Sinhgad Institutes | Date

Assignment No. 1.

Title - To calculate salary of an employee given his basic pay (take as input from user). Calculate gross salary of employee. Let HRA be 10% of basic pay & TA be 5% of basic pay. Let employee pay professional tax as 2% of total salary. Calculate net salary payable after deductions.

Date of completion	
Remark	
Sign. of staff	

Problem Description-

The program take as input salary of an employee given his basic pay from user & calculate gross salary of employee using HRA, TA & TAX

Theory-

Explanation-

Decision making statements in programming languages decides the direction of flow of program execution. Desision making statements available in Python are :

* if statement
* if... else statements
* nested if statements
* if-elif ladder

Syntax:

```
if condition:
        # statements to execute if
        # condition is true
```

Example:

```
i = 10
if (i > 15):
print ("10 is less than 15")
print ("I am Not in if")
```

Output:

```
I am Not in if
```

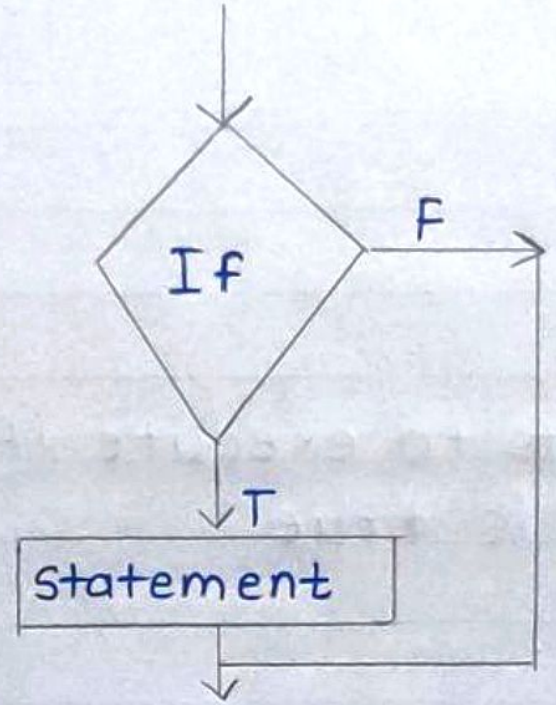
Nested if statements:

Syntax:

```
if (condition 1):
    # Executes when condition 1 is true
    if (condition 2):
            # Executes when condition 2 is true
    # if block is end here
# if block is end here
```

Example:

```
i = 10
if (i = = 10):
        # First if statement
        if (i < 15):
            print ("i is smaller than 15")
    # Nested - if statement
```


If
F
T
Statement

```
# Will only be executed if statement above
# it is true
if (i <12):
    print ("i is smaller than 12 too")
else :
    print ("i is greater than 15")
```

Output :

```
i is smaller than 15
i is smaller than 12 too
```

IF - elif ladder

Syntax :

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

Example :

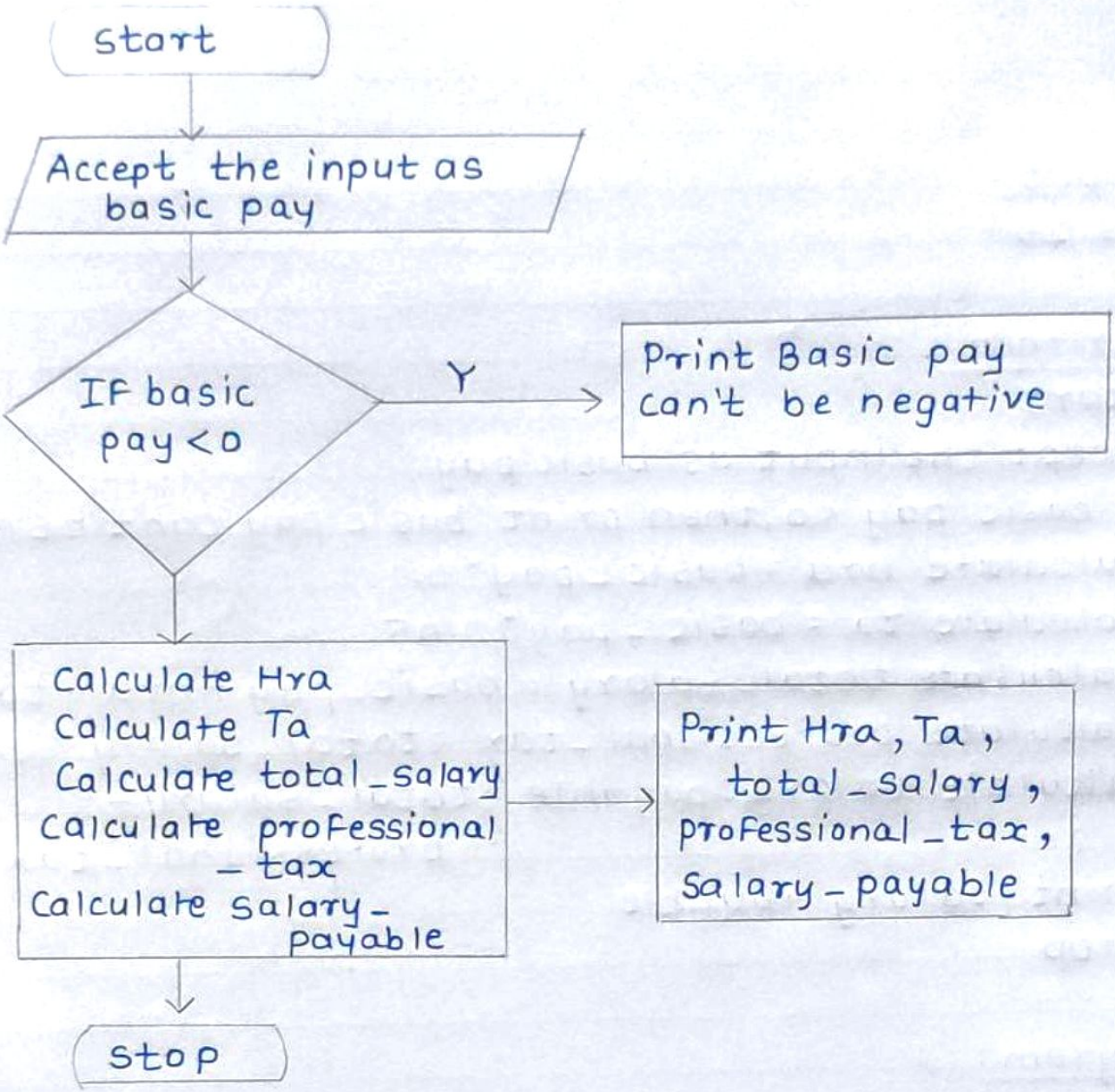
```
i = 20
if (i = =10):
    print ("i is 10")
elif (i = = 15):
    print ("i is 15")
elif (i = = 20)
    print ("i is 20")
else:
    print ("i is not present")
```

Output:
i is 20

Algorithm:
1. Start
2. Accept the input as basic pay.
3. If basic pay <0, then print basic pay can't be negative
4. Calculate hra = basic_pay * 0.1
5. Calculate ta = basic_pay * 0.05
6. Calculate total_salary = basic_pay + hra + ta
7. Calculate professional_tax = total_salary * 0.02
8. Calculate salary_payable = total_salary - professional_tax
9. print Salary Payable
10. stop

Program:
Code:

```
basic_pay = input("Enter your basic pay:")
        basic_pay = Float (basic_pay)
        if basic_pay < 0:        # basic pay cannot be
            less than zero
            print("Basic pay can't be negative.")
            exit()
        hra = basic_pay * 0.1 # hra is 10 % of basic pay
        ta = basic_pay * 0.05 # ta is 5% of basic pay
        total_Salary = basic_pay + hra + ta
        professional_tax = total Salary * 0.02
            # professional tax is 2 % of total salary
        print Salary_payable = total_salary -
```


start
Accept the input as basic pay
IF basic pay<0
Y
Print Basic pay can't be negative
Calculate Hra
Calculate Ta
Calculate total_salary
Calculate professional_tax
Calculate salary_payable
Print Hra, Ta, total_salary, professional_tax, Salary_payable
stop

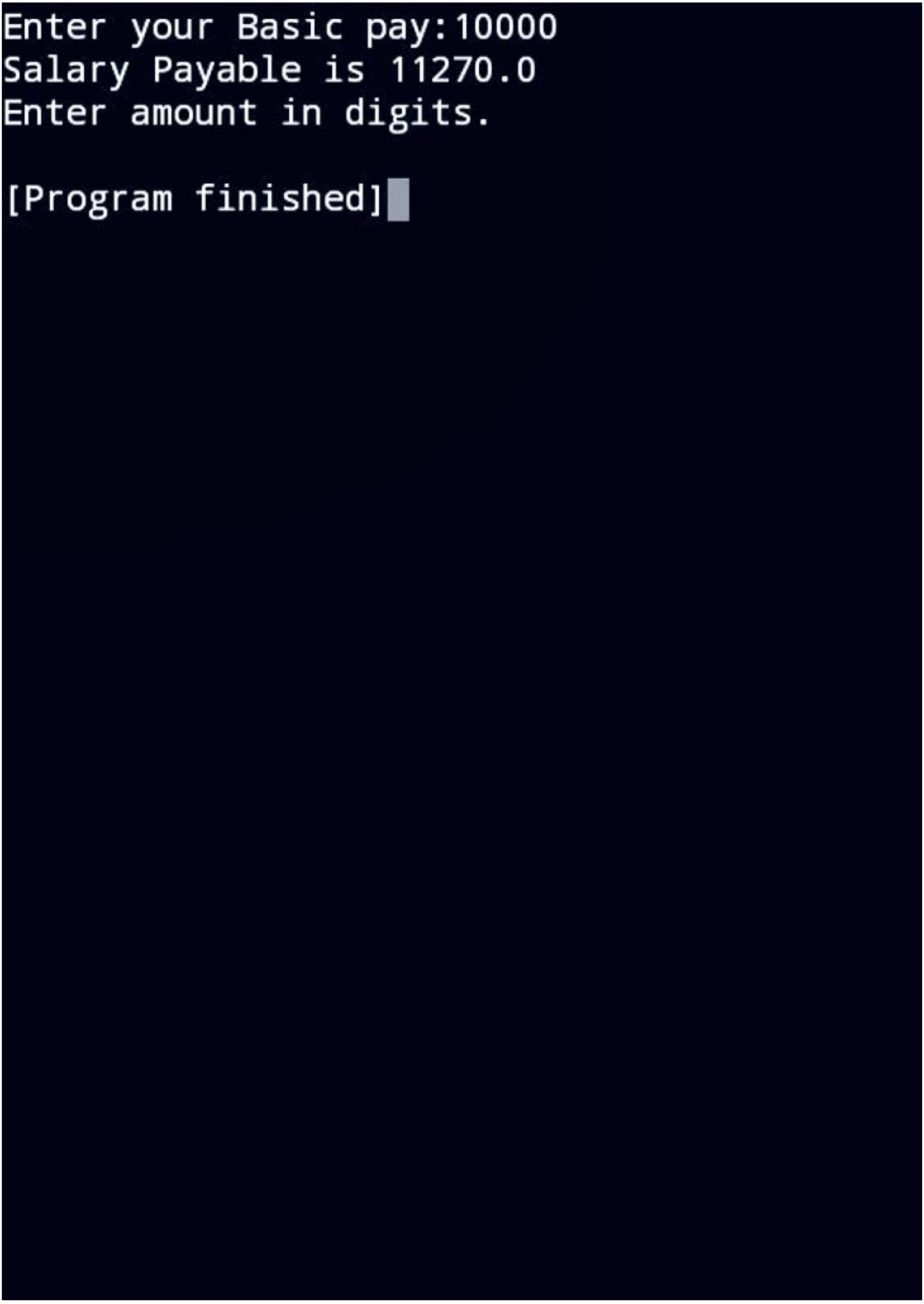
```
                                    -professional_tax
print ("Salary Payable is ", Salary_payable)
print ("Entre amount in digits.")
```

