Name of Faculty : Prof. Shree Gowri SS

Name of Institution : KLS VDIT Haliyal

Department : Electronis and Communication Engg.

Sem : 6th semester B.E Degree Examination July/Aug 2022

Subject : Python Application and Programming
Scheme and Solution

Subject code : 18EC646

[illegible] 14.09.2022

Dr. M M Dixit

HOD

Head of the Department
Dept. of Electronic & Communication Engg.
KLS V.D.I.T., HALIYAL (U.K.)

Prof. Shree Gowri S.S - [illegible]
Subject Teacher

CBCS SCHEME

USN | 2VD19EC042

18EC646

Sixth Semester B.E. Degree Examination, July/August 2022
Python Application Programming

Time: 3 hrs. Max. Marks: 100

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

Module-1

1 a. Explain Conditional Execution , Alternative execution , Chained conditionals and Nested conditionals with examples. **(08 Marks)**

b. Explain the rules of precedence used by Python to evaluate an expression. **(04 Marks)**

c. Write a Python Program to prompt the user for hours and rate per hour for pay computation with time and a half for overtime. To give the employee 1.5 time the hourly rate for hours worked above 40 hours. **(08 Marks)**

OR

2 a. List the features of Python Programming Language (at least FIVE). **(05 Marks)**

b. What are User defined functions? How can we pass parameters in user defined functions? Explain with suitable example. **(06 Marks)**

c. Write a program to prompt for a score between 0.0 and 1.0. if the score is out of range print an error manage and exit. If the score is between 0.0 and 1.0 print a grade using the following table : **(09 Marks)**

Score	> = 0.9	> = 0.8	> = 0.7	> = 0.6	< 0.6
Grade	A	B	C	D	F

Module-2

3 a. With Syntax, explain the finite and infinite looping constructs in Python. What is the need for break and continue statements? Explain with examples. **(08 Marks)**

b. What are String Slices? Explain the Slicing Operator in Python with examples. **(05 Marks)**

c. Write a Python program to accept a file name from the user :
i) Display the number of characters in the file.
ii) Find the frequency of occurrence of the lines which started with a word 'From'. **(07 Marks)**

OR

4 a. List and explain any four built in string manipulation functions supported by Python with examples. **(06 Marks)**

b. Explain file open , file close , file read and file write concepts in Python with examples. **(08 Marks)**

c. Write a Python program to find the largest value from the given set of accepted values. **(06 Marks)**

Module-3

5 a. Lists are mutable. Justify the statement with examples. Discuss the list handling functions in Python with examples. **(08 Marks)**

b. Differentiate between List and Dictionary. **(04 Marks)**

c. Write a Python program to search lines that start with the word 'from' and a character followed by a two digit number between 00 and 99 followed : Print the number if it greater than zero. Assume any input file. **(08 Marks)**

OR

6 a. Compare and contrast tuples with tests. Explain the following operation in tuples with examples : i) Sum of two tuples ii) Slicing operations iii) Compression of two tuples iv) Assignment to variables. **(10 Marks)**

b. Write a Python program that accept a sentence and build dictionary with LETTERS , DIGITS , UPPERCASE , LOWERCASE as key values and their count in sentence as values. **(06 Marks)**

c. Explain the need of Regular expression in Python language, with an example. **(04 Marks)**

Module-4

7 a. Explain Classes and Attributes in Python language with examples. **(06 Marks)**

b. What is the difference between Method and Function? Explain the working of init method with suitable code. **(06 Marks)**

c. Write a function named move – rectangle that takes a Rectangle and two numbers named dx and dy. It should change the location of the rectangle by adding dx to the x co-ordinate of corner and adding dy to the y co-ordinate of corner. **(08 Marks)**

OR

8 a. Show using a Python code how_ str_ method is invoked when you print an object. Explain its working. **(06 Marks)**

b. Illustrate the concept of Pure function and Modifier with examples. **(06 Marks)**

c. What is Operator Overloading? Write Python code to overload "+" , "-" and "*" operator by providing the methods _ _ add_ _ , _ _ sub _ _and _ _ mul _ _. **(08 Marks)**

