Chapter-4: Functions and Modules

4.1 Introduction to Python User defined Function

- **What is a function in Python?**
  - ✓ A function is a block of code which only runs when it is called.
  - ✓ In Python, **a function is a group of related statements that performs a specific task**.
  - ✓ You can pass data, known as parameters; into a function.
  - ✓ A function can return data as a result.
  - ✓ Functions help break our program into smaller and modular chunks.
  - ✓ As our program grows larger and larger, functions make it more organized and manageable.
  - ✓ Furthermore, it avoids repetition and makes the code reusable.

Syntax of Function

```
def function_name (parameters):

        statement(s)
```

Above shown is a function definition that consists of the following components.

- ✓ **Keyword def** that marks the start of the function header.
- ✓ **A function name** to uniquely identify the function. Function naming follows the same rules of writing identifiers in Python.
- ✓ **Parameters** (arguments) through which we pass values to a function. They are optional.
- ✓ A colon (:) to mark the end of the function header.
- ✓ One or more valid python statements that make up the function body. **Statements must have the same indentation level.**
- ✓ return statement is used to return a value from the function.

- **Example of creating a function:**

```
def my_function():
    print("Hello from a 21CE!!")
```

- **How to call a function in python?**

  Once we have defined a function, we can call it from another function, program, or even the Python prompt. To call a function we simply type the function name with appropriate parameters.

- **Example of call a function:**

```
my_function()
```

Output:

SLTIET

```
Hello from a 21CE!!
```

4.2 Passing parameters to a function and returning values from a function

- ✓ Information can be passed into functions as arguments.
- ✓ Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
- ✓ The following example has a function with one argument (name). When the function is called, we pass along a name, which is used inside the function to print the name:

Example of a function with parameter:

```
def my_function (name):

        print ("Hello,” + name + ". Good morning!")
```

calling a function:

```
my_function ('21CE')
```

Output:

```
Hello, 21CE. Good morning!
```

Returning a Values: To let a function return a value, use the return statement:

- **Example of a function with parameter and return value:**

```
def my_function(x):
        return 5 * x
```

calling a function:

```
print (my_function(3))
print (my_function(5))
print (my_function(9))
```

Output:

```
15
25
45
```

Python Default Arguments:

- ✓ Function arguments can have default values in Python.
- ✓ We can provide a default value to an argument by using the assignment operator (=).

SLTIET

Example:

def sum(a,b=10):
    return a+b
print(sum(5))
print(sum(5,5))
 	
Output:
15
10
	def my_function (name, msg="Good morning!"):
	    print("Hello ", name, msg)
	
	my_function ("21CE")
	my_function ("ENGINEERS", "How do you do?")
	Output:
	Hello 21CE Good morning!
	Hello ENGINEERS, How do you do?

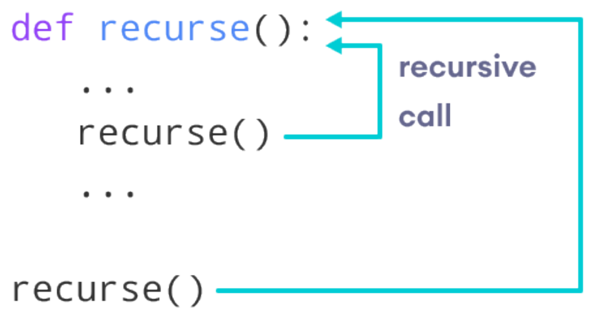
4.3 Recursion

What is recursion?

- ✓ **A Function calls itself is said to be a recursion.**
- ✓ Recursion is the process of defining something in terms of itself.
- ✓ A physical world example would be to place two parallel mirrors facing each other. Any object in between them would be reflected recursively.

Python Recursive Function:

- ✓ In Python, we know that a function can call other functions. It is even possible for the function to call itself. These types of construct are termed as recursive functions.
- ✓ The following image shows the workings of a recursive function called recurse.



```
def recurse():
    ...
    recurse()
    ...

recurse()
```

Example of a recursive function to find the factorial of an integer.

