USN										

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

INTERNAL ASSESSMENT TEST – II

Sub:	Python Programming							Code:	21EC643
Date:	11 / 07 / 2024	Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	ECE

Answer any 5 full questions

		Marks	CO	RBT
1	Explain various modes of opening text files in Python. Write an example program for each mode. Following are the different modes of opening text files in Python. ‘r’ – read mode (default mode) ‘w’ – write mode (existing data will be overwritten) ‘a’ – append mode (write at the end of existing content) ‘x’ – create a new file and open for writing There are three steps to reading or writing files in Python Step 1. Call the open() function to return a File object. Step 2. Call the read() or write() method on the File object. Step 3. Close the file by calling the close() method on the File object. Example 1 : How to Read a Text File x=open('C:\\\\Users\\\\RAVEESH HEGDE\\\\Desktop\\\\Python Class\\\\sample file 1.txt') #By default, file will be opened in read mode x.read() #This will print the content of the file Example 2 : How to Read First 5 Characters of a Text File x=open('C:\\\\Users\\\\RAVEESH HEGDE\\\\Desktop\\\\Python Class\\\\sample file 1.txt') #By default, file will be opened in read mode x.read(5)	10	CO3	L2

#This will print the first 5 characters of the file

Example 3 : How to Read a Text File Line By Line

x=open('C:\\Users\\RAVEESH HEGDE\\Desktop\\Python Class\\sample file 1.txt')

#By default, file will be opened in read mode

x.readlines()

#This will print the content of the file as a list.

#First element of the list will be first line of the text file, second element will be second line of

the text file and so on.

Example 4 : How to Read a Specific Line from a Text File

x=open('C:\\Users\\RAVEESH HEGDE\\Desktop\\Python Class\\sample file 1.txt')

#By default, file will be opened in read mode

x.readlines(5)

#This will print the 5

th line of the text file

#First element of the list will be first line of the text file, second element will be second line of

the text file and so on.

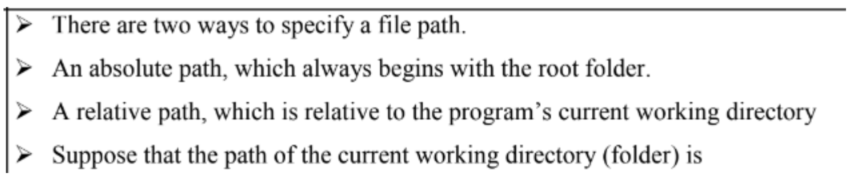
Example 5 : How to Write to a Text File (Overwrite the Contents)

