AISSMS
COLLEGE OF ENGINEERING

ज्ञानम् सकलजनहिताय

Accredited by NAAC with "A+" Grade

Department Of First Year Engineering

Academic Year

2022-23

LAB MANUAL

PROGRAMMING AND PROBLEM SOLVING

Subject Code 110005

Faculty Name - Prof. Bhakti Patil

AISSMS
COLLEGE OF ENGINEERING
ज्ञानम् सकलजनहिताय
Accredited by NAAC with "A+" Grade

DEPARTMENT OF FIRST YEAR ENGINEERING

ACADEMIC YEAR: 2022-2023
LIST OF LAB EXPERIMENTS

CLASS: F.E. **SEMESTER: I**

SUBJECT: PROGRAMMING AND PROBLEM SOLVING

Experiment No.	Problem statement
1	To calculate salary of an employee given his basic pay (take as input from user).Calculate salary of employee. Let HRA be 10 % of basic pay and TA be 5% of basic pay. Let employee pay professional tax as 2% of total salary. Calculate salary payable after deductions.
2	a) To accept an object mass in kilograms and velocity in meters per second and display its momentum. Momentum is calculated as e=mc2 where m is the mass of the object and c is its velocity. b) To check whether input number is Armstrong number or not. An Armstrong number is an integer with three digits such that the sum of the cubes of its digits is equal to the number itself. Ex. 371.
3	a)To accept some N numbers from user and compute and display maximum in list, minimum in list, the count, sum and average of numbers. b) To accept list of N integers and partition list into two sub lists even and odd numbers.
4	To accept student's five courses marks and 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.
5	To simulate simple calculator that performs basic tasks such as addition, subtraction, multiplication and division with special operations like computing x^y

6	Write a python program that accepts a string from user and perform following string operations- i. Calculate length of string ii. String reversal iii. Equality check of two strings iv. Check substring

Experiment No. 1

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

Theory:

FEATURES OF PYTHON :

Simple: Python is a simple and small language Reading a program written in Python feels almost like reading English. This is in fact the greatest strength of Python which allows programmers to concentrate on the solution to the problem rather than the language itself.

Easy to Learn: A Python program is clearly defined and easily readable. The structure of the program is very simple. It uses few keywords and a clearly defined syntax. This makes it easy for just anyone to pick up the language quickly

Versatile: Python supports development of a wide range of applications ranging from simple text processing to WWW browsers to games

Free and Open Source: Python is an example of an open source software. Therefore, anyone can freely distribute it, read the source code, edit it, and even use the code to write new (free) programs

High-level Language: When writing programs in Python, the programmers don't have to worry about the low-level details like managing memory used by the program, etc. They just need to concentrate on writing solutions of the current problem at hand.

Interactive: Programs in Python work in interactive mode which allows interactive testing and debugging of pieces of code. Programmers can easily interact with the interpreter directly at the Python prompt to write their programs.

Dynamic: Python execute dynamically Programs written in Python can be copied and used for development of application If There is any error, it is reported at run-time to allow interactive program development

Extensible: Since Python is an open source software, anyone can add low-level modules to the interpreter. These modules enable programmers to add to or customize their tools to work more efficiently. Moreover, if you want a piece of code not to be accessible for everyone, then you can even code that your program in C or C++ and then use them from your Python program.

Extensive Libraries Python has a huge library that is easily portable across different platforms. Library functions allows programmers to perform Wide range of applications

Easy Maintenance: Code written in Python is easy to maintain

Secure: The Python language environment is secure from altering data. Apart from this, additional security checks can be easily added to implement additional security features

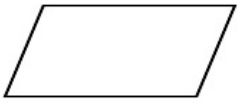
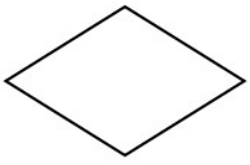
Robust: Python programmers cannot manipulate memory directly. More over errors are raised as an exception. that can be caught and handled by the program code. For every syntactical mistake, a simple interpret message is displayed. All these things makes the language robust.

