Programming:

1. **Write a program to remove duplicates in list?**

```
l=[11,22,44,55,66,11,11,45,45,78,99,45]
l2=[]
for i in l:
    if i not in l2:
        l2.append(i)
print(l2)
```

Output :-

[11,22,44,55,66,45,78,99]

2. **Write a program to remove duplicates in string?**

```
s='hi hello how'
s1=''
for i in s:
    if i not in s1:
        s1+=i
print(s1)
```

Output:-

hai elow

3. **Write a program to print most repeated character in string?**

```
s='hai python'
d={}
for i in s:
        if i not in d:
        d[i]=1
        else:
        d[i]+=1
```

```
A=max(d.values())
for s in d:
        if d[s]==A:
        print(s)
```
Output:-

h

4. **Write a program to print most repeated word in sentence?**

```
s='hai python hai hello hai hello'
l=s.split()
d={}
for i in l:
   if i not in d:
      d[i]=1
   else:
      d[i]+=1
A=max(d.values())
for s in d:
   if d[s]==A:
      print(s)
```

Output:-

hai

5. **Write a program to print most repeated element in list?**

```
l=[11,22,33,44,11,55,66]
d={}
for i in l:
   if i not in d:
      d[i]=1
   else:
      d[i]+=1
A=max(d.values())
```

```
for s in d:
   if d[s]==A:
      print(s)
```
Output:-
11

6. **Write a program to print lengthiest word in sentence?**

```
s='hai python hai hello hai hello '
l=s.split()
d={}
for i in l:
        if i not in d:
        d[i]=len(i)
A=max(d.values())
for s in d:
        if d[s]==A:
        print(s)
```

Output:-

python

Another way

```
s='hai hello how python'
l=s.split()
max=l[0]
for i in l:
   len(i)>len(max)
   max=i
print(max)
```

Output :-
python

Another way

```
s='hai hello python'
a=s.split()
print(max(a,key=len))
```

Output:-
python

7. **Prime number program**

using for loop

```
n=int(input('enter a number'))
if n>1:
        for i in range(2,n):
        if n%i==0:
        print('not prime')
        break
        else:
        print('prime')
else:
        print('not prime')
```

Output:-
Enter a number 11
prime

using while loop

```
n=int(input('enter a number'))
a=2
if n>1:
        while a<n:
        if n%a==0:
        print('not prime')
        break
        a+=1
```

```
        else:
        print('prime')
else:
        print('not prime')
```

Output:-
Enter a number 5
prime

8. **Program to swap two values**

swapping the second and the last value

```
l=[11,22,33,44,55]
l[1],l[-1]=l[-1],l[1]
print(l)
```

Output:-

[11, 55, 33, 44, 22]

swapping the first and the last value

```
l=[11,22,33,44,55]
l[0],l[-1]=l[-1],l[0]
print(l)
```

Output:-

[55, 22, 33, 44, 11]

swapping the fourth and the first value

```
l=[11,22,33,44,55]
l[3],l[-5]=l[-5],l[3]
print(l)
```

Output:-

[44, 22, 33, 11, 55]

9. **Factorial and Fibonacci with recursion and without recursion**

Factorial program using without recursion

using for loop

```
n=int(input('enter a number'))
fact=1
for i in range(1,n+1):
        fact*=i
print(fact)
```

Output:-
Enter a number 5
120

using while loop

```
n=int(input('enter a number'))
fact=1
a=1
while a<n+1:
        fact*=a
        a+=1
print(fact)
```

Output:-
Enter a number 5
120

Fibonacci program using without recursion

```
def fibo(n,a,b):
        if n==1:
```

```
        print(a)
        elif n==2:
        print(a,b)
        else:
        print(a,b)
        for i in range(2,n):
        c=a+b
        print(c)
        a,b=b,c
fibo(10,2,3)
```

