USN [| | | | | | | | |]

21EC643

Sixth Semester B.E. Degree Examination, June/July 2024
Python Programming

Time: 3 hrs. Max. Marks: 100

Note: ***Answer any FIVE full questions, choosing ONE full question from each module.***

Module-1

1 a. List and explain the significant features of Python programming language. **(08 Marks)**

b. Write the math operators in Python from highest to lowest precedence with an example for each. Write the steps how Python is evaluating the expression (5 – 1) * (7 + 1) / (3 – 1) and reduces it to a single value. **(06 Marks)**

c. Demonstrate the syntax of Python functions : print( ), input( ) and str( ) with examples. **(06 Marks)**

OR

2 a. With proper syntax and examples, explain the control statements.
i) if ii) else iii) elif iv) break statement. **(08 Marks)**

b. Explain the local and global scope of the variable with a suitable example. **(06 Marks)**

c. Write a Python snippet to generate the Fibonacci series. **(06 Marks)**

Module-2

3 a. What are the lists? Explain append( ), insert( ) and remove( ) methods with examples. **(08 Marks)**

b. For the following two questions, spam contains the list [‘a’, ‘b’, ‘c’, ‘d’, [3, 4, 7, 2]]
i) What does span [-2] evaluate to?
ii) What does span [4] [1] evaluate to? **(05 Marks)**

c. Demonstrate with example of upper( ), lower( ), and isupper( ) and islower( ) string methods. **(07 Marks)**

OR

4 a. What is a dictionary? Compare dictionaries with lists. Write a program to count the number of occurrences of characters in string. **(08 Marks)**

b. Write a program to implement search and replace multiple occurrences of a given substring in the main string in a list. **(07 Marks)**

c. Define Tuple data type, explain converting types with the list( ) and tuple ( ) Functions. **(05 Marks)**

Module-3

5 a. With example, explain the following Pattern Matching with Regular Expressions.
i) Grouping with Parentheses
ii) Matching Multiple Groups with the Pipe. **(10 Marks)**

b. What are the steps involved in file handling? Also, explain the reading and writing process with suitable examples in Python. **(10 Marks)**

OR

6 a. Explain the basic steps for creating and finding regular expression objects with Python. **(06 Marks)**

b. Writ a python program to accept USN and marks objected. Find maximum, minimum and USN students who scored 100-85, 85-75, 75-60 and below 60 marks separately. **(06 Marks)**

c. Explain the purpose of the following special characters used in optimal matching regular expression: ?, *, +, and {}. Illustrate with example. **(08 Marks)**

Module-4

7 a. Differentiate between class variables and instance variables with suitable examples. **(05 Marks)**

b. Write a program to create a class classed Rectangle with the help of a corner point, width and height. Write the following function sand demonstrate their working :

i) To find and display the center of the rectangle
ii) To display point as an ordered pair
iii) To resize the rectangle
iv) To find area and perimeter of a rectangle. **(10 Marks)**

c. Justify the statement "Objects are mutable" with suitable examples. **(05 Marks)**

OR

8 a. Explain – intit_( ) and – str_( ) methods with an example. **(10 Marks)**

b. Explain operator overloading and polymorphism with examples. **(10 Marks)**

Module-5

9 a. Write a Python program that makes a socket connection to a web server and follows the rules of the HTTP protocol to request a document and display what the server sends back. **(10 Marks)**

b. Illustrate with a python program how to retrieve web pages with urllib. **(10 Marks)**

OR

10 a. What is Service – Oriented Architecture (SOA)? List out the advantages of SOA. **(06 Marks)**

b. Discuss various keys used in the database model. **(06 Marks)**

c. Write the four SQL commands needed to create and maintain data. **(08 Marks)**

* * * * *

CMR INSTITUTE OF TECHNOLOGY, BENGALURU DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING PYTHON PROGRAMMING (21EC643) VI Sem - As Per 2021 Scheme Question Paper

S.No.	Question
1 a)	**List and explain the significant features of Python Programming Language.**
	➢ Free and Open Source ➢ Easy to code ➢ Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc ➢ Easy to Read ➢ Object-Oriented Language ➢ GUI Programming Support ➢ Large Community Support ➢ Easy to Debug ➢ Python is a Portable language ➢ Python is an Integrated language ➢ Interpreted Language: ➢ Large Standard Library ➢ Dynamically Typed Language ➢ Frontend and backend development ➢ Allocating Memory Dynamically
1b	**Write the math operators in Python from highest to lowest Precedencewith an example for each.**

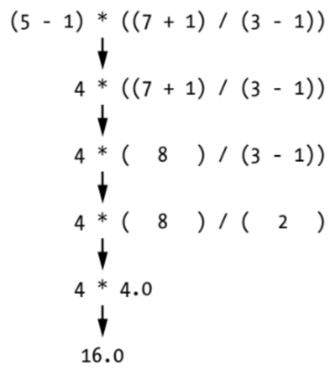
Write the steps how Python is evaluating the expression (5 - 1) * ((7 + 1) / (3 - 1)) and reduces it to a single value.

- Following are the math operators in Python from highest to lowest preference.

Operator	Operation	Example	Evaluates To
**	Exponent	2**3	8
%	Modulus/Remainder	5%2	1
//	Integer Division	7//2	3
/	Division	5/2	2.5
*	Multiplication	5*2	10
-	Subtraction	5-2	3
+	Addition	5+2	7

- We can use parenthesis to override the usual precedence.
- For example 2+3*4 evaluates to 14 as * has higher precedence than +
- But (2+3)*4 evaluates to 20 as 2+3 is within parenthesis.

- Evaluating theexpression (5 - 1) * ((7 + 1) / (3 - 1))



```
(5 - 1) * ((7 + 1) / (3 - 1))
        ↓
      4 * ((7 + 1) / (3 - 1))
        ↓
      4 * (   8   ) / (3 - 1))
        ↓
      4 * (   8   ) / (  2   )
        ↓
      4 * 4.0
        ↓
       16.0
```

-

1c **Determine the syntax of Python Functions: print(),, input(), str() with examples.**

print()

- The print() function displays the string value inside the parentheses on the screen.
- For example, print(‘Hello’) displays Hello on the screen
- The quotes are not displayed on the screen.
- They just mark where the string begins and ends; they are not part of the string value.
- When Python execute the instruction print(‘Hello’), we can say that Python is calling

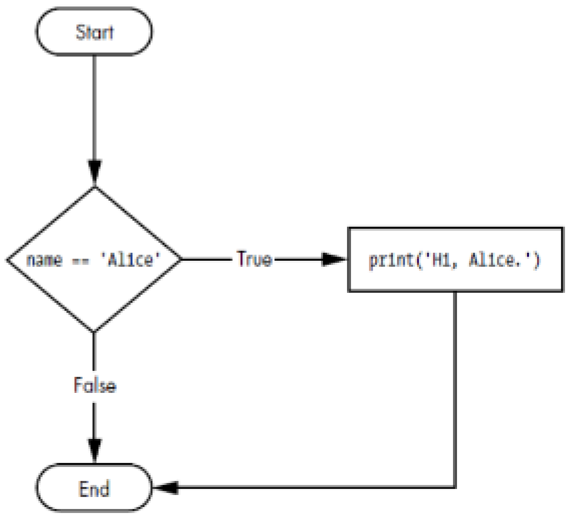
	➢ the print() function and the string value is being passed to the function. ➢ A value that is passed to a function called as an argument. ➢ We can also use print() function to put a blank line on the screen; just call print() with nothing in between the parentheses. Print('Hello"): >>> Hello input() ➢ The input() function waits for the user to type some text on the keyboard and press ➢ enter. For example, >>>name=input() ➢ This instruction takes a user input of string type and assigns it to the variable 'name'. str() ➢ str() function is used to convert integer or float value to string value. ➢ For example, >>>str(-3.14) converts float value -3.14 into string value '-3.14' In the resulting value, -, 3, ., 1, 4 all are considered as string values. ➢ The str() function is useful when we have an integer or float which we want to concatenate to a string.
2a	**With proper syntax and examples, explain the control statements:** **i)if, ii) else, iii) elif, iv) break statement**

a) if()

- The most common type of flow control statement is the if statement.
- An if statement's clause (that is, the block following the if statement) will execute if the statement's condition is True.
- The clause is skipped if the condition is False.
- In Python, an if statement consists of the following:
- The if keyword
- A condition (that is, an expression that evaluates to True or False)
- A colon
- Starting on the next line, an indented block of code (called the if clause)
- For example, consider the following code.

```
if name == 'Alice':
        print('Hi, Alice.')
```

- The flow chart of this code looks as follows.



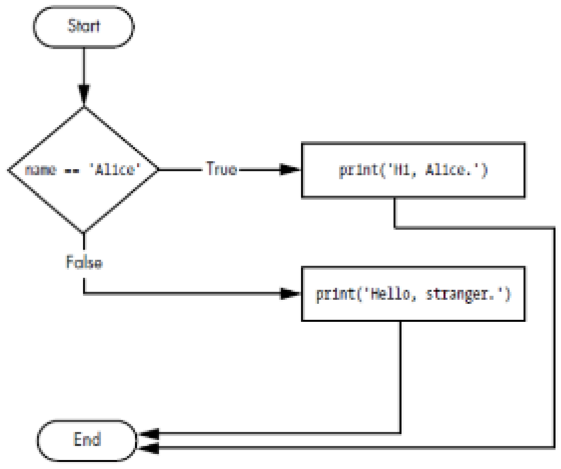
b) else()

- An if clause can optionally be followed by an else statement.
- The else clause is executed only when the if statement's condition is False.
- An else statement always consists of the following :

- The else keyword
- A colon
- Starting on the next line, an indented block of code (called the else clause)
- For example, consider the following code.

```
if name == 'Alice':
        print('Hi, Alice.')
else:
        print('Hello, stranger.')
```

- The flow chart of this code looks as follows.

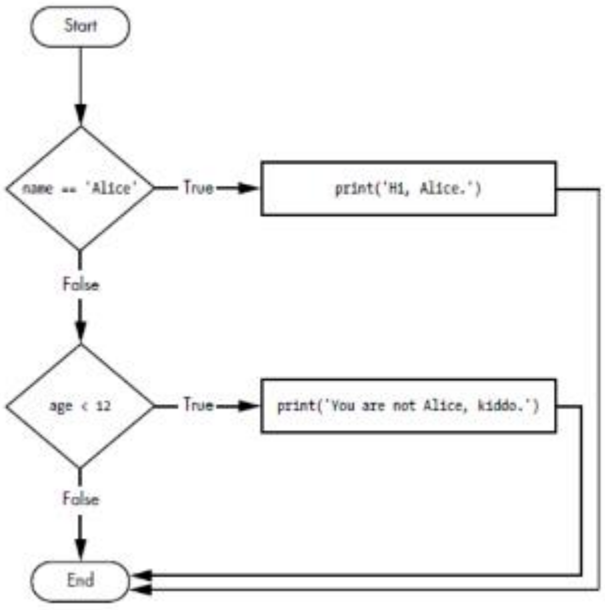


c) elif()

- We may have a case where we want one of many possible clauses to execute.
- The elif() statement helps to do that.
- The elif statement always follows an if or another elif statement.
- It provides another condition that is checked only if the previous conditions were False.
- In code, an elif statement always consists of the following:
- The elif keyword
- A condition (that is, an expression that evaluates to True or False)
- A colon
- Starting on the next line, an indented block of code (called the elif clause)
- Consider the following code.

```
if name==’Alice’:
      print(‘Hi, Alice’)
elif age<12:
      print(‘You are not Alice, kiddo.’)
```

- The flow chart of this code looks as follows.