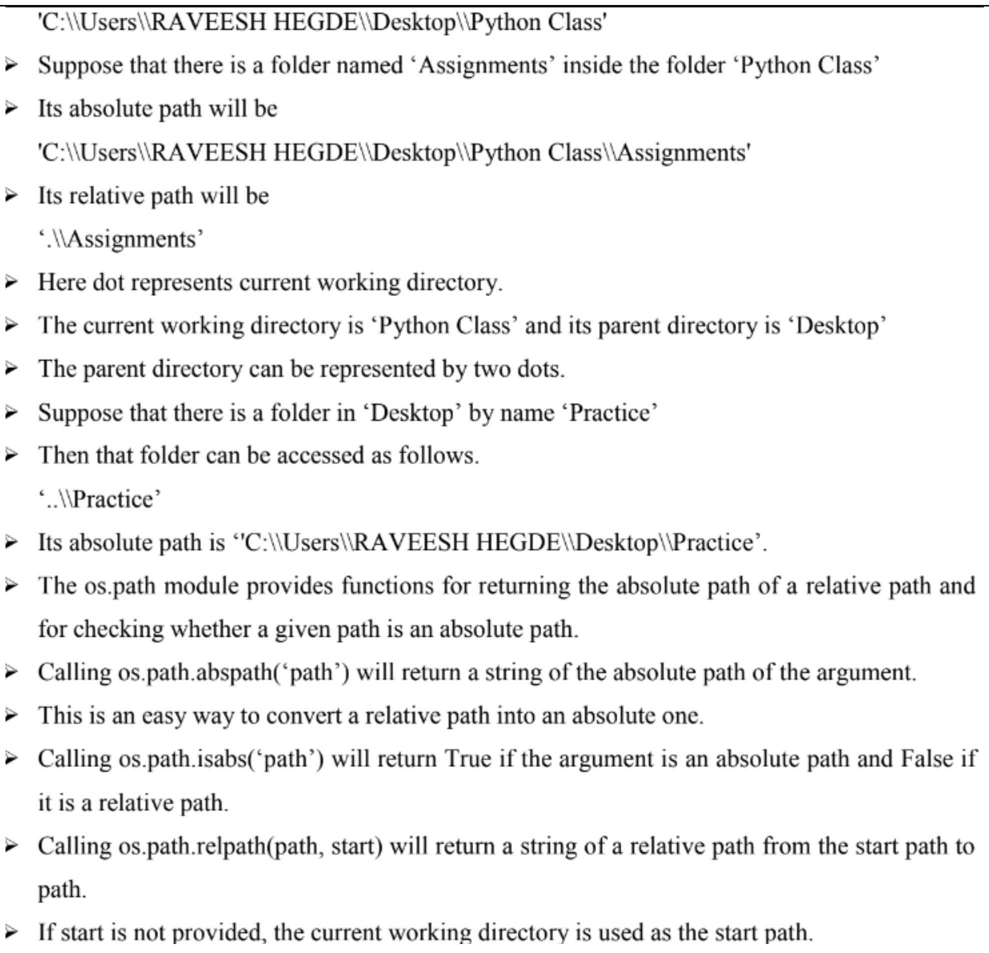
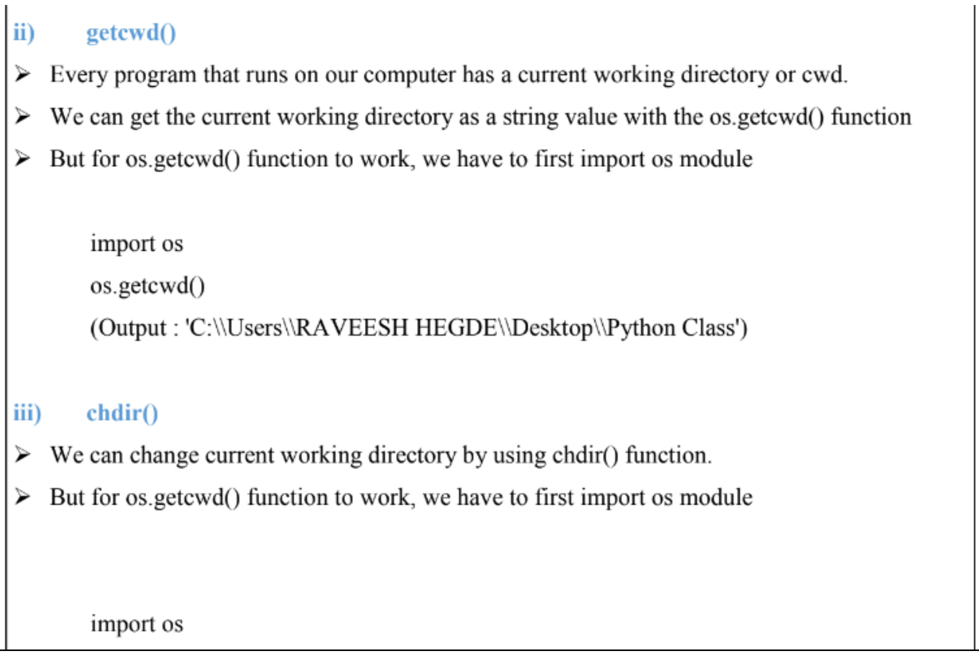
x=open('C:\\Users\\RAVEESH HEGDE\\Desktop\\Python Class\\sample file 2.txt‘,’w’)