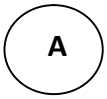
Multi-threaded: Python supports multi-threading, that is executing more than one process of a program simultaneously. It also allows programmers to perform process management tasks

Garbage Collection: The Python runtime environment handles garbage collection of all Python objects. Objects which are currently not in use are deleted.

FLOWCHART :
Flowcharts are the graphical representation of the algorithms.
Using the flowcharts the programmers can find out the bugs in the programming logic and then can go for coding Flowcharts can show errors in the logic and set of data can be easily tested using flowcharts.

Symbols Used In Flowchart:

Flow lines are used to indicate the flow of data. The arrow heads are important for flowlines The flowlines are also used to connect the different blocks in the flowchart.	
These are termination symbols. The start and end of flowchart is represented by keywords Start and Stop/End in the ellipse	
The rectangle indicates the processing. It includes calculations, opening/ closing file.	
Parallelogram indicates input and output	
The diamond indicates the decision. It has one entrance and two exits. One exit indicates the true and other indicates the false.	

The on-page connector connects the two different sections on the same page. A letter is written inside the circle. The off-page connector connects the two different sections on the different pages.	

OPERATORS:

1.Arithmetic Operators:

OPERATOR	DESCRIPTION	SYNTAX
+	Addition: adds two operands	x + y
-	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when first operand is divided by the second	x % y

2.Relational Operators: Relational operators compares the values. It either returns True or False according to the condition.

OPERATOR	DESCRIPTION	SYNTAX
>	Greater than: True if left operand is greater than the right	x > y
<	Less than: True if left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to: True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to: True if left operand is less than or equal to the right	x <= y

3.Logical operators: Logical operators perform Logical AND, Logical OR and Logical NOT operations.

OPERATOR	DESCRIPTION	SYNTAX
And	Logical AND: True if both the operands are true	x and y
Or	Logical OR: True if either of the operands is true	x or y
Not	Logical NOT: True if operand is false	not x

4. Bitwise operators: Bitwise operators acts on bits and performs bit by bit operation.

OPERATOR	DESCRIPTION	SYNTAX
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

5.Assignment operators: Assignment operators are used to assign values to the variables.

OPERATOR	DESCRIPTION	SYNTAX
=	Assign value of right side of expression to left side operand	x = y + z
+=	Add AND: Add right side operand with left side operand and then assign to left operand	a+=b a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b a=a/b
%=	Modulus AND: Takes modulus using left and right operands and assign result to left operand	a%=b a=a%b
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a//=b a=a//b

**=	Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand	a**=b a=a**b

6.Special operators: There are some special type of operators like-

- **Identity operators-**
  ' is' and 'is not' are the identity operators. Both are used to check if two values are located on the same part of the memory. Two variables that are equal does not imply that they are identical.
  - is True if the operands are identical
  - is not True if the operands are not identical

ALGORITHM AND FLOWCHART :

Step 1: Start

Step 2: Input Basic, HRA, TA, DA

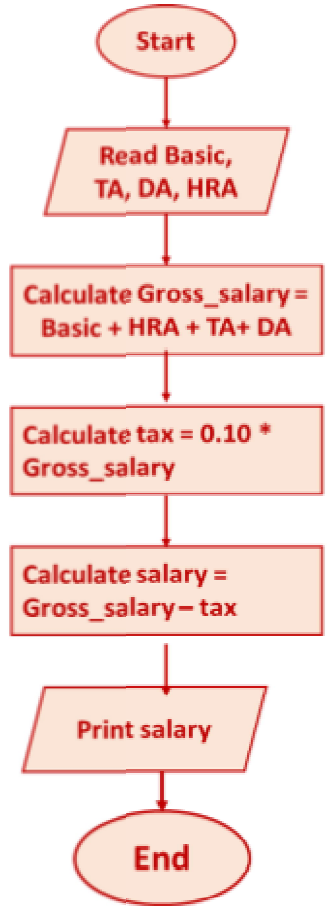
Step 3: Set Gross_salary = Basic + HRA + TA+ DA

