
if (num==0):
        print ("The value is equal to zero")
elif (num>0):
        print ("The number is positive")
else:
        print ("The number is negative")
```

Output:

```
Enter any number : -10
The number is negative
```

Exercise:

1. Write a program to find larger of two numbers.
2. A Company decides to give bonus to all its employees on Diwali. A 5% bonus on salary is given to the male workers and 10% bonus on salary to the female workers. Write a Python program to enter the salary of the employee and gender of the employee. If the salary of the employee is less than Rs/- 10,000 then the employee gets an extra 2% bonus on salary. Calculate the bonus that has to be given to the employee and display the salary that the employee will get.
3. Write a program to find whether a given year is a leap year or not.
4. Write a program to determine whether the character entered is a vowel or not.
5. Write a program to find the greatest number from the three numbers.
6. Write a program that prompts the user to enter a number between 1-7 and then displays the corresponding day of the week.
7. Write a program to calculate tax given the following conditions:
   If income is less than 1,50,000 then no tax
   If taxable income is 1,50,001 – 3,00,000 then charge 10% tax
   If taxable income is 3,00,001 – 5,00,000 then charge 20% tax
   If taxable income is above 5,00,001 then charge 30% tax
8. Write a program to enter the marks of a student in four subjects. Then calculate the total and aggregate, and display the grade obtained by the student. If the student scores an aggregate greater than 75%, then the grade is Distinction. If aggregate is 60>= and <75, then the grade is First Division. If aggregate is 50>= and <60, then the grade is Second Division. If aggregate is 40>= and <50, then the grade is Third Division. Else the grade is fail.
9. Write a program to calculate roots of a quadratic equation.
10. Write a program to take input from the user and then check whether it is a number or a character. If it is a character, determine whether it is in uppercase or lowercase.
11. Write a program that prompts users to enter a character (O, A,B,C,F). Then using if-elif-else construct display Outstanding, Very Good, Good, Average and Fail respectively.
12. Write a program to read two numbers. Then find out whether the first number is a multiple of the

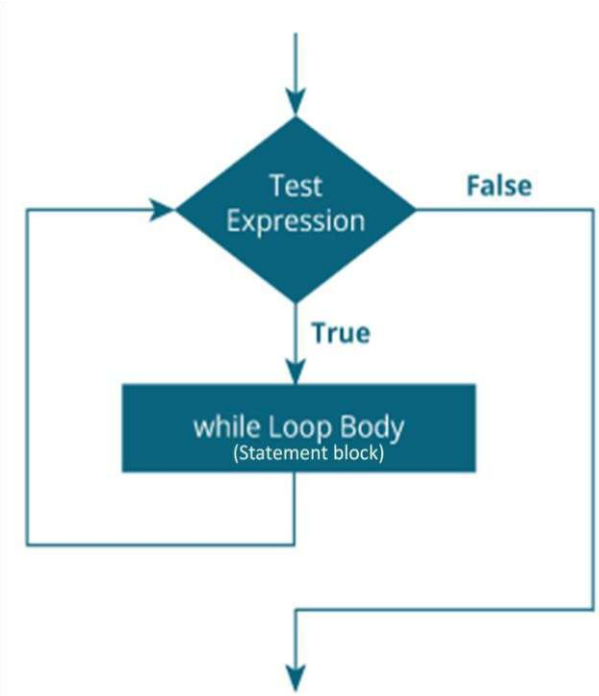
second number.

ITERATIVE CONTROL STATEMENTS/LOOPING STATEMENTS

Iterative statements are decision control statements that are used to repeat the execution of a list/set/group of statements. Python language supports two types of iterative statements **While** loop and **for** loop.

While loop:

The while loop provides a mechanism to repeat one or more statements while a particular condition is True.



While loop, the condition is tested before any of the statements in the statement block is executed. If the condition is True, only then the statements will be executed otherwise if the condition is False, it is jump to next statement which is outside of the while loop.

Note: If the condition of a while loop is never updated and condition never become False, then the computer will run into an infinite loop.

Syntax of While Loop:

```
statement x
while (condition):
        Statement  block
Statement y
```

Example 7: Write a python program to print first 10 numbers using a while loop.

