1) What are List? How to define and acess the element of List.

Ans a) i) The lists are very similar to arrays. it is a sequence of values.

ii) List can contains any type of variable, and they can contain as many variables as you wish.

iii) Lists can also be iterated over in a very simple manner.

iv) The values in a list are called elements or sometimes items

(b) Define and acess the element of list.

i) We can use the index operator [] to acess an element in a list.

ii) Index starts from 0, so a list having 5 elements will have index from 0 to 4.

iii) Trying to acess an element other than this will raise an IndexError. The index must be an Integer

iv) We can't use float or other types, this will result into TypeError.

(v) Example

program

```
Names = ['Varun', 'Tushar', 'Sonali', 'Vedu']
print("Names")
print(Names[0])
print(Names[3])
```

Output:

```
['Varun', 'Tushar', 'Sonali', 'Vedu']
Varun
Vedu
```

(2) How to traverse and delete the elements in the list.

Ans. (i) The for-in statement makes it easy to loop over the items in a list:

```
names = ["Tushar", "Sonali", "Vesho", "Vedu"]
for e in names.
    print(e)
```

Output

```
Tushar
Sonali
Vesho
Vedu
```

(ii) If you need both the index and the item, use the enumerate function:

```
names = ['Varun', 'Stark']
for i,e in enumerate(names):
    print(i,e)
```

Output

```
0 Varun
1 Stark
```

(iii) If you need only the index, use range and len:

```
names = ['Varun', 'Wing']
for i in range(len(names)):
    print(i)
```

Output

```
0
1
```

(b) i) We can use remove method of list to delete the element from the list. The remove method does not return any value but removes the given object from the list.

Ex. s = [1, 2, 3, 4, 5]
s.remove(3)
Output
print(s)
Output
s = [1, 2, 4, 5]

ii) If we want to delet the elements on basis of index then we can use del method.

Ex s = [1, 2, 3, 4, 5, 6]
del s[3]
print(s)
Output
s = [1, 2, 3, 5, 6]

iii) We can also popup the element from the list using pop method by providing the index.

Ex. s = [1, 2, 3, 4]
s.pop(0)
print(s)
Output
s = [2, 3, 4]

(iv) By default pop without any provided any index the last element in the list will be removed.

s.pop()
Output
s = [1, 2, 3]

Write a short note on tuple in Python

Ans i) The tuple is similar structure as list.

ii) The only difference is that list is mutable i.e can be change once assigned and tuple is immutable i.e we cannot change the elements afte assignment.

iii) Tuples are used for grouping data.

iv) Each element or value that is inside of a tuple is called item

v) The tuple is defined with whole set of element enclosed in parentheses.

vi) Each value is numbered (indexed) starting from zero

Example

```
s = ('name', 'Roll', 'angle')
print(s)
>>> ('name', 'Roll', 'angle')
```

What is a tuple how to create and acess it

Ans. i) The tuple is similar structure as List.
ii) The only difference is about flexibility to change the elements once assigned
iii) In tuple we cannot change the elements after assignment, which is done in list
iv) Tuples are used for grouping data.
v) Tuples considers value between parentheses() separated by commas ",".
vi) Empty tuples will appear as:
s = ()
vii) The single tuples value must use a comma as:
s = ('y',)
viii) The tuples are immutable, means we can't alter the values of tuple elements.

(b) python provides several methods to accers elements in tuple.
i) indexing:
→ The tuple items are indexed, so we can use the index operator [] to acess an item in a tuple where the index starts from 0.
→ The index is always an integer. using other types can result in TypeError.
Example

```
st = ('y','a','s','h','r','e','e')
print(st[0])
print(st[5])
>>> y
>>> e
```

ii.) Negative Indexing:

-> The Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item, similarly applied to all other sequence.

-> Example

```
st= ('y','a','s','h','r','e','e')
print(st[-1])
print(st[-6])
>>> e
>>> a
```

iii.) Slicing:

The slicing refers to access the items of selected range. Python provides ":" operator to achieve this with tuples

Examples

```
st= ('y','a','h','r','e','e')
print (st[1:4])
print (st[:-5])
print (st[6:])
print(st[:])
print (st[::-1])
```

Output

```
('a','s','h')
('y','a')
('e')
('y','a','s','h','r','e','e')
('e','e','r','h','s','a','y')
```

Explain basic tuple operation ~~in dictionary~~

Ans) a) Concatenation:-

i) It is combining activity for values. To combine the contents of two tuples, use plus sign (+) between the two tuples being combined, and which contains all of the elements of the first tuple followed by all of the elements of the second tuples.