Output:-

```
2 3
5
8
13
21
34
55
89
144
```

using for loop

```
n=int(input('enter a number'))
a=int(input('enter a number'))
b=int(input('enter a number'))
for i in range(2,n):
        if n==1:
        print(a)
        elif n==2:
        print(a,b)
        else:
        print(a,b)
        c=a+b
        print(c)
```

```
        a,b=b,c
```
Output:-

```
enter a number10
enter a number2
enter a number3
2 3
5
3 5
8
5 8
13
8 13
21
13 21
34
21 34
55
34 55
89
55 89
144
```

Factorial program using with recursion

```
def fact(n):
    if n==0:
        return 1
    else:
        return n*fact(n-1)
num=5
for i in range(1,num+1):
    x=fact(i)
         print(x)
```

Output:-

1
2
6
24
120

For single factorial number

```
def fact(n):
    if n==0:
        return 1
    else:
        return n*fact(n-1)
num=5
x=fact(num)
print(x)
```

Output:-

120

Fibonacci with recursion

```
def fibo(n):
        if n==1 or n==2:
        return n-1
        else:
        return fibo(n-1)+fibo(n-2)
num=10
for i in range(1,num+1):
        x=fibo(i)
        print(x)
```

Output:-

0
1

1
2
3
5
8
13
21
34

Fibonacci number using single fibonacci number

```
def fibo(n):
    if n==1 or n==2:
        return n-1
    else:
        return fibo(n-1)+fibo(n-2)
num=10
x=fibo(num)
print(x)
```

Output:-

34

10.Input:- we are the programmers Output:-programmers the are we

```
s='we are the programers'
l=s.split( )
for i in range(-1,-len(l)-1,-1):
    print(l[i],end=' ')
```

Output:-

programers the are we

Another method

```
s='we are the programmers'
l=s.split( )
rev=l[::-1]
print(' '.join(rev))
```

Output:-

programers the are we

11. Input:- we are the programmers Output:-ew era eht sremmargorp

```
s='we are the programmers'
l=s.split( )
l1=[ ]
for i in l:
    rev=i[::-1]
    l1.append(rev)
print(' '.join(l1))
```

Output:-

ew era eht sremmargorp

12. Write a program to check whether the given number is palindrome or not?

using while loop

```
n=int(input('enter a number'))
rev=0
num=n
while n>0:
    rem=n%10
    rev=rev*10+rem
    n=n//10
if num==rev:
    print('palindrome number')
```

```
else:
    print('not palindrome number')
```

Output:-

Enter a number 11

palindrome

Another method

using type casting

```
n=int(input('enter a number'))
l=str(n)
if l==l[::-1]:
    print('palindrome number')
else:
    print('not palindrome number')
```

Output:-

Enter a number 11

palindrome

13. Write a program to check whether the given string is palindrome or not?

```
s=input('enter a string')
rev=s[::-1]
if rev==s:
    print('palindrome')
else:
    print('not palindrome')
```

Output:-

```
enter a string malayalam
palindrome

enter a string man
not palindrome
```

14. Write a program to print highest value in list?

```
l=[11,22,66,99,88,77]
max=l[0]
for i in l:
    if i>max:
        max=i
print(max)
```

Output:-

```
99
```

Another method

```
l=[11,88,55,98,78,54]
l.sort()
print(l[-1])
```

Output:-

```
98
```

By using bubble sorting technique

```
l=[10,55,18,9,2,5,12]
n=len(l)
for i in range(0,n-1):
    for j in range(0,n-1-i):
        if l[j]>l[j+1]:
```

```
            l[j],l[j+1]=l[j+1],l[j]
print(l[-1])
```

Output:-

55

15. Write a program to print second highest value in list?

```
l=[11,88,99,55,98,78,54]
l.sort()
print(l[-2])
```

Output:-
98
Another method

```
l=[11,88,99,55,98,78,54]
n=len(l)
for i in range(0,n-1):
        for j in range(0,n-1-i):
        if l[j]>l[j+1]:
          l[j],l[j+1]=l[j+1],l[j]
print(l[-2])
```