Module-5

9 a. Define Socket. Explain how socket connection can be established to the internet using Python code over the TCP/IP connection and the http protocol to get the web document. **(08 Marks)**

b. Compare and Contrast the Javascript Object Notation (JSON) and XML. Explain parsing of XML with example. **(06 Marks)**

c. Define Cursor. Explain Connect, Execute and Close command of databases with a snippet code. **(06 Marks)**

OR

10 a. What is Embedded SQL? Explain the importance of SQLite data base. **(04 Marks)**

b. Write a note on Google Geo coding Web service. Using Python supported libraries demonstrate with a snippet code. **(08 Marks)**

c. Write a Python code to read the file from web using urllib and retrieve the data of the file. Also compute the frequency of each word in the file. **(08 Marks)**

* * * * *

Module - 1

1a. Explain conditional execution, alternative execution, chained conditionals and nested conditionals with examples - 08 Marks

Answer: Scheme:

Conditional execution	:	explanation + example - 02M
Alternative execution	:	explanation + example - 02M
Chained conditionals	:	explanation + example - 02M
Nested conditionals	:	explanation + example - 02M
		08M

Solution:

1) conditional execution: Simplest form is if statement

Syntax:
```
if boolean expression:
    Statement
```

Ex:-
```
if x>0:
    print("x is positive")
```

2) Alternative execution: Used when there are two possibilities

Syntax:
```
if boolean_expression:
    Statement_1
else:
    Statement_2
```

Example:
```
if x > 0:
    print("x is positive")
else:
    print("x is negative")
```

3) chained conditionals: chained conditionals are used when there are more than two possibilities

Syntax:
```
if boolean_exp1:
    Statement_1
elif boolean_exp2:
    Statement_2
```

```
elif boolean-exp 3:
        Statement-3
    :
    :
 else:
    last_statement
```

Ex:-

```
if x<y:
    print ("x is less than y")
elif x>y:
    print ('x is greater than y')
else:
    print ('x and y are equal')
```

4) Nested conditionals : one condition can also be nested inside another i.e an if statement contains another if statement either in if block or else block

Syntax.

```
if boolean_exp1:
    if boolean_exp_a:
        Statement_1
    else:
        Statement_2
else:
    last_ Statement
```

1b Explain the rules of precedence used by python to evaluate an expression — 04 Marks

Scheme: PEMDAS —— 4 M

Solution:

1. Parantheses - Have highest precedence and can be used to force an expression to evaluate in the order required i.e expression in parentheses are evaluated first

2. Exponentiation :- has the next highest precedence
3. Multiplication & Division : Multiplication and Division have the same precedence
4. Addition & Subtraction : Addition and subtraction has lower precedence than multiplication & division.

1c. Write a python program to prompt the user for hours and rate per hour for pay computation with time and a half for overtime. To give the employee 1.5 time the hourly rate for hours worked above 40 hours —08 marks

Scheme: reading the i/P — 02 marks
calculating the gross pay — 05 marks
printing the result — 01 marks
08 marks

Solution :

```
Hours = float (input ("Enter working hours of employee: "))
rate  = float (input (" Enter hourly rate: "))
if Hours <= 40:
    pay = Hours * rate
    print (" pay for employee for less than or
                equal to 40 hours:  ", pay)
else:
    pay = (40 * rate + (Hours - 40) * 1.5 * rate)
    print (" pay for employee working greater
              than 40 hours: ", pay)
```

2a List the features of Python programming Language (at least FIVE) ——— 05 Marks

Scheme :- 5 features * 1 = 05 Marks

Solution :- The features of python programming language are :-

1. Easy to Use
2. free and open Source
3. powerful
4. Object oriented and procedure oriented
5. High level and Interpreted language
6. Rich in library.

2b What are user defined functions? How can we pass parameters in user defined function? Explain with suitable example ——— 06 marks

Scheme :-

function definition	— 2 marks
passing parameters	— 2 marks
Explaination with example	— 2 marks
	06 marks

Solution :-

Function :- function is a blocks of statement that contains multiple statements. and name is given.

The statements in the block needs to be repeatedly executed then instead of writting same statements several or for required number of times, it is written only once and called for required times.

function always takes "argument" as input and return "return value" as result.

Function Definition & Calling the function

Syntax :-

```
def  function_name ( parameter_1, parameter_2, .... parameter_n):
        statement (s)
```

def :- keyword

function_name is user defined & it must follow the rules of identifier.