- ✓ Factorial of a number is the product of all the integers from 1 to that number. For example, the

SLTIET

factorial of 6 (denoted as 6!) is 1*2*3*4*5*6 = 720.

Example :

```
def factorial(x):
    if x == 1:

        return 1
    else:
        return (x * factorial(x-1))
```

calling a function:

```
num = 3
```

SLTIET

```
        print("The factorial of", num, "is", factorial(num))
```

Output:

```
        The factorial of 3 is 6
```

- ✓ Our recursion ends when the number reduces to 1. This is called the **base condition**.
- ✓ Every recursive function must have a base condition that stops the recursion or else the function calls itself infinitely.
- ✓ The Python interpreter limits the depths of recursion to help avoid infinite recursions, resulting in stack overflows.
- ✓ By default, the maximum depth of recursion is 1000. If the limit is crossed, it results in RecursionError.

Advantages of Recursion

- ✓ Recursive functions make the code look clean and elegant.
- ✓ A complex task can be broken down into simpler sub-problems using recursion.
- ✓ Sequence generation is easier with recursion than using some nested iteration.

Disadvantages of Recursion

- ✓ Sometimes the logic behind recursion is hard to follow through.
- ✓ Recursive calls are expensive (inefficient) as they take up a lot of memory and time.
- ✓ Recursive functions are hard to debug**.**

4.4 Standard Library: Built-in Functions

- ✓ A Python library is a collection of modules.
- ✓ A Package is a library that can be installed using a Package manager.
- ✓ Python standard libraries make the program simpler and remove the need to rewrite commonly used commands again and again.
- ✓ They can be used by calling/importing at the beginning of python script.

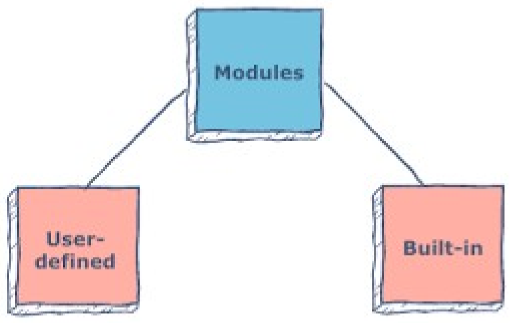
4.5 Modules and Packages

❖ What are modules in Python?

- ✓ Modules refer to a file containing Python statements and definitions.

- ✓ Modules are pre defined files that contain the python codes which include the basic functionality of class, methods ,variables etc.
- ✓ Modules can be divided in to two parts:

SLTIET



- ✓ Python contains built-in modules that are already programmed into the language and user defined modules are created by user.
- ✓ We can define our most used functions in a module and import it, instead of copying their definitions into different programs.
- ✓ A file containing Python code, for example: example.py, is called a module, and its module name would be **example**.

Example of user defined module (module name: example):

```
def add(a, b):

        result = a + b

        print( result)
```

- ✓ Here, we have defined a function add() inside a module named **example**. The function takes in two numbers and returns their sum.

How to import modules in Python?

- ✓ We use the import keyword to do this. To import our previously defined module example, we type the following in the Python prompt.

>>import example

- ✓ This does not import the names of the functions defined in example directly in the current symbol table. It only imports the module name example there.
- ✓ Using the module name we can access the function using the dot (.) operator. For example:

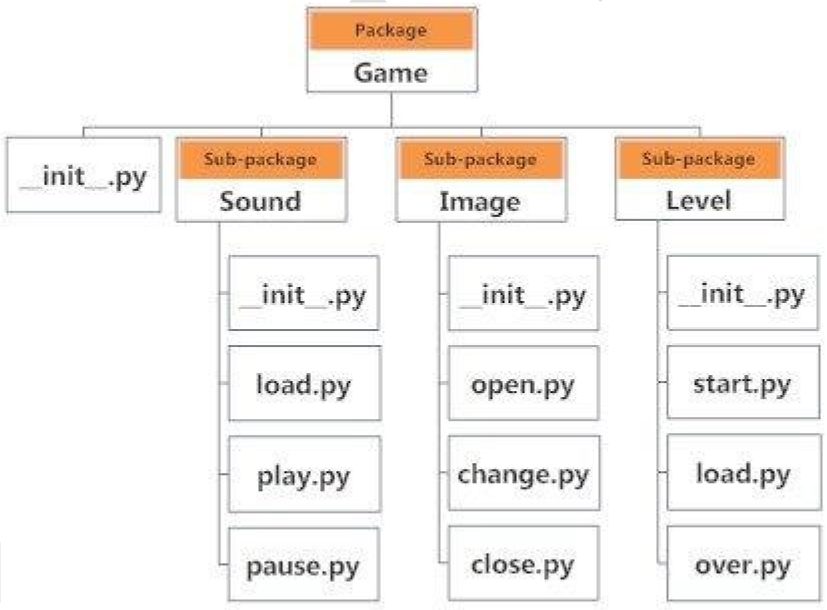
```
>>> example.add(4,5.5)

        9.5
```

SLTIET

- ✓ Python has tons of standard modules. You can check out the full list of Python standard modules and their use cases. These files are in the Lib directory inside the location where you installed Python.

❖ What are packages?

- ✓ We don't usually store all of our files on our computer in the same location.
- ✓ We use a well-organized hierarchy of directories for easier access.
- ✓ **Python has packages for directories and modules for files.**
- ✓ **As a directory can contain subdirectories and files, a Python package can have sub-packages and modules.**
- ✓ A directory must contain a file named __init__.py in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file.
- ✓ Here is an example. Suppose we are developing a **game**. One possible organization of packages and modules could be as shown in the figure below.



Importing module from a package

- ✓ We can import modules from packages using the dot (.) operator.
- ✓ For example, if we want to import the start module in the above example, it can be done as follows:

import Game.Level.start

Now, if this module contains a function named select_difficulty(), we must use the full name to reference it.

Game.Level.start.select_difficulty(2)

SLTIET

❖ rand module - Random numbers generators

- ✓ Python **Random module** is an in-built module of Python which is used to generate random numbers.
- ✓ It can be used to perform some action like to get a random number,select random element from a list, shuffle elements randomly etc.

Example: Printing a random value from a list

import random
 	
list1 = [1, 2, 3, 4, 5, 6]
print(random.choice(list1))
	Output:
	2

The random module has a set of methods, some of methods are listed here:

random( ): It is used to generate random floats between 0 to 1. **Syntax:** random.random() **Example: import random** **print(random.random())** **Output: 0.371793355562307**
uniform( ): It is used to generate random floats between two given parameters. **Syntax:** random.uniform(First number,Second number) **Example: import random** **print(random.uniform(1,10))** **Output: 7.771509161751196**
randint( ): It returns a random integer between the given integers. **Syntax:** random.randint(First number,Second number) **Example: import random** **print(random.randint(1,10))** **Output: 3**
randrange( ): It returns a random integer from the range created by the start,stop and step arguments.The value of start is 0 by default. The value of step is 1 by default. **Syntax:** random. randrange (start,stop,step) **Example:** import random print(random.randrange(1,10,2)) **Output: 5**

SLTIET

choice( ):It returns a randomly selected element from a non empty sequence.

Syntax**:** random.choice(sequence)

Example: Selecting random elements from the list, string, and tuple

```
Example 1: import random
               print(random.choice((1,10,2)))
Output: 2
Example 2:import random
              print(random.choice('computer'))
Output: o
```

choices( ): It returns a list from a non empty sequence.