Output:-
98

16. Write a program to print least value in list?

```
l=[11,88,99,55,98,78,54]
l.sort()
print(l[0])
```

Output:-
11
Another method

```
l=[11,22,66,99,88,77]
min=l[0]
for i in l:
        if i<min:
        min=i
print(min)
```

Output:-

11

17. Sort the given array without using built-in methods?

```
l=[10,55,18,9,2,5,12]
n=len(l)
for i in range(0,n-1):
        for j in range(0,n-1-i):
        if l[j]>l[j+1]:
          l[j],l[j+1]=l[j+1],l[j]
print(l)
```

Output:-

[2, 5, 9, 10, 12, 18, 55]

18. Write a program to swap first and last element in given list?

```
l=[11,22,66,99,88,77]
l[0],l[-1]=l[-1],l[0]
print(l)
```

Output:-

[77, 22, 66, 99, 88, 11]

19. Input:- aaabbcccdaabbb Output:-3a2b3c1d2a3b

```
s='aaabbcccdaabbb'
new=''
c=1
for i in range(1,len(s)):
    if s[i]==s[i-1]:
        c+=1
    else:
        new=new+str(c)+s[i-1]
        c=1
    if i==len(s)-1:
        new=new+str(c)+s[i-1]
print(new)
```

Output:-

3a2b3c1d2a3b

20. Input:- ['kumar', 'somu', 'sonu', 'ram', 'raju'], Output:- {'k': 1, 's': 2, 'r': 2}

```
l=['kumar', 'somu', 'sonu', 'ram', 'raju']
l1=[]
for i in l:
        l1.append(i[0])
d={}
for i in l1:
        if i not in d:
        d[i]=1
        else:
        d[i]+=1
print(d)
```

Output:-

{'k': 1, 's': 2, 'r': 2}

21. Write a program to sort numbers in list without using built-in methods?

```
l=[10,55,18,9,2,5,12]
n=len(l)
for i in range(0,n-1):
        for j in range(0,n-1-i):
        if l[j]>l[j+1]:
          l[j],l[j+1]=l[j+1],l[j]
print(l)
```

Output:-

[2, 5, 9, 10, 12, 18, 55]

Theory:

1. **What are features of python?**

- Python is a high level interpreted and scripting language
- Python is case sensitive language
- Python supports both functional and object oriented for solving programming problems
- Python has more built-in support for libraries
- Python is very easy to learn and write
- Python is dynamically typed language because there is no declaration of variable

Or

- Python is a high-level language.
- Python is an interpreted language.
- Python supports both functional and object oriented approach.
- Python is a dynamically typed language.
- Python is a case sensitive language.

2. What is variable and Identifiers?

Variable:-

- Variable is a name which stores a value.
- The value stored in a variable might get varied.

Identifiers:-

- Identifier is a name, which identifies variables, classes and functions

3. What are rules of identifiers?

- First letter should not be a number.
- We should not use keywords as identifiers.
- Except underscore remaining special characters are not allowed.

4. What is difference between .py and .pyc files?

- .py It is a file with .py extension, which stores the actual code
- .pyc It is a file with .pyc, which stores the compiled code or bytecode

5. What is the difference between list and tuple ?

List & tuple

1. Elements in list is enclosed in a pair of square brackets ([ ]) whereas elements in tuple is enclosed in a pair of parenthesis (( ))
2. List is a mutable data types whereas tuple is an immutable data type
3. List is a variable length data types whereas tuple is a fixed length data type

4. We can initialize a single value in list (,) is not mandatory whereas we can initialize a single in tuple (,) is mandatory.