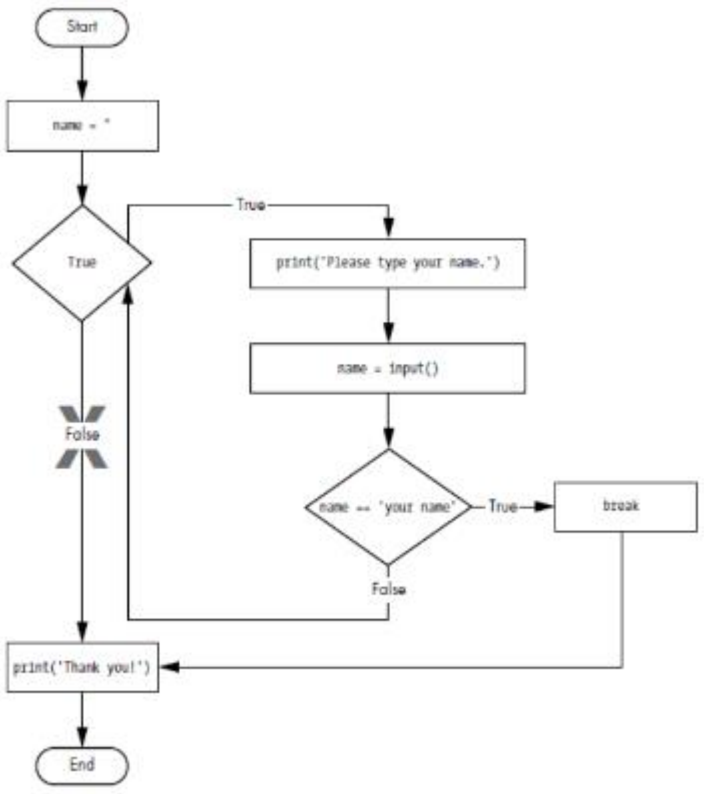


f) break()

- break() statement is useful when we want to break out of a loop early.
- If the execution reaches a break statement, it immediately exits the for loop's or while loop's clause
- In code, a break statement simply contains the break keyword.
- For example, consider the following code.

```
while True:
        print('Please type your name.')
        name = input()
        if name == 'your name':
            break
print('Thank you!')
```

- while True: creates an infinite loop.
- The program execution will always enter the loop and will exit only when a break statement is executed

	 Start name = '' True True print('Please type your name.') name = input() False name == 'your name' True break False print('Thank you!') End
2b	**Explain the local and global scope of the variable with a suitable example.**
	➢ A local scope is created whenever a function is called. ➢ Any variables assigned in this function exist within the local scope. ➢ When the function returns, the local scope is destroyed, and these variables are forgotten. ➢ Scopes matter for several reasons: • Code in the global scope cannot use any local variables. • However, a local scope can access global variables. • Code in a function's local scope cannot use variables in any other local scope. • We can use the same name for different variables if they are in different scopes. • That is, there can be a local variable named spam and a global variable also named spam. ➢ There are 4 rules to tell whether a variable is in local scope or global scope • If a variable is being used in the global scope (that is, outside of all functions), then it is always a global variable. • If there is a global statement for that variable in a function, it is a global variable. • Otherwise, if the variable is used in an assignment statement in the function, it is a local variable. • But if the variable is not used in an assignment statement, it is a global variable.

- For example, consider the following function.

```
#Function definition
def find_sum(a,b):
    c=a+b
    return c
#Main program
x=2
y=3
z=find_sum(x,y)
print(z)
```

- In this example, c is a local variable. Because it is used in an assignment statement within the function find_sum()
- x,y and z are global variables. Because they are defined outside the function.
- If we want to modify the value stored in a global variable from in a function, we must use a global statement on that variable.
- For example, consider the following function.

```
x=3
#Function definition
def fun():
    global x
    x=4

#Main program
fun()
print(x)
```

- Even though value of x is changed within the function, its value is 4 outside the function also.
- Because, within the function, x is declared as a global variable.
- Hence, print(x) prints 4, not 3.

2c **Write a Python Snippet to generate Fibonacci Series.**

Solution

```
n = 10
num1 = 0
num2 = 1
next_number = num2
count = 1

while count <= n:
    print(next_number, end=" ")
    count += 1
    num1, num2 = num2, next_number
    next_number = num1 + num2
print()
```

3a **What is a list? Explain append(), insert() and remove() methods with examples.**

- A list is an ordered and mutable collection of elements.
- A list begins with an opening square bracket and ends with a closing square bracket.
- Values inside the list are called items.
- Items are separated with commas.
- For example,

list1=[2,'cat',2.4]

- The items of the list can be accessed by their index.
- The first value in the list is at index 0, the second value is at index 1 and so on.
- For example, list1[0] returns 2, list1[1] returns 'cat'.
- Indexes can be only integer values, not floats.
- For example, list1[1.0] results in TypeError.
- If we use an index that exceeds the length of the list, then Python will return IndexError.
- For example, list1[10] returns IndexError.
- Lists can also contain other list values.
- The values in these lists of lists can be accessed using multiple indexes.
- For example,

list2=[2,[3,4,5],[6,7,8]]

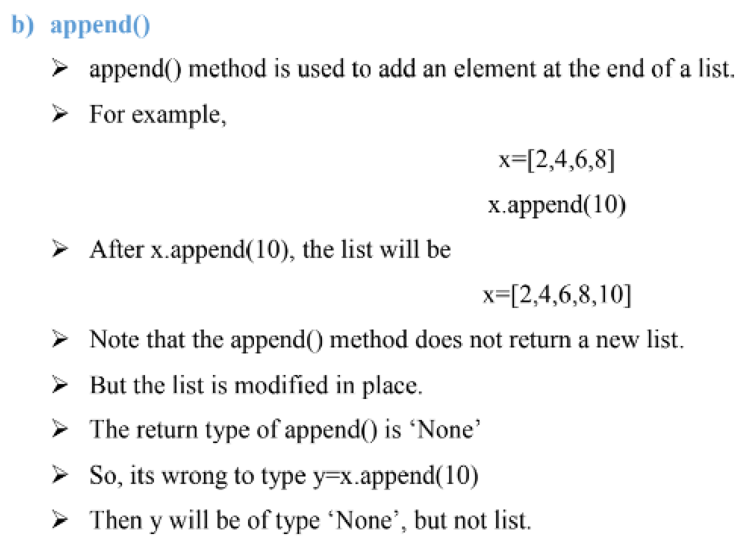
- list2[1] returns [3,4,5].
- list2[1][0] returns 3.
- list2[1][1] returns 4.
- We can also use negative integers for the index.
- The integer value -1 refers to the last index in a list, the value -2 refers to the second-to-last
- index in a list, and so on.
- list3=[2,4,6,8,10]

list3[1:4] returns [4,6,8]

- We can leave out one or both of the indexes on either side of the colon in the slice.
- Leaving out the first index is the same as using 0, or the beginning of the list.
- Leaving out the second index is the same as using the length of the list, which will slice to the end of the list.
- For example, list3[:4] returns [2,4,6,8]

  list3[2:] returns [6,8,10]

  list3[:] returns [2,4,6,8,10



b) append()

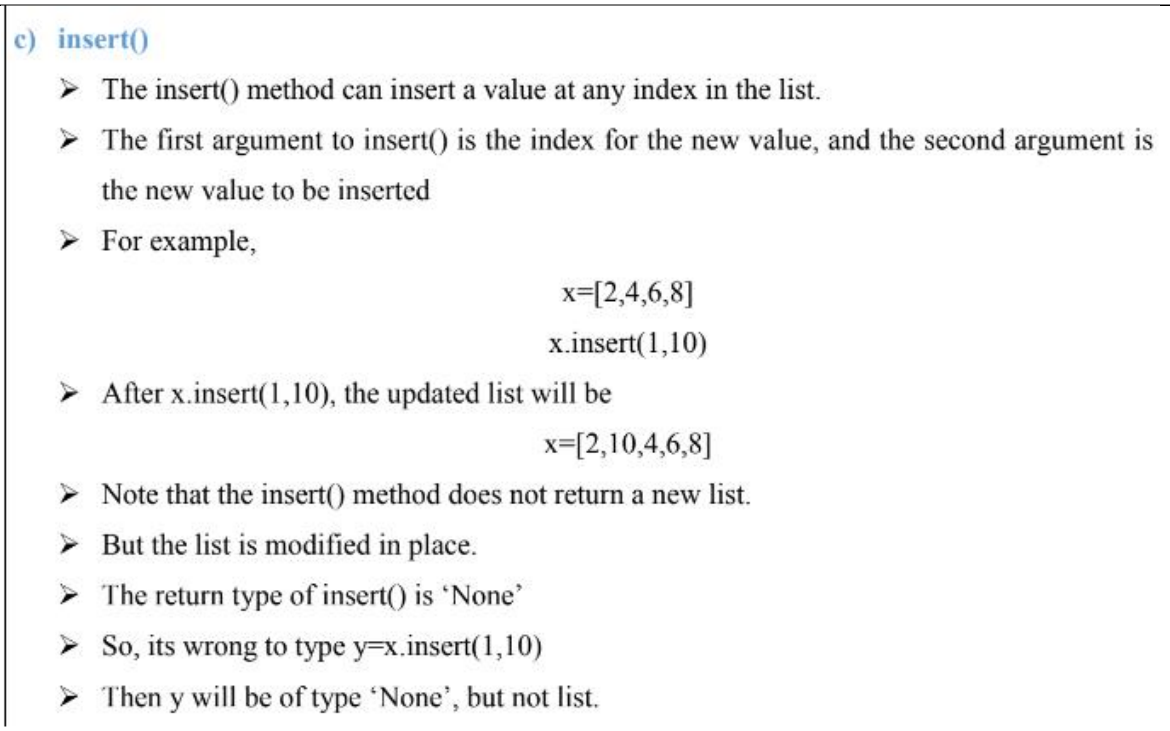
- append() method is used to add an element at the end of a list.
- For example,

```
x=[2,4,6,8]
x.append(10)
```

- After x.append(10), the list will be

```
x=[2,4,6,8,10]
```

- Note that the append() method does not return a new list.
- But the list is modified in place.
- The return type of append() is 'None'
- So, its wrong to type y=x.append(10)
- Then y will be of type 'None', but not list.



c) insert()

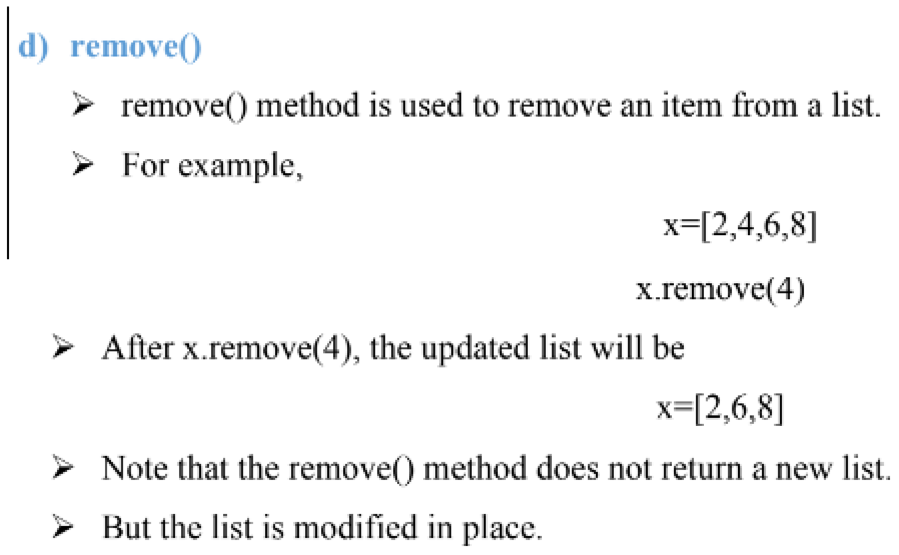
- The insert() method can insert a value at any index in the list.
- The first argument to insert() is the index for the new value, and the second argument is the new value to be inserted
- For example,

```
x=[2,4,6,8]
x.insert(1,10)
```

- After x.insert(1,10), the updated list will be

```
x=[2,10,4,6,8]
```

- Note that the insert() method does not return a new list.
- But the list is modified in place.
- The return type of insert() is 'None'
- So, its wrong to type y=x.insert(1,10)
- Then y will be of type 'None', but not list.



d) remove()

