## Objective:

**Practical No. 2a User Defined Functions**

Write a program to demonstrate working with user defined functions in python.

## Practical Significance

All object-oriented programming languages supports reusability. One way to achieve this is to create a function. Like any other programming languages Python supports creation of functions and which can be called within program or outside program. Reusability and Modularity to the Python program can be provided by a function by calling it multiple times. This practical will make learner use of modularized programming using functions.

## Minimum Theoretical Background

Functions are the most important aspect of an application. A function can be defined as the organized block of reusable code which can be called whenever required.

* 1. **Creating a function:** In Python, we can use **def** keyword to define the function. Syntax:

def my\_function(): function-suite return <expression>

* 1. **Calling a function:** To call the function, use the function name followed by the parentheses.

def hello\_world(): print("hello world")

hello\_world()

## Output:

hello world

* 1. **Arguments in function:** The information into the functions can be passed as the argumenta. The arguments are specified in the parentheses. We can give any number of arguments, but we have to separate them with a comma.

## Example

#defining the function def func (name):

print("Hi ",name);

#calling the function func("ABC")

## Output:

hi ABC

* 1. **return Statement:** The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

## Example

# Function definition is here def sum( arg1, arg2 ):

# Add both the parameters and return them." total = arg1 + arg2

print "Inside the function: ", total return total;

# Now you can call sum function total = sum(10, 20 );

print "Outside the function: ", total

## Output:

Outside the function: 30

## Exercise

1. Write a Python function that takes a number as a parameter and check the number is prime or not.
2. Write a Python function to calculate the factorial of a number (a non- negative integer). The function accepts the number as an argument.
3. Write a Python function that accepts a string and calculate the number of upper case letters and lower case letters.

## Objective:

**Practical No. 2b Modules**

Write Python program to demonstrate use of:

* 1. Built-in module
  2. User defined module.

## Practical Significance

While developing a computer application, more often than not the line of code (LOC) can go beyond developer’s control. Such codes are tightly coupled and hence are difficult to manage. Also one cannot reuse such code in other similar application. Python supports modularized coding which allow developers to write a code in smaller blocks. A module can define functions, classes and variables. A module can also include runnable code. By using module students will be able to group related code that will make the code easier for them to understand and use.

## Minimum Theoretical Background

* 1. **Built-in Modules**

Built-in modules are written in C and integrated with the Python interpreter. Each built-in module contains resources for certain system-specific functionalities such as OS management, disk IO, keyword, math, number, operator etc. The standard library also contains many Python scripts (with the .py extension) containing useful utilities.

To display a list of all available modules, use the following command in the Python console:

>>> help('modules')

## Python - Math Module

Some of the most popular mathematical functions are defined in the math module. These include trigonometric functions, representation functions, logarithmic functions, angle conversion functions, etc. In addition, two mathematical pie and Euler's number constants are also defined in this module.

## Example

>>> import math

>>>math.pi 3.141592653589793

## Python - OS Module

It is possible to automatically perform many operating system tasks. The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

## Example

We can create a new directory using the **mkdir()** function from the OS module.

>>> import os

>>>os.mkdir("d:\\tempdir")

## Python - Random Module

Functions in the **random** module depend on a pseudo-random number generator function random(), which generates a random float number between 0.0 and 1.0.

## Example

>>>import random

>>>random.random() 0.645173684807533

## User-defined Module

The user-defined module is written by the user at the time of program writing.

## Creation a User-defined Module

To create a module just write a Python code in a file with file extension as.py:

## Example

A module in a file with extension as module\_name.py in Python is created. def accept\_int():

val = int(input("Please enter any integer integer: ")) print('The integer value is', val)

## Accessing a User-defined Module

Now we will access the module that we created earlier, by using the import statement.

## Example

import the module in Python. import module\_name

## Output

Please enter any integer integer: 22 The integer value is 22

## Exercise

1. Write a Python program to create a user defined module that will ask your college name and will display the name of the college.
2. Write a Python program that will calculate area and circumference of circle using inbuilt Math Module
3. Write a Python program that will display Calendar of given month using Calendar Module
4. Write a Python program that will calculate square root of number using Math Module