5. For empty list we use square brackets where empty tuple we use tuple () (tuple function)

List	Tuple
List is declared in a pair of square brackets([ ])	Tuple is declared in a pair of parentheses or round brackets(( )).
List is a mutable data type	Tuple is a immutable data type
comma(,) is not mandatory for creating a empty list	comma(,) is mandatory for creating a empty tuple

6. What is difference between list and set?

List & set

1. Elements in list is enclosed in a pair of square brackets ([ ]) whereas set is enclosed in a pair of curly brackets ({ })

2. List is an ordered collection data type. So, we can perform indexing and slicing whereas set is an unordered collection data type. So, we cannot perform indexing and slicing

3. List allows duplicate values where as set is does not allows duplicate values
4. For empty list we use square brackets whereas empty set we use set function

5. List is a homogeneous/heterogeneous data type whereas set is a homogeneous/heterogeneous data type but we can store only immutable data type.

List	Set
List is declared in a pair of square brackets ([ ])	Set is declared in a pair of curly brackets ({ })
List is a ordered collection data type.So,we can perform indexing and slicing	Set is a unordered collection data type.So, we cannot perform indexing and slicing
List is a homogeneous and heterogeneous data type	Set is a homogenous and heterogenous data type but we can store only immutable data types.
List can store duplicate values	Set cannot store duplicate values.

7. Explain about set data type?

- Set is declared in a pair of curly braces ({ }).
- Set is a mutable data type.
- Set is a random ordered collection data type.So, we cannot perform indexing and slicing.
- Set cannot store duplicate values
- Set is a homogenous and heterogenous data type but we can store only immutable data types.

8. Explain about dictionary data type?

- Dictionary is a pair of keys and values declared inside a curly brackets({ }).
- Dictionary is a mutable data type.
- In the Dictionary data is in the form of pairs. So, we cannot perform indexing and slicing.
- Each and every keys and values are separated by colon (:).
- Each and every key value pairs are separated by comma (,).

Properties of keys

- We can store only immutable data types but we are advised to use string because string are defining as keys
- If we try to add duplicate values in dictionary, it will delete the old value and the new value will be assigned to that key

Properties of values

- We can store any type of data as values in dictionary
- We can give duplicate values as values in dictionary

9. How to remove last element in list?

- Syntax is pop(index position)
- Pop method is used to removes the element based on specified index position
- If we give an empty pop () it will delete or remove the last element in the list.

10. How to add value in a tuple?

We can single value in tuple by using type casting.

```
t=(10,20,30,40,50)
print(t)
l=list(t)
print(l)
l.append(80)
print(l)
print(tuple(l))
```

Output:-

```
(10, 20, 30, 40, 50)
[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50, 80]
(10, 20, 30, 40, 50, 80)
```

11. How to add multiple values in tuple?

We cannot add multiple values in tuple. By using type casting we can add elements in tuple.

12. How to convert tuple to list?

Using type casting method we can convert list to tuple

Example:-

t= (10,55,66,88,99)

t= list((10,55,6,88,99))

t= [10, 55, 6, 88, 99]

13. Swap first and last element in list?

Example :-

l=[11,55,66,77]

l[0],l[-1]=l[-1],l[0]

Output:-

[77, 55, 66, 11]

14. Explain the difference between ShallowCopy and DeepCopy?

Shallow Copy	Deep Copy
1. We can define the shallow copy by using slicing operator	1. We can define the deep copy by using importing copy method

2. In shallow copy both the objects are pointing to different memory address but the elements are pointing to same memory address	2. In deep copy both the objects are pointing to different memory address but the elements are pointing to different memory address.
3. If we try to modify one object other objects will not get affected but if we try to modify elements inside the object then the other objects will get affected 4. Ex:- l=[11,22,55,66,77] l1=l[::] print(l) print(l1) print(l is l1) print(l[0] is l1[0]) l1[0]=50 print(l) print(l1) Output [11, 22, 55, 66, 77] [11, 22, 55, 66, 77] False True [11, 22, 55, 66, 77] [50, 22, 55, 66, 77]	3. If we try to modify one object other objects will get affected. 4. Ex:- from copy import deepcopy l=[[1,2,3,4],['x','y','z','a']] l1=deepcopy(l) print(l) print(l1) print(l is l1) print(l[0] is l1[0]) l1[0][1]=50 print(l) print(l1) Output [[1, 2, 3, 4], ['x', 'y', 'z', 'a']] [[1, 2, 3, 4], ['x', 'y', 'z', 'a']] False False [[1, 2, 3, 4], ['x', 'y', 'z', 'a']] [[1, 50, 3, 4], ['x', 'y', 'z', 'a']]

