Recursion in Python :

Question-1: Predict the output

```
houses = ["Eric's house", "Kenny's house", "Kyle's house", "Stan's house"]

def deliver_presents_recursively(houses):

    if len(houses) == 1:
        house = houses[0]
        print("Delivering presents to", house)

    else:
        mid = len(houses) // 2
        first_half = houses[:mid]
        second_half = houses[mid:]

        deliver_presents_recursively(first_half)
        deliver_presents_recursively(second_half)
deliver_presents_recursively(houses)
```

Sol:
```
Delivering presents to Eric's house
Delivering presents to Kenny's house
Delivering presents to Kyle's house
Delivering presents to Stan's house
```

Question-2: Predict the output

```
def sum_recursive(current_number, accumulated_sum):

    if current_number == 11:
        return accumulated_sum

    else:
        return sum_recursive(current_number + 1, accumulated_sum + current_number)
sum_recursive(1, 0)
```

Sol: 55

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

Question-3: Predict the output

```
current_number = 1
accumulated_sum = 0

def sum_recursive():
    global current_number
    global accumulated_sum
    if current_number == 11:
        return accumulated_sum
    else:
        accumulated_sum = accumulated_sum + current_number
        current_number = current_number + 1
        return sum_recursive()
sum_recursive()
```

Sol: 55

Question-4: Predict the output

```
def list_sum_recursive(input_list):
    if input_list == []:
        return 0
    else:
        head = input_list[0]
        smaller_list = input_list[1:]
        return head + list_sum_recursive(smaller_list)
list_sum_recursive([1, 2, 3])
```

Sol: 6

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

Modules in Python:

- Modules refer to a file containing Python statements and definitions.
- A file containing Python code, for example: `example.py`, is called a module, and its module name would be `example`.
- We use modules to break down large programs into small manageable and organized files. Furthermore, modules provide reusability of code.
- We can define our most used functions in a module and import it, instead of copying their definitions into different programs.

Let us create a module. Type the following and save it as `example.py`

```
def add(a, b):
   """This program adds two
   numbers and return the result"""

   result = a + b
   return 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
```

- Using the module name we can access the function using the dot `.` operator. For example:

```
>>> example.add(4,5.5)
9.5
```

Python standard modules:

- We can import a standard module using the `import` statement and access the definitions inside it using the dot operator
- For example, there are many pre-defined functions in standard math module. We need to import math module to support them. Some of them are as follows:

1. ceil() :- This function returns the **smallest integral value greater than the number**. If number is already integer, same number is returned.

2. floor() :- This function returns the **greatest integral value smaller than the number**. If number is already integer, same number is returned.

3. fabs() :- This function returns the **absolute value** of the number

4. exp(a) :- This function returns the value of **e raised to the power a (e**a)** .

5. log(a, b) :- This function returns the logarithmic **value of a with base b**. If base is not mentioned, the computed value is of natural log.

For Example:

```
import math

a = 2.3

# returning the ceil of 2.3
print ("The ceil of 2.3 is : ",
end="")
print (math.ceil(a))

# returning the floor of 2.3
print ("The floor of 2.3 is : ",
end="")
print (math.floor(a))
```

```
import math

# returning the exp of 4
print ("The e**4 value is : ",
end="")
print (math.exp(4))

# returning the log of 2,3
print ("The value of log 2 with
base 3 is : ", end="")
print (math.log(2,3))
```

Import with renaming:

```
# import module by renaming it

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

- We have renamed the `math` module as `m`. This can save us typing time in some cases.
- Note that the name `math` is not recognized in our scope. Hence, `math.pi` is invalid, and `m.pi` is the correct implementation.

Python from...import statement:

- We can import specific names from a module without importing the module as a whole. Here is an example.

```
# import only pi from math module

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

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

Import all names:

- We can import all names(definitions) from a module using the following construct:

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

- Here, we have imported all the definitions from the math module.

Wikipedia:

We can now import Wikipedia in Python using Wikipedia module. Use the incessant flow of knowledge with Python for daily needs.
Install it as:

```
pip install wikipedia
```

And use it as:

```
import wikipedia
result = wikipedia.page("GeeksforGeeks")
print(result.summary)
```

If you wish to get a particular number of sentences from the summary, just pass that as an argument to the summary() function:

```
import wikipedia
print(wikipedia.summary("Debugging", sentences = 2))
```

Emoji:

Emojis have become a way to express and to enhance simple boring texts.
For this, `emoji` module is needed to be installed.
In terminal. Use:

```
pip install emoji
```

To upgrade to the latest packages of emojis. Here's how it can be done:

```
pip install emoji -upgrade
```

```
from emoji import emojize
print(emojize(":thumbs_up:"))
```

Question-1: Predict the output

# A simple module, calc.py def add(x, y): return (x+y)	# importing module calc.py import calc print(add(10, 2))

def subtract(x, y): return (x-y)	print(subtract(10, 2))

Sol: 12
8

Question-2: Predict the output

from math import sqrt, factorial print(sqrt(16)) print(factorial(6))	import math print(math.sqrt(25)) print(math.factorial(3)) print(math.radians(60)) print(math.sin(2)) print(math.cos(0.5)) print(math.tan(0.23))

```
Sol: 4.0
     720
```

```
Sol: 5.0
     6
     1.0471975511965976
     0.9092974268256817
     0.8775825618903728
     0.23414336235146527
```

Question-3: Predict the output of following program using built-in module random and datetime

import random print(random.randint(0, 5)) print(random.random()) print(random.random() * 100) List = [1, 4, True, 800, "python", 27, "hello"] print(random.choice(List))	import datetime now = datetime.datetime.now() print ("Current date and time : ") print (now.strftime("%Y-%m-%d %H:%M:%S"))

```
Sol: 4
     0.49365113728919396
     18.53126970352004
     hello
```

```
Sol: Current date and time :
     2021-01-08 13:04:59
```

Modules in Python

Question-4 (H.W):

Implement a recursive function in Python for the sieve of Eratosthenes.

The sieve of Eratosthenes is a simple algorithm for finding all prime numbers up to a specified integer. It was created by the ancient Greek mathematician Eratosthenes. The algorithm to find all the prime numbers less than or equal to a given integer n:

1. Create a list of integers from two to n: 2, 3, 4, ..., n
2. Start with a counter i set to 2, i.e. the first prime number
3. Starting from i+i, count up by i and remove those numbers from the list, i.e. 2*i, 3*i, 4*i, etc..
4. Find the first number of the list following i. This is the next prime number.
5. Set i to the number found in the previous step
6. Repeat steps 3 and 4 until i is greater than n. (As an improvement: It's enough to go to the square root of n)
7. All the numbers, which are still in the list, are prime numbers

Sol:

```
from math import sqrt
def sieve(n):
    # returns all primes between 2 and n
    primes = list(range(2,n+1))
    max = sqrt(n)
    num = 2
    while num < max:
        i = num
        while i <= n:
            i += num
            if i in primes:
                primes.remove(i)
        for j in primes:
            if j > num:
                num = j
                break
    return primes
print(sieve(100))
```