Syntax**:** random.choices(sequence)

Example: Selecting random elements from the list, string, and tuple

```
Example 1: import random
               print(random.choices((1,10,2)))
Output: [2]
Example 2:import random
              print(random.choices('computer'))
Output: [‘o’]
```

shuffle(): It is used to shuffle a sequence (list). Shuffling means changing the position of the elements of the sequence

Syntax: random.shuffle(sequence)

```
Example: import random
           a=[10,11,12,13,14]
           print('before shuffle',a)
         random.shuffle(a)
         print('after shuffle',a)
Output:
         before shuffle [10, 11, 12, 13, 14]
         after shuffle [10, 12, 14, 11, 13]
```

❖ math module – Mathematical functions

✓ Mathematical functions are defined in the math module.

✓ The math module has set of methods and constants.

✓ This module includes trigonometric functions , logarithm functions ,angle conversion functions etc.

Math constant:

Constant	**Description**	**Example**

math.e	Return Euler’s number (2.718281828459045)	import math
		print(math.pi)
		print(math.e)
		print(math.tau)

math.pi	Return PI (3.141592653589793)	
math.tau	Return tau (6.283185307179586)	

SLTIET

Numeric Functions:

ceil(): Rounds a number up to the nearest integer Example : import math   print(math.ceil(8.23)) Output : 9
floor():Rounds a number down to the nearest integer Example : import math   print(math.floor(8.23)) Output : 8
sqrt(): It returns the square root of the number. Example: import math   print(math.sqrt(25)) Output: 5
pow(): It receives two arguments, raises the first to the second and give result. Example: import math   print(math.pow(2,4)) Output**:** 16.0
factorial(): It returns the factorial of a given number. Example: import math   print(math.factorial(4)) Output : 24
gcd(): It is used to find the greatest common divisor of two numbers passed as the arguments. Example :import math   print(math.gcd(10,35)) Output :5
fabs(): It returns the absolute value of the number. Example :import math   print (math.fabs(-10)) Output : 10.0
fmod(): It returns the reminder of x/y. Example : import math   print (math.fmod(10,3)) Output : 1.0
exp(): It returns a float number after raising e to the power of a given number. (e**x) Example : import math   print (math.exp(2)) Output: 7.38905609893065

❖ Datetime module - Date and time functions

- ✓ Python Datetime module comes built into Python, so there is no need to install it externally.
- ✓ Python Datetime module supplies classes to work with date and time.
- ✓ The date and time objects can be either categorized as **aware or naïve** depending on whether it

SLTIET

include timezone information.

- ✓ **Aware object** contains enough information to identify itself to other date/time objects.
- ✓ **naive object** does not contain enough information to identify itself to other date/time objects.
- ✓ The DateTime module is categorized into 6 main classes –

date – An idealized naive date, assuming the current Gregorian calendar. Its attributes are year, month and day.
time – An idealized time, independent of any particular day, assuming that every day has exactly 24*60*60 seconds. Its attributes are hour, minute, second, microsecond, and tzinfo.
datetime – Its a combination of date and time along with the attributes year, month, day, hour, minute, second, microsecond, and tzinfo.
timedelta – A duration expressing the difference between two date, time, or datetime instances to microsecond resolution. **tzinfo –** It provides time zone information objects.
timezone – A class that implements the tzinfo abstract base class as a fixed offset from the

UTC (New in version 3.2).

✓ We can use dir() function to get a list containing all attributes of a module.

import datetime
print(dir(datetime))
	Output
	['MAXYEAR', 'MINYEAR', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'date', 'datetime', 'datetime_CAPI', 'sys', 'time', 'timedelta', 'timezone', 'tzinfo']

Example:1 Get current date and time(Using datetime class)

import datetime
d=datetime.datetime.now()
print(d)
	Output
	
	2022-11-28 12:40:36.467347

Example:2 Get current date (Using date class)

import datetime
d=datetime.date.today()
print(d)
	Output
	
	2022-11-28

❖ **matplotlib module – Plotting functions**

✓ Matplotlib is a library for creating static, animated and interactive visualization in Python.