15. What is difference between get method and set-default method?

1. If the specified key is present it will display the value	1. If the specified key is present it will display the value
2. If the specified key is not present it will take the default value	2. If the specified key is not present it will update the dictionary key and values
3. Syntax :- get(keyname,[default value])	3. Syntax :-set default(keyname,[default value])
4. If the default value is not present it will give None as an output Ex:- d={'name':'hari','age':5} d {'name': 'hari', 'age': 5} d.get('name') 'hari' d.get('age') 5 d.get('AGE',15) 15	4. If the default value is not present it will update the actual dictionary Ex:- d={'name':'mahesh','address':'hyderabad','state':'telangana'} d {'name': 'mahesh', 'address': 'hyderabad', 'state': 'telangana'} d.setdefault('name') 'mahesh' d.setdefault('age',55) 55 d {'name': 'mahesh', 'address': 'hyderabad', 'state': 'telangana', 'age': 55}

16.What is difference between remove and pop?

Pop	Remove
Pop method is used for removing the element based on index position	Remove method is used for removing the element based on value

Pop method will return the value	Remove method will not return the value
Pop method will take one argument	Remove method will take one mandatory argument
Syntax for pop:- pop(index position)	Syntax for remove:- remove(value)

17. What is difference between remove and discard?

pop	**remove**	**discard**
Pop method will delete random element	Remove method is used for delete the element based on value	Remove method is used for delete the element based on value
In pop method if we give index position it will throws a key error	If the specified element is present it will delete the value	If the specified element is present it will delete the value
	If the specified element is not present it will throws a key error	If the specified element is not present it will not perform any operation
	Syntax for remove:- remove(value)	Syntax for discard:- discard(value)

18. Explain if, elif and else?

if condition:-

It is used to control the flow of statements for one condition

Ex:-

```
a=int(input('enter a value'))
if a>10:
    print('enter into the block')
    print(a)
```

```
    print('come out of the block')
```

Output:-

```
enter a value15
enter into the block
15
come out of the block
```

Here the a value is greater than 10.so,that it will enter into the if block and it will execute the output

```
a=int(input('enter a value'))
if a>10:
    print('enter into the block')
    print(a)
    print('come out of the block')
```

Output:-

```
enter a value5
```

Here the value (a) is not greater than 10. So,that it will not enter into the if block

If-else block:-

It is used to control the flow of statements for two condition

In if else block only one block will get executed

If block is not get executed then else block will get executed

Ex:-

```
a=int(input('enter a value'))
if a%2==0:
```

```
    print('even number')
else:
    print('odd number')
```

Output:-

enter a value6

even number

Here the given condition is satisfied. So,that if block is executed

```
a=int(input('enter a value'))
if a%2==0:
    print('even number')
else:
    print('odd number')
```

Output:-

enter a value5

odd number

Here the given condition is not satisfied. So,that else block is executed

if-elif block:-

It is used to control the flow of statements for multiple conditions

In if elif block only one block will get executed

Ex:-

```
a=int(input('enter a value'))
b=int(input('enter b value'))
c=int(input('enter c block'))
```

```
if a>b:
    print('a is maximum')
elif b>c:
    print('b is maximum')
else:
    print('c is maximum')
```

Output:-

```
enter a value10
enter b value15
enter c block25
c is maximum
```

Here the a,b values are not maximum.So, that else is executed

```
a=int(input('enter a value'))
b=int(input('enter b value'))
c=int(input('enter c block'))
if a>b:
    print('a is maximum')
elif b>c:
    print('b is maximum')
else:
    print('c is maximum')
```

Output:-

```
enter a value10
```

enter b value36

enter c block25

b is maximum

Here the a is not maximum. So,that elif block is executed

```
a=int(input('enter a value'))
b=int(input('enter b value'))
c=int(input('enter c block'))
if a>b:
    print('a is maximum')
elif b>c:
    print('b is maximum')
else:
    print('c is maximum')
```

Output:-

enter a value95

enter b value25

enter c block45

a is maximum

Here a value is maximum. So,that if block is executed

Note: If we write any condition, that should end with a block(:) after give one tab indentation and then write the statements or code of a program.