-> Examples

```
s=(22,3,'Sonali')
t=('Kushar',13,14,12)
print(s+t)
```

Output

```
(22,3,'Sonali','Kushar',13,14,12)
```

b) Repetition:

The asterisk (*) is overloaded for tuple to serve as a repetition operator. The result of applying repetition to a tuple is a single tuple, with the elements of the original tuple repeated as many times as you specify:

-> Example

```
s=('Varun','Varun')
t=('Varun','Varun')*2
print(t)
```

Output

```
('Varun','Varun','Varun','Varun')
```

c.) in Operator:

i) It is used to check the existence of element in the tuple.

ii) We have to provide a value on the left hand side of the in operator, and a tuple on the right hand side; an expression so constructed will return True if the value is in the tuple and False otherwise.

iii) The left hand side can be a literal value, or an expression; the value on the right hand side can be a tuple, or an expression that evaluates to a tuples

Example:

```
>>> fruit = ('a', 'p', 'p', 'l', 'e')
>>> print ('a' in fruit)
>>>  True
>>> print ('b' in fruit)
>>> false
>>> print ('g' not in fruit)
>>>  True
```

What is dictionary? How to create and acess it

Ans.) i.) python provide dictionary structure to deal with an unordered set of key and value pairs.

ii.) This is a container which contains data in curly braces.

iii.) The pair i.e. key and value is known as item

iv.) The key passed to the item must be unique. The key and the value is separated by a colon(:).

v.) This pair is known as item. items are seprated from each other by a comma (,).

vi.) Different items are enclosed within a curly braces and this creates dichonary

vii.) The values in the dichonary are mutable. However the key is immutable and must be unique for every item.

(viii.) The key is used to acess the specific value. associative array since the key works as Index and they are decided by the user.

Ex. program.

```
mydata = {}
mydata[1] = 'Tush'
mydata[2] = 'Varun'
mydata['name'] = 'Yusha'
mydata[4] = 'Vedu'
print (mydata)
print (mydata[2])
print (mydata['name'])
```

Output

```
{1: 'Tush', 2: 'Varun', 4: 'Vedu', 'name': 'Yusha'}
Varun
Yusha
```

Write short note on operation in dichonary

Ans.)

1.) len(d): It returns the number of stored entries in dichonary, i.e. the number of (key, value) pairs.

2.) del d[k]: It deletes the key k together with its associated value.

3.) ~~del d~~ k in d: This returns True, if a key k exists in dichonary d.

4.) k not in d: This returns True, if a key k doesn't exist in the dichonary d.

5.) Example:

program

```
d={'Name':'Veshu', 'Age':1, 'School':'VHP'}
print(len(d))
print(1 in d)
print(2 not in d)
print(99 in d)
del d['school']
print(d)
del d
```

Output

```
3
False
True
False
{'Name':'Veshu', 'Age':1}
```

Write a program to delete the element form the dictionary

Ans program:

```
d={'Name':'Varun','Age':1,'School':'H.N.H.S'}
print(d)
del d['Age']
print(d)
d.clear()
print(d)
print("Dichonary Deleted")
```

Output

```
{'Name':'Varun','Age':1,'School':'H.N.H.S'}
{'Name':'Varun','School':'H.N.H.S'}
{}
Dichonary Deleted.
```

Write a program to show Various Opreation in dictionary

Ans.

program:

```
student = {'name': 'varun', 'class': 'second year', 'roll no': 17}
print ("Studen :", student)
student ['last name'] = 'jaiswar'
print ("last name exist in student :", 'last name'
        in student)
del student ['last name']
print ("Student :", student)
print ("last name does not exist in student :",
        'last name' not in student)
student.clear()
print ("student :" student)
del student
```

Output

```
student : {'name': 'varun', 'class': 'second year', 'roll no': 17 }
student : {'name': 'varun', 'class': 'second year',
        'roll no': 17, 'last name': 'jaiswar'}
last name exist in student : True
Student : {'name': 'varun', 'class': 'second year', 'roll no.': 17}
last name does not exist in student : True
student : { }
```

What is a file and what are it's operating modes

Ans. 1) In windows there are different types of files such as Text, Image files. etc for a computer file is a named location on hard disk to store related information.