Calling the function

Defining a function does not execute it, calling a function actually performs the actions specified in the function.

Syntax :- function_name (arg_1, arg_2, ...)

Ex :-

```
def attention ( ):
    print (" Hi ")

def greet ( ):
    print (" welcome ")

>>> attention ( )
>>> greet ( )
```

op: Hi welcome

Ex :-

```
def print_twice (spam):
    print (spam)
    print (spam)

print_twice (17)
```

O/P

```
17
17
```

Q. Write a program to prompt for a score between 0.0 and 1.0 if the score is out of range print an error manage and exit. If the score is between 0.0 and 1.0 print a grade using the following table — 09 Mark

Score	>= 0.9	>= 0.8	>= 0.7	>= 0.6	< 0.6
Grade	A	B	C	D	F

[illegible]

Scheme : reading input — 01 Marks

writing blocks of ~~excep~~ statement using ~~tryes~~ except — 06 Marks

printing the O/P — 02 Marks

08 Marks

Solution :

```
score = float(input(" enter the score: "))
try
    if score >1.0 or score <0.0:
        print(" Score is out of range")
    elif score >= 0.9:
        print(" Grade is A")
    elif score >= 0.8:
        print(" Grade is B")
    elif score >= 0.7:
        print(" Grade is C")
    elif score >= 0.6:
        print(" Grade is D")
    elif score <0.6:
        print(" Grade is F")
```

Module - 2

3a: With syntax, explain the finite and infinite looping constructs in Python. What is the need for break and continue statements? Explain with examples — 08 Marks

Scheme: finite looping (Syntax + Example) — 02 marks
Infinite looping (Syntax + Example) — 02 marks
Break and continue + Example — 04 marks
08 marks

Solution:

Finite loop:- If we know the number of iterations in prior then the loop is called as finite loop.

Example:- for loop

Syntax:-

```
for iteration_variable in sequence:
        Statement_1
        Statement_2
            :
        Statement_n
    Statement_after_for loop
```

Example:-

```
names = ["ECE", "VTU", "Belgum"]
for x in names:
        print ("welcome", x)
print ("Done")
```

Indefinite looping:- The number of iterations are unknown in prior Ex:- while loop

Syntax :- while boolean_exp :
```
                  Statement _1
                  Statement _2
                      :
                  Statement _n
          statement _ after _ while
```
break statement :- In order to come out from the infinite loop break statement is used

Syntax : while testexpression :
```
                  if condition :
                      break
```
continue statement : break statement helps to come out from the infinite loop. where as continue statement is used to skip the current iteration of the loop

Example :
```
    while True :          # infinite loop
        line = input ('>')
        if line == 'done' :
            break
        print (line)
    print ('Done')
```
36. what are string slices ? Explain string slicing operator in python with examples ——— 05 marks

Scheme : String Slices — 01 Marks
String operators — 04 marks
05 Marks

Solution : string slice :- portion or segment of a string is called a string slice.

Syntax :- string-name [start : end[:step]]

String slice operators

Example :-

```
>>> beverage = " Black Coffee"
>>> beverage = [ : 5]        # start index not specified
>>> beverage [ 0 : 5 : 2]    # start, step & end index is specified
>>> beverage [ : : 2]        # start, ~~stop~~ end index is not specified
>>> beverage [5 : ]          # start index is specified
                               step & end index is not specified
>>> beverage [4 : 4]         # start & end index are equal
>>> beverage [5 : 20]        # end index value is more than end of string.
```

30. Write a python program to accept a file name from the user : i) Display the number of characters in file
ii) find the frequency of occurence of the lines which started with a word 'From'

— 07 marks

Scheme: accept file name from user — 01 mark
Display the number of Characte — 02 marks
finding the frequency of lines with word 'From' — 04 marks
07 marks

Program:

```
fname = input (" Enter the filename: ")
count = 0
try :
    fhand = open (fname)
except :
    print (" file cannot be opened")
    exit ()
~~for~~ for line in fhand :
    if line . startswith ('From'):
        Count = count + 1
print (" Line Count : " , count)
```

4a. List and explain any four built in string manipulation functions supported by python with examples — 06 marks.