Installation steps for matplotlib:

Step 1: Make sure that Python and pip is preinstalled on system.Type following command in command prompt to check python is installed or not.

```
python    -V
```

If python is successfully installed , then version of python will be displayed. To check pip, type following command in command prompt.

```
pip -V
```

If pip is successfully installed, then version of pip will be displayed

Step 2: Install Matplotlib.

Matplotlib can be installed using pip. Write following command to install Matplotlib.

```
pip install matplotlib
```

Step 3: Check if it is installed successfully.

To verify that Matplotlib is installed on your system, write following command in the command prompt. If it is successfully installed than version of Matplotlib will be displayed.

```
C:\Users\vpmp\AppData\Local\Programs\Python\Python310\Scripts > pip show matplotlib
```

For Importing matplotlib :

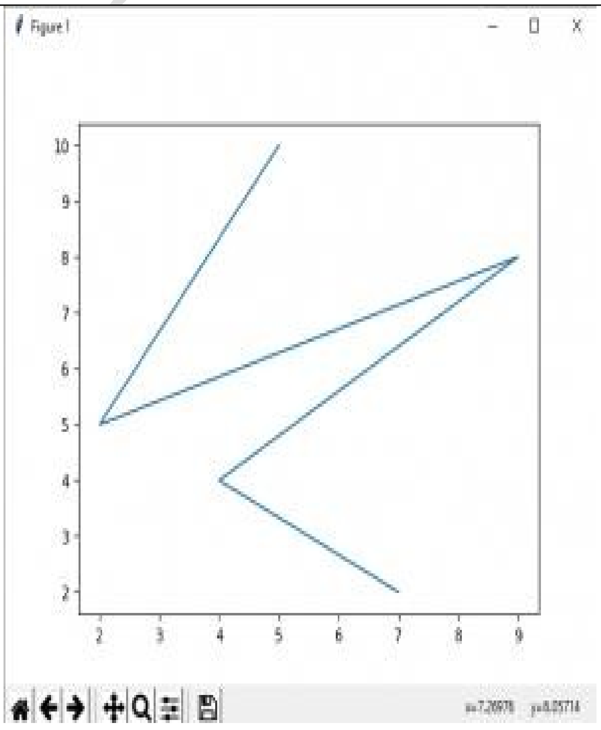
```
from matplotlib import pyplot as plt
or
import matplotlib.pyplot as plt
```

Types of plots in Matplotlib :

✓ Various plots can be created using Matplotlib like Bar Graph, Line Plot, Histogram, Scatter Plot, Area Plot etc.

1. **Line plot :**

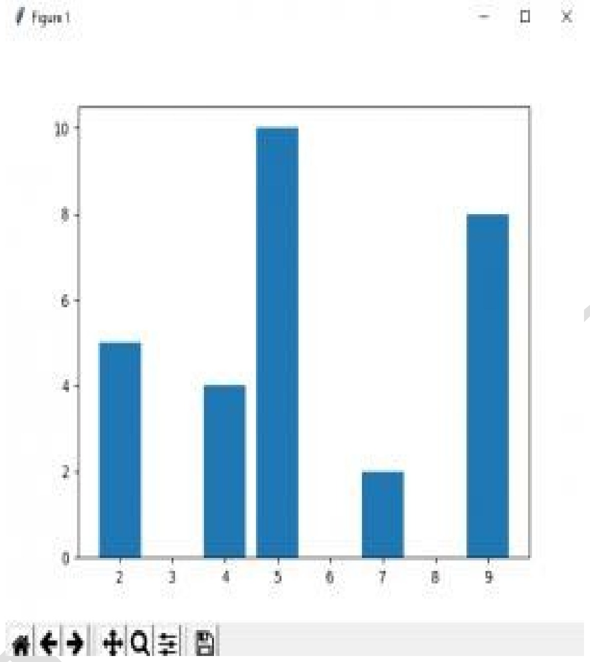
EXAMPLE:	OUTPUT:

from matplotlib import pyplot as plt
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]
plt.plot(x,y)
# function to show the plot 	
plt.show()
	

2. **Bar plot :** It uses bar to compare data.