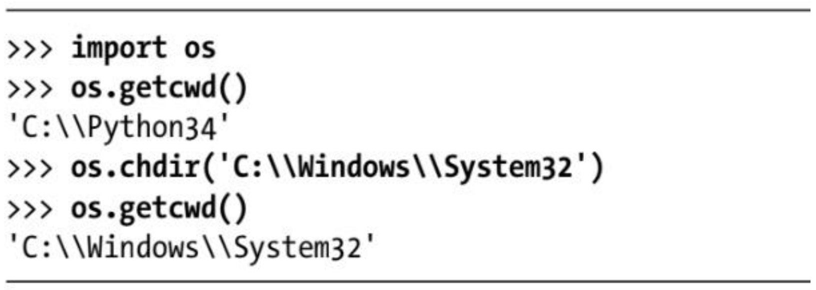
#’w’ will open the file in write mode

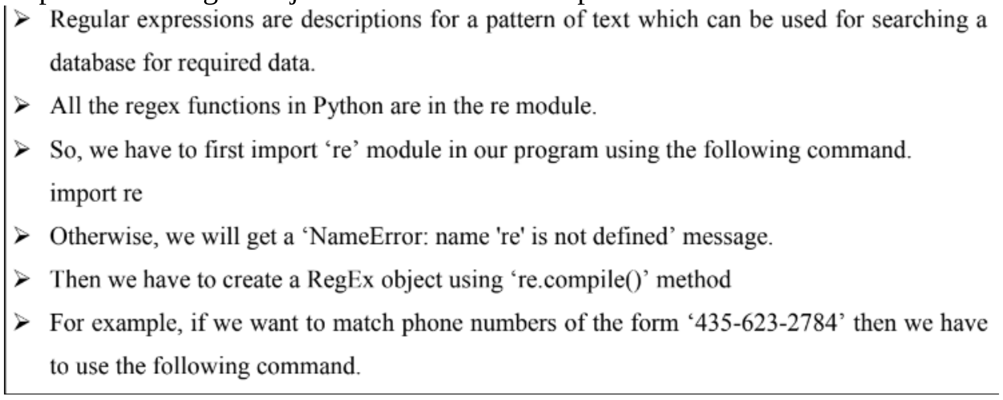
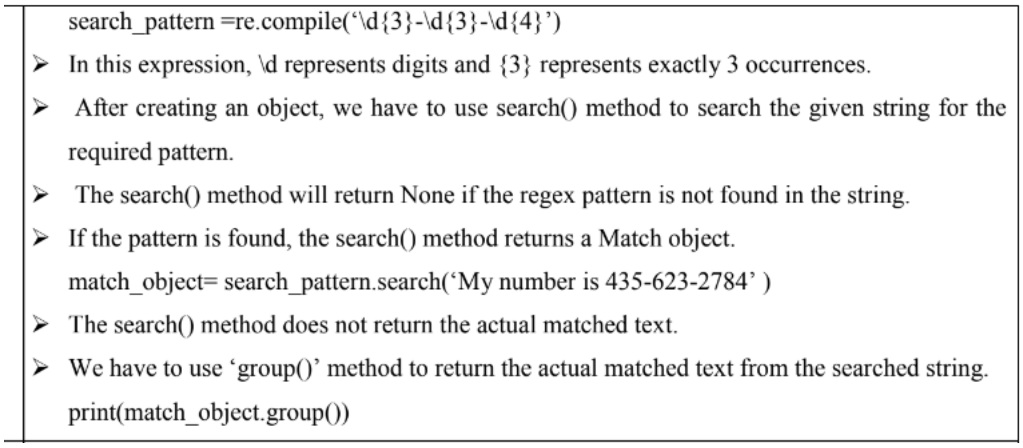
#If the specified file doesn’t exist, then it will not return an error.

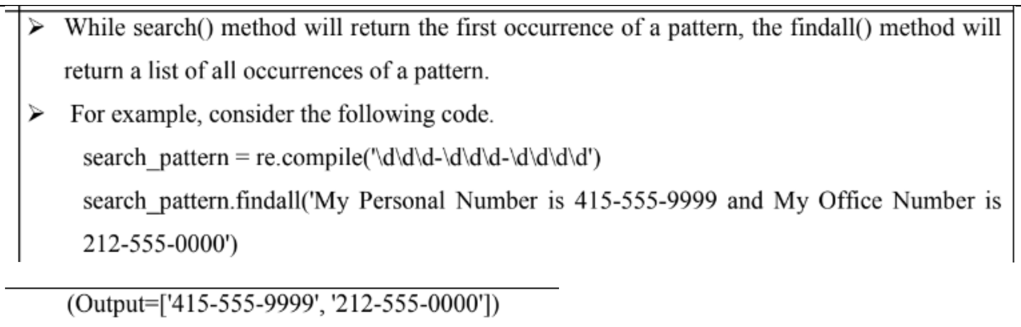
#But it will create a new file with the required name