2) All type file modes can be easily applied on file at the time of opening it.

a. r mode
Open text file for reading. The stream is positioned at the start of the file

b) r+ mode
Open text file for reading the stream is positioned at the beginning of the file.

c) w mode
Truncate to zero length or create text for writing. The stream is positioned at the begginning of the file.

d) w+ mode
Open for reading. The file is created if it does not exist. The stream is positioned at the start of the file.

e) a mode
Open for writing. The is created if it does not exist. The stream is positioned at the end of the file. Subsequent writes to the file will always end up at the current end of file.

f) a+ mode
Open for reading and writing. The file is created if it does not exist. The stream is positioned at the end of the file. Sopsequent writes to the file will always end up at the current end of file.

g) rb+modes
Opens a file for reading and writing in binary format.
The pointer placed at the beginning of the file.

h) rb mode
Opens a file for reading only in binary format.
The file pointer is placed at the beginning of the file
This is the default mode.

i) wb mode
Opens a file for writing only in binary format.
Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
wbt mode

j) Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new for reading and writing.

What is text file? write a program to show how to create a Text file.

Ans. A text file is computer file that only contains text and has no special formatting such as bold text, italic text, images, etc. It is sequence of characters stored on a permanent medium like a hard drive, flash memory, or CD-ROM

(i) Text files are structured as a sequence of lines. where each line includes a sequence of characters

(ii) Each line is terminated with a special character, called the EOL or End of Line characters.

(iii) There are several types, but the most common is the comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun.

(iv) For creating the text file just use open() function with appropriate arguments as explained earlier.

Example

```
f= open ("t1.txt", "w")
f.write ("Hello")
f.write ("Tushar Nigga");
f.cofs
f.close
```

Write() method is used to add line in the text file.

Write short note on file object Attributes and Explain with a program.

Ans. i) ~~Write~~ After opening the file and arrigned it to a file object, we can get all information about the file

ii) file attributes.

a) file.closed
Returns true if file is closed, false otherwise

b) file.mode
Returns access mode with which file was opened

c) file.name
Returns name of the file

iii) program

t2.txt
Tushar Sambare
Airoli

f1.py
```
f=open("t2.txt","rt")
print("Name of the file:",f.name)
print("Closed or not:",f.closed)
print("Opening mode:"f.mode)
f.close()
print("Closed or not:",f.closed)
```

Output
```
Name of the file: f2.txt
closed or not: false
Opening mode: rt
closed or not: True.
```

Write short note on file methods with suitable Example

Ans. following are few methods

1. F.read(): for getting a string containing all characters in the file, you can use f.read() as shown below.

Ex. F= open ('t2.txt', 'r')
print (f.read())

Output
Tushar Sambare
Airoli

2. F.readline: This function will read line by line from file. The first call of readline() function will return the first line of your file and if we make subsequent call to same method it will return, return successive liner. Basically, it will read a single line from the file and return a string containing characters upto \n.

Ex F= open ('t2.txt', 'r'))
print (f.readline())
Output:
Tushare Sambare

3. F. readlines(): This function returns the complete file as a list of strings each separated by \n.

Ex. F= open ('t2.txt', 'r'))
print (f.readlines())
Output
["Tushar Sambare \n', 'Airoli \n']

4. f.write(): This method takes one parameter, which is the string to be written. To start a new line after writing the data, add a \n character to the end.

```
f= open("t1.txt","w")
f.write("Hello")
f.write("Tushare Sambare");
f.close()
```

Output

Hello Tushare Sambare

5. f.tell(): This method tells you the current cursor position within the file, in the other words, the next read or write will occur at that many bytes from the beginning of the file. This method returns long as per position

Ex.
```
f= open("demo.txt","r"))
print(f.tell())
```

Output

0

6. f.seek(offset[,from]): This method changes the position. The offset argument tells the number of bytes to be moved. This form argument specifies the reference position from where the bytes are to be moved.

