28-11-25

Date
Page No.

Q WAP to Print "welcome to Python world!"
Ans Print ("welcome to Python world")

Q WAP to print "Hello world!"
Ans Print ("Hello world!")

Q WAP to print your name, age, and address
Ans name = "Rajesh Rana"
age = 21
address = "Bhadrak"
Print ("Name :", name)
Print ("Age :", age)
Print ("Address:", address)

Q WAP to accept ~~two~~ numbers, Print the sum and product.
Ans a = 10
b = 20
print("a + b:", (a+b))
print ("a * b:", (a * b))

Q WAP to swap two numbers without using 3rd variable.
Ans a = 10
b = 20
Print ("originally: a =", a, ", b =", b)
a = a + b
b = a - b
a = a - b
Print ("After swapping: a =", a, ", b =", b)

Date
Page No.

Q WAP to swap two variables using 3rd variable

Ans
```
a = 10
b = 20
print ("Before swapping: a = ", a, ", b = ", b)
temp = a
a = b
b = temp
print ("After swapping: a = ", a, ", b = ", b).
```

Q WAP to find area and perimeter of a circle.

Ans
```
r = 10
area = 3.14159 * r * r
perimeter = 2 * 3.14159 * r
print ("Area is ", area)
print ("Perimeter is ", perimeter)
```

Q WAP to input your name, age and address and print.

Ans
```
name = input ("Enter your name")
age = int (input ("Enter your age"))
address = input ("Enter your address")
print ("Name : ", name)
print ("Age : ", age)
print ("Address : ", address)
```

Q WAP to input yours marks for 3 subjects then find sum and percentage.

Ans
```
a = float (input ("Enter 1st mark : "))
b = float (input ("Enter 2nd mark : "))
c = float (input ("Enter 3rd mark : "))
sum = a+b+c
per = sum / 3
print ("sum = ", sum)
print ("percentage = ", per)
```

Date
Page No.

Q WAP to swap with and without 3rd variable.

Ans
```
a= int (input ("Enter 1st variable"))
b=int (input ("Enter 2nd variable"))
print ("Before swapping : a =", a, ", b=", b)
a=a+b
b=a-b
a=a-b
print ("After swapping : a =", a, ", b=", b)


a= int (input ("Enter 1st number"))
b=int (input ("Enter 2nd number"))
Print ("Before swapping: a =", a, ", b=", b)
temp =a
 a=b
 b=temp
Print("After swapping : a =", a, ", b=", b)
```

Q WAP to find the area and perimeter of a triangle.

Ans
```
a= int (input ("Enter 1st side"))
b= int (input ("Enter 2nd side"))
c=int (input ("Enter 3rd side"))
sum = a+b+c
s = sum/2    #semi-perimeter
area = (s*(s-a)*(s-b)*(s-c))**0.5
Print ("Area :", area)
print ("Perimeter :", sum)
Print ("semi-Perimeter :", s)
```

Date
Page No.

Q WAP to calculate the simple interest by accepting the PTR using user input.

Ans
```
p=float(input("Enter the principal amount:"))
t=float(input("Enter the time in year:"))
r=float(input("Enter the rate of interest"))
print("simple interest:",((p*t*r)/100))
```

7.12.25

What is Python?

Ans Python is a general purpose, dynamic, high-level and interpreted programming language.

Guido van Rossum 1991 (module-3 + ABC language)

Fea-

Note: No need to manage hardware: In low-level languages (like C or assembly), the programmer must handle: Memory allocation, memory deallocation, Data types strictly, Hardware-level operations python automatically handles these tasks.

→ It supports object oriented programming approach.
→ Provides lots of high-level data structures.
→ Functional or procedural programming styles.
→ easy to learn
→ versatile scripting language
→ dynamic typing
→ an ideal language for scripting.

Python is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD etc.

Date
Page No.

we don't need to use data types to declare variables.

Python makes the development and debugging fast.