	x.write(‘This is the new content’) x.close() **Example 6 : How to Write to a Text File (Append to the Existing Content)** x=open('C:\\Users\\RAVEESH HEGDE\\Desktop\\Python Class\\sample file 2.txt‘,’a’) #’a’ will open the file in append mode #If the specified file doesn’t exist, then it will not return an error. #But it will create a new file with the required name x.write(‘This is the new content’) x.close() **Example 7 : How to Create a New Text File** x=open('C:\\Users\\RAVEESH HEGDE\\Desktop\\Python Class\\sample file 3.txt‘,’x’) #This will create a file with the given name x.write(‘This is the new content’) x.close()			
2a	Explain the concept of file path. Discuss absolute and relative file paths.  ➢ There are two ways to specify a file path. ➢ An absolute path, which always begins with the root folder. ➢ A relative path, which is relative to the program’s current working directory ➢ Suppose that the path of the current working directory (folder) is	5	CO3	L2

	 'C:\\Users\\RAVEESH HEGDE\\Desktop\\Python Class' ➢ Suppose that there is a folder named 'Assignments' inside the folder 'Python Class' ➢ Its absolute path will be 'C:\\Users\\RAVEESH HEGDE\\Desktop\\Python Class\\Assignments' ➢ Its relative path will be '.\\Assignments' ➢ Here dot represents current working directory. ➢ The current working directory is 'Python Class' and its parent directory is 'Desktop' ➢ The parent directory can be represented by two dots. ➢ Suppose that there is a folder in 'Desktop' by name 'Practice' ➢ Then that folder can be accessed as follows. '..\\Practice' ➢ Its absolute path is ''C:\\Users\\RAVEESH HEGDE\\Desktop\\Practice'. ➢ The os.path module provides functions for returning the absolute path of a relative path and for checking whether a given path is an absolute path. ➢ Calling os.path.abspath('path') will return a string of the absolute path of the argument. ➢ This is an easy way to convert a relative path into an absolute one. ➢ Calling os.path.isabs('path') will return True if the argument is an absolute path and False if it is a relative path. ➢ Calling os.path.relpath(path, start) will return a string of a relative path from the start path to path. ➢ If start is not provided, the current working directory is used as the start path.			
2b	Explain how we can get the current working directory and change the current working directory in Python.  **ii) getcwd()** ➢ Every program that runs on our computer has a current working directory or cwd. ➢ We can get the current working directory as a string value with the os.getcwd() function ➢ But for os.getcwd() function to work, we have to first import os module import os os.getcwd() (Output : 'C:\\Users\\RAVEESH HEGDE\\Desktop\\Python Class') **iii) chdir()** ➢ We can change current working directory by using chdir() function. ➢ But for os.getcwd() function to work, we have to first import os module import os	5	CO3	L2

	
	`>>> import os`
	`>>> os.getcwd()`
	`'C:\\Python34'`
	`>>> os.chdir('C:\\Windows\\System32')`
	`>>> os.getcwd()`
	`'C:\\Windows\\System32'`			

		Marks	CO	RBT
3a	Explain how RegEx object can be created and pattern can be matched.  ➢ Regular expressions are descriptions for a pattern of text which can be used for searching a database for required data. ➢ All the regex functions in Python are in the re module. ➢ So, we have to first import 're' module in our program using the following command. import re ➢ Otherwise, we will get a 'NameError: name 're' is not defined' message. ➢ Then we have to create a RegEx object using 're.compile()' method ➢ For example, if we want to match phone numbers of the form '435-623-2784' then we have to use the following command.  search_pattern =re.compile('\d{3}-\d{3}-\d{4}') ➢ In this expression, \d represents digits and {3} represents exactly 3 occurrences. ➢ After creating an object, we have to use search() method to search the given string for the required pattern. ➢ The search() method will return None if the regex pattern is not found in the string. ➢ If the pattern is found, the search() method returns a Match object. match_object= search_pattern.search('My number is 435-623-2784') ➢ The search() method does not return the actual matched text. ➢ We have to use 'group()' method to return the actual matched text from the searched string. print(match_object.group())	5	CO3	L2
3b	Explain findall() and search() method with respect to RegEx.	5	CO3	L2