Ex.
```
f=open("t2.txt","r+")
st=f.read(6)
print("Read String is :",st)
pos=f.tell();
print("Current file position:",pos)
position=f.seek(0,0)
sf=f.read(6)
print("Again read String is:",st)
f.close()
```

Output:
Read String is: Tushar
Current file position: 6
Again read String is: Tushar

7. os.rename (current_F_name, newf_name):
This method is provided by os module of python. Using this method we can rename exishing file by providing two arguments as first the name of exishing file and second the name as that we want to assign to the file.

Ex.
```
import os
os.rename ("t2.txt", "st.txt")
print ("File Renamed Successfully")
```
Output
File Renamed Sucessfully

8. os.remove ("fileName.txt"): This method is proveded by os module of Python. By using this method you can delete files by supplying the name of the file to be deleted as the argument.

Ex
```
import os
File.remove ("text2.txt")
print ("File Deleted successfully")
```
Output
File Deleted Successfully

Write a program to shows to use tell() and seek() method of file.

Write a short note on directories.

Ans. (i) A directory is a file system cataloging structure which contains references to other computer files, and possibly other directories, in windows environment we generally refer it as folders.

(ii) The python provides various useful built in methods to deal with directories.

(iii) The Python's os module has several methods that help you create, remove and change directories by providing several useful functions as explained below

(a) os.mkdir("Name"): This method of os module is used to create directories in the current directory by providing the name of directory.

Ex.
```
import os
os.mkdir("test")
print("Directory Created Sucessfully")
```
Output
Directory Created Sucessfully

(b) os.chdir("dirPathName"): By using this method we can change the working directory by providing the new path of diretory as an argument. We can use both forward slash(/) or the backward slash(\) to separate path elements.

Ex.
```
import os
os.chdir('C:\\Python34')
print(os.getcwd())
```
Output
C:\Python34

(c.) os.getcwd(): To get the name of current working directory we can use this method. This method prints the name of current working directory in the form of a string. We can also use the getcwd() method to get it as bytes object.

Ex.
```
import os
os.mkdir("styu")
print(os.getcwd())
```
Output
```
C:\Python34
```

(d.) rmdir("dirName"): This function is used to remove or delete the directory mentioned in the argument. It is mandatory to remove all the contents of the directory first before removing it.

Ex
```
import os
os.rmdir("styu")
print("Directory Deleted Successfully")
```
Output
Directory Deleted Sucessfully.

(e.) os.rename: Using this method we can rename existing directory by providing two arguments as first the name of existing file and second the name as that we want to assign to the file.

Ex.
```
import os
os.rename("f1","f2")
print("Directory Rename Successfully")
```
Output
Directory Rename successfully.

(f.) os.listdir("dirName"): Using this method we can list the directories present under the directory name provided as argument.

Ex.

```
import os
        print(os.listdir("t2"))
        print(os.listdir("t2/sonali"))
```

Output

```
['sonali']
['vedu', 'yashu']
```

What are Exception? Explain built in Exception?

Ans. (i) Exception is an abnormal condition in a program code. It is an error that happens during execution of a program.

(ii) When ~~exception~~ error occurs python generates an exception that can be handled, which avoids our program to crash.

(iii) Python has many built-in exception which forces you program to output an error when something in it goes wrong.

(iv) When there exceptions occur, it causes the current process to stop and passes it to the calling process until it is handled. if not, then your program will crash

(v) following are some summaried built in exception supported by Python.

a) Exception
Base class for all exception.

b) Stop Iteration
Raised when the next() method of an iterator does not point to any object.

c) System Exit
Raised by the sys.exit() function.

d) Arithmetic Error
Base class for all errors that occur for numeric calculation

Write a program try--finally with multiple excepts

Ans.

Write a program try--except-else

Ans.

program:

```
def divide(x, y):
    try:
        result = x // y
    except ZeroDivisionError:
        print ("Sorry! You are dividing by zero")
    else:
        print ("Yeah! Your answer is :", result)
divide(3, 2)
divide(3, 0)
```

Output

```
Yeah! Your answer is : 1
Sorry! You are dividing by zero
```

What are Regular expression. Explain any five patterns.

Ans. (i) The regular exprerrion can be defined as a specific kind of text pattern. There patterns you can use with many mordern applications and programming langvages to have intersehing and fast results.