- remove() method is used to remove an item from a list.
- For example,

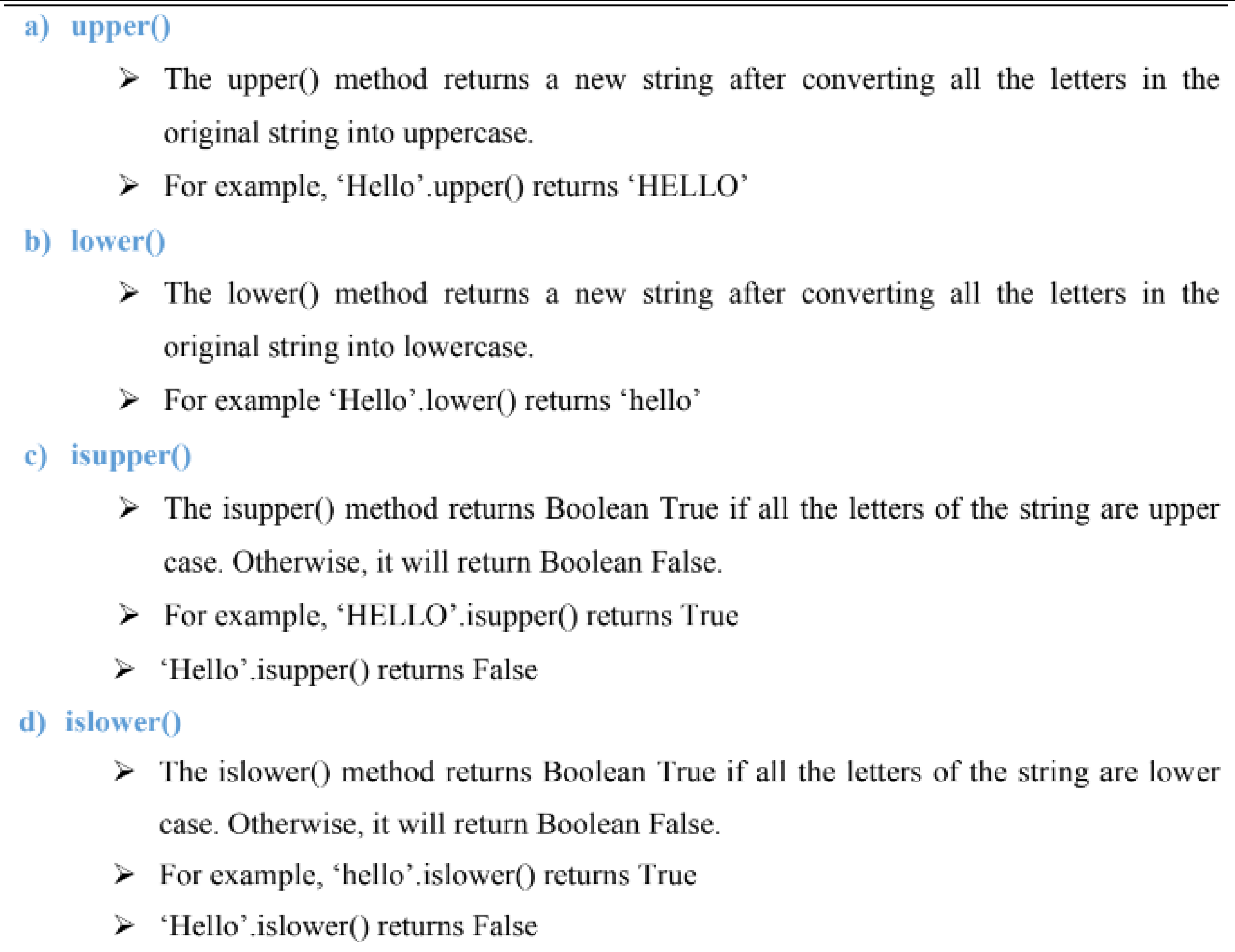
```
x=[2,4,6,8]
x.remove(4)
```

- After x.remove(4), the updated list will be

```
x=[2,6,8]
```

- Note that the remove() method does not return a new list.
- But the list is modified in place.

3b	**For the following two questions, spam contains the list ['a','b','c','d',[3,4,7,2]]** **i) What does spam[-2] evaluate to ?** **ii) What does spam[4][1] evaluate to?**
	➢ i) spam[-2]='d'

	➢ ii) spam[4][1]=4
3c	**Demonstrate with example of upper(), lower(), isupper() and islower() string methods?**
	 **a) upper()** ➢ The upper() method returns a new string after converting all the letters in the original string into uppercase. ➢ For example, ‘Hello’.upper() returns ‘HELLO’ **b) lower()** ➢ The lower() method returns a new string after converting all the letters in the original string into lowercase. ➢ For example ‘Hello’.lower() returns ‘hello’ **c) isupper()** ➢ The isupper() method returns Boolean True if all the letters of the string are upper case. Otherwise, it will return Boolean False. ➢ For example, ‘HELLO’.isupper() returns True ➢ ‘Hello’.isupper() returns False **d) islower()** ➢ The islower() method returns Boolean True if all the letters of the string are lower case. Otherwise, it will return Boolean False. ➢ For example, ‘hello’.islower() returns True ➢ ‘Hello’.islower() returns False
4a	**What is a dictionary? Compare dictionaries with List? Write a program to count the number of occurrences of characters in a string**
	➢ Like a list, dictionary is also a collection of many values. ➢ But in a list, indexes are always integer values. ➢ In a dictionary, index can be any immutable type of data such as integer and string. ➢ The indexes for dictionaries are called keys. ➢ Another difference between list and dictionary is that the list is ordered, but the dictionary is unordered. ➢ For example, x={‘Name’:’Alice’,’Age’:10,’Gender’:’Female’}

x[‘Name’] returns ‘Alice’

x[‘Age’] returns 10

- We can append a new value to the dictionary as follows.

  x[‘Fav_Color’]=’Blue’

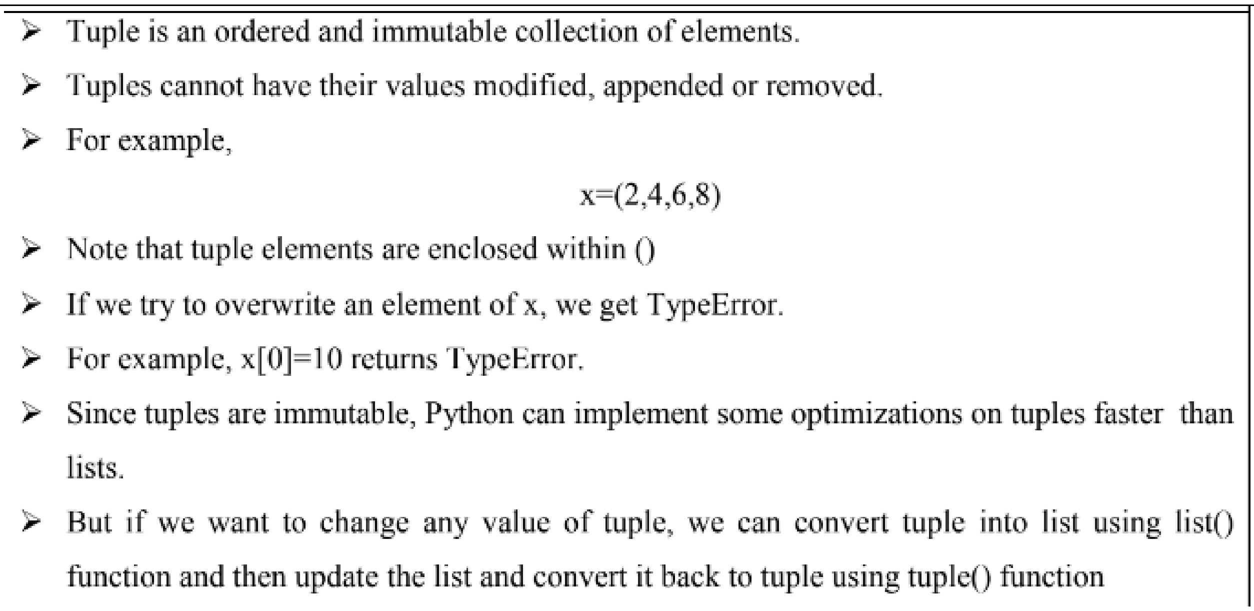
- Now the updated dictionary will be

  x={‘Name’:’Alice’,’Age’:10,’Gender’:’Female’,’Fav_Color’:’Blue’}

```
#write a python program to check number of occurrence of each characters in string
s="IamIndian"
s=s.lower()
s=list(s)
i=0
d={}
s1=sorted((set(s)))
for i in s1:
    d[i]=s.count(i)
print(d)
```

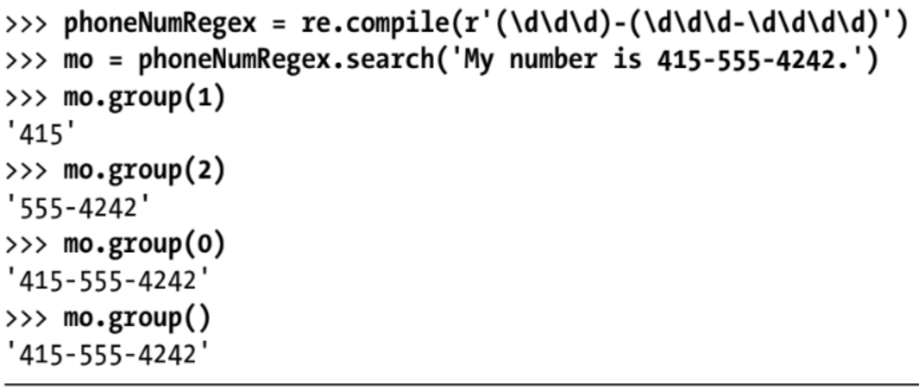
4b **Write a program to implement search and replace multiple occurrences of a given substring in the main string in a list.**

```
#code for replacing all occurrences of substring s1 with new string s2

test_str="geeksforgeeks"
s1="geeks"
s2="abcd"

#string split by substring
s=test_str.split(s1)
new_str=""

for i in s:
    if(i==""):
        new_str+=s2
    else:
        new_str+=i

#printing the replaced string
print(new_str)
```

4c	**Define Tuple data type, explain in converting types with the list() and tuple() Functions.**
	 ➢ Tuple is an ordered and immutable collection of elements. ➢ Tuples cannot have their values modified, appended or removed. ➢ For example, x=(2,4,6,8) ➢ Note that tuple elements are enclosed within () ➢ If we try to overwrite an element of x, we get TypeError. ➢ For example, x[0]=10 returns TypeError. ➢ Since tuples are immutable, Python can implement some optimizations on tuples faster than lists. ➢ But if we want to change any value of tuple, we can convert tuple into list using list() function and then update the list and convert it back to tuple using tuple() function The list() function creates a list object. Example: x = list(('apple', 'banana', 'cherry')) print(x) Output: ['apple', banana', 'cherry'] The **Python tuple() function** is a built-in function in Python that can be used to create a tuple. Example: l = [1,2,3] print(tuple(l)) Output: (1, 2, 3)
5a	**With example, explain the following Pattern Matching with RegularExpressions.** **(i) Grouping with Parentheses** **(ii) Matching Multiple Groups with the Pipe**

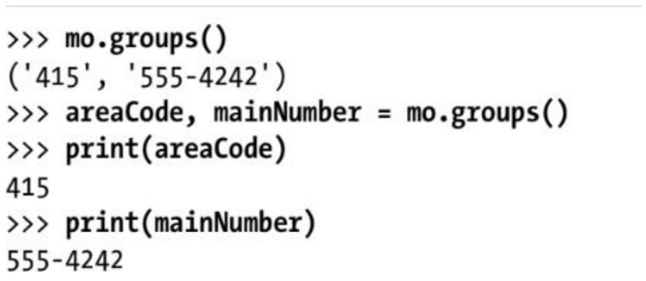
Grouping with Parentheses

- **improves power matching capabilities.**
- Adding parentheses will create groups in the regex: (\d\d\d)-(\d\d\d-\d\d\d\d).
- Then we can use the group() match object method to grab the matching text from just one group.
- The first set of parentheses in a regex string will be group 1.
- The second set will be group 2.
- By passing the integer 1 or 2 to the group() match object method, we can grab different parts of the matched text.
- Passing ( ) or nothing to the group() method will return the entire matched text.



```
>>> phoneNumRegex = re.compile(r'(\d\d\d)-(\d\d\d-\d\d\d\d)')
>>> mo = phoneNumRegex.search('My number is 415-555-4242.')
>>> mo.group(1)
'415'
>>> mo.group(2)
'555-4242'
>>> mo.group(0)
'415-555-4242'
>>> mo.group()
'415-555-4242'
```

- If we would like to retrieve all the groups at once, use the groups() method



```
>>> mo.groups()
('415', '555-4242')
>>> areaCode, mainNumber = mo.groups()
>>> print(areaCode)
415
>>> print(mainNumber)
555-4242
```