Step 4: Set tax = 0.10 * Gross_salary

Step 5: Set salary = Gross_salary – tax

Step 6: Print salary

Step 7: End



PROGRAM :

```python
#Experiment 1
#To calculate salary of an employee given his basic pay (take as input from user).
#Calculate salary of employee. Let HRA be 10 % of basic pay and TA be 5% of basic pay.
#Let employee pay professional tax as 2% of gross salary. Calculate salary payable after deductions.


basic_pay=float(input("Enter basic pay:"))
hra=0.1*basic_pay
ta=0.05*basic_pay
gross_sal=basic_pay+hra+ta          #Gross salary of employee
pt=0.02*gross_sal                   #Professional Tax
total_sal=gross_sal-pt              #Total salary
print("House Rent Allowance: ",hra)
print("Travelling Allowance: ",ta)
print("Gross Salary: ",gross_sal)
print("Professional Tax: ",pt)
print("Total Salary Payable: ",total_sal)
```

```
Enter basic pay:30000
House Rent Allowance:  3000.0
Travelling Allowance:  1500.0
Gross Salary:  34500.0
Professional Tax:  690.0
Total Salary Payable:  33810.0
```

Conclusion :

Hence, we studied basics of Python Programming and implemented given problem statement in Python.

Experiment No. 2

Aim : a) To accept an object mass in kilograms and velocity in meters per second and display its momentum. Momentum is calculated as e=mc2 where m is the mass of the object and c is its velocity.

b) To check whether input number is Armstrong number or not. An Armstrong number is an integer with three digits such that the sum of the cubes of its digits is equal to the number itself. Ex. 371.

Theory:

Data types are used to define type of variable.

1. **Numeric:**
   Integer: It holds signed integers Ex. 222
   Float: It Holds floating point precision number. Ex. 23.21
   Complex: Holds complex number. Ex: 5+6j
2. **String:** String is a collection of characters. In python, we can use single quote, double quote or triple quote to define a string.
   + operator is used for string concatenation and * operator is used for repetition
   Eg: str1= "AISSMS".
3. **List:** It holds different types of data in list, enclosed in square brackets [ ]. Items separated with commas. List is mutable, i.e. we can add, delete or updates list elements.
   Eg: list1=[10, 20, 'aaa', 'bbb', 33.43, 22.32]
4. **Tuple:** It holds different types of data in list, enclosed in parenthesis ( ). Items separated with commas. Tuple is immutable. i.e. we cannot add, delete or updates tuple elements. Tuples are Read-only.
   Eg: tup1= (10, 20, 'aaa', 'bbb', 33.43, 22.32)
5. **Dictionary:** It is a collection of elements in the form of key-value pair. Elements are enclosed in curly brackets.
   Eg: dict1={ 1: "Red", 2: "Blue", 3: "Green", 4: "White", 5: "Black" }

Identifier is a collection of alphanumeric characters. Identifiers are used to give name to the programming elements like variable, tuples, lists, functions, classes etc.

Rules:

1. Each Identifier starts with an alphabet or underscore and then followed by any number of alphabets, digits or underscore.
2. Other than underscore no special symbol is allowed to form the Identifier name.
3. Identifier shouldn't start with digit.
4. In a program variable name must be unique.
5. Identifiers are case sensitive so the identifiers area, AREA and Area considered as different Identifier.
6. A Identifier should not contain any comma or blank space.
7. Identifier name should not same as the keywords otherwise compiler will generate an error
8. The length of an identifier varies from language to language. But it is a good practice to keep the identifier name short and specific.

Eg. Correct identifier: sum, area_of_circle, my_list1

Algorithm & Flowchart:

To be drawn by students

Program:

```
#Experiment 2
#To accept an object mass in kilograms and velocity in meters per second and display its momentum.
#Momentum is calculated as e=mc2 where m is the mass of the object and c is its velocity.

m=float(input("enter the mass of the object"))
c=float(input("enter the speed of the object"))
e=m*c*c
print("the Energy of the object is: ",e)
```

```
enter the mass of the object20
enter the speed of the object10
the Energy of the object is:  2000.0
```

Conclusion :

Hence, we studied data types in python and implemented given problem statement in Python.

Experiment No. 3

Aim :

a)To accept some N numbers from user and compute and display maximum in list, minimum in list, the count, sum and average of numbers

b) To accept list of N integers and partition list into two sub lists even and odd numbers.