Scheme: List of built-in string function — 02 Marks
Explaination of each — 04 x 1 = 04 mark
06 marks

Solution:- List of built-in string function

1. str()
2. len()
3. max()
4. min()
5. capitalize()
6. lower()
7. upper()

1. str():- This function returns a String for the passed argument. If the argument is not provided then it returns empty string.

Ex:- >>> str(10) # i/p is int but o/p is String

o/p '10'

~~>>> print~~

2> capitalize():- method returns a string where the first character is upper case, & the rest is lower case

Syntax:- string.capitalize()

Example:- txt = "python is FUN"

x = txt.capitalize()

print(x)

3> lower():- This method returns a string where all characters are lowercase

Syntax:- string.lower()

Ex:- txt = "Hello My friends"

x = txt.lower()

print(x)

47 upper() :- returns a string where all characters are in upper case

Syntax :- string.upper()

Example :-
```
txt = " Hello My Friends"
x = txt.upper()
print(x)
```

46 :- Explain file open, file close, file read and file write concepts in python with examples — 08 Marks

scheme :

file open	—	02 marks
file close	—	02 marks
file write	—	02 marks
file read	—	02 marks
		08 Marks

Solution :-

file open :- when you open a file, you are asking the OS to find the file key name and make sure the file is existing

Syntax :- open (filename, mode) # open() is function

Ex :-
```
fhand = open ("mbox.txt')
print (fhand)
```

file close :- Closing a file will free up the resources that were tied with the file

Syntax :- filename.close()

Ex :- fhand.close()

Reading file

The content of the file can be read, first open the file, which returns the file handle also known as object method i.e read method is associated with object i.e we can read the content of file

example:-

```
fhand = open(" filename.txt")
print (fhand. read ())
```

Writing file:- In order to write information in a file write() method is used with the file handler or object and parantheses pass the information that we need to write. After writing the file, the file must be closed using close() method.

Ex:-
```
fhand = open ('filename.txt', 'w')
fhand. write (' hi how are you')
fhand . close ()
```

4c. Write a python program to find the largest value from the given set of accepted values — 06 marks

Solution:

```
largest = None
print ('Before', largest)
for itervar in [ 3, 41, 12, 9, 74 15]:
    if largest is None or itervar > largest:
        largest = itervar
    print ('Loop: ', itervar, largest)
print ('largest: ', largest)
```

Module - 3

5a. Lists are mutable. Justify the statement with examples. Discuss the list handling functions in python with examples — 08 Marks

Scheme :- Lists are mutable with example - 03 marks
List handling functions — 05 Marks
08 Marks

Lists are Mutable :- After creating a list, items in the list are modified i.e items can be modified by replacing the older item with a newer item in its place without assigning the list to a completely new variable.

Syntax for accessing the elements of a list

Ex :- numbers = [10, 20, 25, 40, 50]
index 0 1 2 3 4

```
numbers [2] = 30
print (numbers)
[10, 20, 30, 40, 50]
```

List handling functions :- 1) len() 2) max() 3) min() 4) sum() 5) list()

Example :- num = [3, 41, 12, 9, 74, 19]

```
>>> print ( len (num))
6
>>> print (max (num))
74
>>> print ( min (num))
3
>>> print ( sum (num))
154
```

list() : list is a function i.e it is built-in python function that creates an empty list

Example .. s = 'spam'

```
t = list(s)
print (t)
['s', 'p', 'a', 'm']
```

5b:- Differentiate between list an dictionary —— 04 marks

Scheme: Diffeence between list & dictionary - 04 marks

List	Dictionary
1. Ordered sequence of objects	1. Unordered sets
2. Elements are accessed via their position	2. Elements are accessed via keys
3. list are Mutable	3. Dictionary are immutable
4. Elements are placed in []	4. Elements are placed in { }

5C. Write a python program to search lines that start with the word 'from' and a character followed by a two digit number between 00 and 99 followed : print the number if it is greater than zero. Assume any input file —— 08 Marks

Scheme: program - 08 marks

Program:
```
import re
fhand = open('mbox.txt')
for line in fhand:
 line = line.rstrip()
  x = re.findall('^from: *([0-9][0-9]):', line)
  if line(x) > 0
```

6a Compare and contrast tuples with lists. Explain the following operation in tuples with examples
i) Sum of two tuples ii) Slicing operations
iii) Compression of two tuples iv) Assignment to variables
— 10 Marks

Scheme: Comparision between tuples & lists — 02M
Explaination with Example Sum of tuples - 02M
slicing operation - 02M
Comparision of two tuples - 02M
Tuple assignment to variable - 02M
08M

Solution:-
comparision between tuples and lists

Tuples	List
1. Elements in the tuple are enclosed in ( )	1. Elements in the list are enclosed in [ ]
2. tuple are immutable	2. List are mutable.