(ii) Matching Multiple Groups with the Pipe

- The | character is called a pipe.
- We can use it to match one of many expressions.
- For example, the regular expression ‘Raveesh|Mahesh’ will match either 'Raveesh' or ‘Mahesh'.
- When both ‘Raveesh’ and ‘Mahesh’ occur in the searched string, the first occurrence of matching text will be returned as the Match object.

	import re
	search_pattern = re.compile (r'Raveesh|Mahesh')
	match_object= search_pattern.search(' My teachers are Raveesh and Mahesh')
	match_object.group()
	(match_object.group() returns 'Raveesh' as it is the first match.)
	match_object= search_pattern.findall(' My teachers are Raveesh and Mahesh')
	(findall function will return a list containing 'Raveesh' and 'Mahesh')

5b	**What are the steps involved in file handling? Also, explain the reading and writing process with suitable examples in Python.**
	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)

#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

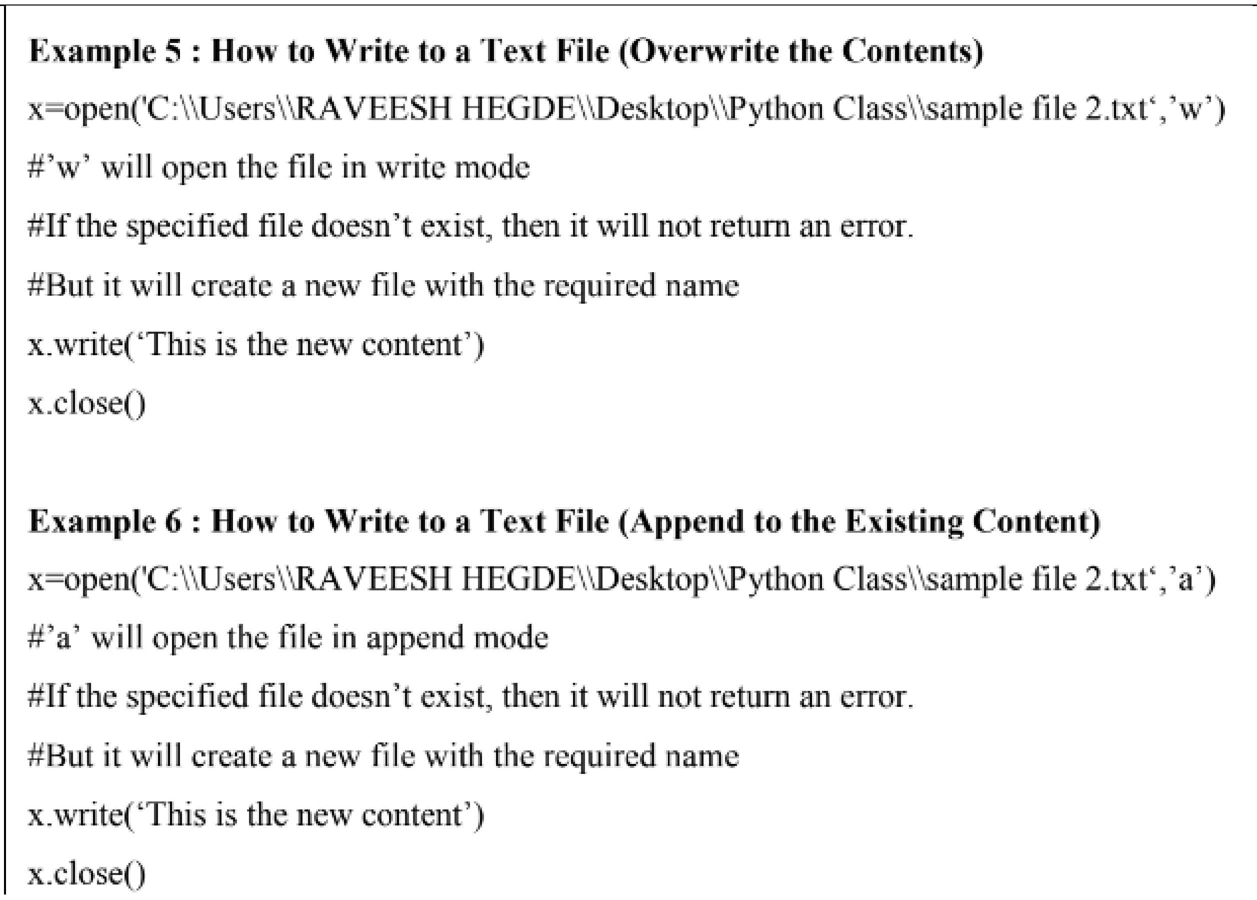
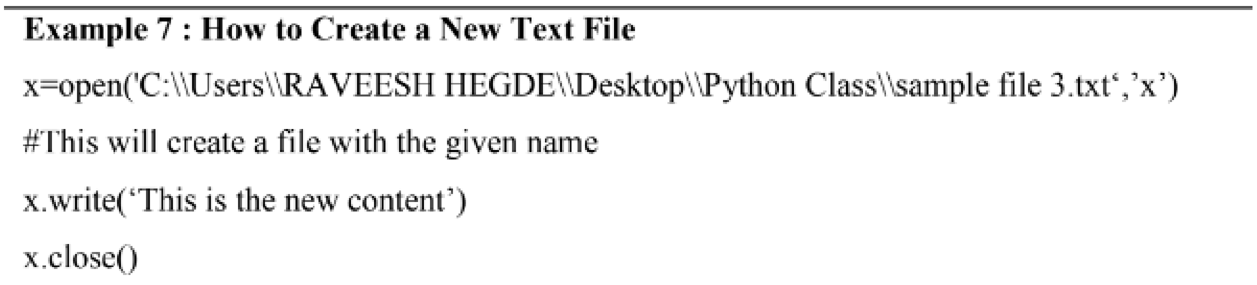
x=open('C:\\Users\\RAVEESH HEGDE\\Desktop\\Python Class\\sample file 1.txt')

#By default, file will be opened in read mode

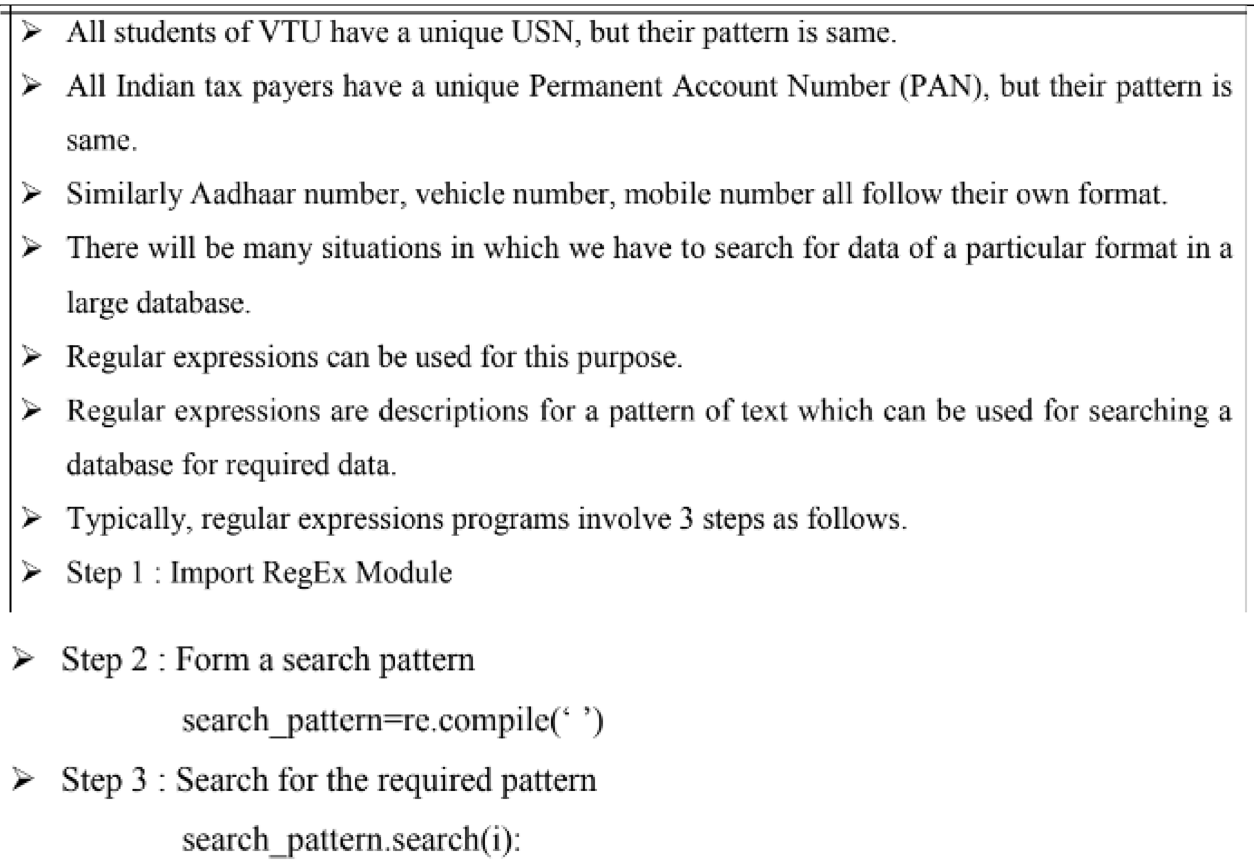
x.readlines(5)

#This will print the 5th 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()

6a	**Explain the basic steps for creating and finding regular expression objects with Python**

	 ➢ All students of VTU have a unique USN, but their pattern is same. ➢ All Indian tax payers have a unique Permanent Account Number (PAN), but their pattern is same. ➢ Similarly Aadhaar number, vehicle number, mobile number all follow their own format. ➢ There will be many situations in which we have to search for data of a particular format in a large database. ➢ Regular expressions can be used for this purpose. ➢ Regular expressions are descriptions for a pattern of text which can be used for searching a database for required data. ➢ Typically, regular expressions programs involve 3 steps as follows. ➢ Step 1 : Import RegEx Module ➢ Step 2 : Form a search pattern search_pattern=re.compile(' ') ➢ Step 3 : Search for the required pattern search_pattern.search(i):
	➢
6b	**Write a Python program to accept USN and marks obtained. Find maximum, minimum and USN students who scored 100-85,85-75,75-60 marks separately.**

```
import re
n= int(input("enter number of students: "))
marks=[]
USN=[]
Good=[]
Medium=[]
Average=[]
poor=[]
i=0
while i<n:
    print("Enter USN OF ", i+1, " student: ")
    item=input()
    USNRegex = re.compile(r'\d\w\w\d\d\w\w\d\d\d')
    if USNRegex.search(item)==None:
        print("Not correct USN")
        continue
    else:
        USN.append(item)
        marks.append(int(input("enter marks :")))
        i+=1
```

```
dict={}
for i in range(n):
    if marks[i] >85:
        Good.append(USN[i])
    elif marks[i]<=85 and marks[i]>=75:
        Medium.append(USN[i])
    elif marks[i]<75 and marks[i]>=60:
        Average.append(USN[i])
    else:
        poor.append(USN[i])
print(" Maximum mark is ", max(marks))
print(" Minimum mark is ", min(marks))
print(" USNs  who got Marks between 100-85 mark is ", * Good)
print(" USNs  who got Marks between 85-75 mark is ", * Medium)
print(" USNs who got  Marks less than 60 mark is ", * poor)
```

Output:

```
enter number of students: 4
Enter USN OF  1  student:
1cr21ee213
enter marks :32
Enter USN OF  2  student:
xx
Not correct USN
Enter USN OF  2  student:
1cr21ec255
enter marks :83
Enter USN OF  3  student:
1cr23089
Not correct USN
Enter USN OF  3  student:
1cr23ec214
enter marks :74
Enter USN OF  4  student:
1cr21ec034
enter marks :75
 Maximum mark is  83
 Minimum mark is  32
 USNs  who got Marks between 100-85 mark is
 USNs  who got Marks between 85-75 mark is  1cr21ec255 1cr21ec034
 USNs who got  Marks less than 60 mark is  1cr21ee213
```

6c	Explain the purpose of the following special characters used in optimal matching regular expression:?,*,+ and {}. Illustrate with example.