Python History:
Python was invented by Guido van Rossum in 1991 at CWI in Netherland. The Python programming language is successor of ABC language.
Python = ABC + Modula-3.

Features of python:
→ Interpreted language
→ cross platform language
→ open-source language
→ object oriented language
→ Large standard library
→ GUI Programming support
→ Dynamic memory Allocation
→ Integrated easily with C++, Java.

Applications: (where python is used)
→ web application development.
→ GUI applications
→ S/W development process
→ Scientific and numerical calculations
→ multimedia application
→ CAD, CAM application
→ Data science and data Mining
→ mobile APPLICATIONS
→ ERP, Ecommerce

Date
Page No.

→ Artificial Intelligence and machine learning.
→ Image processing Applications.
→ Speech Recognitions.

Advantages of Python:
→ Easy to write and understand
→ Huge Library available
→ Less code to solve a problem.

To install Python, you can:
go to the python website
click the "Download Python" button.
Select the latest version (add python to the path)
Select the "windows installer" option
click the "Download" button
Run the Installer
Follow the on-screen instructions

Check the installation:-
Open CMD and types:
python --version

How to Execute the Python code:-
Using interactive mode: In >>> Prompt we can type individual code.
print("Hello world!")

Script mode: Here we can write a program using some editor Notepad and then we can execute the program. from commands Prompt. code is saved with .py extension

Date
Page No.

steps: Open IDLE → New File → write code → save → Run (F5)

using IDE: Here we can write a program, save, debug and execute the programs.

Python Identifiers.

Python identifiers refer to a name used to identify a variable, function, module, class, module or other objects.

There are few rules to follow while naming the python variable.

→ A variable name must start with either an English letter or underscore (_).

→ A variable name cannot start with the number.

→ Special characters are not allowed in the variable name.

→ The variable's name is case sensitive.

→ Identifier name must not be similar to any keywords defined in the language.

Examples of valid identifiers: a123, _n, n_9, etc.

Examples of invalid identifiers: 1a, n=4, n 9, if, etc.

Let's understand the following examples:-

```
num=10
print(num) #10
-a=100
Print (a) #100
X_Y=1000
print(X_Y) #1000
```

Date
Page No.

Python keywords :-

Python keywords are special reserved words that convey a special meaning to the compiler/interpreter.

```
>>> import keyword
>>> print (keyword.kwlist)
>>> print (len(keyword.kwlist))   # 35
```

In Python-2 keywords are 31 and in Python 3.3 it is 33. Latest is 35.
Python Latest Version is 3.14.0

Python Lines and indentation :-

(Basic syntax)

There is no use of curly braces ({}) or semicolon(;) in Python programming language. But Python uses the indentation to define a block of code.

For ex :-

```
def func():
    Statement 1
    Statement 2
    ---------
    Statement n
```

EX-1 :-

```
lst1 = [1,2,3,4,5]
for i in lst1:
    print (i)
    if i==4:
        break
print ("End of for loop")
```

output :-

```
1
2
3
4
End of for loop
```

Date
Page No.

multiline statement in python:

In python, a statement is normally written in one line, but sometimes it becomes too long. Python allows you to continue a statement across multiple lines.
they can be created using the backslash (\), or automatically inside parentheses (), brackets [], braces {} or using triple-quoted strings.

→ 1. using Backslash (\) – Explicit Line continuation.
You can break a long statement using a backslash:

```
total = 10+20+30+\
        40+50
```

→ 2. using Parentheses – Implicit Line continuation
If you use (), [] or {}, python automatically allows multiple lines without a backslash. using parenthesis:

```
result = (
        10 + 20+
        30+ 40
        )
```

using lists :-

```
numbers = [
        10,
        20,
        30,
        40
        ]
```

using dictionary :-

```
Student = {
    "name": "Rajesh",
    "age": 21
    "course": "Python"
}
```

→ 3. Multiline strings (Triple Quotes) :-

```
msg = """ This is
    a multiline
    string in python. """
```

Date
Page No.