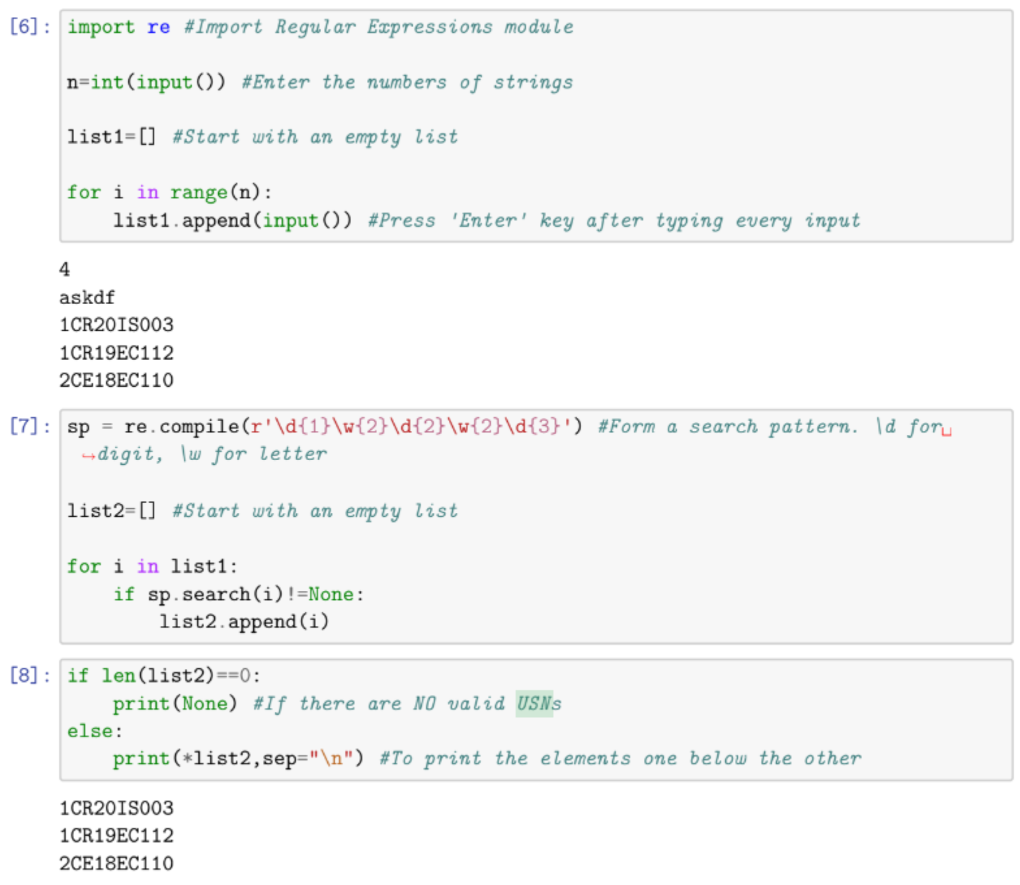
- While search() method will return the first occurrence of a pattern, the findall() method will return a list of all occurrences of a pattern.
- For example, consider the following code.

  search_pattern = re.compile('\d\d\d-\d\d\d-\d\d\d\d')

  search_pattern.findall('My Personal Number is 415-555-9999 and My Office Number is 212-555-0000')

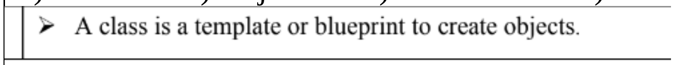
(Output=['415-555-9999', '212-555-0000'])

4	You are given a list of strings. Write a Python program to print only the valid USNs present in the list using Regular Expressions. An example of a valid USN : 1CR21EC285	10	CO3	L3

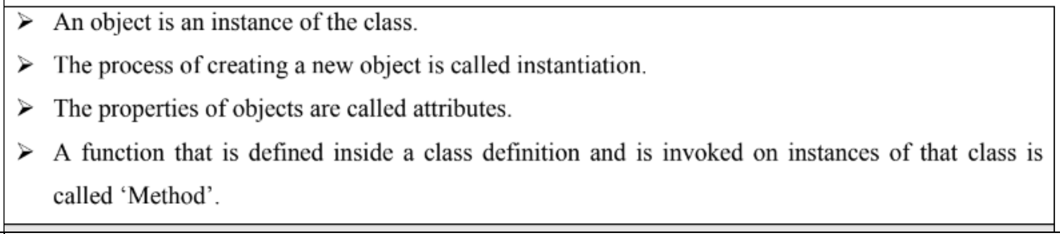


```
[6]: import re #Import Regular Expressions module

     n=int(input()) #Enter the numbers of strings

     list1=[] #Start with an empty list

     for i in range(n):
         list1.append(input()) #Press 'Enter' key after typing every input

     4
     askdf
     1CR20IS003
     1CR19EC112
     2CE18EC110

[7]: sp = re.compile(r'\d{1}\w{2}\d{2}\w{2}\d{3}') #Form a search pattern. \d for␣
      ↪digit, \w for letter

     list2=[] #Start with an empty list

     for i in list1:
         if sp.search(i)!=None:
             list2.append(i)

[8]: if len(list2)==0:
         print(None) #If there are NO valid USNs
     else:
         print(*list2,sep="\n") #To print the elements one below the other

     1CR20IS003
     1CR19EC112
     2CE18EC110
```