Output:

1] input : 10000
O/P: Salary Payable is 11270.0

2] input : 0
O/P : Salary Payable is 0.0

3] input - 10000
O/P : Basic pay can't be negative


Enter your Basic pay:10000
Salary Payable is 11270.0
Enter amount in digits.
[Program finished]

Sinhgad Institutes | Date

Assignment No. 2

Title- To accept N numbers from user. Compute & display maximum in list, minimum in list, sum & average of numbers.

Date of completion-	
Remark-	
Sign. of staff -	

Problems Description-

The program accept N input numbers from user & perform following operation on list compute & display maximum number in list, minimum number in list, sum & average of numbers.

Theory- Lists are just like the arrays, declared in other languages. Lists need not be homogenous always which makes it is a most powerful tool in Python. A single list may contain Data Types like integers, strings, as well as objects. Lists are also very useful for implementing stacks & queues. Lists are mutable & hence, they can be altered even after their creation.

In Python, list is a type of container in Data Structures, which is used to store multiple data at the same time. Unlike sets, the lists in python are ordered and have a definite count. The elements in a list are indexed according to a definite sequence & the indexing of a list is done with 0

being the first index. Each element in the list has its definite place in the list, which allows duplicating of elements in the list, with each element having its own distinct place & credibity

Creating a List -

List in Python can be created by just placing the sequence inside the square brackets []. Unlike sets, list doesn't need a built-in function for creation of list. A list may contain duplicate values with their distinct position & hence, multiple distinct or duplicate values can be passed as sequence at the time of list creation.

```
# Python program to demonstrate.
# Creation of List.

# Creating a List
 List []
print ("Intial blank List:")
print (List)
```

Output:

```
Intial blank list:
 []
```

```
# Creating a list with
# mixed types of values
# (Having numbers & strings)
List = [1, 2 'Geeks', 4, 'For', 6, 'Geeks']
```

```
print ("\n list with the use of Mixed Values:")
print (list)
```

Output:

```
List with the use of mixed values:
[1, 2, 'Geeks', 4, 'For', 6, 'Geeks']
```

Adding Elements to a list

Elements can be added to the list by using built in append() function. Only one element at a time can be added to the list by using append() method for addition of multiple elements with the append() method, loops are used. Tuples can also be added to the List with the use of append method because tuples are immutable. Unlike sets, lists can also be added to the existing list with the use of append() method. append() method only works for addition of elements at the end of the List, for addition of element at the desired position, insert() method is used. Unlike append() which takes only one argument, insert() method requires two arguments (position, value) Other than append() & insert() methods, there's one more method for addiction of elements, extend(), this method is used to add multiple elements at the same time at the end of the list.

Built-in function with List-

max()- return maximum element of given list
min()- return minimum element of given list

sum()- Sums up the numbers in the list

COMPUTE AND DISPLAY AVERAGE OF NUMBERS

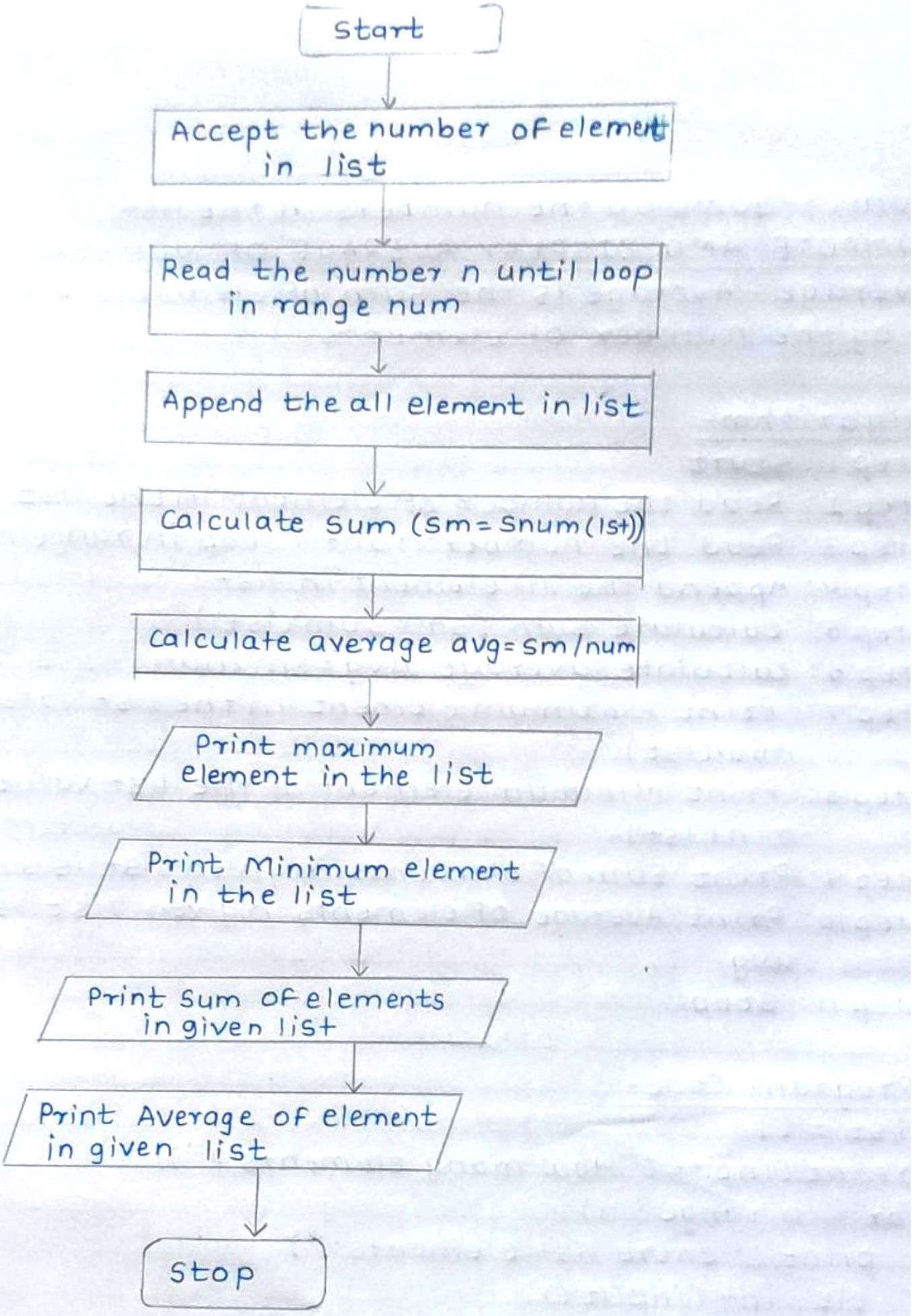
Average - Average is the sum of elements divided by the number of elements.

Algorithm-

Step 1: Start
Step 2: Read the number of element in the list.
Step 3: Read the number n until loop in range num.
Step 4: Append the all element in list.
Step 5: Calculate sum (sm=sum(lst)).
Step 6: Calculate average avg=sm/num
Step 7: Print Maximum element in the list value max(lst).
Step 8: Print Minimum element in the list value min(lst).
Step 9: Print Sum of elements in given list is sum
Step 10: Print Average of elements in given list is avg.
Step 11: Stop.

Program Code-

```
lst = []
n = int(input("How many elements ?"))
For i in range (n):
   print ("entre list elements:")
    ele = int(input())
   lst.append (ele)
   sm = sum (lst)
   avg = sm/n
```


Start
Accept the number of element in list
Read the number n until loop in range num
Append the all element in list
Calculate sum (sm=snum(lst))
Calculate average avg=sm/num
Print maximum element in the list
Print Minimum element in the list
Print sum of elements in given list
Print Average of element in given list
Stop

```
print ("Maximum element is:", max (lst))
print ("Minimum element is:", min (lst))
print ("Summation of element is:", sum (lst))
print ("Average of element is:", avg)
```

Output:

```
How many elements ? 5
Entre list elements:
 5
Entre list elements:
 2
Entre list elements:
 7
Entre list elements:
 9
Entre list elements:
 3
Maximum element is: 9
Minimum element is : 2
Summation of element is : 26
Average of element is : 5.2
```

```
Type "help", "copyright", "credits" or "license" for more information.
>>> lst=[]
>>> n=int(input("Hown many elements?"))
Hown many elements?5
>>> for i in range(n):
...  print("Enter list elements:")
...  ele=int(input())
...  lst.append(ele)
...  sm=sum(lst)
...  avg=sm/n
...
Enter list elements:
5
Enter list elements:
2
Enter list elements:
7
Enter list elements:
9
Enter list elements:
3
>>> print("Maximum elements is:",max(lst))
Maximum elements is: 9
>>> print("Minimum elements is:",min(lst))
Minimum elements is: 2
>>> print("summation of elements is:",sum(lst))
summation of elements is: 26
>>> print("Average pf elements is:",avg)
Average pf elements is: 5.2
>>>
```

Assignment No. 3

Date

Title- To accept student's Five course marks & compute his/her result. student is passing if he/she scores marks equal to and above 40 in each course. If student scores aggregate greater than 75%, then the grade is distinction. If aggregate is 60>= and <75 then the grade if 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.