=>4. Using semicolon(;) to write multiple statements in one line
(Not recommended, but allowed.)
a=10; b=20; Print(a+b)

Comments in Python
Comments are essential for defining the code and help us and other to understand the code.
Python provides the facility to write comments in two ways - single line comment and multi-line comment.
Single-Line comment :-
Single-Line comment starts with the hash # character followed by text for further explanation.
#defining the marks of a student marks=90
we can also write a comment next to a code statement. Consider the following examples.

Name="Rajesh" #the name of a student is Rajesh
marks=90 #defining student's marks
Branch="Computer Science" #defining student branch

Multi-Line comments:-
we can use """ character to the multiple lines.
For example:-
we can also use another way,
""" """
This is an example
of multi-line comment
using triple-quotes
""" """

Date
Page No.

variable & Assigning values to variables.

Variable: A name that is used to store a value in memory.

Assignment: The process of giving a value to a variable using the = operator.

Python supports simple assignment a=10

multiple assignment:-

a,b,c=10,20,30

same value to multiple variables:-

x=y=z=100

compound assignment:-

x=5

x+=3 # x=x+3

x*=2 # x=x*2

Rules for Python variable Names:-

→ must start with a letter or underscore(_)

→ cannot start with a number

→ can contain letters, digits and underscore

→ case sensitive (age and Age are different)

→ Cannot be a Python keyword (if, while, for, etc)

valid:- name, age2, _total, marks_1

Invalid:- 2name, total $, for

Date
Page No.

the datatype of a variable can be changed:
Ex:-

```
a=25
print(a, type(a))
a="hello"
print(a, type(a))
a=123.456
print(a, type(a))
```

So, the datatype of a variable is purely based on the assigned data to it.

Object References:-

How the Python interpreter works when we declare a variable. The process of treating variables is somewhat different from many other programming languages. In Python, variables are a symbolic name that is a reference or pointer to an object. The variables are used to denote objects by that name.
Let's understand the following example:

```
a=50
```

a → [50]

In the above image, the variable a refers to an integer object.
Suppose we assign the integer value 50 to a new variable b.

```
a=50
b=a
```

a → [50] ← b

The variable b refers to the same object that a points to because Python does not create another object.

Date
Page No.

Let's assign the new value to b. Now both variables will refer to the different objects.

a = 50
b = 100

a → [50] b → [100]

Python manages memory efficiently if we assign the same variable to two different values.

Python variable Types :-

There are two types of variables in Python - Local variable and Global variable. Let's understand the following variables.

Local variables :-

Local variables are the variables that declared inside the function and have scope within the function.

Let's understand the following example.

Ex :-

```
# Declaring a function
def add():
    #Defining local variables. They has scope only within
     a function
    a = 20
    b = 30
    c = a+b
    print("The sum is:", c)

#calling a function
add()
```

output: The sum is 50

Date
Page No.

EX2:-

```
def add():
    #Defining local variables. They has scope only within
    a function
    a=20
    b=30
    c=a+b
    print("The sum is:",c)

#calling a function
add()
print(a+b) # it is invalid as a, b are local for
function add()
```

Global variable:-

Global variables can be used throughout the program, and its scope is in the entire program.
A variable declared outside the function is the global variable by default.
Python provides the global keyword to use global variable inside the function. If we don't use the global keyword, the function treats it as a local variable. Let's understand the following example.

```
count=0 # Global variable
def update-count():
    global count #this is global variable.
    #count=10 this is local variable
    count=count+1 #modifying global variable.
    print("Inside function, count=", count)
update-count()
print("outside function, count=", count)
```

Date
Page No.

```
update_count()
print("outside function, count=", count)
```

Delete a variable:-

We can delete the variable using the del keyword. The syntax is given below.