19. Explain about break, continue and pass?

Break

It is used for terminating the loop

Ex:

```
for i in range(1,10):
        print(i)
        if i==5:
        break
```

Output

1 2 3 4 5

Continue

Continue is used for skipping the current iteration

Ex:

```
For i in range(1,11)
        print(i)
        if i==3:
        continue
```

Output

1 2 4 5 6 7 8 9 10

Pass

Pass is used for creating the empty blocks

Ex :-

```
For i in range(1,10)

        pass

If a>10:

        Pass

def sum ():

        Pass
```

20. What is difference between for loop and while loop?

For loop	While loop
For loop is used for when we know how times we have to run the loop	While loop is used for when we don't know how many times we have to run the loop
If we want to run the loop n times it will run for that many times	While loop will run infinity times until unless we give a condition
In for loop if the condition is not true it will not enter into the loop	In the while loop if the condition is true the loop will get iterated(iterate means repeating the execution).until unless the condition is satisfied.

Ex:-
s='hello'
for i in s:
print(i)
 	
Output:-
h
e
l
l
o
	Ex:-
	a=5
	while a<10:
	  print(a)
	  a+=1
	Ex:-
	5
	6
	7
	8
	9

21. What is a function?

1. Function is an independent block of instructions which is used to perform specific task
2. Function is used for decreasing the code redundancy and increasing the code reusability
3. Syntax for getting the address of function is Function name
4. Syntax for call the function is Function name ( )

- Until and unless we call the function, the statements will not get executed
- We can call the function anyplace and anywhere check before it has been declared or not

Types of functions

1. Built-in function
2. User defined function

Built-in function

These are already developed by the python developers

EX:-
Print, type, id, Len, input.

User defined function

User defined functions are defined by using def keyword

Advantages:-

1. Reusability
2. Debugging

Ex:-

```
def sum ( ): # functionname ex:- sum
        print(‘hello’)
        print(‘hai’)

sum( )
```

Output:-

hello

hai

Ex:-

```
def details(name,age):
    print(f'Name = {name}')
    print(f'Age ={age}')
details('hari',25)
```

Output:-

Name = hari

Age =25

22. What is difference between positional and keyword arguments?

Positional arguments	Keyword arguments
Positional arguments are defined by using proper position	Keyword arguments are defined by using no proper position

No.of arguments in function call is equal to the no.of arguments in function declaration
 	
Ex:-
def add(a,b):
  print(a)
  print(b)
 	
add(10,50)
 	
Output:-
10
50
 	
Ex:-
 	
def details(name,age,mobile):
  print(f'Name = {name}')
  print(f'Age ={age}')
  print(f'Mobile ={mobile}')
 	
details('hari',25,6352417485)
 	
Output:-
 	
Name = hari
Age =25
Mobile =6352417485
	No.of arguments in function call is equal to the no.of arguments in function declaration
	
	Ex:-
	def add(a,b):
	  print(b)
	  print(a)
	
	add(10,50)
	
	Output:-
	50
	10
	
	Ex:-
	
	def details(age,name,mobile):
	  print(f'Name = {name}')
	  print(f'Age ={age}')
	  print(f'Mobile ={mobile}')
	details(15,'loki',9673154863)
	
	Output:-
	
	Name = loki
	Age =15
	Mobile =9673154863

23. Explain about default arguments?

In default arguments the default values are provided in function declaration

If we not provide any values in function call it will take the default values

No.of arguments in function call less than or equal to the no.of arguments in function declaration

