Page No.
Date

College Name:

DELHI GLOBAL INSTITUTE OF TECHNOLOGY

Name: BAZGHA RAZI

Course Code: PCC-CSE-207G

Subject: Python Programming

Session: 2019-2023

Page No.
Date

Ans 1a) List : A list is a collection of items in a particular order.

It is used to store the sequence of various data types in python.

List are mutable data type i.e., we can modify list elements.

The items in a list is separated by a comma (,) and written in between the two square brackets.

Syntax : l = [item1, item2, ---, item3]

Example : l = ["Saurav", 19, "ABC", "DEF"]

Built in functions of List

i) len() → It is used to calculate the length of the list.

Example :
```
l = [1, 2, 3, 4, 5)
print(len(l))
```

Output : 5

ii) append() → It is used to add an element at the end of list.

Example: l = [1, 2, 3, 4, 5]
l.append(6)
print(l)

Output: [1, 2, 3, 4, 5, 6]

iii) count() → It is used to calculates the total occurence of given element of list.

Example: l = [1, 2, 3, 2, 2, 4, 3, 2]
print(l.count(2))

Output: 4

iv) insert(): It is used to insert an element at specified position in list.

Example: l = [1, 2, 4, 5, 6]
l.insert(2, 3)
print(l)

Output: [1, 2, 3, 4, 5, 6]

v) sum(): It is used to calculates sum of all the elements of list.

Example: l = [1, 2, 3, 4, 5, 6]
print (sum(l))

Output: 21

WAP to create, insert and delete element from list

```
l = [1,2,3,4,5]                    # List is created.
print (l)                          # Output: [1, 2, 3, 4, 5]

l.insert(2,6)
print (l)                          # Output: [1, 2, 6, 3, 4, 5]

l.remove(6)
print (l)                          # Output: [1, 2, 3, 4, 5]
```

Page No.
Date

Ans 1c) We can make certain mistake while writing a program that lead to errors when try to run it. In python, an error can be syntax error or logical error.

Syntax Error : Occur when proper syntax is not used in the programs.

Example :
```
a = 10
b = 5
printf("Addition of a & b :", a + b)
```
↳ Syntax error occur because printf is not the correct syntax for printing anything in python.

Logical Error : Occurs at runtime or during the execution of code. It is also known as exception.

Exception

Python has many built-in exceptions which enables our program to output an error message when something in it goes wrong.

When these exceptions occur, it causes the current process to stop and passes it to the calling process until it is handled. If not handled, our program will crash.

Page No.
Date

Python support exception handling using various blocks:

i) try
ii) except
iii) else
iv) finally

→ Try: In this block we will write the code in which exception will generate.

→ Except: When in try block exception occur then the control moves to the exception block. We use multiple except block for multiple errors.

→ Else: If there is no exception in the try block then after executing all the codes of try block control moves to the else block. It is an optional block.

→ Finally: The codes written in this block will always execute whether there is any exception occur in the try block or not.

Try and Except blocks are mandatory in exception handling. But else and finally are the optional blocks.

Page No.
Date

Example:

```
a = [1, 2, "Saurav", 7]
try:
    print("Welcome!")
    print(a[10])
    print("Hello Exception Handling")
except:
    print(a[2])
    print("Exception Caught")
else:
    print("There is no exception")
finally:
    print("I am in finally block")
```

Output:

```
Welcome!
Saurav
Exception Caught
I am in finally block.
```

Page No.
Date

Ans26) Function : It is block of organised, reusable code that is used to perform a single, related action.

Python has many built in functions but we can also created own functions. These types of functions are called user define function.

Syntax : def function_name (parameters):
function_block.
return expression.

Example : def wish_user():
print ("Hello!!")

wish_user()

Output : Hello!!

Arguments in function.

Arguments : These are types of information which can be passed into the function. These are written between the parenthesis. If we have to pass more than one argument then it can be separated by a comma (,).

Types of arguments are as follows:

i) Keyword Argument: It is used to pass the arguments in the random order.

The name of the arguments is treated as the keywords and matched in the function calling. If the same match is found, the values of the arguments are copied in the function definition.

Example1: def function (name, message)
print ("Printing message with", name, "and", message)

function (name = "Saurav", message = "Hello")

Output: Printing message with Saurav and Hello.

Example 2: def function (name, message)
print ("Printing message with", name "and", message)

function (message = "Hello", name = "Saurav")

Output: Printing message with Saurav and Hello.

ii) Required Argument : It is the argument which is required to be passed at the time of function calling with the exact match of ~~either~~ their positions in the function call.

Example : def function (name) :
    message = "Hello" + name.
    return message.

name = input ("Enter the name : ")
print (function (name))

Output : Enter the name : Sourav
Hello Sourav

iii) Default Argument : Python allows us to initialize the arguments at the function definition. If the value of any of the arguments is not provided at the time of function call, then the argument can be initialized with the value given in the definition even if the argument is not specified at the function call.

Page :
Date :

Example: def function (name, age = 23):
print ("My name is ", name,
"and age is ", age)

function (name = "Saurav")

Output: My name is Saurav and age is 23.

iv) Variable - length Argument

Python provides the offer the comma - separated values which are internally treated as tuples at the function call. By using variable length arguments, we can pass any number of argument. It is defined by using (*args)

Example: def function (*names):
print ("Printing all the names:")
for name in names:
print (name)
function ("Saurav", "Meenu", "Bhawna")

Output: Printing all the names:
Saurav
Meenu
Bhawna.

Page :
Date :

WAP to perform linear search of an element in the list.

```
def search(list, n):
    for i in range(len(list)):
        if list[i] == n
            return i+1
    return("Not found")

list = [1, 2, 3, 4, 5, 1, 4, 6]
n = int(input("Element to be Search :"))
print(search(list, n))
```

Output : Element to be Search : 6
8 # Element found at 8 position

Ans 3a) Method Overridding.

In python, method overriding has an ability that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super class or parent class.

Example:

```
class A:
    def n(self):
        print("In class A")

class B(A):
    def n(self):
        print("In class B")

class C(A):
    def n(self):
        print("In class C")
class D(B,C):
    pass

object = D()
object.n()      # Output: In class B
A.n(object)     # Output: In class A
C.n(object)     # Output: In class C
```

Operator Overloading.

It is used to perform specific operations on the given operands. The operation that any particular operator will perform on any predefined data type is already defined in python.

Each operator can be used in a different way for different types of operands.

Example:- '-' operator is used for subtracting two integers or float operands. A single operator has different functions to perform.

Example of Operator Overloading.

```
class demo:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __add__(self, e):
        r = self.a + e.a
        s = self.b + e.b
        print("\n", r, s)
```

```
def __sub__(self, e):
    a = self.a - e.a
    s = self.b - e.b
    print("\n", a, s)

x = demo(6, 4)
y = demo(3, 2)
x + y
x - y
```

Output: 9 6
3 2