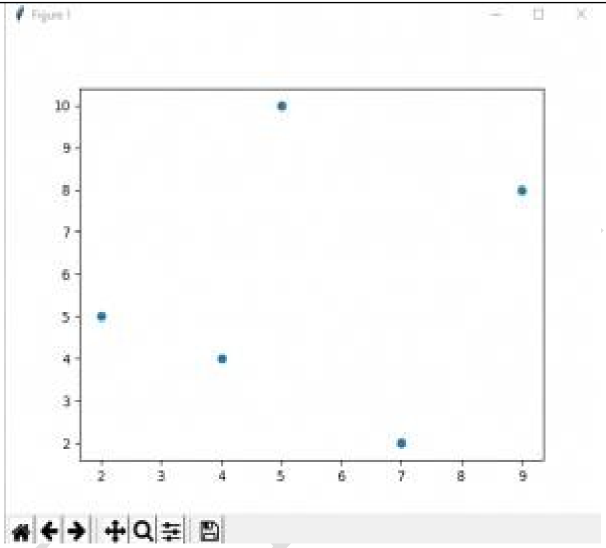
EXAMPLE:	OUTPUT:

SLTIET

from matplotlib import pyplot as plt
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]
plt.bar(x,y)
plt.show()
	

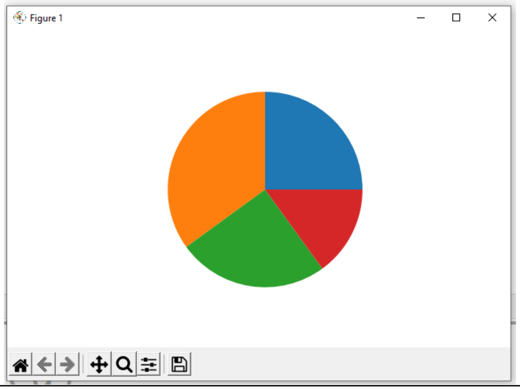
SLTIET

EXAMPLE:	OUTPUT:

from matplotlib import pyplot as plt
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]
plt.scatter(x, y)
plt.show()
	

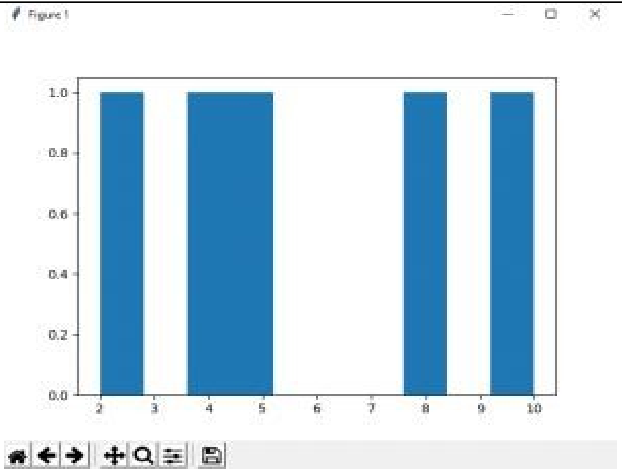
1. **Pie Chart:** It refers to circular graph which is broken down into segments. It is used to show the percentage or proportional data where each slice of pie represents a category.

EXAMPLE:	OUTPUT:

import matplotlib.pyplot as plt
y =[25, 35, 25, 15]
plt.pie(y)
plt.show()
	

5. **Histogram :**

EXAMPLE:	OUTPUT:

from matplotlib import pyplot as plt
y = [10, 5, 8, 4, 2]
plt.hist(y)
plt.show()
	

SLTIET

2. **Scatter Plot :** The data is displayed as a collection of points.

4.6 Create and import custom user defined module

- ✓ We can define our most used functions in a module and import it, instead of copying their definitions into different programs.