Syntax - del <variable_name>

In the following example, we create a variable x and assign value to it. We deleted variable x, and print it, we get the error "variable x is not defined"

EX:-

```
#Assigning a value to x
x = 6
print(x)
#deleting a variable.
del x
print(x)
```

Note: First print statement executed and we get output 6. After that del is executed and the when again print statement executes the following error is displayed.

output: 6

```
Traceback (most recent call last):
  File "C:/Users/DEVANSH SHARMA/PycharmProjects/Hello/
    multiProcessing.py", line 389, in
    print(x)
NameError: name 'x' is not defined
```

Date
Page No.

Datatypes :-

In Python the variable declaration is not necessary. As and when a variable is required, we can use an Identifier. So, when we write an Identifier and assign a data then according to the type of data the datatype of the variable will be decided.

EX:

```
x = 250
print(type(x))
```

output : Int

data types are 7 types :-

Numbers
String
List
TUPLE
Dictionary
Set
Boolean

Numbers type :-

Python allows int, float, long, complex

* Here we can store a large integer value as per the memory available.
* float type supports by default up to 15 decimal points.
* long supports up to Python 2.X and after that it is merged with Int type.

Example : Assignment of data

```
x = 10; //single assignment
x = y = z = 40; //multiple assignments
x, y, z = 20, 30, 40; //multiple assignment with different values.
```

Date
Page No.

x = 5+7j
print (type (x))
o/p : <class, 'complex'>

Example :-
x = 1 #int
y = 2.8 #float
z = 1j #complex

Example :-
print (type (x))
print (type (y))
print (type (z))

Example
Integers :-
x = 1
y = 35655222548887771
z = -3255522
print (type (x))
print (type (y))
print (type (z))

Example
Floats :
x = 1.10
y = 1.0
z = -35.59
print (type (x))
print (type (y))
print (type (z))

Example :-
Complex :
x = 3+5j
y = 5j
z = -5j
print (type (x))
print (type (y))
print (type (z))

~~Strings~~

Date
Page No.

String Type:-

Python has a set of built-in methods that you can use on strings.
Strings can be created by enclosing characters inside a single quote or double-quotes. Even triple quotes can be used in Python but generally used to represent multiline strings and docstrings.

```
#defining strings in Python
#all of the following are equivalent
my_string = 'Hello'
print (my_string)

my_string = "Hello"
print (my_string)

my_string = '''Hello'''
print (my_string)

#triple quotes string can extend multiple lines.
my_string = """Hello, welcome to the world of Python"""
print (my_string)
```

String slicing :-

It allows to interact with subset of characters with slice operator [] and :

Ex1:
```
>>> str = "hello world"
>>> print (str)
```
o/p: hello world

Ex2:
```
>>>print (str[0])
```
o/p = 0

Ex3:
```
>>> print (str[2:7])
```
o/p = llo w

//here it prints the characters from index 2 to less than index 7.

Date
Page No.

Example:-
The len() function returns the length of a string:
a = "Hello, world!"
print(len a)
o/p : 13

Example:-
Get the characters from the start to position 5 (not included)
b = "Hello, world!"
print(b[:5])
o/p : Hello

Example:-
Get the characters from position 2, and all the way to the end:
b = "Hello, world!"
print(b[2:])
o/p : llo, world!

Example: Use negative indexes to start the slice from the end of the string:
a = "Hello, world!"
print(a[:-1])
o/p : !dlrow, olleH

EX:-
b = "Hello friends"
print(b[::-2])
o/p : sndfolH

EX:-
b = "Hello, world!"
print(b[2:-2])
o/p = orld

EX:-
The upper() method returns the string in upper case
a = "Hello, world!"
print(a.upper())

Date
Page No.

EX:-
The lower() method returns the string in lower case:
a = "Hello, world!"
print(a.lower())

EX:-
The strip() method removes any whitespace from the beginning or the end:
a = " Hello, world! "
print(a.strip()) #returns "Hello, world!"

EX:-
The replace() method replaces a string with another string:
a = "Hello, world!"
print(a.replace("H", "J"))

EX:-
The split() method splits the string into substrings if it finds instances of the separator:
a = "Hello, world!"
print(a.split(",")) #returns ['Hello', 'world!']