Theory:

For Loop:

It is used to repeat a task until a particular condition is True. The for loop usually known as a determine or definite loop because the programmer knows exactly how many times the loop will repeat.

When for loop is used a range sequence is specified. The items of sequence are assigned to the loop control variable one after the other. The for loop is executed for each item in the sequence. With every iteration of the loop, a check is made to identify whether the loop control variable has been assigned all the values in the range. If all the values have been assigned, the statement block of the loop is executed else, the statement comprising the statement the statement block of the for loop are skipped and control jumps to immediate statement following for loop body.

Syntax: for var in sequence:

Statement(s)

While Loop:

While LOOP provide a mechanism to repeat one or more statements while particular condition is true. In while loop, the condition is tested before any of the statement block is executed..If the condition is true, only then the statements will be executed otherwise if the condition is false, the control will jump to statement y, that is immediate statement outside the while loop block.

Syntax: statement x

while(condition):

Statement(s) block

statement y

Program:

```python
#Experiment 3
#To accept some N numbers from user.
#compute and display maximum in list, minimum in list, the count, sum and average of numbers.
lst=[]                          #create empty list
num=int(input("enter the number of elements in the list"))  #Number of elements in list
for n in range(num):
    a=int(input("enter the element in the list"))
    lst.append(a)
print("contents of the list are: ",lst)
summation=0
for i in range(0,num):
    summation+=lst[i]
print("Sum of elements is: ",summation)
average=summation/num
print("The average is: ",average)
print("Minimum element from the list is: ",min(lst))
print("Maximum element from the list is: ",max(lst))
```

```
enter the number of elements in the list5
enter the element in the list10
enter the element in the list15
enter the element in the list2
enter the element in the list4
enter the element in the list5
contents of the list are:  [10, 15, 2, 4, 5]
Sum of elements is:  36
The average is:  7.2
Minimum element from the list is:  2
Maximum element from the list is:  15
```

Conclusion : Hence we studied loop statements in python and implemented given problem statement.

Experiment No. 4

Aim : To accept student's five courses marks and 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.

Theory:

Decision making is required when we want to execute a code only if a certain condition is satisfied.

The if…elif…else statement is used in Python for decision making.

Python if Statement Syntax
if test expression:
    statement(s)

Here, the program evaluates the test expression and will execute statement(s) only if the text expression is True.

If the text expression is False, the statement(s) is not executed.

In Python, the body of the if statement is indicated by the indentation. Body starts with an indentation and the first unindented line marks the end.

Python interprets non-zero values as True. None and 0 are interpreted as False.

Python if Statement Flowchart

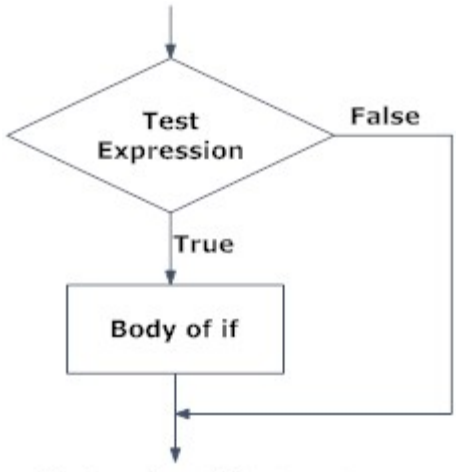


Fig: Operation of if statement

Example: Python if Statement

Eg: If the number is positive, we print an appropriate message

```
num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")
num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

The body of if is executed only if this evaluates to True.

When variable num is equal to 3, test expression is true and body inside body of if is executed.

If variable num is equal to -1, test expression is false and body inside body of if is skipped.

The print() statement falls outside of the if block (unindented). Hence, it is executed regardless of the test expression.

Python if...else Statement

Syntax of if...else