Date of Completion	
Remark	
Sign of Staff	

Problem Description:

The program is to accept student's Five course marks and compute his/her result. student is passing if he/she scores marks equal to & above 40 in each course. If student scores aggregate greater than 75%, then the grade is distinction. If aggregate is 60>= and <75 then the grade if First division. If aggregate is 50>= and <60 then grade is second division. If aggregate is 40>= and 50, then the grade is third division.

Theory-

Decision Making in Python

There come situations in real life when we need to make some decisions & based on these

decision. we decide what should we do next. Smilar situation arise in programming also where we need to make some decisions & based on these decisions we will execute the next block of code.

Decision making statements in programming language decides the direction of flow of programe excution. Decision making statement available in python are:

- If statement
- If ... else statements
- nested if statements
- if-elif ladder

1] **if statement:**

if statement is the most simple decision making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e. if a certain condition is true then a block of statement is executed otherwise not.

Syntax:

```
if condition:
    # Statements to execute if
    # condition is true
```

Here, condition after evaluation will be either true or false. if statement accepts boolean values- if the value is true then it will execute

the block of statements below it otherwise not. We can use condition with bracket '(')' also.

As we know, python uses identation to identify a block. So the block under an if statement will be identified as shown in the below example

```
if condition:
    Statement 1
Statement 2

# Here if the condition is true, if block
# will consider only statement 1 to be inside
# its block
```

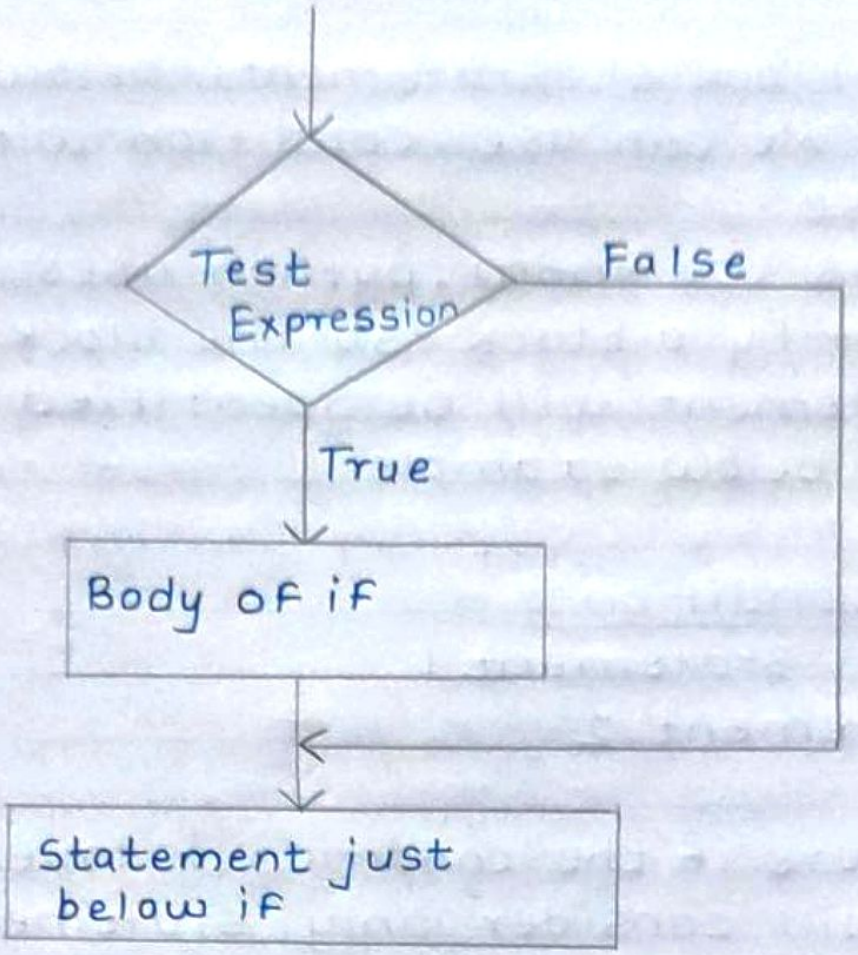
Example:

```
# python program to illustrate IF statement

i = 10
if (i>15)
    print ("10 is less than 15")
print ("I am Not in if")
```

Output:

```
I am Not in if
```

As the condition present in the if statement is false. So, the block below the if statement is not executed.


Test Expression
False
True
Body of if
Statement just below if

2] if-else

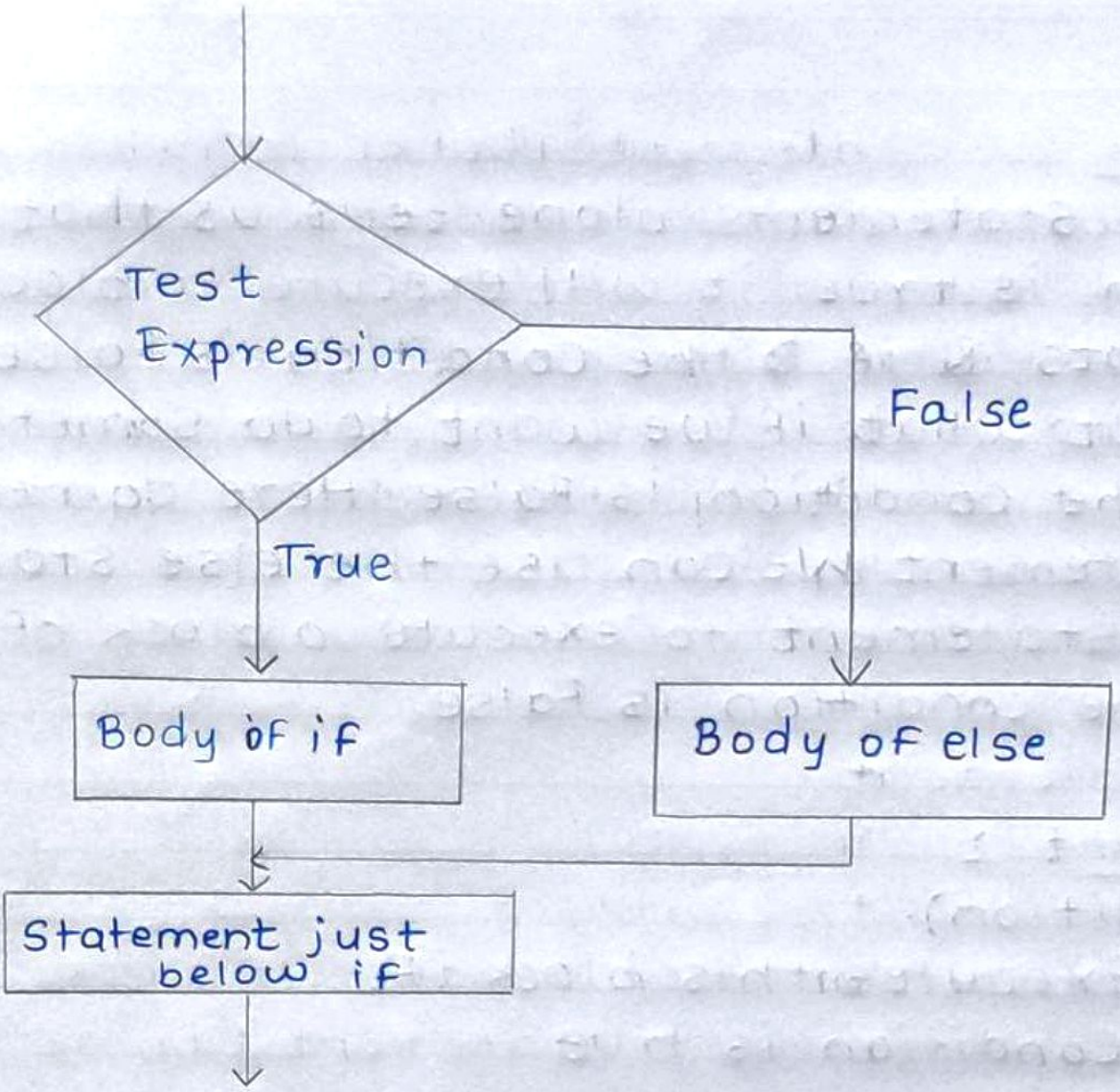
The if statements alone tells us that if a condition is true it will execute a block of statements & if & the condition is false is won't. But what if we want to do something else if the condition is false. Here comes the else statement. We can use the else statement with if statement to execute a block of code when the condition is false.

Syntax-

```
if (condition):
        # Executes this block if
        # condition is true
else:
        # Executes this block if
        # condition is false
```

Example-

```
# python program to illustate IF else statement
#!/usr/bin/python
i = 10
if (i == 10):
        # First if statements
        if (i < 15):
                print ("i'm in if Block")
else:
        print ("i is greater than 15")
        print ("i'm in else Block")
print ("i'm not in if & not in else Block")
```


Test
Expression
False
True
Body of if
Body of else
Statement just
below if

Output:
i is greater than 15
i'm in else block
i'm not in if & not in else Block

The block of code following the else statement is executed as the condition present in the if statement is false after call the statement which is not in block (without spaces).