Sum of two tuples :-

```
t = ('a', 'b', 'c', 'd')
t = ('A',) + t [1:]
print (t)
('A', 'b', 'c', 'd')
```

Slicing operations

```
t = ('a', 'b', 'c', 'd')
print (t [1: 3])
('b', 'c')
```

tuple assignment

```
fruits = ("apple", "orange", "cherry")
green, yellow, red = fruits
print (green)
print (yellow)
print (red)
```

comparing tuples

```
t1 = (1, 2, 3)
t2 = (1, 4, 3)

t3 = t1 == t2

print (t3)
False.
```

6b Write a python program that accept a sentence and build dictionary with LETTERS, DIGITS, UPPERCASE, LOWERCASE as key values and their count in sentence as values — 06 Marks

Scheme :- program — 06 Marks

program :-

```
str = input (" enter the sentence: ")
cletter =0
cdigit =0
clower =0
cupper =0
d = {}
for c in str:
    if c.isalpha()
       cletter = cletter +1
```

```
if c.islower():
        clower = clower +1
if c.isupper():
        cupper = cupper +1
d['letter'] = cletter
d['lower'] = clower
d['upper'] = cupper
d['digit'] = cdigit
print(d)
```

6c: Explain the need of Regular expression in python language, with an example — 04 marks

scheme :- Explaination of RE — 02 marks
Example — 02 marks
04 marks

solution :- A regular expression is a special sequence of characters that helps you to match or find other strings or set of strings, using a specialised syntax held in a pattern.

Example :-

```
import re
txt = "The rain in spain"
x = re.findall("ai", txt)
print(x)
```

o/p: ["ai", "ai"]

Module - 4

7a. Explain Classes and Attributes in python language with examples — 06 Marks.

Scheme :- explaination about classes & attributes - 03 M
Examples - 03 M
06 M

Solution :-

Class : class is a "blueprint" for creating objects. A class is an object constructor. Class is a prototype i.e it is like a factory for creating object

Syntax for creating class

```
Class classname:
    <statement_1>
        :
    <statement_2>
```

Ex :-

```
class point:
    pass
print (point)
```

Attributes :- Attributes are variables
There are two types of attributes
class attributes
Object attributes

Class attributes :- classname.classattribute = value
Syntax to access class attribute :- Objectname. classattribute

Object attribute.
syntax to assign value → Objectname. attributename = value
Syntax to access instance attribute :- Objectname. attribute name

7b What is the difference between method and function. Explain the working of init method with suitable code
—06 Marks

Scheme:- difference b/w^n method & function — 02M
working of init method — 02M
Suitable code — 02M
06M

Solution:- Method is a function that is defined inside a class definition & is invoked on instances of that class. Method is a function that is ~~defined~~ associated with a particular class while a function is not.

init method:- __init__ method
The init method is an initilization method. It is a special method that gets invoked automatically when an object of a class is instantiated. i.e all the attributes of the init method are going to become the attributes of the object instantiated.

Syntax:-
```
def _init_ (self, par1, par2, ... parn):
    Statements
```

Example:-
```
class Time:
    def __init__(self, hour=0, min=0, sec=0):
        self.hour = hour
        self.min = min
        self.sec = sec
    def print_time(self):
        print('%.2d : %.2d : %.2d' % (self.hour,
                  self.min, self.second))
```

```
t1 = Time (9, 45, 56)
t1. print_time ()

t2 = Time (10, 23, 45)
t2. print_time()
```

o/p:

```
09 : 45 : 56
10 : 23 : 45
```

7c. Write a function named move rectangle that takes a rectangle and two numbers named dx and dy. It should change the location of rectangle by adding dx to x- coordinate of corner and adding dy to the y-co-ordinate of corner — 08 Marks