```
if test expression:
    Body of if
else:
    Body of else
```

The if..else statement evaluates test expression and will execute body of if only when test condition is True.

If the condition is False, body of else is executed. Indentation is used to separate the blocks.

Python if..else Flowchart

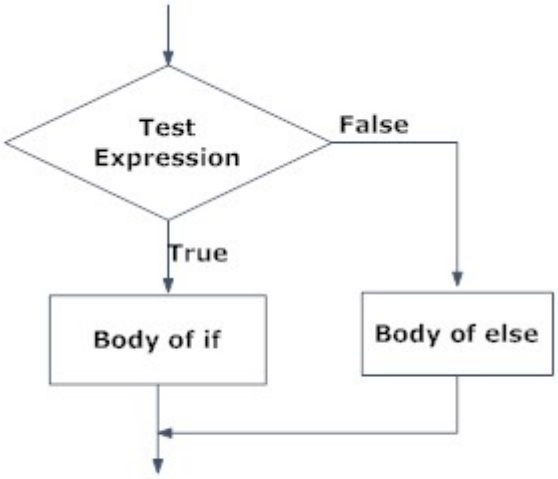


Fig: Operation of if...else statement

Example of if...else

Program checks if the number is positive or negative and displays an appropriate message

```
num = 3

# Try these two variations as well.

# num = -5

# num = 0

if num >= 0:

    print("Positive or Zero")

else:

    print("Negative number")
```

In the above example, when num is equal to 3, the test expression is true and body of if is executed and body of else is skipped.

If num is equal to -5, the test expression is false and body of else is executed and body of if is skipped.

If num is equal to 0, the test expression is true and body of if is executed and body of else is skipped.

Python if...elif...else Statement

Syntax of if...elif...else

```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```

The elif is short for else if. It allows us to check for multiple expressions.

If the condition for if is False, it checks the condition of the next elif block and so on.

If all the conditions are False, body of else is executed.

Only one block among the several if...elif...else blocks is executed according to the condition.

The if block can have only one else block. But it can have multiple elif blocks.

Flowchart of if...elif...else

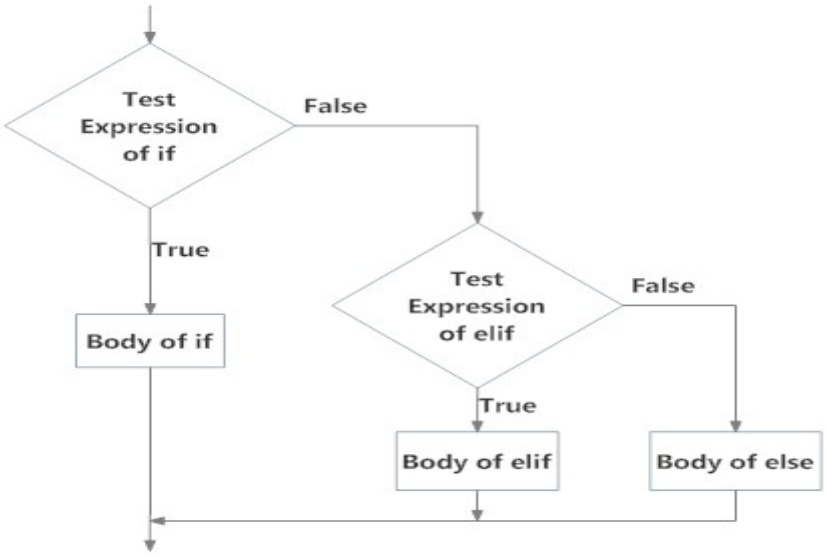


Fig: Operation of if...elif...else statement

Program:

```
#Experiment 4
#To accept student's five courses marks and 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.

sub1=int(input("Enter the marks for first subject"))
sub2=int(input("Enter the marks for second subject"))
sub3=int(input("Enter the marks for third subject"))
sub4=int(input("Enter the marks for fourth subject"))
sub5=int(input("Enter the marks for fifth subject"))
total=sub1+sub2+sub3+sub4+sub5
avg=total/5

print("Percentage :",avg," % ")

if(avg>75):
    print("You have passed with distinction")
elif(avg>60 and avg<75):
    print("You have passed with first grade")
elif(avg>50 and avg<60):
    print("You have passed with second grade")
elif(avg>40 and avg<50):
    print("You have passed")
else:
    print("you have failed")
```

```
Enter the marks for first subject75
Enter the marks for second subject80
Enter the marks for third subject89
Enter the marks for fourth subject83
Enter the marks for fifth subject82
Percentage : 81.8  %
You have passed with distinction
```

Conclusion : Hence, we studied Conditional statements in python.