```
i=0
while (i<=10):
        print (i)
        i=i+1
```

Output:

```
1
2
3
4
5
6
7
8
```

9
10

Example 8: Write a python program to print first 10 numbers using a while loop on the same line.

```
i=0
while (i<=10):
        print (i, end= ' ')
        i=i+1
```

Output:

1 2 3 4 5 6 7 8 9 10

Example 9: Program to display the Fibonacci sequence up to n-th term where n is provided by the user

```
nterms = 10
n1 = 0
n2 = 1
count = 0
if nterms <= 0:
        print("Please enter a positive integer")
elif nterms == 1:
        print("Fibonacci sequence upto",nterms,":")
        print(n1)
else:
        print("Fibonacci sequence upto",nterms,":")
        while count < nterms:
                print(n1,end=' , ')
                nth = n1 + n2
                n1 = n2
                n2 = nth
                count += 1
```

Output:

Fibonacci sequence upto 10 :

0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

Note: In simple terms, end specifies the values which have to be printed after the print statement has been executed. In above example, the values printed on the same line, used end with a separator. You can specify any separator like tab (\t), space, comma, etc., with end.

Exercise:

1. Write a program to calculate the sum and average of first 10 numbers.

2. Write a program to print 20 horizontal asterisks(*).
3. Write a program to calculate the sum of numbers from m to n.
4. Write a program to read the numbers until -1 is encountered. Also count the negative, positive and zeros entered by user.
5. Write a program to read the numbers until -1 is encountered. Find the average of positive number and negative numbers entered by the user.
6. Write a program to find whether the given number is an Armstrong Number or not.
7. Write a program to enter a decimal number. Calculate and display the binary equivalent of this number.
8. Write a program to enter a binary number and convert it into decimal number.
9. Write a program to read a character until a * is encountered. Also count the number of uppercases, lowercase and numbers entered by the users.
10. Write a program to enter a number and then calculate the sum of the digits.
11. Write a program to calculate GCD of two numbers.
12. Write a program to print the reverse of a number.
13. Write a program to read a number from the user and find whether it is a palindrome number or not.
14. Write program using a while loop that asks the user for a number and prints a countdown from that number to zero.
15. Write a program that prompts users to enter numbers. Once the user enters -1, it displays the count, sum and average of even numbers and that of odd numbers.

For loop:

The for loop provides a mechanism to repeat a task(set of statements) until a particular condition is true.

The For loop is known as a determine loop because the programmer knows exactly how many times the loop will repeat. The number of times the loop has to be executed can be determined mathematically checking the logic of the loop.

The for …… in statement is a looping statement used in python to iterate over a sequece(list, tuple, string) of objects i.e., go through each item in a sequence. Sequence means an ordered collecion of itmes.

Syntax of for Loop

for val in sequence:

    Body of for (Statement Block)

Note: The for loop is widely used to execute a single or a group of statements.

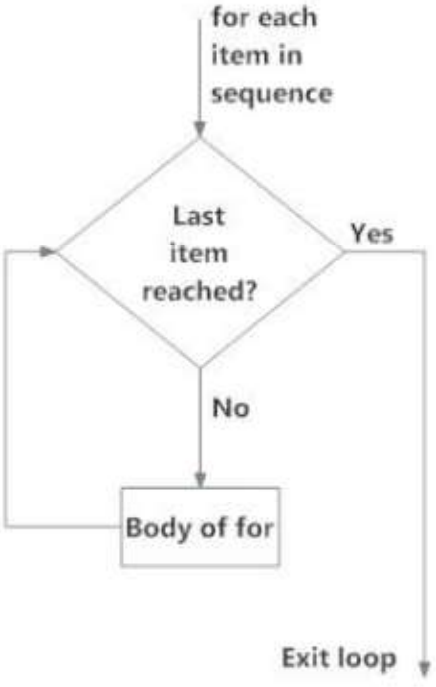


Fig: operation of for loop

The range function():

The range() is a built-in function in Python that is used to iterate over a sequence of numbers. The syntax of range() is

range(beginning, ending, [step size])

The range() produces a sequence of numbers starting with beginning (inclusive) and ending with one less than the numeber end. The step argument is optional. By default, every number in range is incremented by 1 but we can specify a different increment using step.

Note: Step size can be either positive or negative but it cannot be equal to zero.

Example 10:

Program:

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

Output: 1 2 3 4

Example 11:

Program:

```
for i in range(1, 10, 2):
    print (i,    end= "   " )
```

Output: 1 3 5 7 9

Example 12:

Program:

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

Output: 0 1 2 3 4 5 6 7 8 9

Example 13:

Program:

```
for i in range(1, 15):
    print (i,    end= "   " )
```

Output: 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Example 14:

Program:

```
for i in range(1, 20,3):
        print (i,    end= "   " )
```

Output: 1 4 7 10 13 16 19

Exercise – For Loop:

1. Write a program using for loop to calcuate the average of first n natural numbers.
2. Write a program to print the multiplication table of n, where n is entered by the user.
3. Write a program using for loop to print all the numbers from m-n thereby classifying them as even or odd.
4. Write a program using for loop to calculate factorial of a number.
5. Write a program using for loop to caculate pow(x,n)
6. Write a program that display all leap years from 1800 – 1900
7. Write a program to sum the series – 1+ 1/2 + 1/3 + ---- + 1/n.
8. Write a program to sum the series - $1 + 1/2^2 + 1/3^2 + ----+ 1/n^2$.
9. Write a program to sum the series – 1/2 + 2/3 + 3/4 +----------+ n/(n+1).
10. Write a program to sum the series -$1/1+ 2^2/2+3^2/3+$ ----------n^2/n.
11. Write a program to calculate sum of cubes of numbers from 1-n.
12. Write a program to sum of squares of even numbers.

Nested Loops:

Nested loops, that is, loops can be placed inside other loops.This feature works any loop like while loop as well as for loop, but it is most commonly used with the for loop, because this is easiest to control.

Exercise:

1. Write a program to print the following pattern.

```
* * * * *
*         *
*         *
*         *
* * * * *
```

Python Loop Control Statements:

Loop control statements used to exit from the loop or skip specific part of the loop when it meets a specific conditions. In python we have following loop control statements.

1) Break Statement

2)Continue Statement

3) Pass Statement

break STATEMENT:

Break can be used to unconditionally jump out of the loop. It terminates the execution of the loop. Break can be used in while loop and for loop. Break is mostly required, when because of some external condition, we need to exit from a loop.

Example 15: Program to using the break statement

```
i=1
while (i<=10):
        print (i, end=" ")
        if (i==5):
                break
        i=i+1
print ()
print ("completed")
```

Output:

```
1  2  3  4  5
completed
```

Continue Statement:

This statement is used to tell Python to skip the rest of the statements of the current loop block and to move to next iteration, of the loop. Continue will return back the control to the beginning of the loop. This can also be used with both while and for statement.

Example 16: Program to using the continue statement

```
for i in range(1,11):
            if (i==5):
                  continue
             print (i, end=" ")
print ("\n completed")
```

Output:

```
1 2  3  4  6 7  8   9   10
```

Exercise:

1. Write a program to classify a given number as prime or composite.
2. Write a program to read the numbers until -1 is encountered. Count the number of prime numbers and composite numbers entered by the user.
3. Write a Python program that prints all the numbers from 0 to 6 except 3 and 6.

4. Write a Python program to Print first 100 prime numbers.

Pass Statement:

The pass statement is used when a statement is required syntactically but no command or code has to be executed. It specifies a null operation or No Operation (NOP) statement. Nothing happens when the pass statement is executed.

Example17: Program to demonstrate pass statement

```
for letter in “Hello”:
        pass          #The statement is doing nothing
        print (“PASS:”, letter)
print (“Done”)
```

Output:

```
PASS: H
PASS: E
PASS: L
PASS: L
PASS: O
Done
```

The pass statement is used as a placeholder. For example, if we have a loop that is not implement yet, but we may wish to write some piece of code in it in the future. In such cases, pass statement can be written because we cannot have an empty body of the loop. Though the pass statement will not do anything but it will make the program syntactically correct.

Note: The difference between comment and pass statements is, pass is null statement that is executed by python interpreter, comment is non-executable statement that is ignored(not executed) by python interpreter.

The else STATEMENT used with Loops:

Unlike C and C++, in python have the **else** statement associated with a loop statements.

1) In for-else statement, the **else** statement is executed when the loop has completed iterating. **Break** statement can be used to stop a for loop. In such case, the else statement is ignored. Remaining cases, for loop’s **else** part is executed if no break occurs.

Example 18:

```
for i in range(1,11):
        print (i, end="")
else:
```

```
        print ("\n  Done")
```

Output:
1 2 3 4 5 6 7 8 9 10
Done

Example 19:

```
for i in range(1,20):
        if (i==5):
                break
        print (i, end="")
else:
        print ("\n  Done")
```

Output:
1 2 3 4

2) **else** statement with the while loop, the else statement is executed when the conditions becomes False. The while loop can be terminated with a break statement. In such case, the else part is ignored. Hence, a while loop's else part runs if no break occurs and the condition is false.

Example 20:

```
i=1
while (i<0):
        print (i)
        i=i-1
else:
        print (i, "is not negative so loop did not execute")
```

Output:
1 is not negative so loop did not execute

Example 21:

```
i=1
while (i<10):
        if (i==6):
                break
        print (i, end=" ")
        i=i+1
else:
        print ("Completed")
```

Output:
1 2 3 4 5

Assignment-III

1. Write a detail notes on conditional/selectional branching statements supported by python?
2. Explain *for* and *while* loop in python along with syntax, flowchart and an example?
3. Write a program to print the following pattern.

RGUKT 1 – 1 2 3 4 5
RGUKT 2 – 1 2 3 4 5
RGUKT 3 – 1 2 3 4 5
RGUKT 4 – 1 2 3 4 5
RGUKT 5 – 1 2 3 4 5

4. Write a program to print the following pattern.

****
****
****
****
****
****

5. Write a program to print the following pattern.

*
**
***
****
*****

6. Write a program to print the following pattern.

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

7. Write a program to print the following pattern.

1
22
333
4444

55555

8. Write a program to print the following pattern.

```
1
2 3
4 5 6
7 8 9 10
```

9. Write a program to print the following pattern.

```
    1
   12
  123
 1234
12345
```

10. Write a program to print the following pattern.

```
          1
        1  2  1
     1  2  3  2  1
  1  2  3  4  3  2  1
1 2  3  4   5  4  3  2 1
```

11. Write a program to print the following pattern.

```
          1
        2  2
      3  3  3
    4  4  4  4
  5  5  5  5  5
```

12. Write a program to print the following pattern.

```
$ * * * *
* $     *
*   $   *
*     $ *
* * * * $
```

Assignment-IV

1. Explain range() function along with an example?

2. Explain break statement with the help of an example?

3. Explain continue statement with the help of an example?

4. Explain about pass statement in Python?

5. Write short notes on for-else and while-else?

6. Write a Python program to print prime numbers between 500 to 600.

Unit-4

Functions

Function is a group of related statements that perform a specific task. Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable. It avoids repetition and makes code reusable.

Syntax of Function

```
def function_name(parameters):          #Keyword def marks the start of function header.
    statement(s)
```

Basically, we can divide functions into the following two types:

1. Built-in functions - Functions that are built into Python.
2. User-defined functions - Functions defined by the users themselves.

Built-in Functions

Functions that come built into the Python language itself are called built-in functions and are readily available to us.

Functions like *print(), input(), len()* etc. that we have been using, are some examples of the built-in function. There are 68 built-in functions defined in Python 3.7.4

Examples:

Name	Description	Example
max(x, y, z, ….)	It returns the largest of its arguments: where x, y and z are numeric variable/expression.	>>>max(80, 100, 1000) 1000 >>>max(-80, -20, -10)