Ques ⇒ (4) Discuse if-else statement :-

Ans ⇒ if-else :⇒ if-else statement is used to execute both the true part and the false Part of a given condition. if the condition is true, if block code is executed and if the condition is false, the else block of code is executed.

Syntax :-

```
if (condition):
    statement
else:
    statement
```

Ex:- Program:-

```
num = int(input("Enter a number:"))
if num > 0 :
    Print("num, "is Positive.")
else:
    Print("num, "is Negative.")
```

Output:-

Enter a number.
4
4 is positive.

Ques-(8) String in python:-

Answer-> String:->

String is a sequance data type in python.
"String is a sequance of characters."

for Ex:->
"Hello", etc.

Syntax:-

data_type = "Statement".

Ex:-> Program:->

```
name = "Shivam"
Print(name)
message = "[redacted] is a boy."
Print(messege)
```

output
Shivam
is a boy.

Output :-

Shivam
is a boy.

Section - (B)

(List)

Ques (6) List :-

Answer → List :→ List is a collection of data enclosed in ([]) square bracket and separated with comma.

→ Creating List in python :-

A list can be creat by placing the element separated by comma (,) between the square brackets ([]).

```
# create an empty list.

Empty_List = [ ].

# Create a List of Integers.

Integer_List = [1, 2, 3, 4, 5]

# Create a list of string.

String_List = ["mango", "Banana"]
```

⇒ **Accessing element in list in python:-**

for accessing the ~~list~~ Element in the list. the index is mentioned within the index operator ([]) proceded by the list's Name.

Ex:-

```
my_list = [1, 2, 3, 4]
Print ("Accessing Element :")
Print (my_list)
```

Output:-

```
Accessing Element :
1
2
3
4
5
```

⇒ **Updating List in python:-**

By using assignment operator (=) we can change or update the List Element.

Ex:-

```
mylist = [1, 2, 3, 5]
Print(mylist)
my_list[5] = 4
Print(my_list)
```

Output:-

```
[1, 2, 3, 5]
[1, 2, 3, 4]
```

⇒ Add Element in List in python :-

By using the append(), extend() and Insert() method we can add element in the list.

Ex :⇒

```
Integers = [1,5,6,9]
Print (Integers)
Integer.insert(1,3)
Print(Integers)
```

Output :-

```
[1,5,6,9]
[1,3,5,6,9]
```

⇒ Delete Element from list in Python :-

We can delete one or more elements from the list in python using del Keyword.

Ex :⇒

```
my_list = [1,2,3,4,5,6,7]
Print(my_list)
del my_list[3]
Print(my_list)
```

Output :⇒

```
[1,2,3,4,5,6,7]
[1,2,3,5,6,7]
```

List operations in python :->

(i) Concatination.
(ii) Repeating the element of the list.
(iii) membership check of an element in a list.
(iv) finding the length of a list.
(v) Iterating through the list.

* -> Characteristics of a List in Python:-

(i) Ordered:
(ii) mutable.
(iii) Dynamic.
(iv) Duplicate Element.

(i) Ordered :- List maintain the order in which the data is inserted.
(ii) mutable :-> List can be changed. It means we can modify the list.
(iii) Dynamic :- List can expend or shrink automatically to accommodate the item accordingly.
(iv) Duplicate Element :- List allows us to store duplicate data.

Question (7) Write about Loops in python :-

Ans :) **Loops :)**

Looping means repeating a condition over and over until a particular condition is satisfide.

Types of loop in Python :)

There are three type of loop in python :-

(i) while loop.
(ii) for loop.
(iii) Nested loop.

(i) **while loop :)**

while loop is used to execute a block of statement repeatedly until a given condition is satisfied.

Syndax :)

```
while Expression:
    Statement(s)
```

ex :)

Program :)

```
Count = 0
while (count < 3):
    Count = count + 1
    Print(count)
```

output :-

0
1
2

(ii) for loop :->

The for loop is used to repeat a specific block of code to given number of time.