3] nested-if

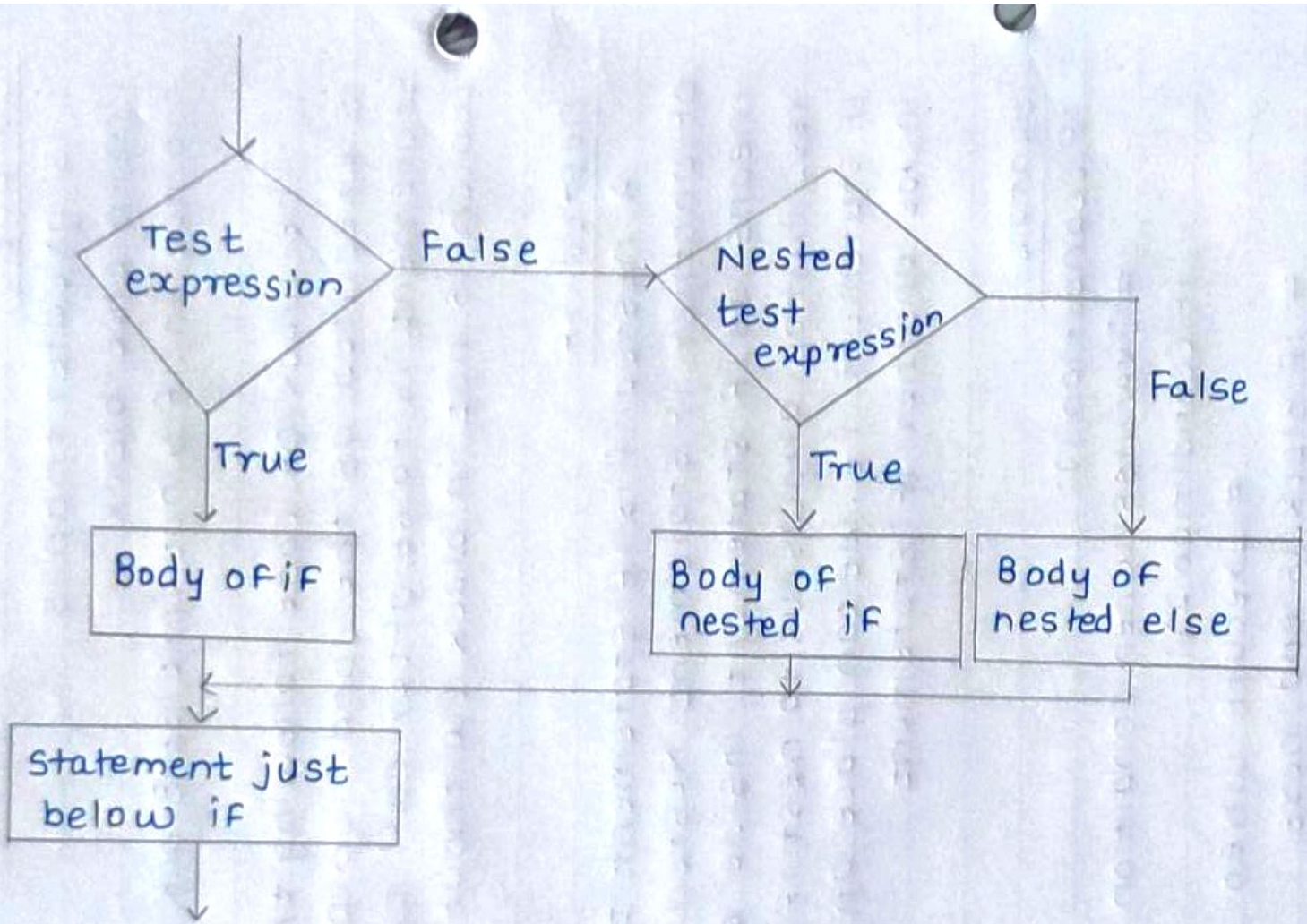
A nested if is if statement that is the target of another if statement. Nested if statements means an if statement inside another if statement Yes, Python allows us to nest if statement within if statement. i.e, we can place an if statement inside another if statement.

Syntax:

```
if (condition 1):
        # Executes when condition 1 is true
        if (condition 2):
                # Execute when condition 2 is true
        # if Block is end here
# if block is end here
```

Example-

```
# python program to illustrate nested if statement
#!/usr/bin/python
i = 10
```


Test expression
False
Nested test expression
False
True
True
Body of if
Body of nested if
Body of nested else
Statement just below if

```
if (i==10)
        # First if statement
        if (i<15):
                print ("i is smaller than 15")
        # Nested - if statement
        # Will only be executed if statement above
        # it is true
        if (i<12):
                print ("i is smaller than 12 too")
        else:
                print ("i is greater than 15")
```

Output:

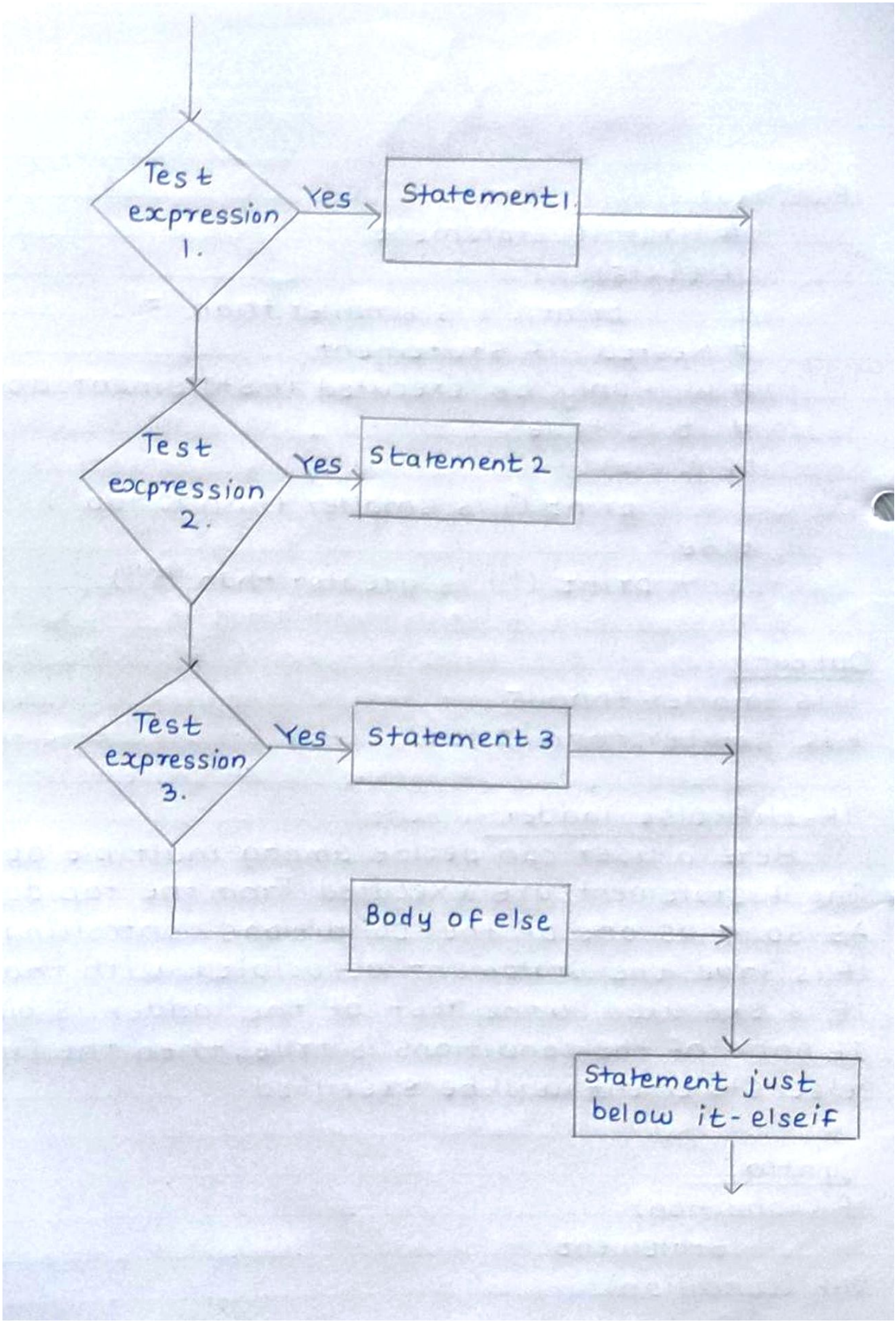
```
i is smaller than 15
i is smaller than 12 too
```

4] if-elif-else ladder

Here a user can decide among multiple options The if statement are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, & the rest of the ladder is bypass. If none of the conditions is true, then the final else statement will be executed.

Syntax:

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


Test expression 1.
Yes
Statement1.
Test expression 2.
Yes
Statement 2
Test expression 3.
Yes
Statement 3
Body of else
Statement just below it- elseif

:

```
else
    statement
```

Example:

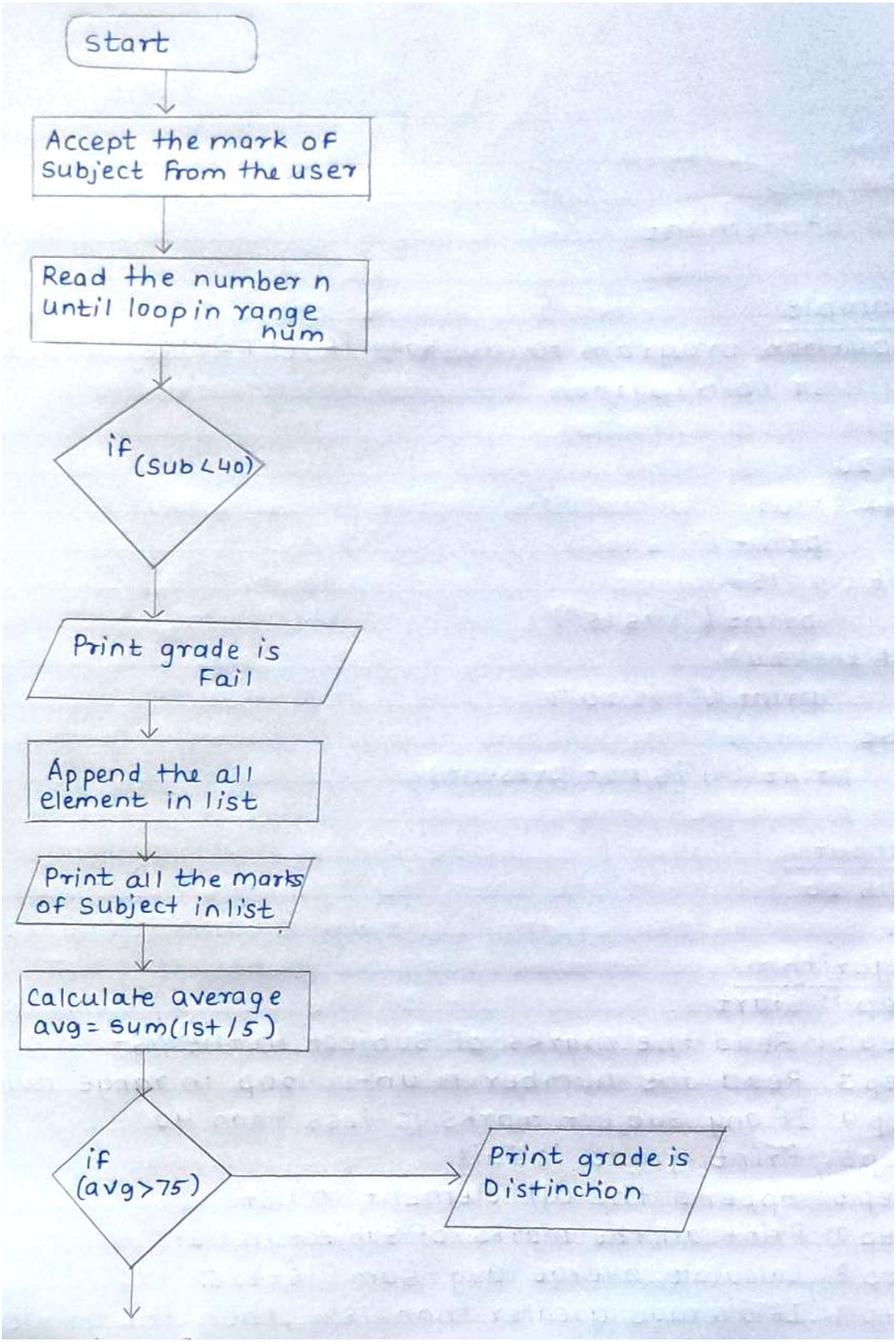
```
# python program to illustrate if-elif-else ladder
#!/usr/bin/python

i=20
if (i==10):
    print ("i is 10")
elif (i==15):
    print ("i is 15")
elif (i==20):
    print ("i is 20")
else
    print ("i is not present)
```

Output:

```
i is 20
```

Algorithm:

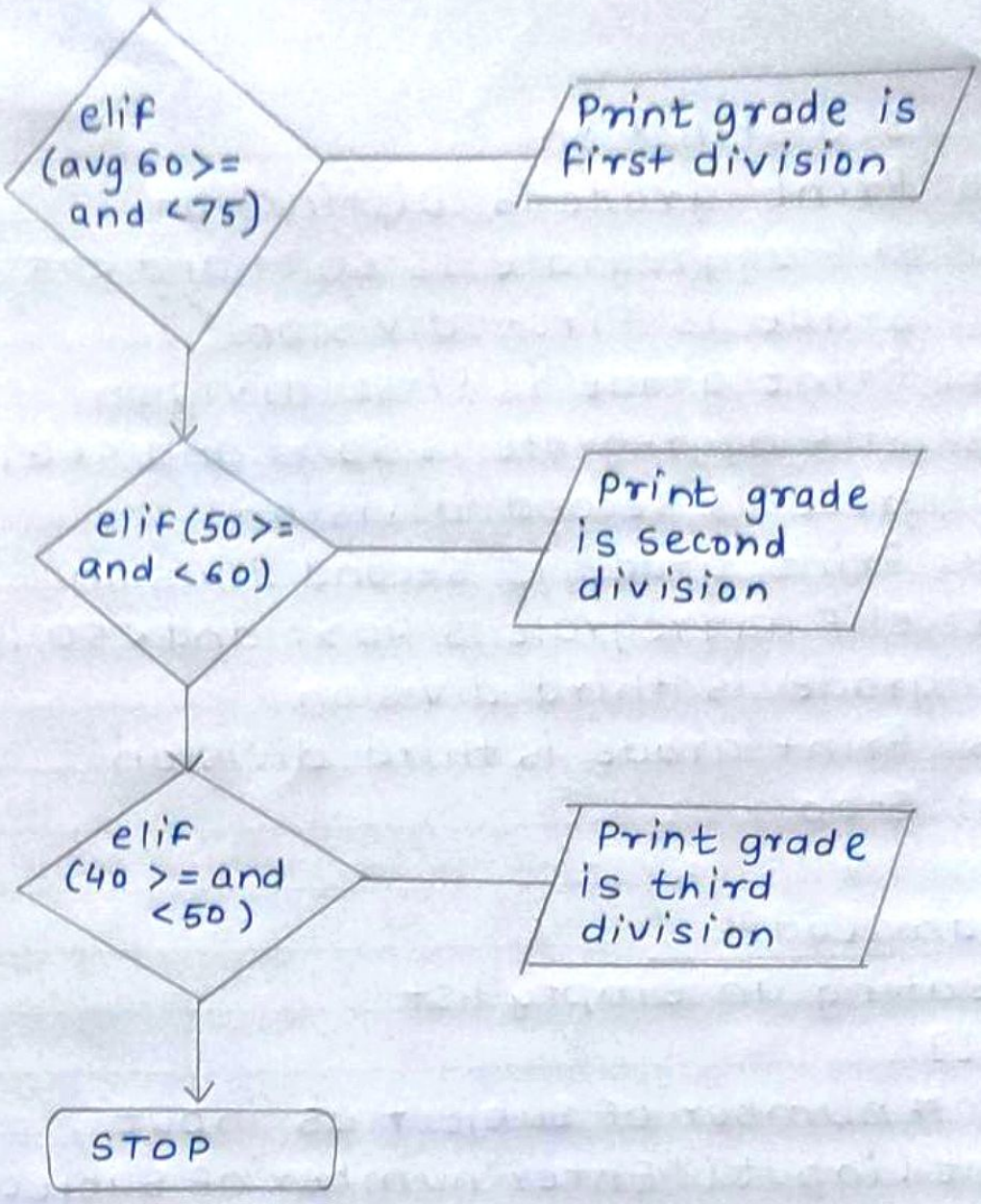
Step 1: Start.
Step 2: Read the marks of subject in the list.
Step 3: Read the number n until loop in range num
Step 4: If any subject marks is less than 40
Step 5: Print grade is Fail
Step 6: Append the all element in list.
Step 7: Print all the marks of subject in list.
Step 8: Calculate averge avg = sum (lst)/5
Step 9: If average greater than 75%, then the grade


Start
Accept the mark of Subject from the user
Read the number n until loop in range num
if (sub<40)
Print grade is Fail
Append the all element in list
Print all the marks of subject in list
Calculate average avg = sum(lst/5)
if (avg>75)
Print grade is Distinction

is distinction.
Step 10: Print grade is Distinction
Step 11: elif aggregrate is 60>= and <75 then the grade is first division.
Step 12: Print grade is first division.
Step 13: elif aggregrate is 50>= and <60, then the grade is second division.
Step 14: Print grade is second division.
Step 15: elif aggregrate is 40>= and <50, then the grade is third division.
Step 16: Print grade is third division.
Step 17: Stop

Program code:

```
# creating an empty list
lst = []
    # number of subject as input
n = int(input("Enter number of subjects: "))
    # iterating till the range
for i in range(0,n):
    sub = int(input("Enter mark of subject: "))
    if(sub<40):
        print("Grade: Fail")
        exit();
    lst.append(sub) # adding the element
    print(lst)
avg = sum(lst)/5
if(avg>=75):
    print("Grade: Distinction")
elif(avg>=60 & avg<75):
    print("Grade: First division")
```


elif (avg 60>= and <75)
Print grade is first division
elif (50>= and <60)
Print grade is second division
elif (40 >= and <50)
Print grade is third division
STOP

```
elif (avg > 40 and avg < 50):
    print ("Grade : Third division")
else:
    print ("Grade : Fail")
```

Output:

```
Enter number of subjects : 5
Enter mark of subject : 67
 [67]
Enter mark of subject : 89
 [67,89]
Enter mark of subject : 90
 [67,89,90]
Enter mark of subject : 45
 [67,89,90,45]
Enter mark of subject : 78
 [67,89,90,45,78]
Grade : First division
 >>>
```

```
Enter number of subject 5
enter marks of maths 67
enter marks of physics 89
enter marks of electrical 90
enter marks of mechanical 45
enter marks of progamming 78
total marks: 369
average marks: 73.8
grade:first:division

[Program finished]
```

Sinhgad Institutes | Date

Assignment No 4.

Title- To accept two numbers from user & compute Smallest divisior & greatest common divisior of these two numbers.

Date of completion	
Remark	
Sign. of staff	

Problem Description:

The program takes the first two numbers of the series along with the number of terms needed and prints the fibonacci series.

Theory-

GCD - The greatest number which divides each of the two or more numbers is called the highest common Factor (HCF). It is also called the Greatest common Measure (GCM) & Greatest common divisior (GCD). HCM & LCM are different.

Example - The highest common factor of 60 & 75 is 15 because 15 is the largest number which can divided both 60 & 75 exactly.

There's another way to find out HCF

- by using prime factorization method or
- by dividging the numbers or division method.

GCD Example

Find the GDC of 45 & 40

Sinhgad Institutes | Date

Step 1: Find the divisors of given numbers:

The divisors of 45 are : 1, 3, 5, 9 & 45
The divisors of 40 are : 1, 2, 4, 5, 7, 8, 10, 20 & 40

Step 2: Find the greatest number that these two lists share in common. In this example the GCD is 5

LCM : Least common multiple which is also mentioned as LCM is the basics of mathematics. This topic, we have learned in our primary classes. Basically, the common multiple is a no. which is a multiple of two or more numbers. L.C.M is used to determine the least common factor or multiple of any two or more given integers. For example, L.C.M of 16 & 20 will be $2 \times 2 \times 2 \times 2 \times 5 = 80$, where 80 is the smallest common multiple for numbers 16 & 20

Along with the least common multiple, you must have heard about the highest common factor, H.C.F., which is used to derive the highest common multiple factors of any two or more given integers. It is also called as Greatest Common Divisior. For example, the H.C.F of 2, 6, 8 is 2, because all three numbers can be divided with the highest number 2 commonly. H.C.F & L.C.M & both have equal importance in maths.