Module Name: example

```
# Python Module example
    def add(a, b):
       result = a + b
       print(result)
```

Here, we have defined a function add() inside a module named example. The function takes in two numbers and print their sum.

How to import modules in Python?

- ✓ We can import the definitions inside a module to another module or the interactive interpreter in Python.
- ✓ We use the import keyword to do this. To import our previously defined module example, we type the following in the Python prompt.

```
import example
example.add(4,5.5)
```

- ✓ This does not import the names of the functions defined in example directly in the current symbol table.
- ✓ It only imports the module name example there.
- ✓ Using the module name we can access the function using the dot (.) operator.
- ✓ Standard modules can be imported the same way as we import our user-defined modules.

❖ Various ways to import modules

Python import statement: We can import a module using the import statement and access the definitions inside it using the dot operator.

Example.

```
import math
print("The value of pi is", math.pi)
```

Output:

```
The value of pi is 3.141592653589793
```

SLTIET

Python from...import statement: We can import specific names from a module without importing the module as a whole.

Example: import only pi from math module

```
from math import pi
print("The value of pi is", pi)
```

Output:

```
The value of pi is 3.141592653589793
```

• Here, we imported only the pi attribute from the math module.

• In such cases, we don't use the dot operator. We can also import multiple attributes as follows:

```
from math import pi,e
print("The value of pi is", pi)
print("The value of e is", e)
```

Output:

```
The value of pi is 3.141592653589793
The value of e is 2.718281828459045
```

Import with renaming: We can import a module by renaming it as follows:

Example: import module by renaming it

```
import math as m
print("The value of pi is", m.pi)
```

Output:

```
The value of pi is 3.141592653589793
```

Import all names: We can import all names (definitions) from a module.

Example: import all names from the standard module math

```
from math import *
print("The value of pi is", pi)
```

Here, we have imported all the definitions from the math module. This includes all names visible in our scope except those beginning with an underscore(private definitions).

SLTIET

❖ Creating User Defined Package:

✓ Create a directory with the name EMP in home directory.

✓ Create a Python source file with name p1.py in EMP directory with following code.

```
def getName():
    name=['a','b','c']
    return name
```

- ✓ Now create one more python file with name p2.py in EMP directory with following code.

```
def getSalary():
    salary=[1000,2000,3000]
     return salary
```

- ✓ Now create __init__.py without any code in EMP directory.
- ✓ __init__.py converts Emp directory to Emp package with two modules p1 and p2.

❖ Importing a package in Python

- ✓ To use the module defined in Emp package, Create one file Test.py in home directory with code:

```
from Emp import p1
```

Now run Test.py to see the Output:

```
from Emp import p2
print(p1.getName())
print(p2.getSalary())
```

Output:
```
['a', 'b', 'c']
[1000, 2000, 3000]
```

❖ PIP-Package Installer for Python:

- ✓ PIP is the package manager for Python packages.
- ✓ It is a package management system used to install and manage software packages written in Python.
- ✓ We use pip to install packages that do not come with Python.
- ✓ Python pip comes preinstalled on 3.4 or older version of Python.
- ✓ To check whether pip is installed or not, type below command in terminal. It shows the version of pip which is already installed in the system.

```
pip  -version
```

- ✓ **Pip use PyPI as the default source for packages. So type following command on command prompt for installation of packages.**

```
pip install packagename
```

- ✓ Pip will look for that package on PyPI and if found ,download it and install the package on local system.
- ✓ For **uninstalling a package** type following command on command prompt.

```
pip uninstall packagename
```

Download and Install PIP:

- ✓ Download the get-pip.py file and store it in the same directory as python is installed.
- ✓ Change the current path of the directory in the command line to the path of the directory where get-pip.py file exists.
- ✓ Type following command in command prompt and wait through the installation process.

```
python  get-pip.py
```

- ✓ Type following command to check whether pip is installed successfully or not. If successfully installed then display current version of it.

SLTIET

```
pip -V
```

SLTIET