?	Zero or One Occurrence
*	Zero or More Occurrence
+	One or More Occurrence
{2}	Exactly Two Occurrences
{2,3}	Two or Three Occurrences
{2,10}	Two to Ten Occurrences Including Two and Ten

- The '?' symbol can be used to match a regex pattern optionally.
- The '?' symbol will match either zero or one occurrence of a pattern in a string.
- For example, consider the following code.

```
search_pattern= re.compile('Ravi(sh)?')
match_object =search_pattern.search('My teacher is Ravi')
match_object.group()
(Output : 'Ravi')
match_object = search_pattern.search('My teacher is Ravish')
 match_object.group()
(Output : 'Ravish')
```

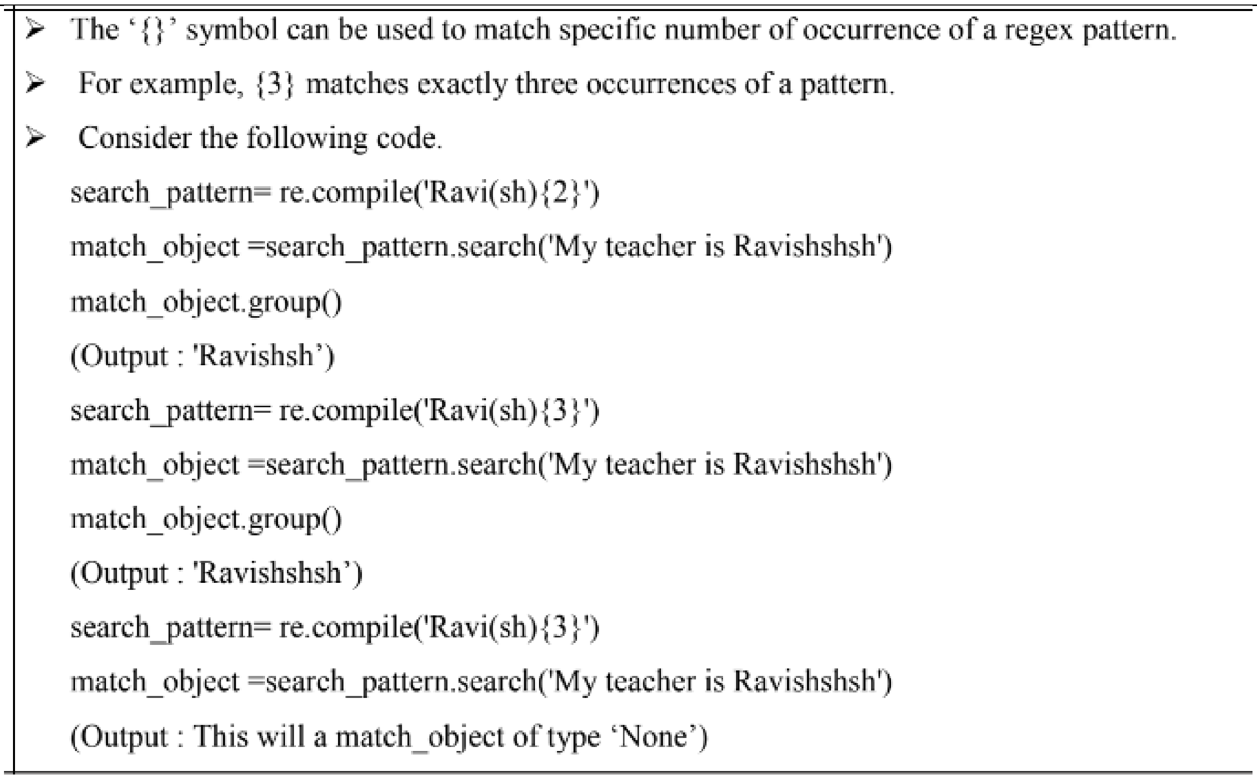
- The ‘*’ symbol can be used to match zero or more occurrence of a regex pattern.
- The pattern can be completely absent or repeated over and over again.
- For example, consider the following code.

```
search_pattern= re.compile('Ravi(sh)*')
match_object =search_pattern.search('My teacher is Ravi')
match_object.group()
(Output : 'Ravi’)
match_object = search_pattern.search('My teacher is Ravish')
 match_object.group()
(Output : 'Ravish‘)
match_object = search_pattern.search('My teacher is Ravishshshshsh')
 match_object.group()
(Output : 'Ravishshshshsh’)
```

- The ‘+’ symbol can be used to match one or more occurrence of a regex pattern.
- The pattern must be present at least once to return a match object.
- For example, consider the following code.

```
search_pattern= re.compile('Ravi(sh)+')
match_object =search_pattern.search('My teacher is Ravi')
(Output : This will return a match_object of type ‘None’)
```

```
match_object = search_pattern.search('My teacher is Ravish')
match_object.group()
(Output : 'Ravish‘)
match_object = search_pattern.search('My teacher is Ravishshshshsh')
match_object.group()
(Output : 'Ravishshshshsh’)
```



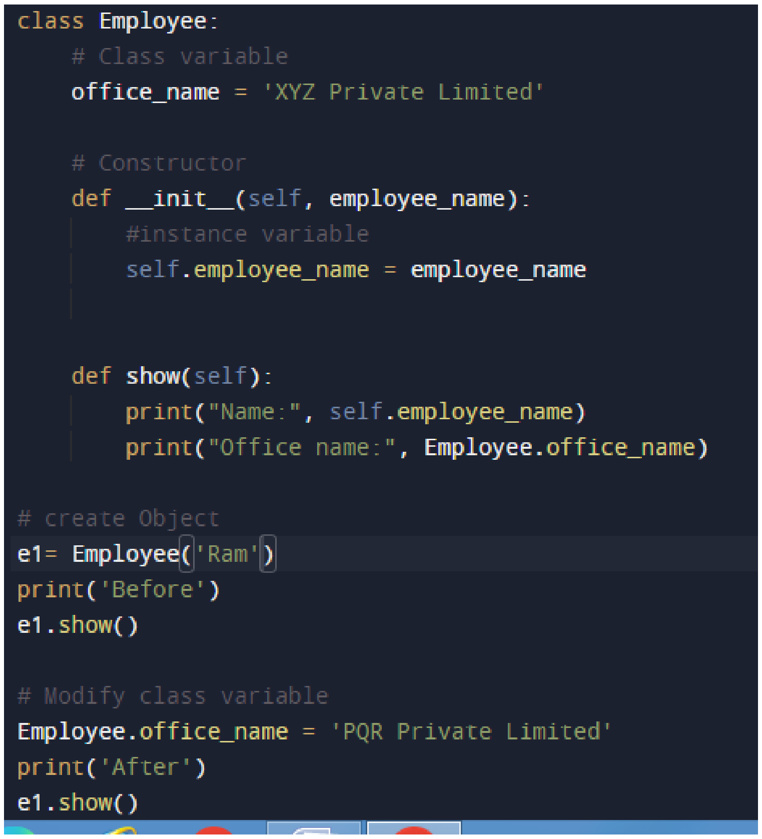
- The '{}' symbol can be used to match specific number of occurrence of a regex pattern.
- For example, {3} matches exactly three occurrences of a pattern.
- Consider the following code.

```
search_pattern= re.compile('Ravi(sh){2}')
match_object =search_pattern.search('My teacher is Ravishshsh')
match_object.group()
(Output : 'Ravishsh')
search_pattern= re.compile('Ravi(sh){3}')
match_object =search_pattern.search('My teacher is Ravishshsh')
match_object.group()
(Output : 'Ravishshsh')
search_pattern= re.compile('Ravi(sh){3}')
match_object =search_pattern.search('My teacher is Ravishshsh')
(Output : This will a match_object of type 'None')
```

7a **Differentiate between class variables and instance variables with suitable examples**

Class variables are defined within the class but outside of any class methods.	Instance variables are defined within class methods, typically the constructor.
Changes made to the class variable affect all instances.	Changes made to the instance variable does not affect all instances.
Class variables can be initialized either inside the class definition or outside the class definition.	Instance variables are typically initialized in the constructor of the class.
Class variables are accessed using the class name, followed by the variable name.	Instance variables are accessed using the instance name, followed by the variable name.
Class variables are useful for storing data that is shared among all instances of a class, such as constants or default	Instance variables are used to store data that is unique to each instance of a class, such as

values.	object properties.



```
class Employee:
    # Class variable
    office_name = 'XYZ Private Limited'

    # Constructor
    def __init__(self, employee_name):
        #instance variable
        self.employee_name = employee_name


    def show(self):
        print("Name:", self.employee_name)
        print("Office name:", Employee.office_name)

# create Object
e1= Employee('Ram')
print('Before')
e1.show()

# Modify class variable
Employee.office_name = 'PQR Private Limited'
print('After')
e1.show()
```

Output:

Before

Name: Ram

Office name: XYZ Private Limited

After

Name: Ram

Office name: PQR Private Limited

7b	**Write program to create a class named Rectangle with the help of a corner point, width and height. Write the following functions and demonstrate their working.** **i) To find and display the center of the rectangle** **ii) To display point as an ordered pair** **iii) To resize the rectangle** **iv) To find area and perimeter of a rectangle**

```
class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y
class Rectangle:
    def __init__(self,cornerpt,width,height):
        self.cornerpt=cornerpt
        self.width=width
        self.height=height
    def findcenter(self):
        center=Point(0,0)
        center.x=self.cornerpt.x+self.width/2
        center.y=self.cornerpt.y+self.height/2
        print("Center is", center.x, center.y)
        return center

    def areaperimeter(self):
        area=self.width*self.height
        perimeter=2*(self.width + self.height)
        print("Area of Rectangle is", area )
        print("Perimeter of Rectangle is", perimeter )
    def resize(self, widthn,heightn):
        self.width=widthn
        self.height=heightn
    def cornerpts(self):
```

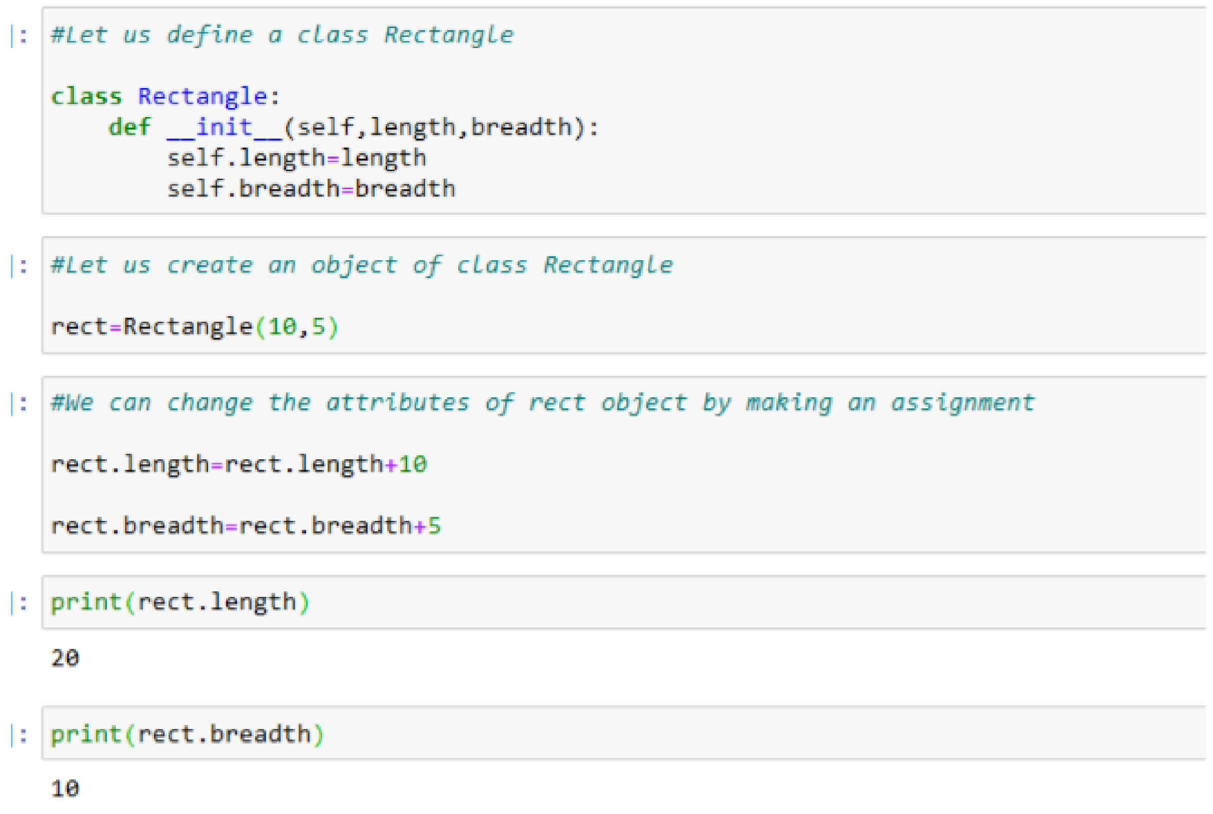
```
    A=(self.cornerpt.x,cornerpt.y)
    B=(self.cornerpt.x +self.width,cornerpt.y)
    C=(self.cornerpt.x ,cornerpt.y+self.height)
    D=(self.cornerpt.x +self.width,cornerpt.y+ self.height)
    print(A, B, C, D)

cornerpt=Point(0.0,0.0)
rect=Rectangle(cornerpt,100.0,200.0)
rect.cornerpts()
rect.findcenter()
rect.areaperimeter()
rect.resize(50.0,40.0)
rect.areaperimeter()
```