5a	Define the following terms with respect to object oriented programming. i) Class ii) Object iii) Attribute iv) Method	4	CO4	L2



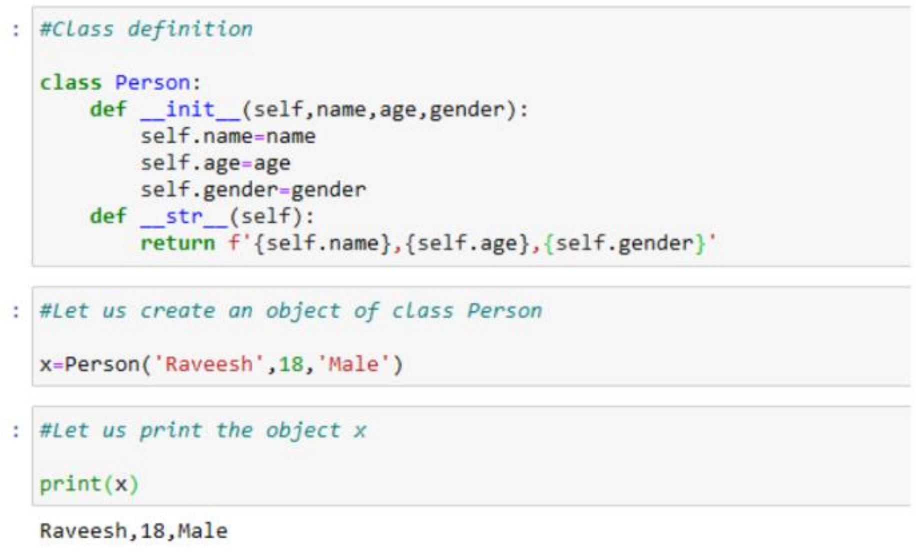
- A class is a template or blueprint to create objects.



- An object is an instance of the class.
- The process of creating a new object is called instantiation.
- The properties of objects are called attributes.
- A function that is defined inside a class definition and is invoked on instances of that class is called ‘Method’.

5b	Explain init() and str() methods with an example python program.	6	CO4	L2

- All classes have a function called _ _init_ _( ), which is executed by default when the object is created.
- The _ _init_ _( ) function assigns values to object properties or attributes.
- The _ _init_ _( ) function is called automatically when an object of that class is created.
- The 'self ' parameter is a reference to the current object.
- It is used to access the variables or attributes that belong to the class.
- The _ _str()_ _ method is an optional method that can be added to a class to return a human readable string representation of the object.
- For example, consider the following program.