Scheme : program : 08 marks

Program:

```
Class point():
    " Represents a point in class"

Class Rectangle ():
    " Represents a rectangle attribute width, length,
      height, corner (lower left corner) "

box = rectangle()
box. width = 100.0
box. height = 200.0
box. corner = point()
box. corner. x = 0.0
box. corner. y = 0.0
```

```
def more_rectangle (rect, dx, dy):
    box.corner.x + = dx
    box.corner.y + = dy
more_rectangle (box, 50, 100)
```

8a Show using a python code how _str_ method is invoked when you print an object. Explain its working — 06 marks

Scheme: _str_method invoking — 03 M
-Explaination — 03 M
06 M

str method returns a string representation of an object. It is used for printing the types in python. But in order to display class objects rather than using user defined function like print_time() i.e which uses print () function, we can use _str_ method
__str__ method will be invoked automatically when we want to print an object.

Example:-

```
class Time :
    def __str__ (self) :
        return '%.2d: %.2d: %.2d' %.( self.hour,
                                     self.min, self.second)

~~classe time:~~

time = Time (9, 45)
print (time)
```

8b:- Illustrate the concept of pure function and Modifier with example —— 06 marks

Scheme:-

Illustration of pure function with an example - 03M

Illustration of modifier with an example — 03M

06M

Solution:-

Pure function:- pure function is a function which does not modify any of the object passed as an argument to the user defined function i-e such functions are called as pure function

Ex:-

```
class Time:
    " Represents the time of the day"

def print_time (time):
    print("%.2d: %.2d :%.2d" % (time.hour, time.min, time.second)


def add_time (t1, t2):
    sum = Time()
    sum.hour = t1.hour + t2.hour
    sum. min = t1.min + t2.min
    sum.second = t1.second + t2.second
    if sum.second >= 60:
        sum.second -= 60
        sum.min += 1
    if sum.min >= 60:
        sum.min -= 60
        sum.hour += 1

    return sum

t1 = Time ()
t1. hour = 10
t1. min = 84
t1. sec = 25

print ("Time 1 is : ")
print_time(t1)

t2 = Time()
t2. hour = 2

t2. min = 12

t2. sec = 41
print ("Time2 is : ")
print_time(t2)
t3 = add_time (t1, t2)
print (" sum of two time : ")
print_time(t3)
```

Modifiers :- If the user defined function modifies the attributes of the object which is passed as argument of user defined function. Such functions are called as modifiers.

Ex :-
```
class time :
    " Represents time"
```

```
t = time()
t. hours = 0
t. min = 0
t. sec = 0
def increment (t, second):
    t. sec + = second
    if t. sec > =60:
        t. sec - = 60
        t. min + = 1

    if t. min > =60:
        t. min - =60
        t. hour + =1
increment (t, 72)
print (t. sec)
print (t. min)
print (t. hour)
```

8c what is operator overloading? write Python code to overload "+", "_" and "*" operator by providing the methods --add--, --sub--, --mul-- — 08 marks

Scheme:

Operator overloading — 02 marks
Python code for overloading — 06 marks
08 marks

Solution:

Operator overloading: Ability of an existing operator to work on programmer defined type i.e class.

Python program

```
class point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
    def __add__(self, p2):
        p3 = point()
        p3.x = self.x + p2.x
        p3.y = self.y + p2.y
        return p3
    def __sub__(self, p2)
        p5 = point()
        p5.x = self.x - p2.x
        p5.y = self.y - p2.y
        return p5
    ~~p1 = point~~
    def __str__(self):
        return "(%d %d)" % (self.x, self.y)
p1 = point (2, 3)
p2 = point (4, 5)
print ("p1 is:  ", p1)
print ("p2 is:  ", p2)
p4 = p1 + p2
p6 = p1 - p2
print ("sum is:  ", p4)
print ("diff is:  ", p6)
```

99. Define socket. Explain how socket connection can be established to the internet using python code over TCP/IP connection and the http protocol to get the web document — 08 marks

Scheme : Defn of socket — 02 m
Making connection to web browser — 03 m
Displaying what server sends back — 03 m
08 m

Socket :- is a bidirectional data path to a remote system i.e data can be read from the socket and write to the same socket.

```
import socket
mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect (('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org/remo.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())
mysock.close()
```