Ex:-
Merge variable a with variable b into variable c:
a = "Hello"
b = "world"
c = a+b
Print(c)

Date
Page No.

Ex:-
x = "python"
x = x.capitalize()
Print(x)
o/p: Python

Ex:-
x = "python"
x = x.upper()
Print(x)
o/p: PYTHON

String Format:-
But we can combine strings and numbers by using the format() method!. The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

Ex:-
Use the format() method to insert numbers into strings:
age = 36
txt = "My name is John, and I am {}"
Print(txt.format(age))

Ex:-
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
Print(myorder.format(quantity, itemno, price))

* with string we can use + and * operators + is used for concatenation and * is used for ~~concatenation~~ repetition.

Ex 3:
>>> a = "Manohar"
>>> b = "Singh"
>>> print(a+b)
o/p: ManoharSingh

Ex 4):-
>>> print(a*3)
o/p: ManoharManoharManohar

Date
Page No.

EX:-

```
a="manohar"
Print(len a)
a.upper()
```

EX:-

```
#index must be in range
my_string="hello"
>>>Print(my_string[15])
...
IndexError: string index out of range.
```

EX:-

```
x="hello"
Print(x[20:])
```

o/p: NO OUTPUT as out of range

EX:-

```
#index must be an integer
>>> my_string[1.5]
...
TypeError: String indices must be integers
```

Example:-

Loop through the letters in the word "banana"

```
for x in "banana":
    Print(x)
```

o/p:

b
a
n
a
n
a

Date
Page No.

Example:
check if "free" is present in the following text:
txt = "The best things in life are free!"
print("free" in txt)
o/p: True

Example:-
Print only if "free" is present:
txt = "The best things in life are free!"
if "free" in txt:
    print("Yes, 'free' is present.")
o/p: Yes, 'free' is present.

EX:-
#Iterating through a string
count = 0
for letter in 'Hello world':
    if (letter == 'l'):
        count += 1
Print(count, 'letters found')

In Python the strings can be manipulated with index positions or subscripts.

EX:-
>>> word = "Python"
>>> word[-1]
o/p: 'n'

EX:-
>>> word[0]
o/p: 'P'
we can even interact with a range of characters:
>>> word[2:4]
o/p: 'tho'
*In string type in the range specification, the starting and ending position is optional.

Date
Page No.

Ex:-
>>>word[:2]
o/p: 'Py'

Ex:-
>>>word[2:]
o/p: 'thon'

Ex:-
>>>word[:2]+word[2:]
o/p: 'Python'

Ex:-
>>> word[:]
'Python'
As string is immutable so we cannot modify the content. If we do then it shows error as:
Ex:-
>>> word = 'Python'
>>> word[0] = 'q'
TypeError: 'str' object does not support item assignment

5.12.25

>List1 = [1,4,8,16,25]
>List1.append(36)
>List1
o/p: [1,4,8,16,25,36]

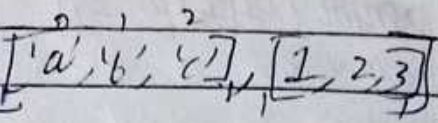
>List1.append(7**2)
>List1
o/p: [1,4,8,16,25,36,49]

>List1[3:5] = ['D','E']
>List1
o/p: [1,4,8,'D','E',36,49]

Date
Page No.

Nesting of list type:

```
>list1 = ['a', 'b', 'c']
>list2 = [1,2,3]
>list3 = [list1, list2]
>list3[0][1]
o/p: 'b'
```



Concatenation :-

```
>list1 = [1, 4, 8, 16, 25]
>list2 = [36, 49, 64, 81, 100]
>list3 = list1 + list2
>list3
o/p: [1, 4, 8, 16, 25, 36, 49, 64, 81, 100]
```

change the content of a list :-

```
>list1 = [1, 4, 8, 16, 25]
>list1[3] = 999
>list1
o/p: [1, 4, 8, 999, 25]
```

```
>list1 ['a', 'b', 'c', 'd', 'e', 'f']
>list1[2] = 'C'
>list1
o/p: ['a', 'b', 'C', 'd', 'e', 'f']
```

Datatype of list, l1 :-

```
l1 = ['a', "murali", 10, 20, 1.23]
>type(l1)
o/p: <class 'list'>
>type(l1[2])
o/p: <class 'int'>
>type(l1[0])
o/p: <class 'str'>
```

Date
Page No.