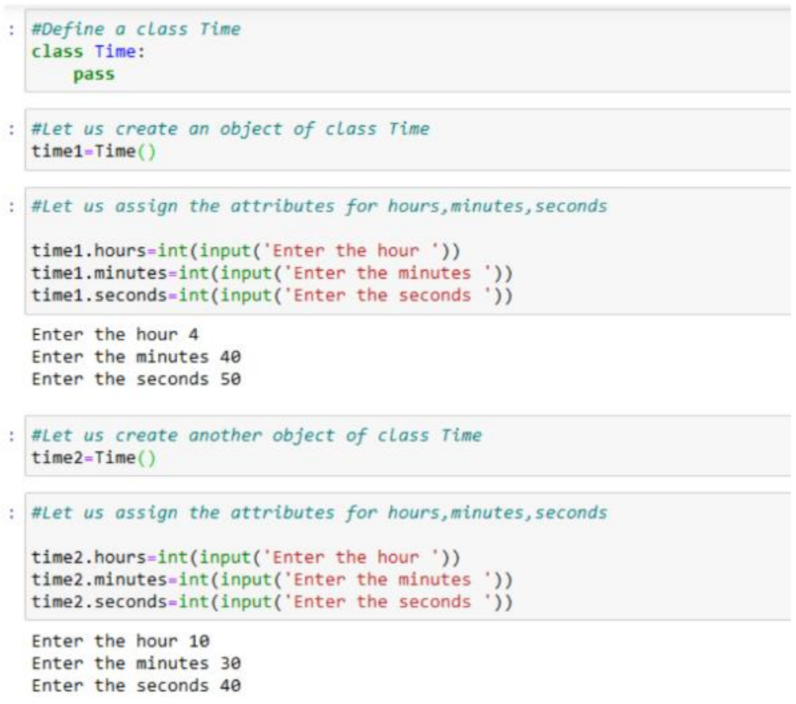
```
#Class definition

class Person:
    def __init__(self,name,age,gender):
        self.name=name
        self.age=age
        self.gender=gender
    def __str__(self):
        return f'{self.name},{self.age},{self.gender}'
```

```
#Let us create an object of class Person

x=Person('Raveesh',18,'Male')
```

```
#Let us print the object x

print(x)
```

```
Raveesh,18,Male
```

6	Create a Time class with hour, min and sec as attributes. Develop a function to add two Time objects.	10	CO4	L3



```
#Define a class Time
class Time:
    pass
```

```
#Let us create an object of class Time
time1=Time()
```

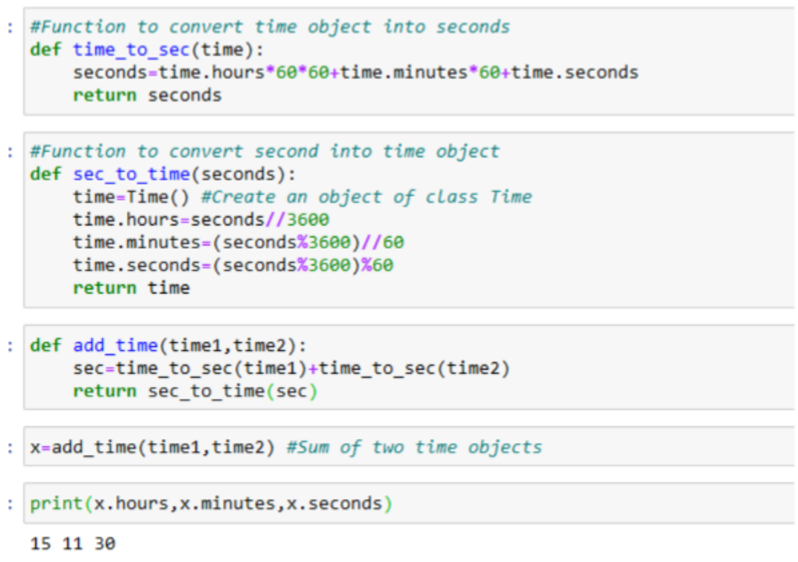
```
#Let us assign the attributes for hours,minutes,seconds

time1.hours=int(input('Enter the hour '))
time1.minutes=int(input('Enter the minutes '))
time1.seconds=int(input('Enter the seconds '))
```

```
Enter the hour 4
Enter the minutes 40
Enter the seconds 50
```

```
#Let us create another object of class Time
time2=Time()
```

```
#Let us assign the attributes for hours,minutes,seconds

time2.hours=int(input('Enter the hour '))
time2.minutes=int(input('Enter the minutes '))
time2.seconds=int(input('Enter the seconds '))
```

```
Enter the hour 10
Enter the minutes 30
Enter the seconds 40
```



```
: #Function to convert time object into seconds
  def time_to_sec(time):
      seconds=time.hours*60*60+time.minutes*60+time.seconds
      return seconds

: #Function to convert second into time object
  def sec_to_time(seconds):
      time=Time() #Create an object of class Time
      time.hours=seconds//3600
      time.minutes=(seconds%3600)//60
      time.seconds=(seconds%3600)%60
      return time

: def add_time(time1,time2):
      sec=time_to_sec(time1)+time_to_sec(time2)
      return sec_to_time(sec)

: x=add_time(time1,time2) #Sum of two time objects

: print(x.hours,x.minutes,x.seconds)

  15 11 30
```

CI CCI HoD-ECE

HoD-ECE