Output:

```
(0.0, 0.0) (100.0, 0.0) (0.0, 200.0) (100.0, 200.0)
Center is 50.0 100.0
Area of Rectangle is 20000.0
Perimeter of Rectangle is 600.0
Area of Rectangle is 2000.0
Perimeter of Rectangle is 180.0
```

7c	**Justify the statement" Objects are mutable" with suitable examples.**

- We can change the state of an object by making an assignment to its attributes.
- For example, let us create a class rectangle with length and breadth as attributes.
- Let us create an object of type class Rectangle and then change its length and breadth.



```
#Let us define a class Rectangle

class Rectangle:
    def __init__(self,length,breadth):
        self.length=length
        self.breadth=breadth
```

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

rect=Rectangle(10,5)
```

```
#We can change the attributes of rect object by making an assignment

rect.length=rect.length+10

rect.breadth=rect.breadth+5
```

```
print(rect.length)
```

```
20
```

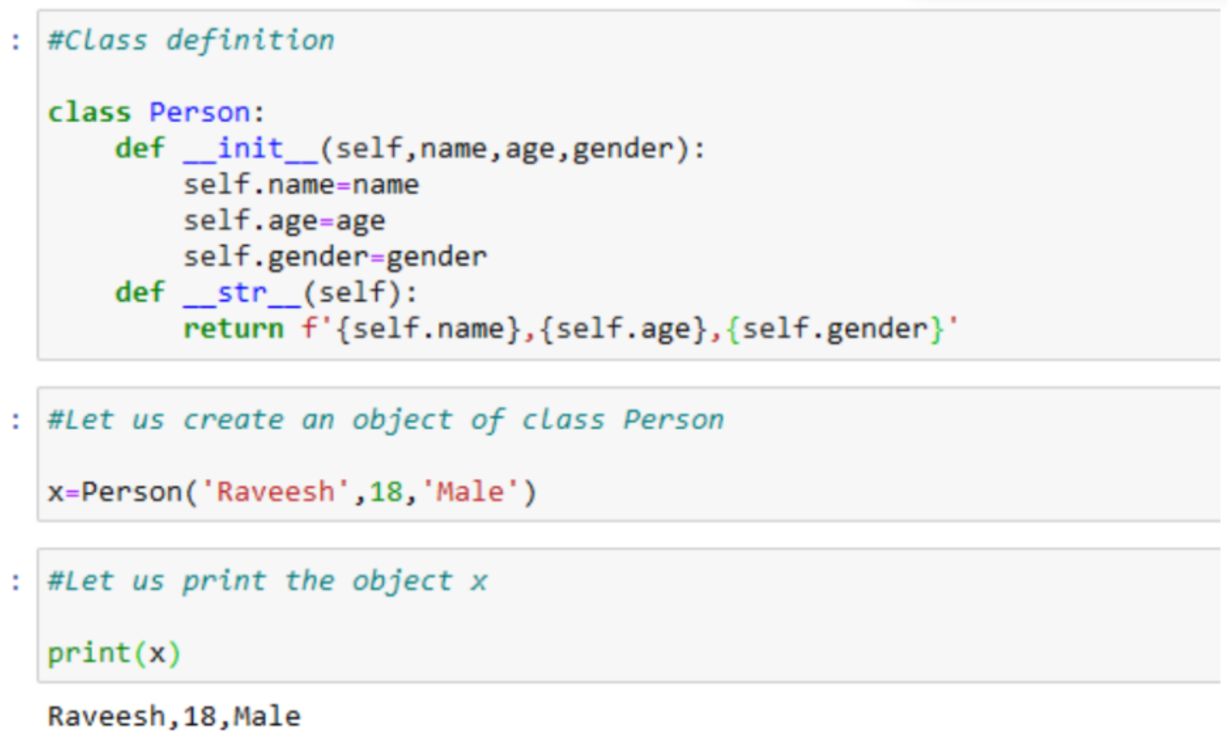
```
print(rect.breadth)
```

```
10
```

8a **Explain – init( ) and – str( ) methods with an example.**

All classes have a function called _ _init_ _( ), which is executed by default when the object iscreated.

- 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}'
```

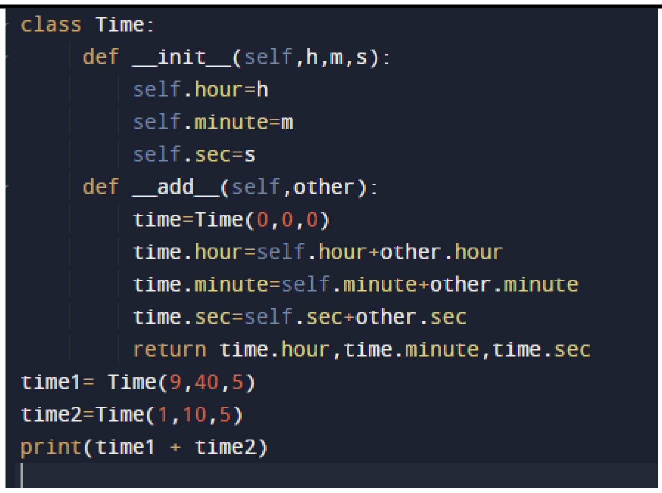
```
: #Let us create an object of class Person

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

```
: #Let us print the object x

  print(x)
```

```
Raveesh,18,Male
```

8b **Explain operator overloading and polymorphism with examples.**

- ➢ operator overloading.
- ➢ Changing the behavior of an operator so that it works with programmer defined types is called operator overloading.
- ➢ For every operator in Python there is a corresponding special method, like _ _add_ _

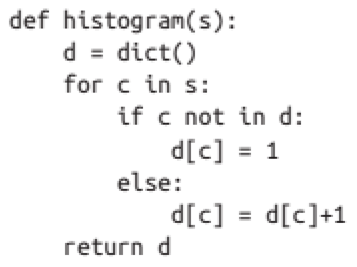


```
class Time:
    def __init__(self,h,m,s):
        self.hour=h
        self.minute=m
        self.sec=s
    def __add__(self,other):
        time=Time(0,0,0)
        time.hour=self.hour+other.hour
        time.minute=self.minute+other.minute
        time.sec=self.sec+other.sec
        return time.hour,time.minute,time.sec
time1= Time(9,40,5)
time2=Time(1,10,5)
print(time1 + time2)
```

- ➢ Output:

- (10, 50, 10)

- Polymorphism
- If x and y are integers, then x+y performs arithmetic addition of x and y.
- If x and y are strings, the x+y performs string concatenation.
- Same operator or function name can be used to perform different operations.
- This property is known as polymorphism.
- Functions that work with several types are called polymorphic.
- Many of the functions we wrote for strings also work for other sequence types.
- Forexample,



```
def histogram(s):
    d = dict()
    for c in s:
        if c not in d:
            d[c] = 1
        else:
            d[c] = d[c]+1
    return d
```

- This function also works for lists, tuples, and even dictionaries,

9a **Write a Python program that makes a socket connection to a web server and follows the rules of the HTTP protocol to request a document anddisplay what the server sends back.**

```
import socket
mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'.encode()
mysock.send(cmd)
while True:
data = mysock.recv(512)
if len(data) < 1:
break
print(data.decode(),end='')
mysock.close()
```

- First the program makes a connection to port 80 on the server www.py4e.com.

- program is playing the role of the “web browser”,
- the HTTP protocol says we must send the GET command followed by a blank line. \r\n signifiesan EOL (end of line),
- so \r\n\r\n signifies nothing between two EOL sequences. That is the equivalent of a blank line.
- Once we send that blank line, we write a loop that receives data in 512-character chunks from the socket and prints the data out until there is no more data to read (i.e., the recv() returns an empty string).

9b **Illustrate with a python program how to retrieve web pages with urllib.**

- Using urllib, we can treat a web page much like a file.
- We simply indicate which web page we would like to retrieve and urllib handles all of the HTTP protocol and header details.
- Once the web page has been opened with urllib.urlopen, we can treat it like a file and read through it using a for loop.
- When the program runs, we only see the output of the contents of the file.
- The headers are still sent, but the urllib code removes the headers and only returns the data to us.