>Print (l1[1][0:2])
o/p: mv

>Print (l1[1][0:2:1])
o/p: mv

>List1 = [1, 2, 3, 4]
>List1 *3
o/p: [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4] 4, 2, 4, 2

>[illegible]
>Print(l1[0])
o/p: a
>Print(l1[2])
o/p: lo
>Print(l1[1])
o/p: Murali

Negative indexing:-
my-list = ['P', 'r', 'o', 'b', 'e']
Print(my-list[-1])
o/p: e

Print(my-list[-5])
o/p: P

>fruits = ['apple', 'banana', 'gua', 'orange', 'banana']
>fruits.count('banana')
o/p: 2

>fruits.insert(2, 'cherry')
>fruits
o/p: ['apple', 'banana', 'cherry', 'gua', 'orange', 'banana']

Date
Page No.

> fruits.append('xyz')
> fruits.
o/p: ['apple', 'banana', 'cherry', 'gva', 'orange', 'banana', 'xyz']

> fruits.remove('cherry')
> fruits.
o/p: ['apple', 'banana', 'gva', 'orange', 'banana', 'xyz']

> fruits.sort()
> fruits.
o/p: ['apple', 'banana', 'banana', 'gva', 'orange', 'xyz']

> fruits.reverse()
> fruits.
o/p: ['xyz', 'orange', 'gva', 'banana', 'banana', 'apple']

> x = fruits.pop()
> print(x)
o/p: ['xyz', 'orange', 'gva', 'banana', 'banana']

Sort the list descending:
> thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
> thislist.sort(reverse = True)
> Print (thislist)
o/p: ["Pineapple", "orange", "mango", "kiwi", "banana"]

Creating a tuple:-
> thistuple = ("apple", "banana", "cherry")
> Print (thistuple)
o/p: ("apple", "banana", "cherry")

list [start : end-1 : jump]

Date
Page No.

Length :-
To determine how many items a tuple has, use the len() function.
EX:-

>thistuple = ("apple", "banana", "cherry")
>print (len (thistuple))
o/p : 3

Data type :-
>print (type (mytuple))
o/p : <class 'tuple'>

print the last item of the tuple :-
>thistuple = ("apple", "banana", "cherry")
>print (thistuple[-1])
o/p : "cherry"

Display 3, 4, 5 th item :-
>thistuple = ("apple", "banana", "cherry", "orange",
"kiwi", "melon", "mango")
>print (thistuple[3:6])
o/p : ["orange", "kiwi", "melon")

Display beginning 4th position/index :-
>print (thistuple[:4])
o/p : ["apple", "banana", "cherry", "orange"]

Display from "cherry" to the end :-
>print (thistuple[2:])
o/p :

print (thistuple[-4:-1])
o/p : ["orange", "kiwi", "melon"]

(4) 3 2 1 — excluded.
-1 -1 = -2

Date
Page No.

Removing element from tuple:-
> thistuple = ("apple", "banana", "cherry")
> y = list(thistuple)
> y.remove("apple")
> thistuple = tuple(y)
> print(thistuple)
o/p :- ("banana", "cherry")

Del the tuple completely:-
> del thistuple

Dictionary:-
> thisdict = { "brand": "Ford",
"model": "mustang",
"year": 1964 }
> print(thisdict)
o/p : { "brand": "Ford", "model": "mustang", "year": 1964 }

keys():-
> x = thisdict.keys()
o/p : dict_keys(["brand", "model", "year"])

values():-
> y = thisdict.values()
o/p : dict_values(["Ford", "mustang", 1964])

Inserting values :-
> thisdict["color"] = "Blue"
> print(thisdict.keys())
o/p : dict_keys(["brand", "model", "year", "color"])

```
for i in dict:
    dict[i]
```

Date
Page No.