9b. Compare and contrast the Javascript Object Notation (JSON) and XML. Explain parsing of XML with example — 06 marks

Scheme: Difference – 03 M
Parsing of XML – 03 M
06 M

Solution

Comparison between JSON and XML

JSON	XML
1. JSON object has a type	1. XML data is type less
2. It has no display capability	2. It offer the capability to display data
3. Supports only UTF-8 encoding formats	3. Supports various encoding format
4. It is less secured	4. It is more secured

Parsing of XML with an example

Program:

```
~~import urllib.request, urllib.parse, urllib.error~~
~~from bs4 import~~
import xml.etree.ElementTree as ET
data = '''
<person>
  <name> Chuck </name>
  <phone type = "intl">
     +1734  303  4456
  </phone>
```

```
<email hide = "yes"/>
</person>"'

tree = ET. fromstring (data)
print ('Name:', tree. find ('name'). txt)
print ('Attr :', tree. find ('email'). get('hide'))
```

10 a What is Embedded SQL. Explain the importance of SQL lite data base — 04 Marks

Scheme :- Embedded SQL — 02 Marks
Importance of SQL lite - 02 marks
04 Marks

Embedded SQL is a method of combining the the computing power of a programming language and the database manipulation capabilities of SQL.

SQ Lite is self - contained means it does not need any external dependence like an OS ~~of~~ or external library. The feature of SQlite help especially in embedded devices like Android.

9c Define cursor. Explain connect, Execute and close command of databases with a snippet code —— 06 Marks.

Scheme:- Explaination — 04 marks
Snippet code — 02 marks
06 marks

Solution:-
Cursor is used to interact with the database
Connect:- connect operation makes a connection to the database stored in the file. If the file doesnot exist, it will be created. Once we have cursor we begin to execute command on the contents of the database using the execute method
At the end of operation close command is used to close the connection.

Snippet code:

```
import sqlite3
 conn = sqlite3.connect('music.sqlite')
 cur = conn.cursor()
 cur.execeete('DROP TABLE IF EXISTS Tracks')
 Cur.execute ('CREATE TABLE Tracks (title TEXT, plays
                                        INTEGER)')
 conn.close()
```

10b Write a note on Google geo coding web service. Using python supported libraries demonstrate with a snippet code —— 08 Marks.

Scheme: Google geo coding web service — 03 marks
Demonstration using snippet code — 05 marks
08 marks.

Solution :-

Google Geo coding web service is the process of converting addresses into geographic co-ordinate, which can be used to place makers on a map or position the map.

Google has an excellent web service that allows us to make use of their large database of geographic information. The geogoding service is free but rate limited so it cannot be used to make unlimited calls to the API in a commerical applications.

Snippet code

```
import urllib.request, urllib.parse, urllib.error
import json
service url = 'http://maps.google apis.com/maps/api/
                    geocode/json?'

 while True:
      address = input('Enter location')
      if len (address) < 1:
           break
      url = service url + urllib.parse.urlencode
                  ({'address': address})

      print ('Retrieving', url)
      uh = urllib.request.urlopen(url)
      data = uh.read().decode()
      print('Retrieved, len(data), 'character')
```

```
try:
    js = json.loads(data)
except:
    js = None

if not js or 'status' not in js or js['status'] != 'OK'
    print('====Failure To Retrieve ====')
    print(data)

    continue
print(json.dumps(js, indent=4))
lat = js["results"][0]["geometry"]["location"]["lat"]
lng = js["results"][0]["geometry"]["location"]["lng"]
print('lat', lat, 'lng', lng)
location = js['results'][0]['formatted_address']
print(location)
```

10 C Write a python code to read file from web using urllib and retrieve the data of the file also compute the frequency of each word in the file - 08 marks

Scheme:- Python code to read file from web & retrieve - 04 marks

Computing the frequency of each word - 04 marks

08 marks.

Solution :-

Code to read the file from web using urllib

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

example to retrieve the data & compute the frequency of each word in the file as follows :

```
import urllib . request, urllib.parse, urllib. error
fhand = urllib. request. urlopen('http: // data. pr4e. org/
                                                romeo. txt')
counts = dict()
 for line in fhand:
        words = line. decode (). split()
        for word in words;
            counts [word] = counts. get( word, 0) +1
 print( counts)
```