Ex:-

```
def hai(a=44,b=88):
    print(a)
    print(b)
hai(10)
```

Output :-

10

88

Ex:-

```
def details(name='raju',age=20,mobile=9674124596):
    print(f'Name = {name}')
    print(f'Age ={age}')
    print(f'Mobile ={mobile}')
details('anand',30,9673154863)
```

Output:-

Name = anand

Age =30

Mobile =9673154863

24. What is return keyword in python?

- Return is used for returning some values from function space to main space
- In Return we any data type
- Return is the final block of statement
- After return we cannot write single line of statement
- In return if we give multiple elements the output format is in tuple
- We can call the return function in three ways
    1. Assign the value to a variable.
    2. Call the function inside a print function
    3. We can call using conditional statements

Ex:-

```
def add(a,b):
    print(a)
    print(b)
    return 50
a=add(10,20)
print(a)
```

Output:-

```
10
20
50
```

```
def add(a,b):
    print(a)
    print(b)
    return 50

print(add(10,20))
```

Output:-

```
10
20
50
```

```
def add(a,b):
    print(a)
    print(b)
    return 50,40,80

print(add(10,20))
```

Output:-
10
20
(50, 40, 80)

25. Explain Variable length arguments and Variable length keyword arguments?

Variable length arguments	Variable length keyword argument
1. *args defines the variable length arguments or *prefix to the variable defines the variable length arguments	1. **args defines the variable length keyword argument or **prefix to the variable defines the variable length keyword arguments
2. We can define a function with o to any n number of values	2. We can define 0 to n number of key value pairs
3. *args will pack the individual elements into a tuple	3. **args will pack the individual elements into a dictionary
4. If we do not provide any values in function call it will give an empty tuple. Ex:- def hello(*args): print(args) hello(10,50,66,88) Output:- (10, 50, 66, 88)	4. If we do not provide any values in function call it will give empty dictionary. Ex:- def hello(**kwargs): print(kwargs) hello(a=10,b='hello',c='hai') Output:- {'a': 10, 'b': 'hello', 'c': 'hai'}

26. What is recursion and advantages of using recursion?

- Recursive function is a process of calling a function within itself until the terminating condition is satisfied
- Recursive function is used when we want to execute the same set of statements repeatedly
- Advantages of recursive function is to avoid loops.

Ex:-

Print the sum of n natural numbers by using recursion

```
def sum(n):
    if n==0:
        return 0
    return n+sum(n-1)
print(sum(5))
```

Output:- 15

Print the factorial numbers by using recursion

```
def fact(n):
    if n==0:
        return 0
    return n*fact(n-1)
print(fact(5))
```

Output:-

120

27. What is lambda function?

- We can define the lambda function using lambda keyword
- Lambda functions are used for simple expressions
- Lambda functions are anonymous single line statements
- We can give n number of input but we can get only one expression
- By default lambda function will return output of the given expression

(Optional)

- Syntax lambda arguments : expression
- We want to define a lambda function we have to define it to a variable
- variablename= lambda arguments : Expression

28. What is difference between map function and filter function?

Map function	Filter function
1. Map function are built in function.It will process and access all the elements to the iterable data type	1. Filter function are the built in function will take one value at a time and it will pass the function as a parameter if the function returns true it will add the element in the filter object and if the function return false it will not add the element in the filter object
2. Map function will take two arguments one is function name and another one is iterable data type. Syntax:- map(function name, iterable data type) 3. Map function returns the map object	2. Filter function will take two arguments one is function name and another one is iterable data type. Syntax:- filter(function name, iterable) 3. Filter function will return the filter object

4. Map object is an iterator	4. Filter object is an iterator

Ex:-
In one way
 	
def sum(n):
  return n*2
print(list(map(sum,[11,22,88,99,77]))
))
 	
Output:-
[22, 44, 176, 198, 154]
 	
If we want to write in single line we use lambda function
 	
print(list(map(lambda n:n*2,[11,22,88,99,77])))
 	