Items :-

```
> x = thisdict.items()
> Print(x)
o/p : dict_items([('brand', 'Ford'), ('model', 'mustang'),
      ('year', 1964), ('color', 'Blue')])
```

Copy :-

```
> mydict = thisdict.copy()
> Print(mydict)
o/p : {"brand": "Ford",
       "model": "mustang",
       "year": 1964,
       "color": "Blue"}
```

Type conversion in Python :-

```
x = '1010'
c = int(x,2)   # converting binary to integer
Print(c)       # int(value, Base)
o/p : 18
```

sizeof Datatype :-

```
import sys
sys.getsizeof(int())
              (float())
              (str())
              (list())
              (set())
              (tuple())
              (dict())
```

Date
Page No.

1 Program to find the size of datatype :-

```
import sys
print("size of integer :", sys.getsizeof(int()))
```

q. 1 - 25

1. Create a list with different datatype and print
2. access the 3rd element of the list
3. Add elements to a list.
4. Insert element at a specific position
5. Remove an item from a list by using remove method.
6. Copy a list
7. concaterate 2 list and print

Ans

```
l = ["Rajesh Rana", 21, 8.82]
print("Elements in the list are :- ")
```

8. count the occurance of an element
9. convert list to tuple
10. sort a list

Ans

①
```
l = ["Rajesh Rana", 21, 8.82]
print("Elements in the list are :- ")
for i in l:
    print(i, end=", ")
```
output :-
Elements in the list are :-
Rajesh Rana, 21, 8.82

②
```
l = ["Rajesh Rana", 21, 8.82]
print("Accessing the 3rd element :- ", l[2])
```
output :- Accessing the 3rd element :- 8.82

③
```
l = ["Rajesh Rana", 21, 8.82]
l.append("Bhadrak")
l.append(756100)
print(l)
```
output :-
["Rajesh Rana", 21, 8.82, "Bhadrak", 756100]

Date
Page No.

(4) l = ["Rajesh Rana", 21, 8.82]
l.insert(2, "Bhadrak")
Print(l)
output:-
"Rajesh Rana", 21, "Bhadrak", 8.82

l = ["Rajesh Rana", 21, 8.82]

(5) l.remove(21)
Print(l)
output:- ["Rajesh Rana", 8.82]

6 l = ["Rajesh Rana", 21, 8.82]
l2 = l.copy()
Print(l2)
output :- ["Rajesh Rana", 21, 8.82]

7 l1 = ["Rajesh Rana"]
l2 = [21, 8.82]
l3 = l1 + l2
Print(l3)
output :- ["Rajesh Rana", 21, 8.82]

8 l1 = [1, 2, 1, 3, 4, 5, 2, 3, 5, 1, 4]
d = { }
for i in l1:
    if i not in d:
        d[i] = 1
    else:
        d[i] = d[i] + 1
Print("The occurance of an elements are :-")
for i in d:
    Print(i, ":", d[i])

Date
Page No.

output :-
1 : 3
2 : 2
3 : 2
4 : 2
5 : 2

9 l = ["Rajesh Rana", 21, 8.82]
t = tuple(l)
print(t)
output :- ("Rajesh Rana", 21, 8.82)

~~l = "Rajesh Rana"~~

10 l = [1, 3, 4, 10, 2, 5]
print(l.sort())
output :- [1, 2, 3, 4, 5, 10]

unit - 2

statements are of 3 types :-
→ sequential statements
→ selection control statements
→ Loop control statements.

selection control statements :-
if statement :
It can used in 4 ways :
1) if statement,
2) if ... else,
3) elif
4) nested if.