(ii) You can use them to verify whether input fits into the text pattern to find text that matches the pattern within a larger body of text, to replace text matching the pattern with other text or rearranged bits of the matched text, to spilt a block of text into a list or subtexts etc.

(iii) Python re module functions support set of strings that matches it, Except for control characters (+, ?, ., *, ^, $, (), [], {}, |, \), all character match themselves.

(iv) following are some regular expression

a) . (aperiod) - matches any single character except newline '\n'

(b) \w - matches a "word" character: letter or digit or underbar [a-zA-Z0-9_]

(c) \W = matches any non-word character.

(d) \b = boundary between word and non-word

(e) \s = matches a single whitespace character - space, newline, return, tab.

Explain math function with suitable Examples.

Ans. (i) This function attempts to match RE pattern to string with optional flags.

Syntax:

re.match(pattern, string, flags=0)

Where, pattern = The regular expression to be matched
string = Given string in which pattern is matched
flag = You can specify different flags using bitwise OR (|).

ii) Example program.

```
import re
l = "Using Regular Expression is simple in Python"
st = re.match(r'(.*) is (.*?) .*', l, re.M|re.I)
if st:
  print("st.group():", st.group())
  print("st.group(1):", st.group(1))
  print("st.group(2):", st.group(1))
else:
  print("No match")
```

Output.

st.group(): Using Regular Expression is simple in Python
st.group(1): Using Regular Expression
st.group(2): simple

What is OOP and explain concepts of OOP.

Ans) (i) Object Oriented Programing is a technique that combine data and instructions for processing that data into an object that can be used within the program.

(ii) Object-oriented programming provides concepts that helps modelling complicated system of real world into manageable software solutions.

(iii) It is paradigm that provides many concepts such as inheritance, Object, Polymorphism etc

(iv) followings are important concepts of OOP.

(a) Object:

Object is anything in the real world that contains some unique characterishics. It has state and behavior. It may be physical and logical. For example:

Ex. mouse, keyboard, chair, table etc.

(b) Class:

The class is a collection of objects. It is a logical entity that has some specific attributes and methods.

Ex. If you have a student class then it should contain an attributes and method i.e. an name, standard, age etc.

(c) Method:

The method is defined as a group of statements enclosed together to achieve specific task. It is associated with an object. In Python method is not unique to class instances. Any objects type can have their own methods.

(d.) Inheritance :-

1) Inheritance can be defined as characteristic transferring technique. a feature of one class can be shared with the another class through inheritance.
2) It means one object acquire the properties and behaviours of parent object.
3) The new class is known as derived class or child class and form which it inherits the properties is called base class or parent class.

(e.) Polymorphism:

1) Polymorphism is the ability to present the same interface for differing underlying forms.
2) Poly means many and Morphs means form, shape.
3) Polymorphic functions or methods can be applied to arguments of different types, and they can behave differently depending on the type of the arguments to which they are applied.

(f.) Encapsulation: Encapsulation is used to restrict access to methods and variables.

(2) Here code and data are wrapped together within a single unit from being modified by accident.

(g.) data abstraction:

1) Hiding the implementation details is called as data abstraction.
2) It shows functionalities. while hiding the internal details.
3) Abstracting something means to give names to things, so that the name capture the core of the what a function or a whole program does.

Write short notes on class in python, with program.

Ans. (i) In python, class definitions begin with a class keyword, followed keyword class followed class name ':' colon. After that every line segment we can have different statements such as variable declaration or functions

(ii) Syntax:

```
class class_name:
    [statement 1]
    [statement 2]
    [statement N]
```

(iii) program:

```
class st():
    def show(self):
        print("hello")

s=st()
s.show()
```

Output

hello

Write a short program to demonstrate use of inheritance.

Ans.

program:

```
class st1:
        def s1(self):
                print ("Base class")
class st2(st1)
        def s2(self):
                print ("Derived class")
t= st2()
t.s1()
t.s2()
```

Output

```
Base class
Derived class
```

Write a short program to demonstrate use of multiple inheritance.

Ans

program:

```
class Class1:
    def m(self):
        print("In Class1")
class Class2(Class1):
    def m(self):
        print("In Class2")
class Class3(Class1):
    def m(self):
        print("In Class3")
class Class4(Class2, Class3):
    pass
obj = Class4()
obj.m()
```

Output

```
In Class2
```