```
import urllib.request
fhand =urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
for line in fhand:
print(line.decode().strip())
```

10a What is Service-Oriented Programming(SOA)? List out the advantages of SOA.

- When we begin to build our programs where the functionality of our program includes access to services provided by other programs, we call the approach a Service-oriented architecture (SOA).
- A SOA approach is one where our overall application makes use of the services of other applications.
- A non-SOA approach is where the application is a single standalone application which contains all of the code necessary to implement the application.

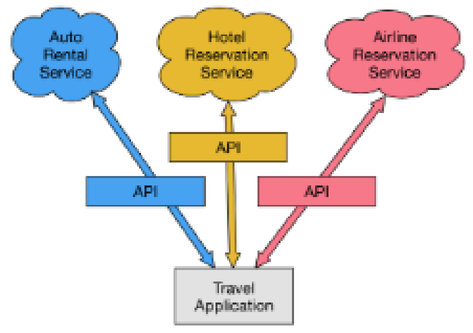


Figure 13.2: Service-oriented architecture

Advantages Of SoA

- Always maintain only one copy of data (this is particularly important for things like hotel reservations where we do not want to over-commit)
- The owners of the data can set the rules about the use of their data.
- With these advantages, an SOA system must be carefully designed to have good performance and meet the user's needs.
- When an application makes a set of services in its API available over the web, we call these web services.

10b **Discuss various keys used in the database model.**

Logical Key

- A logical key is a key that the real world users might use to look up a row.
- For example, if we built a table to store the details of twitter users, the name field is a logica key.
- It is the screen name for the user and we look up a user's row in the program using the name

field.

- Since the logical key is how we look up a row from the outside world, we cannot allow

multiple rows with the same value in the table.

So we usually add UNIQUE constraint to a logical key.

Primary Key

- A primary key is usually a number that is assigned automatically by the database.
- It generally has no meaning outside the program and is only used to link rows from dfferent

tables together.

- When we want to look up a row in a table, usually searching for the row using the primary key is the fastest way to find the row.
- Since primary keys are integer numbers, they take up very little storage and can be compared or sorted very quickly.
- In twitter user's data model, the id field is an example of a primary key.

Foreign key

- When a primary key is used in another table, it is called a "foreign key".
- The connection between the primary and foreign key then creates a "relationship" between records contained across multiple tables.
- A foreign key is usually a number that points to the primary key of an associated row in adifferent table.

10c Write the four SQL commands needed to create and maintain data.

Following are the 4 main operations in a database management system.

1) Create

2) Read

3) Update

4) Delete

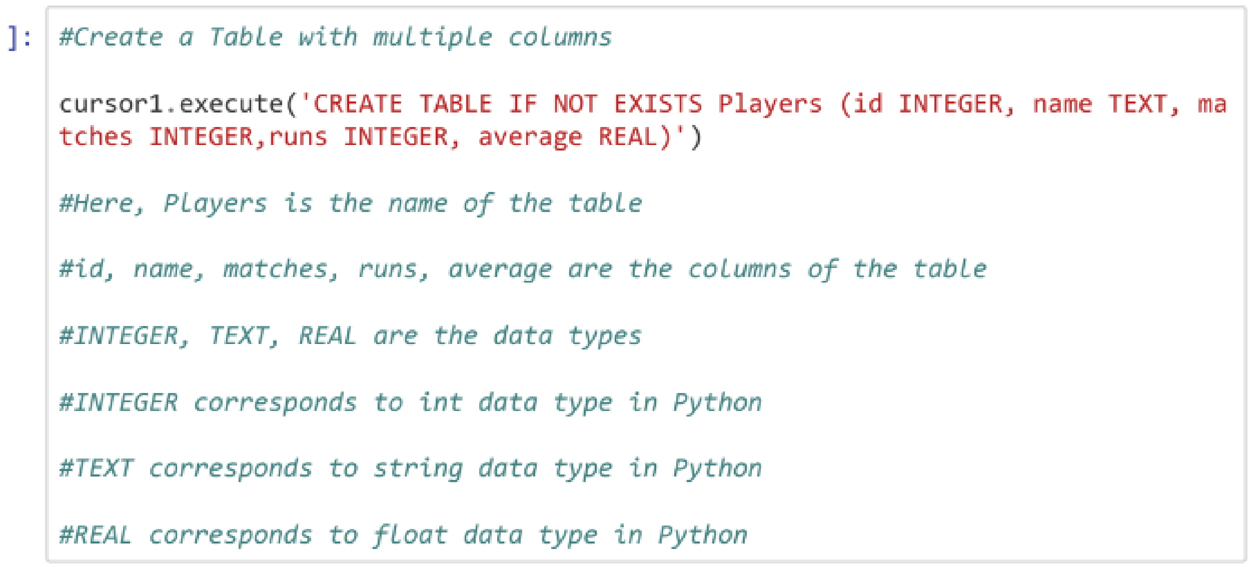
- These operations are popularly known as CRUD operations.
- Following are the main SQL commands used in CRUD operations.

1) INSERT

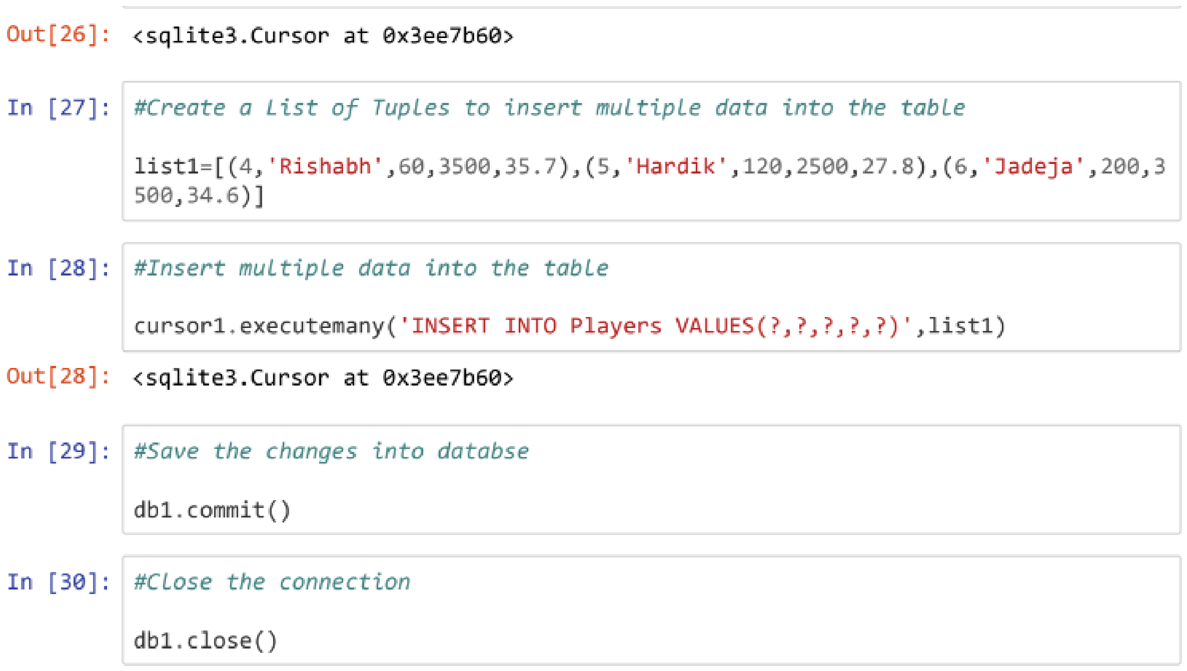
2) SELECT

3) UPDATE

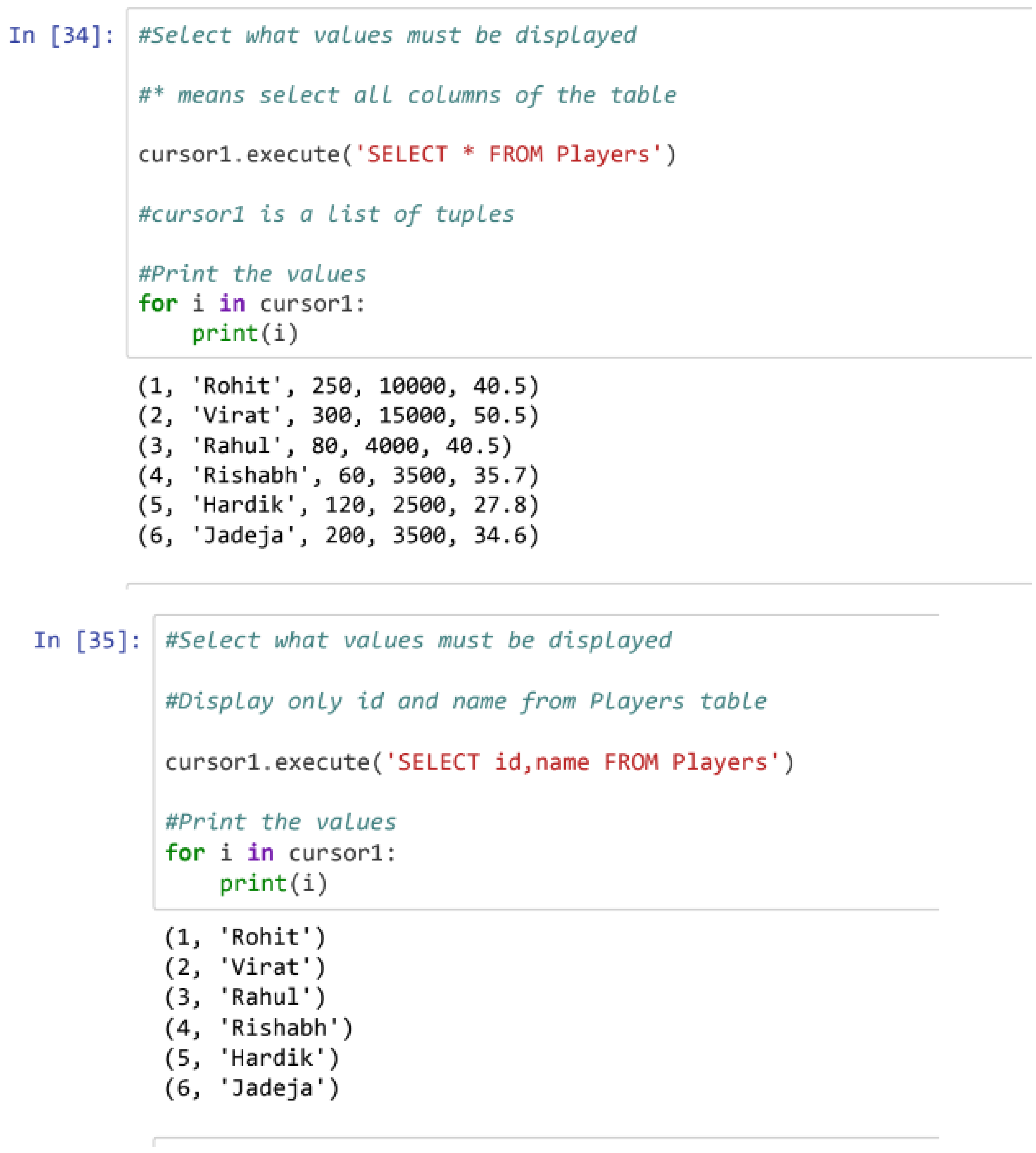
4) DELETE



```
]: #Create a Table with multiple columns

   cursor1.execute('CREATE TABLE IF NOT EXISTS Players (id INTEGER, name TEXT, ma
   tches INTEGER,runs INTEGER, average REAL)')

   #Here, Players is the name of the table

   #id, name, matches, runs, average are the columns of the table

   #INTEGER, TEXT, REAL are the data types

   #INTEGER corresponds to int data type in Python

   #TEXT corresponds to string data type in Python

   #REAL corresponds to float data type in Python
```

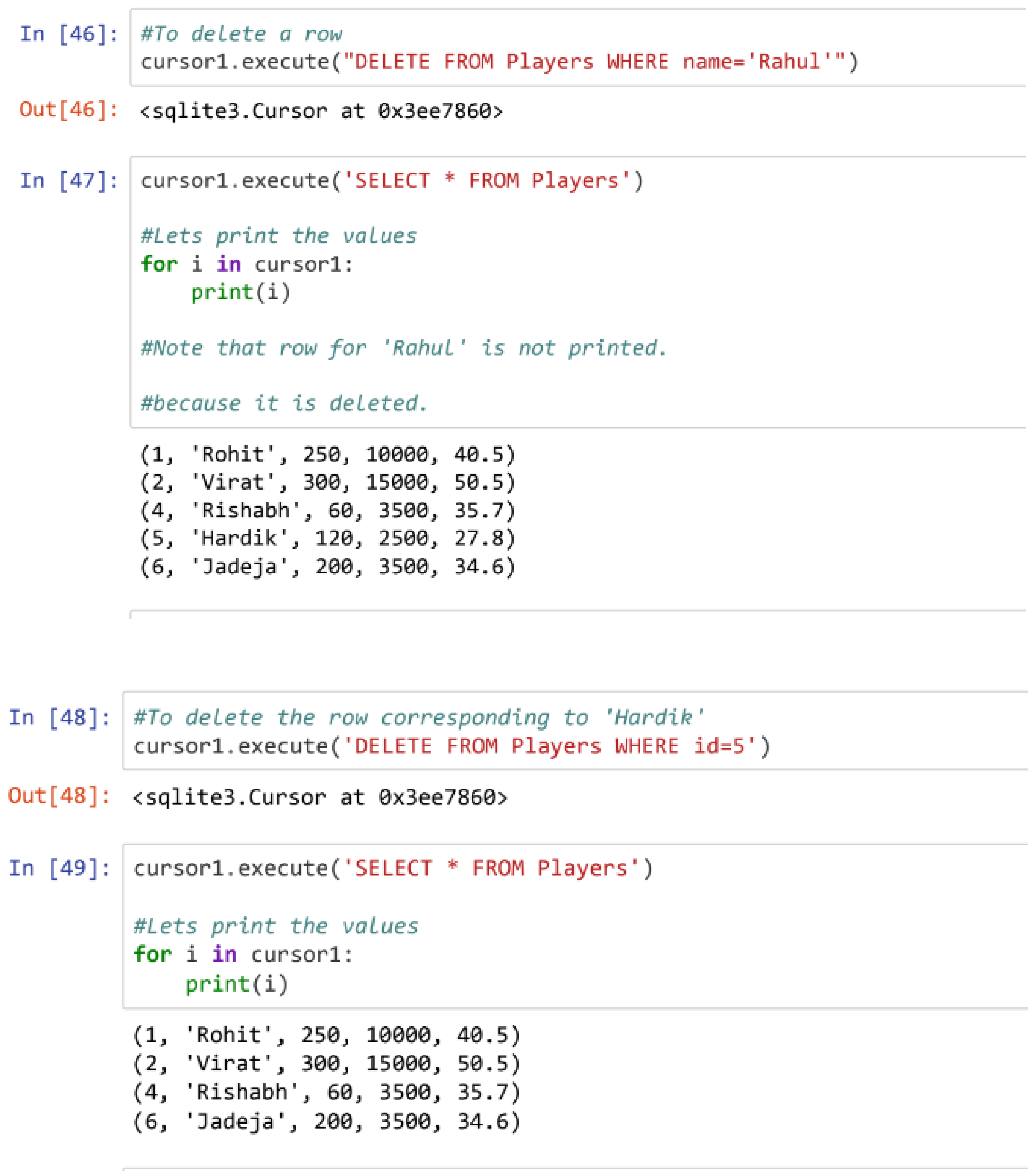