Syntax :->

```
for variable_Name in sequence:
    body of loop.
```

Ex :-> Program :->

```
n = 4
for i in range(0, n):
    Print(i)
```

Output :->

```
0
1
2
3
```

(iii) Nested Loop :->

A loop inside another loop are called nested loop.

Syntax :->

```
for variable_name in sequence:
    for variable_name in sequence:
        Statement(s)
    Statement (s)
```

Ex :→ Program :→

```
for i in range(1,5):
    for j in range(i):
        Print(i, end=' ')
    Print()
```

Output :→

```
1
2 2
3 3 3
4 4 4 4
```

Question → (10) Program to ~~[illegible]~~ Print Sum of n natural number in Python :-

Answer :→ Program :→

```
num= int(input("Enter a number:"))
if num <0:
    Print("Please Enter a positive num:")
else:
    Sum = 0
while(num >0):
    Sum += num
    num -= 1
Print("The result is:", Sum)
```

Input / Output :→

```
Enter a number : 15
The result is : 120
```

Imp. topic

Python :- Dictionary

(i) Operators :-> Already Done P- 04 to 8

(ii) List :-> Alread Done Page :- 11 to 14.

(iii) Dictionary :-> Dictionary

A ~~python~~ is a unordered collection of data. It does not allow duplication. Each element in the dictionary is in the form of "key: value" pairs. Dictionary element should be enclosed with {} and key: value pair seperated by commas.

Syntax :->

Dictionary_Name = {'key': 'value'}

Example :->

Courses = { "Course": "Python", "fee": 4000, "Duration": "30 days"}
Print (Course)

Output :->

{'Course': 'Python',
'fee' : 4000 ,
'Duration': '30 days'}

(P.T.O)

(i) Create Dictionary in Python :-

We can create dictionary with key : value pair, by using {} and seperated the value by comma.

Syntax :-

Dictionary_Name = { " key" : "value" }

Example :-

Course = { " Course" : "Python", "fee" : 4000 }

Output :- Print ('course', 'fee')

{'Course' : 'Python', 'fee' : 4000}

(ii) Accessing element in Dictionary :-

We can access the value of the dictionary using key.

for example :-

my-dict = { 'Course' : 'Python', 'fee' : 4000, }

Print (my-dict [' course'])

Print (my_dict [' fee'])

Output :-

Python

4000.

(P.T.O)

(iii) Add Element in Dictionary :->

Dictionary is mutable which means we ~~we~~ can add, delete their key : value pair.

for example :->

```
my_dict = {'Course': 'Python', 'fee': 4000}
# Adding Element.
my_dict['duration'] = '60 days'
print (my_dict)
```

Output :->

```
{'Course': 'Python', 'fee': 4000, 'Duration': '60 days'}
```

(iv) Update Element in Dictionary :-

Dictionary is mutable so we can update it.

for example :->

```
my_dict = {'Course': 'Python', 'fee': 4000,
           'Duration': '60 days'}
my_dict['duration'] = '40 days'
print (my_dict)
```

Output :->

```
{'Course': 'Python', 'fee': 4000, 'Duration': '45 days'}
```

(v) Deleting element in Dictionary :->

we can delete element from the dictionary.

for Example :->

```
my-dict={'Course': 'Python', 'fee': 4000,
         'Duration': '60 days'}
Print(my-dict)
 del my-dict['Duration']
Print(my-dict)
```

Output :=>

```
{'Course': 'Python', 'fee': 4000, 'Duration':
    '60 days'}

{'Course': 'Python', 'fee': 4000}
```

String

(iv) String :=>

String is a sequence data type in Python.
"String is a seqvance of characters."

Syntax :-

~~data type~~
var = " Statement"

Example :=>

```
Name = "Shivam maurya"
Print (name)
```

Output :=>

Shivam maurya.

(i) Assigne a string in python :->

```
a = 'Shivam'
b = "maurya"
Print (a)
Print (b)
```

Output :=>

Shivam
maurya

(ii) Access the element of a string :-

Characters in a string can be accessed using their index number.