Experiment No. 5

Aim : To simulate simple calculator that performs basic tasks such as addition, subtraction, multiplication and division with special operations like computing x^y and x!

Theory :

Function is a block of instructions/code that perform specific task. Every function is specified by a name. Python enables its programmers to break up a program into segments commonly known as functions, each of which can be written more or less independently of the others. Every function in the program is supposed to perform a well-defined task.

Need of Function

- **Reuse of code**

Code reuse is one of the most prominent reasons to use functions. Large programs usually follow the **DRY** principle, that is, Don't Repeat Yourself principle. Once a function is written, it can be called multiple times within the same or by a different program wherever its functionality is required.

Correspondingly, a bad repetitive code abides by the WET principle, i.e., Write Everything Twice, Consider a program that executes a set of instructions repeatedly 'n' times, though not continuously. In such case, the instructions had to be repeated continuously for n times they can better be placed within a loop.

But if these instructions have to be executed abruptly from anywhere within the program code, then instead of writing these instructions everywhere they are required, a better way is to place these instructions in a function and call that function wherever required.

Function definition and call in Python:

- Function can be defined using keyword **def**
- Function call invokes a function

Example:

function definition

```
def myfunction():
    print("test function is defined here")

myfunction()
```

function call

Program:

```
#Experiment 5
#To simulate simple calculator that performs basic tasks such as addition, subtraction, multiplication and division
#with special operations like computing x^y and x!

def addition(a,b):
    print("Addition is: ",a+b)
def sub(a,b):
    print("Subtraction is: ",a-b)
def mul(a,b):
    print("Multiplication is: ",a*b)
def div(a,b):
    print("Division is: ",a/b)
def expo(a,b):
    print("a to the power b is: ",a**b)
def factorial(a):
    fac=1
    i=a
    while(i>0):
        fac=fac*i
        i=i-1
    print("Facorial of",a,"is: ",fac)
```

```
print("CALCULATOR".center(65))
while True:
    ch=int(input(" 1.Addition\n 2.Subtraction\n 3.Multiplication\n 4.Division\n 5.Exponent\n 6.Factorial\n Enter your choice:  "
    if(ch==1):
        a=int(input("Enter First number: "))
        b=int(input("Enter Second number:  "))
        addition(a,b)
    elif(ch==2):
        a=int(input("Enter First number: "))
        b=int(input("Enter Second number:  "))
        sub(a,b)
    elif(ch==3):
        a=int(input("Enter First number: "))
        b=int(input("Enter Second number:  "))
        mul(a,b)
    elif(ch==4):
        a=int(input("Enter First number: "))
        b=int(input("Enter Second number:  "))
        div(a,b)
    elif(ch==5):
        a=int(input("Enter First number: "))
        b=int(input("Enter Second number:  "))
        expo(a,b)
    elif(ch==6):
        a=int(input("Enter a number: "))
        factorial(a)
    else:
        print("Sorry!!!Incorrect choice!!!!")
        break
```

```
                    CALCULATOR
 1.Addition
 2.Subtraction
 3.Multiplication
 4.Division
 5.Exponent
 6.Factorial
 Enter your choice:  3
Enter First number: 12
Enter Second number:  4
Multiplication is:  48
 1.Addition
 2.Subtraction
 3.Multiplication
 4.Division
 5.Exponent
 6.Factorial
 Enter your choice:  6
Enter a number: 7
Facorial of 7 is:  5040
 1.Addition
 2.Subtraction
 3.Multiplication
 4.Division
 5.Exponent
 6.Factorial
 Enter your choice:  7
Sorry!!!Incorrect choice!!!!
```

Conclusion : Hence, we studied and implemented functions in python.