LCM of two integers a & b, usually denoted by LCM (a, b), is the smallest positive integer

that is divisible by both a & b. For two integer a & b to know if there are any smallest numbers d so that d/a and d/b doesn't have a remainder. Such a number is called a Least common multiplier.

LCM Example-

Find the LCM of 6 & 8

The multiples of 6 are : 6, 12, 18, 24, 30, ...
The multiples of 8 are : 8, 16, 24, 32, 40, ...

So, the lowest common Multiple is 24.
Also, we can calculate the LCM using the following formula :

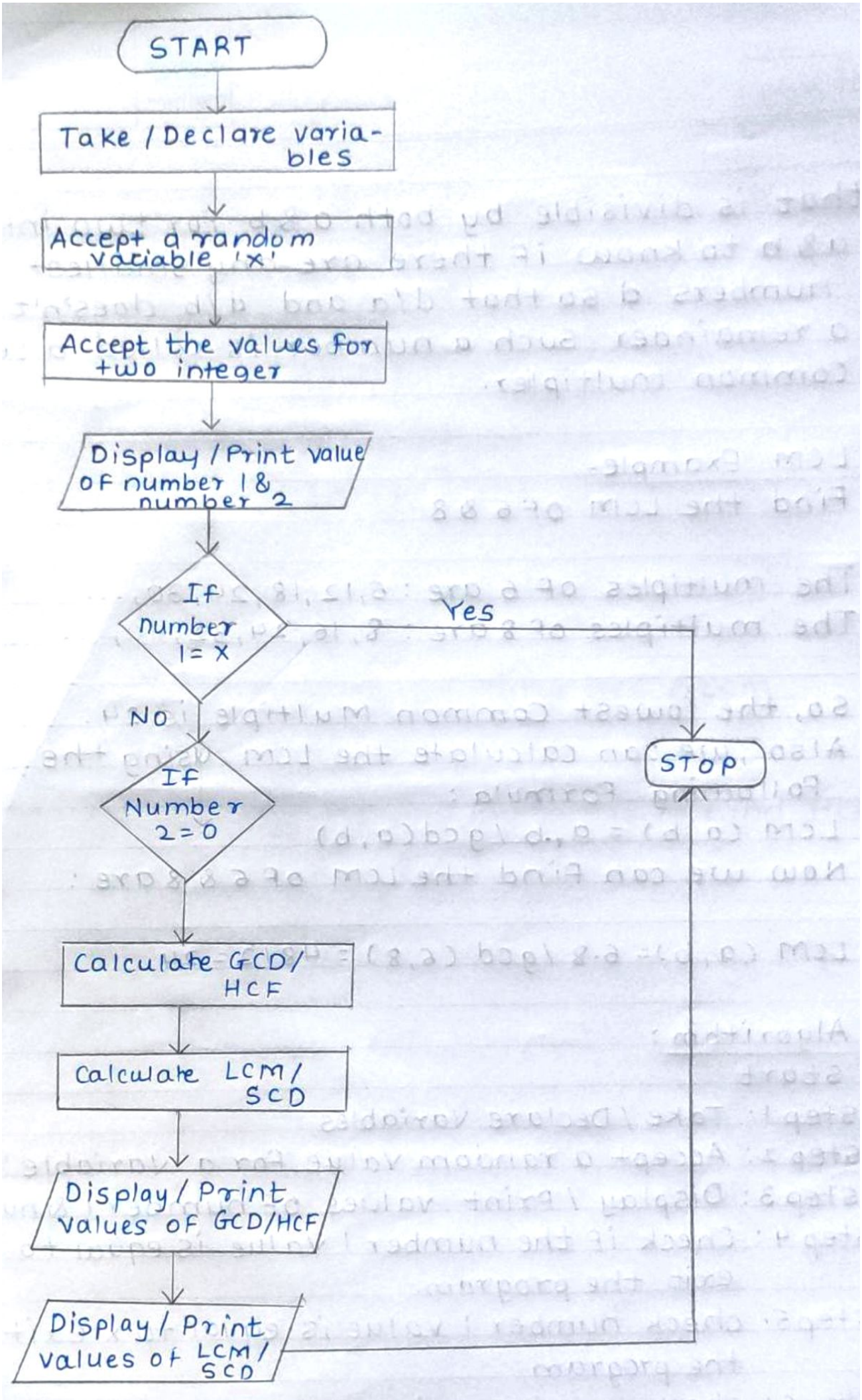
LCM (a, b) = a, b / gcd (a, b)

Now we can find the LCM of 6 & 8 are :

LCM (a, b) = 6.8 / gcd (6, 8) = 48 / 2 = 24

Algorithm :

start
step 1 : Take / Declare Variables
step 2 : Accept a random value for a variable 'x'
step 3 : Display / Print values of number 1 & number 2
step 4 : Check if the number 1 value is equal to x exit the program.
step 5 : check number 1 value is equal to x exit the program.
step 6 : check number 2 is not equal to 0 while


START
Take / Declare variables
Accept a random vaciable 'x'
Accept the values for two integer
Display / Print value of number 1 & number 2
If number 1 = X
Yes
No
If Number 2 = 0
STOP
Calculate GCD / HCF
Calculate LCM / SCD
Display / Print values of GCD/HCF
Display / Print values of LCM / SCD

number 2 != 0
Step 7: Calculate GCD / HCF
Step 7: Calculate LCM / SCD with (number 1 * number 2) / hcf;
Step 8: Display / Print the value of GCD / HCF
Step 8: Display / Print the value of LCM / SCD
STOP

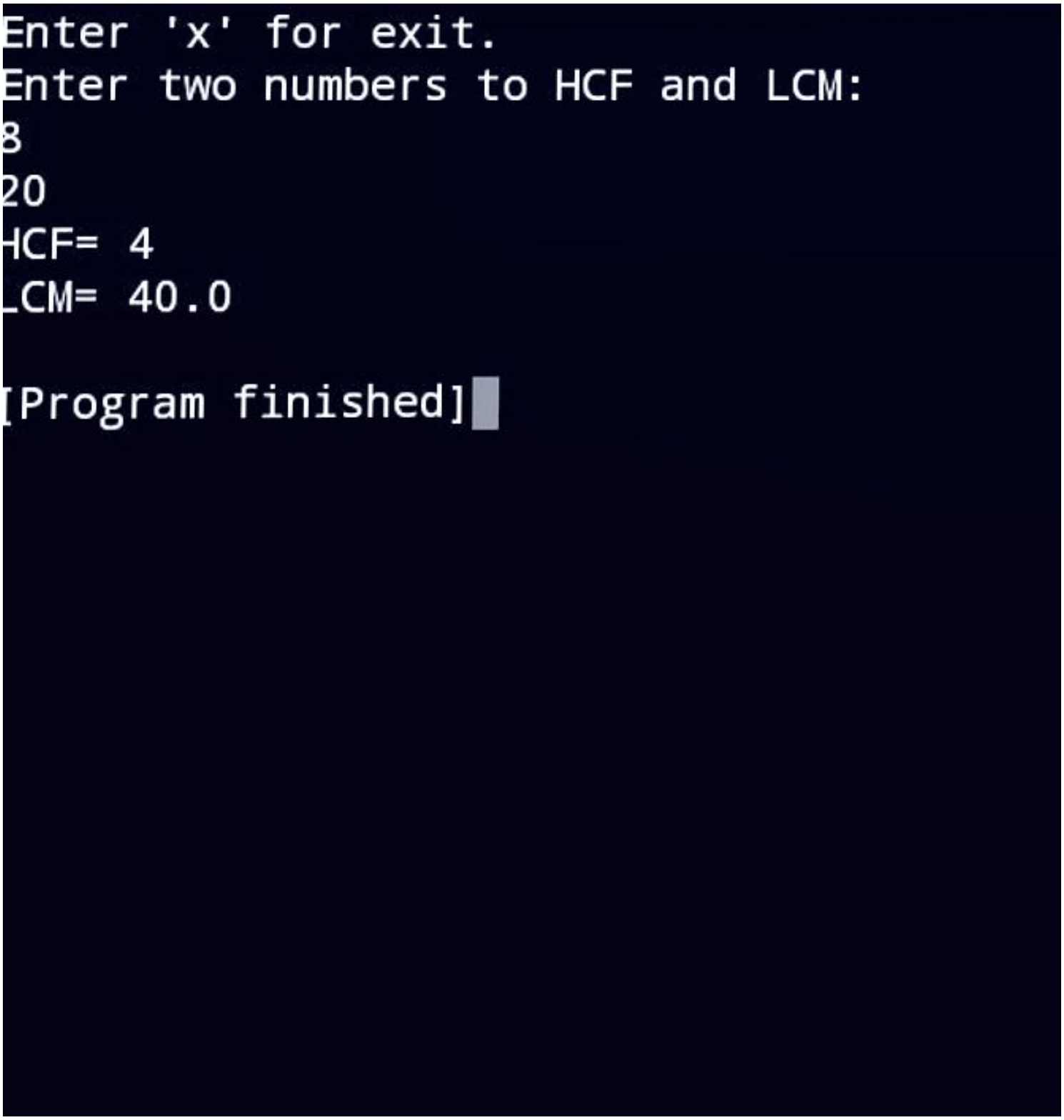
Program:
Code:

```
# Python Program - Find HCF & LCM

print ("Enter 'x' for exit. ");
print (" Enter two numbers to find HCF & LCM: ");
num 1 = input ();
if num 1 == 'x';
        exit();

else :
    num 2 = input ();
    number 1 = int (num 1);
    number 2 = int (num 2);
    temp 1 = number 1;
    temp 2 = number 2;
while temp 2 != 0;
     t = temp 2;
     temp 2 = temp 1 %. tem 2;
     temp 1 = t;
     hcf = temp 1;
     lcm = (number 1 * number 2) / hcf;
```

```
print ("HCF =", hcF);
print ("LCM = ", lcm)
```

Output:

```
Enter 'x' For exit.
Enter two number to Find HCF & LCM:
8
20
HCF = 4
LCM = 40.0
```


Enter 'x' for exit.
Enter two numbers to HCF and LCM:
8
20
HCF= 4
LCM= 40.0
[Program finished]

Date

Assignment No.5

Title: To accept a number from user and print digits of number in a reverse order.