Syntax :- var_name[index]

Example :-

```
a = 'shivam'
Print(a[0])
Print(a[1])
Print(a[2])
Print(a[3])
Print(a[4])
Print(a[5])
```

Output :-

```
s
h
i
v
a
m
```

(iii) Add the element into string :-

we can add string / char by using Join()

Ex :-

```
a = "shivam"
b = " maurya"
Print(" ".join([a, b])
```

Output ⇒ shivam maurya.

(iv) Removing / Replacing element from string :-

we can delete string / char by using replace() method.

Tuple

Example:-

```
name = "Shivam! maurya!"
my_name = name.replace("!", "")
print(my_name)
```

Output:- shivam maurya.

Tuple:

(v) Tuple:-> Tuple is an ordered collection of python object.
Tuple is sequencial data type in python.

Syntax:->

```
var_name = ('val1', 'val2' ---)
```

Example:->

```
Tuple = ('appli', 'mango', 'orange')
Print(Tuple).
```

Output:->

('apple', 'mango', 'orange')

(1) Create a Tuple in python:->
we can create a tuple by using "( )" and separated the value using comma.

Syntax:-> Tuple_Name = ('value1, 'value2')

Example :- fruit = ('Apple', 'mango', 'Orange')
Print (fruit)

Output :- ('Apple', 'mango', 'Orange')

(ii) Accessing element in tuple :-
we can access the element of tuple with the help of index.

Example :-
fruit = ('Apple', 'mango', 'Orange')
print(fruit[0])

Output :-
('Apple')

(iii) Adding a new element in tuple :-
By using Appened() we can add a new element into tuple.

Example :-
fruit = ('Apple', 'mango', 'Orange')
fruit[3] . appened ('Grapes')
Print (fruit)

Output :-

('Apple', 'mango', 'Orange', 'Grapes')

(iv) Deleting item from tuple :->

we can not delete item from the Tuple because it is immutable.

But we can delete Whole Tuple By using del.

Example :->

```
fruit = ('Apple', 'mango', 'Orange')
Print (fruit)
del fruit
Print (fruit)
```

Output :->

```
('Apple', 'mango', 'Orange')
NameError : name 'fruit' is not defined.
```

Set :

3) Set :-

Set is an unordered collection of data that is iterable mutable and has no duplicate element.

① Creating a set :->

we can create sets by placing all the element inside curly braces ({}), separated by comma.

A set can have any number of items (integer, float, tuple, string, etc)

Ex :->

```
# Creat an empty set.

empty_set = set()

# Creat an empty dictionary

Empty_dictionary = { }
# check the data type of Empty_set.
print ('Data type', type (Empty_set))
# check the data type of Dict_set
print ('Data type ', type (Dictionary_set))
```

Output :-

```
Data type <class 'set'>
Data type <class 'dict'>.
```

(ii) Duplicate items in a set :-

set does not allows duplicate element because it is muteable.

Ex :-

```
numbers = { 2, 4, 6, 6, 2, 8 }
Print (numbers)
```

Output :-> { 2, 4, 6, 8 }

Here we can see that there is no duplicate element in the set.

(iii) Add item in set :->

In python, We can use the add() method to add item in a set.

Ex :-

```
num = { 2, 4, 5, 10 }
print ('Before adding item', number)
# using add () method
number.add(6)
Print('After adding item', number)
```

Output :-

Before adding item: { 2, 4, 5, 10 }
After adding item : { 6, 2, 4, 5, 10 }

(iv) update item in set :→

By using update() method we can update set.

Ex :→

```
numbers = { 2, 4, 6, 8, 10}
my_num = [1, 3, 5, 7, 9]
numbers.update(my_numbers)
print(numbers)
```

Output :→

{1, 3, 5, 7, 9, 2, 4, 6, 8, 10}

(v) Remove item from set :—

We can remove items from set by using discard() method.

Ex :→

```
numbers = {2, 3, 5, 6, 4}
Print('Initial set:', numbers)
# remove '2' from set
removeValue = numbers.discard(2)
Print('Set after removing items:',
        numbers)
```

Output :→

{3, 5, 6, 4}