Experiment No. 6

Aim : Write a python program that accepts a string from user and perform following string operations- i. Calculate length of string ii. String reversal iii. Equality check of two strings iv. Check substring

Theory:

Python does not support a character type; these are treated as strings of length one, thus also considered a substring. To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring.

There are many operations that can be performed with **string** which makes it one of the most used datatypes **in Python**. Concatenation of Two or More **Strings**. Joining of two or more **strings** into a single one is called concatenation. The + operator does this **in Python**.

	Description	**syntax**	**example**	**Output**
Concatenation	You can add two strings i.e. joining of 2 strings called concatenation. '+' operator is used. Two strings whether created using single or double quotes are concatenated in the same way.	String1 + string 2	str1="Ro" str2="shan" print(str1+str2)	Roshan
Repetition	When a string is multiplied with an integer n, the string is repeated n times. '*' operator is used.	string1*integer	str1="Hi" print(str1*3)	HiHiHi
Slice	To extract subset of string, slice operator[:] is used. You need to specify index or range of index.	var[start : end]	Str="Python is Easy!!!" Print(str[0]) Print(str[-1]) Print(str[2: 9])	 P ! thon is
Range	It is built-in function.. Range function produces a sequence of numbers starting with **beg** and ending with one less than the number **end.** The step argument is optional. By default, beg is 0 and step is 1.	range(beg, end, step)	for x in range(1,10, 2) Print (x)	1 3 5 7 9

Program :

```
#Experiment 6
#Write a python program that accepts a string from user and perform following string operations:
#i. Calculate length of string ii. String reversal iii. Equality check of two strings iv. Check substring

def strlen(s):
    l=len(s)
    print("Length of the string is:",l)

def strrev(s):
    rev=s[::-1]
    print("Reverse string is: ",rev)

def strcheck(s1,s2):
    if s1==s2:
        print("Both the strings are equal.")
    else:
        print("Strings are not equal")

def substr_check(s1,s2):
    if s1 in s2:
        print("substring",s1,"is present in the string",s2)
    else:
        print("No substring found!!")

def find_substr(s1,s2):
    index=s2.find(s1)
    print("String",s1, "found at index",index)

print("STRING OPERATIONS".center(65))
```

```
while True:
    ch=int(input(" 1.Length of string\n 2.Reverse string\n 3.Equality check\n 4.Check Substring\n 5.Find substring\n Enter your
    if(ch==1):
        a=str(input("Enter the string: "))
        strlen(a)
    elif(ch==2):
        a=str(input("Enter the string: "))
        strrev(a)
    elif(ch==3):
        a=str(input("Enter First string: "))
        b=str(input("Enter Second string:  "))
        strcheck(a,b)
    elif(ch==4):
        a=str(input("Enter substring to be checked: "))
        b=str(input("Enter main string:  "))
        substr_check(a,b)
    elif(ch==5):
        a=str(input("Enter substring to be found: "))
        b=str(input("Enter main string:  "))
        find_substr(a,b)
    else:
        print("Sorry!!!Incorrect choice!!!!")
        break
```

```
                    STRING OPERATIONS
  1.Length of string
  2.Reverse string
  3.Equality check
  4.Check Substring
  5.Find substring
  Enter your choice:  1
Enter the string: programming
Length of the string is: 11
  1.Length of string
  2.Reverse string
  3.Equality check
  4.Check Substring
  5.Find substring
  Enter your choice:  2
Enter the string: programming
Reverse string is:  gnimmargorp
  1.Length of string
  2.Reverse string
  3.Equality check
  4.Check Substring
  5.Find substring
  Enter your choice:  3
Enter First string: green
Enter Second string:  rgeen
Strings are not equal
  1.Length of string
  2.Reverse string
  3.Equality check
  4.Check Substring
  5.Find substring
  Enter your choice:  4
Enter substring to be checked: on
Enter main string:  python
substring on is present in the string python
  1.Length of string
  2.Reverse string
  3.Equality check
  4.Check Substring
  5.Find substring
  Enter your choice:  5
Enter substring to be found: tho
Enter main string:  python
String tho found at index 2
  1.Length of string
  2.Reverse string
  3.Equality check
  4.Check Substring
```

Conclusion : Hence, we studied string and string operations in python