Date of Completion	
Remark	
Sign of staff	

Problem description:

The program is to accept a number (e.g - integer) & with separating the digit of number will reverse the original number & display the reversed number.

Theory:

Here we read a number from user and reverse that number using some arithmetic operation like mood (%) and floor division (//) to Find each digit & build the reverse of it. The following python program reverses a given multi-digit number. In this example, we simply print the digits in reverse order without removing reductant zeros. For example, if the input number is "1000", the reversed number displayed will be "0001" instead of "1!"

Algorithm:

START

1. Read an input number using input() or raw_

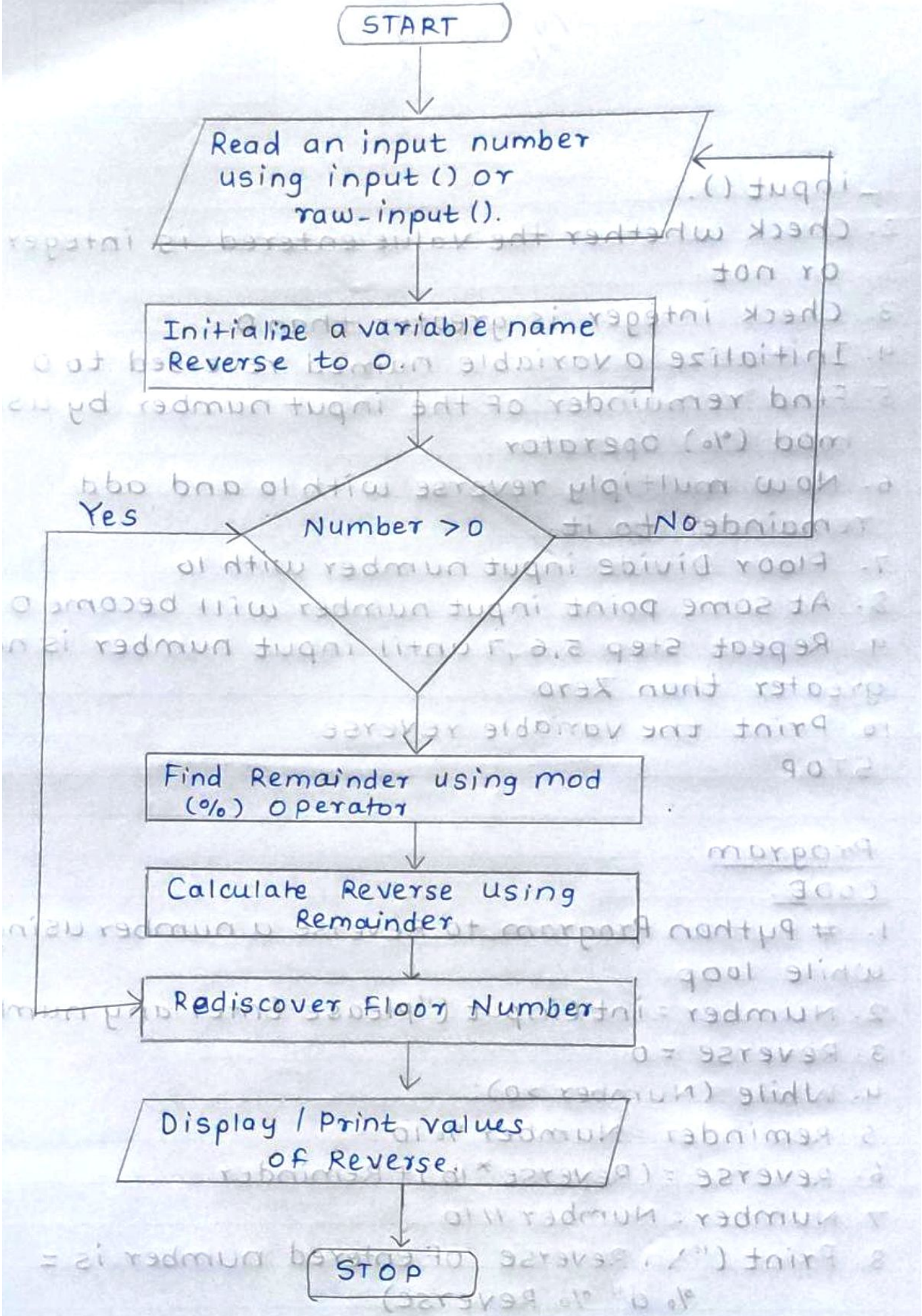
_input().
2. Check whether the value entered is integer or not.
3. Check integer is greater than 0
4. Initialize a variable named reversed to 0
5. Find remainder of the input number by using mod (%) operator
6. Now multiply reverse with 10 and add remainder to it.
7. Floor Divide input number with 10
8. At some point input number will become 0
9. Repeat step 5,6,7 until input number is not greater than Zero.
10. Print the variable reverse.
STOP

Program:

CODE:

```
1. # Python Program to Reverse a number using while loop
2. Number = int(input("please Enter any number."))
3. Reverse = 0
4. While (Number >0):
5. Reminder = Number % 10
6. Reverse = (Reverse * 10) + Reminder
7. Number = Number // 10
8. Print ("\n Reverse of entered number is =
          % d" % Reverse)
```

Output: Enter a number : 9823
Reverse number is: 3289


START
Read an input number using input() or raw_input().
Initialize a variable name Reverse to 0.
Number > 0
Yes
No
Find Remainder using mod (%) Operator
Calculate Reverse using Remainder
Rediscover Floor Number
Display / Print values of Reverse
STOP

```
Please Enter any Number.9823
\Reverse of entered number is =3
\Reverse of entered number is =32
\Reverse of entered number is =328
\Reverse of entered number is =3289

[Program finished]
```

Date :

Assignment. No. 6

Title : To accept from user the number of fibonacci numbers to be generated & print the fibonacci series.

Date of completion	
Remark	
Sign. of staff	

Problem Description:

The program takes the first two number of the series along with the number of terms needed & prints the fibonacci series.

Theory:

Fibonacci Series generates subsquent number by adding two previous numbers. Fibonacci series start from two numbers - F_0 & F_1. The initial value of F_0 & F_1 can be taken 0, 1 or 1, 1 respectively (generally starting from 0 & 1). That is

$F_0 = 0$ & $F_1 = 1$

Fibonacci series satisfies the following mathematical conditions -

$$F_n = F_{n-1} + F_{n-2}$$

So a Fibonacci numbers in the following integer sequence, called the Fibonacci sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144...

Following numbers can be calculated as

$F_2 = F_0 + F_1 = 0+1 = 1$

$F_3 = F_1 + F_2 = 1+1 = 2$

$F_4 = F_2 + F_3 = 2+1 = 3$

$F_5 = F_3 + F_4 = 3+4 = 5$ like wise

Algorithm-

START

Step 1: Take / Declare Variables F_1, F_2, F_3, i, n

step 2: Set / Initialize $F_1 = 0, F_2 = 1$

Step 3: Display / Print values of F_1 & F_2

Step 4: Accept the value of n from user

Step 5: for i = 2 to n

Step 6 : $F_3 = F_1 + F_2$

Step 7: Display / Print the value of F_3

Step 8: Set / Assign F_2 to F_1 & F_3 to F_2

(ie- $F_1 = f_2$ & $F_2 = F_3$)

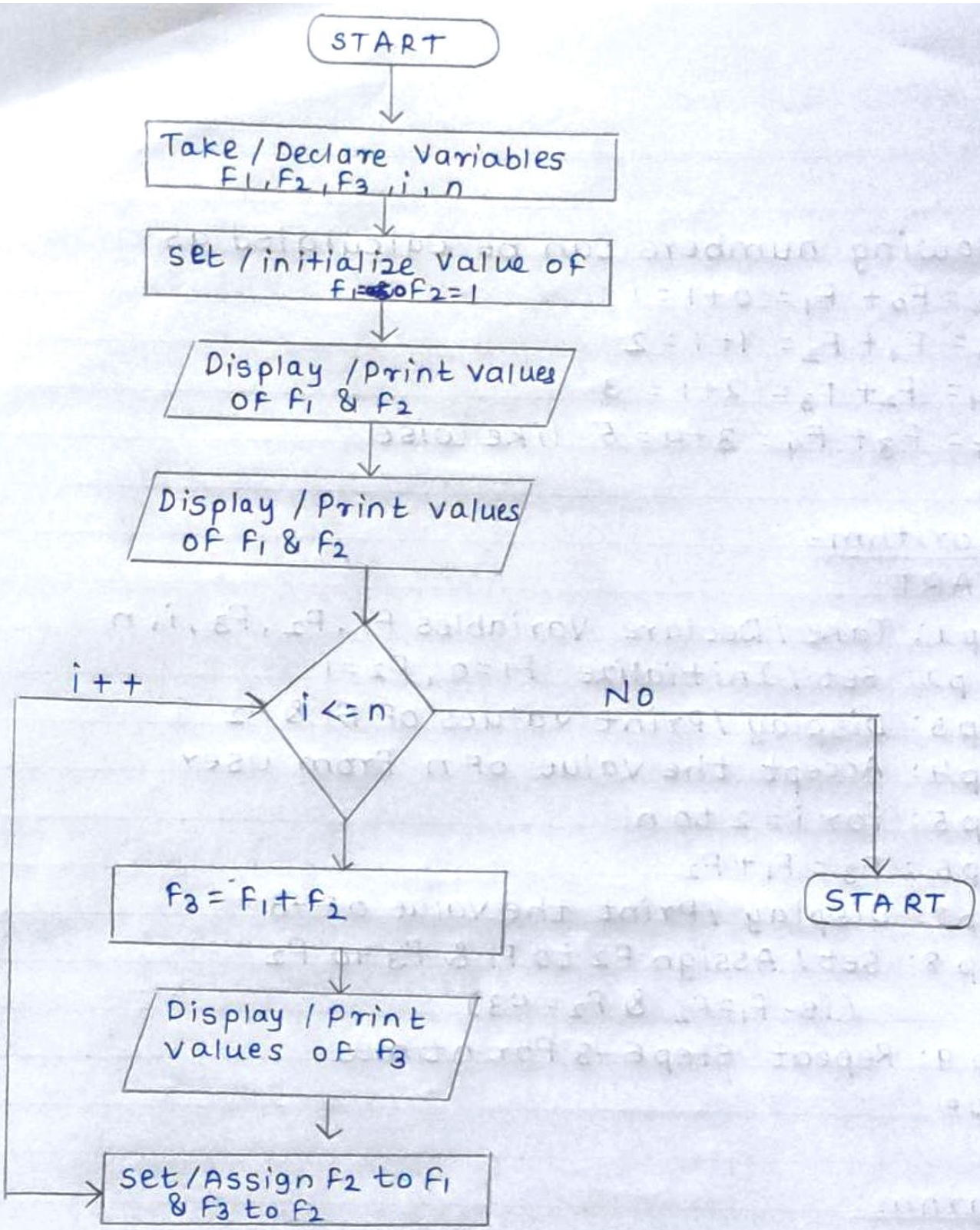
Step 9: Repeat Step 6-8 For n times

STOP

Program:

Code:

```
1. def recur_fibo(n):
2.     if n<=1:
3.         return n
4.     else:
5.         return (recur_fibo(n-1) + recur_
                   fibo(n-2))
6. # take input from the user
7. nterms = in(input("How many terms?"))
```


START
Take / Declare Variables
F1, F2, F3, i, n
set / initialize value of
f1=0 f2=1
Display / Print values
of f1 & f2
Display / Print values
of f1 & f2
i++
i <= n
No
f3 = f1 + f2
START
Display / Print
values of f3
set/Assign f2 to f1
& f3 to f2

```
8. check if the number of terms is valid
9. if nterms <= 0:
10.         print ("please enter a positive integer")
11. else:
12.     print ("Fibonacci sequence:")
13.     for i in range (n terms):
14.                 print (recur_fibo3 (i))
```

Output:

Fibonacci sequence:

0
1
1
2
3
5
8
13
21
34
55
89

Here, we store the number of terms in n terms. We initialize the first term to 0 & the second term to 1.

If the number of terms is more than 2, we use a while loop to find the next term in the sequence by adding the preceding two terms. We then interchange the variable (update it) and continue on with the process.

```
How many terms?12
Fibonacci sequence:
0
1
1
2
3
5
8
13
21
34
55
89

[Program finished]
```

Assignment No.7

Title: Write a python program that accepts a string from user & perform following string Operations - i. Calculate length of string ii. string reversal iii. Equality check of two strings iv. Check palindrome v. Check substring.

Date of completion	
Remark	
Sign. of staff	

Problem Description:

The program accepts a string from user & finds length of string, reversal of string. checks if is string is palindrome. Again it accepts substring & check if the substring is found or not & it also accepts second string & checks the equality of two strings.

Theory:

Strings in python can be created using single quotes or double quotes or even triple quotes. Python treats single quotes the same as double quotes. In python a string is displayed with the print () function.

String len () method: The method len() return the length of the string.

Syntax : len (str)

Sinhgad Institutes | Date :

Eg. Print ("length of the input string is :", len(str))

Reverse string using slicing : logic : str[::-1]
Eg. txt = str[::-1]
print ("\n Reversed string is :", (txt))

Palindrome string : A string is said to be palindrome if reverse of the string is same as string. For example, "radar" is a palindrome, but "radix" is not palindrome.

Substring : To check if a python string contains a substring is to use the in operator. It returns a boolean (either True or False)
Eg. str 1 = hello python
str 2 = python
if str 2 in str 1 :
    print ("Substring found !")
else :
    print ("Substring Not found!")

Equality of two strings : In python, the compraison operator (==) is used to check if the strings are identical.
Eg- str = hello python
str 2 = python
if list(str) == list(str2):
    print ("\n Both strings are equal to each other")
else:
    print ("\n Both strings are not equal.")

Page-(2)

Date :

Sinhgad Institutes

Algorithm:

START

Step 1: User inputs the string & it gets stored in variable str

Step 2: Using len() Function length of string is calculated

Step 3: Display / Print length of string

Step 4: Using Reversal Function string is reversed

Step 5: Display / Print Reversed string

Step 6: string is compared with reversed string to check palindrome

Step 7: Display / Print whether string is palindrome or not

Step 8: User inputs the substring is & it gets stored in variable substring

Step 9: Using in Function substring is checked

Step 10: Display / Print whether substring is Found or not

step 11: User inputs another string & it get stored in variable str 2

Step 12: Both strings are connected to check Equality of strings

step 13: Display / Print whether both strings are equal or not

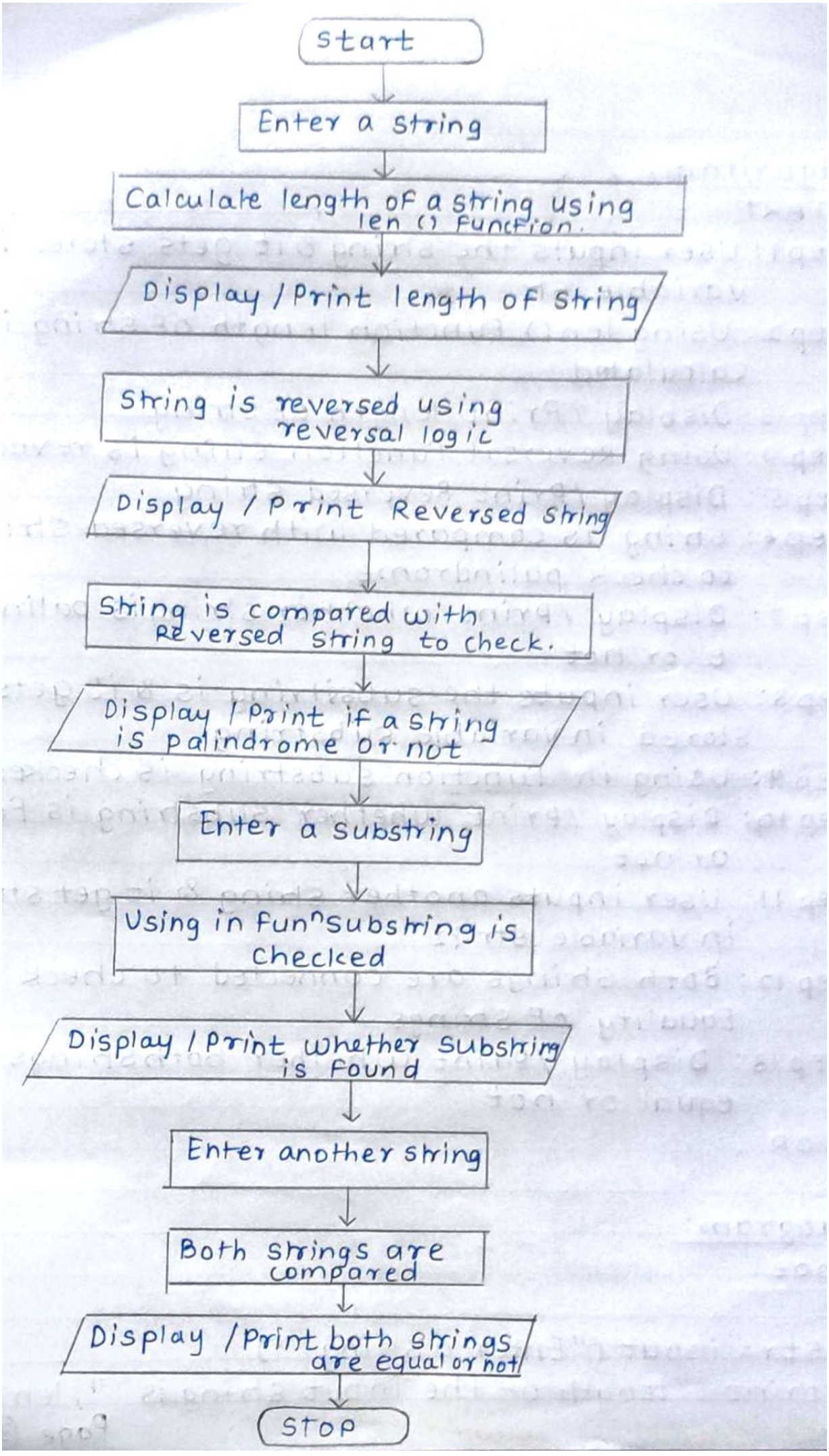
STOP

Program:

Code:

```
1. str= input ("Enter a string:")
2. print ("length of the input string is :", len (str))
```

Page-③


Start
Enter a string
Calculate length of a string using len () function.
Display / Print length of string
String is reversed using reversal logic
Display / Print Reversed string
String is compared with Reversed string to check.
Display / Print if a string is palindrome or not
Enter a substring
Using in fun^n substring is checked
Display / Print whether substring is found
Enter another string
Both strings are compared
Display / Print both strings are equal or not
STOP

```
3. txt = str[::-1]
4. print ("\n Reversed string is:", (txt))
5. if list (str) == list (txt):
6.         print ("it is palindrome")
7. else:
8.         print ("It is not palindrome")
9. substr = input ("Enter a substring:")
10.        if substr in str:
11.                print ("substring Found!")
12. else:
13.        print ("substring Not Found!")
14. str2 = input ("Enter second string:")
15. if list (str) == list (str2):
16.        print ("\n both strings are equal to
                   each other:")
17. else:
18.        print ("B\n both strings are not equal:")
```

Output:

```
Enter a string : welcome
Length of the input string is:7

Reverserd string is: emoclew
 It is not palindrome
Enter a substring : come
substring Found!
Enter second string : python

Both strings are not equal
```

Here, user inputs 1 string & find length of string Then string is reversed & displayed. Next the string is palindrome or not is checked & displays message Next user inputs substring to check substring is present or not & displays message. User inputs second string to check equality of two strings and displays message.

```
Enter a string:Welcome
Length of the input string is: 7

Reversed string is: Welcome
It is palindrome
Enter a substring:come
substring Found!
Enter second string:python

Both string are not equal.

[Program finished]
```