```
Out[26]: <sqlite3.Cursor at 0x3ee7b60>

In [27]: #Create a List of Tuples to insert multiple data into the table

         list1=[(4,'Rishabh',60,3500,35.7),(5,'Hardik',120,2500,27.8),(6,'Jadeja',200,3
         500,34.6)]

In [28]: #Insert multiple data into the table

         cursor1.executemany('INSERT INTO Players VALUES(?,?,?,?,?)',list1)

Out[28]: <sqlite3.Cursor at 0x3ee7b60>

In [29]: #Save the changes into databse

         db1.commit()

In [30]: #Close the connection

         db1.close()
```

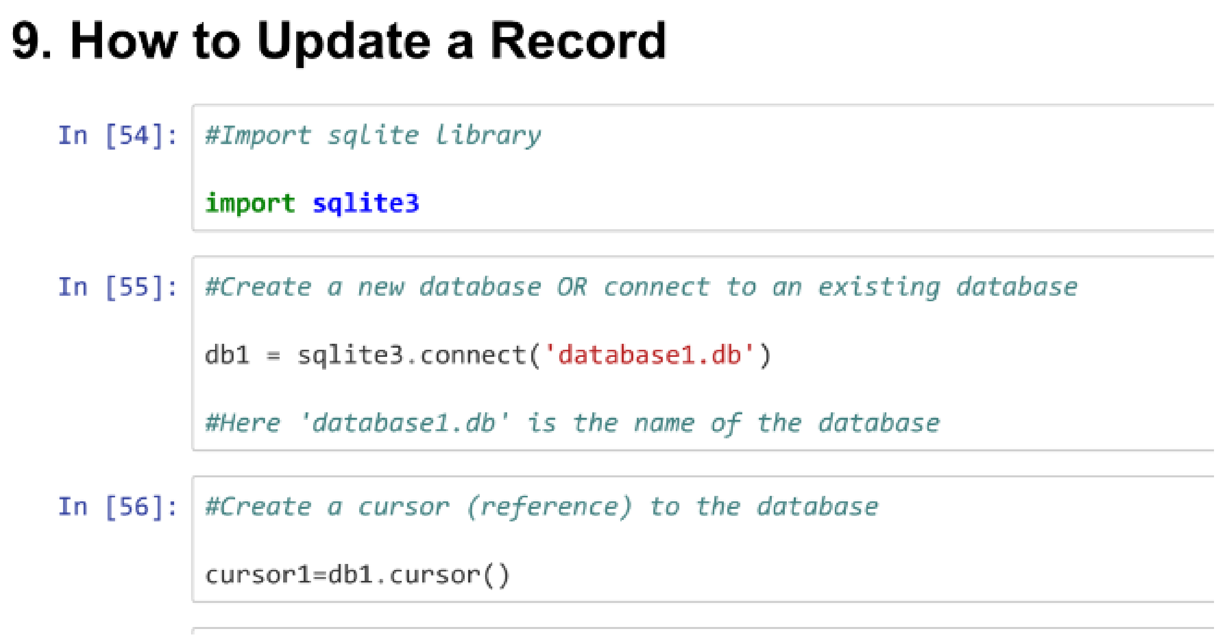


```
In [34]: #Select what values must be displayed

         #* means select all columns of the table

         cursor1.execute('SELECT * FROM Players')

         #cursor1 is a list of tuples

         #Print the values
         for i in cursor1:
             print(i)

         (1, 'Rohit', 250, 10000, 40.5)
         (2, 'Virat', 300, 15000, 50.5)
         (3, 'Rahul', 80, 4000, 40.5)
         (4, 'Rishabh', 60, 3500, 35.7)
         (5, 'Hardik', 120, 2500, 27.8)
         (6, 'Jadeja', 200, 3500, 34.6)
```

```
In [35]: #Select what values must be displayed

         #Display only id and name from Players table

         cursor1.execute('SELECT id,name FROM Players')

         #Print the values
         for i in cursor1:
             print(i)

         (1, 'Rohit')
         (2, 'Virat')
         (3, 'Rahul')
         (4, 'Rishabh')
         (5, 'Hardik')
         (6, 'Jadeja')
```



```
In [46]: #To delete a row
         cursor1.execute("DELETE FROM Players WHERE name='Rahul'")

Out[46]: <sqlite3.Cursor at 0x3ee7860>

In [47]: cursor1.execute('SELECT * FROM Players')

         #Lets print the values
         for i in cursor1:
             print(i)

         #Note that row for 'Rahul' is not printed.

         #because it is deleted.

         (1, 'Rohit', 250, 10000, 40.5)
         (2, 'Virat', 300, 15000, 50.5)
         (4, 'Rishabh', 60, 3500, 35.7)
         (5, 'Hardik', 120, 2500, 27.8)
         (6, 'Jadeja', 200, 3500, 34.6)

In [48]: #To delete the row corresponding to 'Hardik'
         cursor1.execute('DELETE FROM Players WHERE id=5')

Out[48]: <sqlite3.Cursor at 0x3ee7860>

In [49]: cursor1.execute('SELECT * FROM Players')

         #Lets print the values
         for i in cursor1:
             print(i)

         (1, 'Rohit', 250, 10000, 40.5)
         (2, 'Virat', 300, 15000, 50.5)
         (4, 'Rishabh', 60, 3500, 35.7)
         (6, 'Jadeja', 200, 3500, 34.6)
```

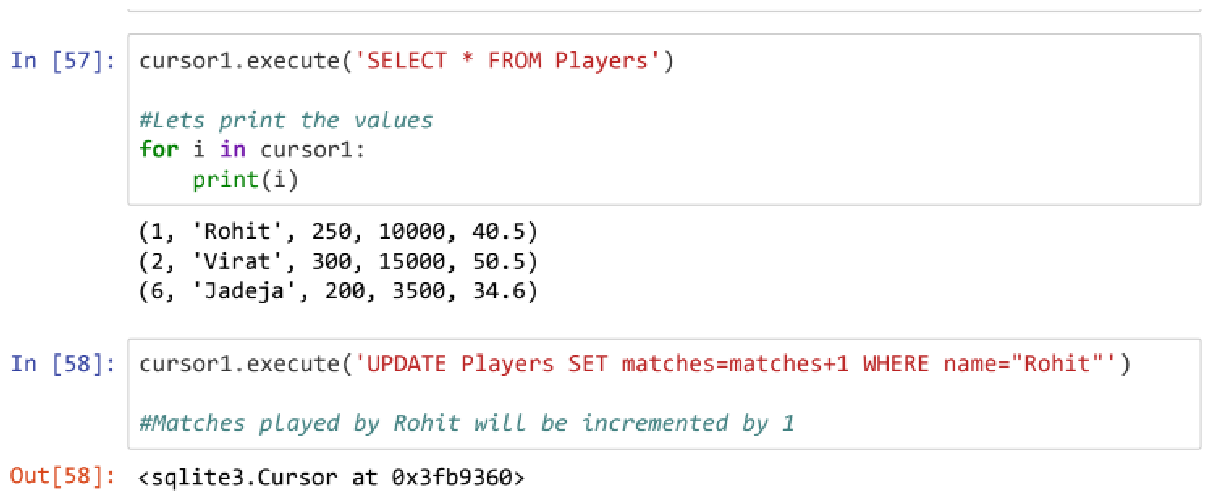


9. How to Update a Record

```
In [54]: #Import sqlite library

         import sqlite3
```

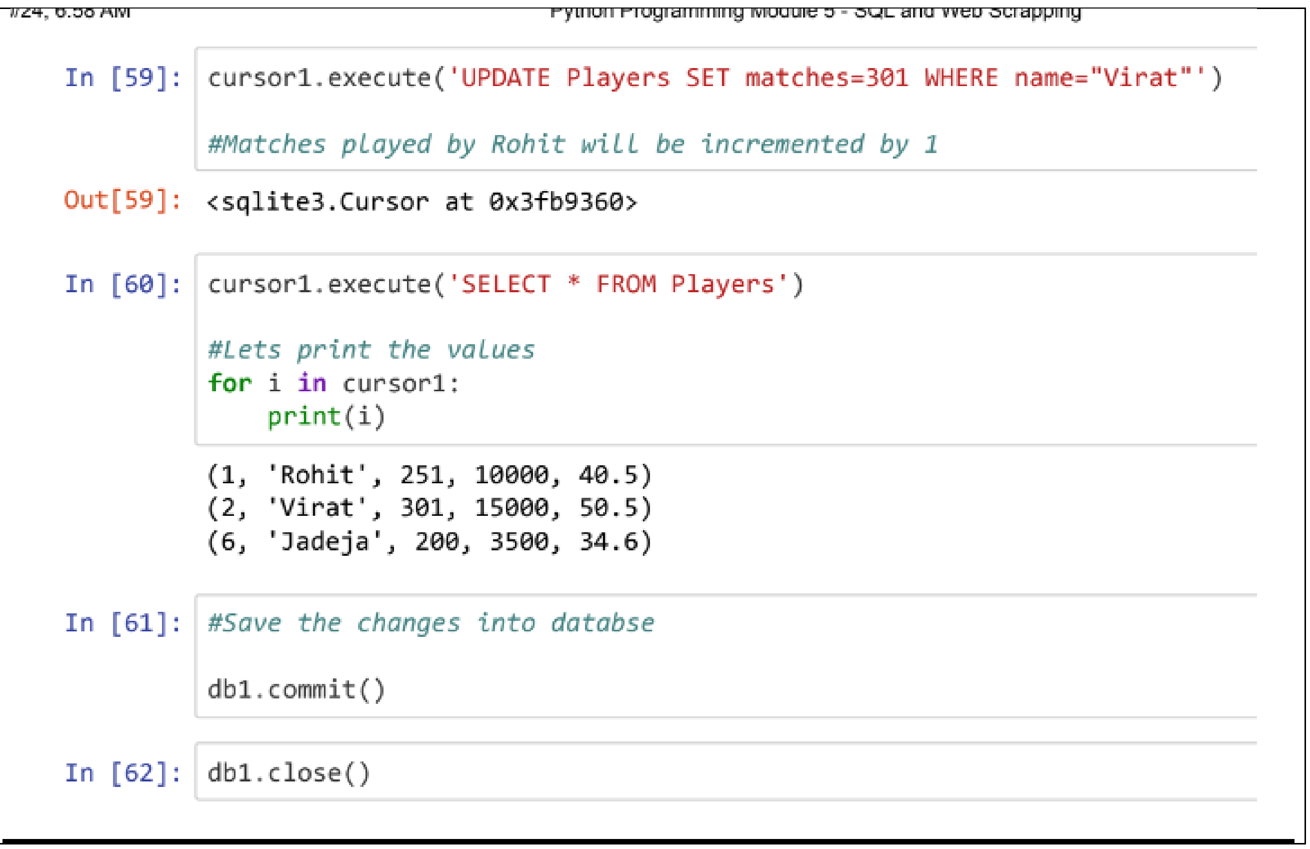
```
In [55]: #Create a new database OR connect to an existing database

         db1 = sqlite3.connect('database1.db')

         #Here 'database1.db' is the name of the database
```

```
In [56]: #Create a cursor (reference) to the database

         cursor1=db1.cursor()
```



```
In [57]: cursor1.execute('SELECT * FROM Players')

         #Lets print the values
         for i in cursor1:
             print(i)

         (1, 'Rohit', 250, 10000, 40.5)
         (2, 'Virat', 300, 15000, 50.5)
         (6, 'Jadeja', 200, 3500, 34.6)
```

```
In [58]: cursor1.execute('UPDATE Players SET matches=matches+1 WHERE name="Rohit"')

         #Matches played by Rohit will be incremented by 1

Out[58]: <sqlite3.Cursor at 0x3fb9360>
```



```
In [59]: cursor1.execute('UPDATE Players SET matches=301 WHERE name="Virat"')

         #Matches played by Rohit will be incremented by 1

Out[59]: <sqlite3.Cursor at 0x3fb9360>

In [60]: cursor1.execute('SELECT * FROM Players')

         #Lets print the values
         for i in cursor1:
             print(i)

         (1, 'Rohit', 251, 10000, 40.5)
         (2, 'Virat', 301, 15000, 50.5)
         (6, 'Jadeja', 200, 3500, 34.6)

In [61]: #Save the changes into databse

         db1.commit()

In [62]: db1.close()
```