Output:-
[22, 44, 176, 198, 154]
	Ex:-
	In one way
	
	def is_hai(n):
	  if n%2==0:
	  return True
	  else:
	  return False
	print(list(filter(is_hai,[11,22,88,99,77]))))
	
	Output:-
	[22, 88]
	
	If we want to write in single line we use lambda function
	
	print(list(filter(lambda n:n%2==0,[11,22,88,99,77])))
	
	Output:-
	[22, 88]

29. What is _name_?

- __name__ is a special method.
- It is used to evaluate the current module.
- If we won't import any module ten the value of __name__ is __main__.
- If we import any module then the value of __main__ is that current imported module

30. What are Global, Local and Non local variables?

Global variables

Global variables are defined in main space and modified,accessing in the function space.

We cannot directly modify or access. Using global keyword we can modify or access the variable

global var1,var2,.....varN

Ex:-

```
a,b=10,50

def hello( ):

    global a,b

    print(a,b)

    a-=8

    b+=5

    print(a,b)

hello()
```

Output:-

10 50

2 55

Local variables

Local variables are defined in function space and accessed,modified in function space itself.

Ex:-

```
def hai( ):

    a=10

    print(a)

hai()
```

Output:-

Non-Local variables

Non-local variables are defined outside the function space and accessed,modified in inner function space

We cannot directly modify or access. Using non-local keyword we can modify or access the variable

non-local var1,var2,.....varN

Ex:-

```
def add( ):
    a,b=40,50
    print(a,b)
    def sum():
        nonlocal a,b
        print(a,b)
        a+=100
        b-=20
        print(a,b)
    sum()
    print(a,b)
add()
```

Output:-

```
40 50
40 50
140 30
```

31. What is class?

Class is a blueprint that is used to create an object

Class can be created by using class keyword.

32.What is object?

Object is a real time entity that has states and behaviours

33. How to create a class?

If we create a class a dictionary will be created.

Here the states and methods are considered as keys and their respective values are considered as values

Syntax for creating a class

Class classname( ):

( ) are optional

Ex:-

```
class A:
    x=10
    y=20
    def hai(self):
        Code
```

Here x,y,hai are keys and 10,20,code are values

Ex:-

```
class A:
  x=10
```

```
    y=20
    def hai(self):
        pass

obj=A( )
print(obj.x)
print(obj.y)
```

Output:-
10
20

34. How to create a object?

- Object is a real time entity
- Syntax for creating an object
- objectname=classname(arguments)
- arguments are optional
- If we create a object a dictionary will be created under object name
- It will take all the properties of class dictionary to object dictionary
- It will checks for the constructor if it is present it will be called automatically

35. What is constructor?

- Constructor is special method which will be called automatically when we create an object
- __init__ is a constructor for the creation of object
- Self is a mandatory arguments which is used while we create an object
- Self will take the instance address of an object

36. What is self?

- Self is a mandatory arguments used when we create an object or constructor.
- Self will give the instance address of an object
- Self is a mandatory argument we have to pass for the __init__

37. Explain the difference between class variables and instance variables?

Class variable	Instance variable or object variable
Class variable is defined inside a class and outside the method	Object variable is defined inside a class and inside the method
Class variables are used for generic properties	Object variables are used for specific properties

41. Access modifiers in python

- Python uses _(underscore) symbol to determine the access modifiers
- Access modifiers have an important role to secure the data from unauthorised users.

Access modifiers are of three types:-

1. public
2. protected
3. private

1. Public

- The members that are declared public can be accessible any part of the program

- By default every variable is public

2. Protected

- The members that are declared protected can be accessible within a class,outside the class and within the package

- To indicate variable protected we use single underscore(_) before the variable

3. Private

- The members that are declared private can be accessible within a class

- To indicate variable protected we use double underscore(__) before the variable

Ex:-

```
class C:
    def __init__(self):
        self.x=10
        self._y=20
        self.__z=30
    def get(self):
        return self.__z
    def set(self,m):
        self.__z=m
obj=C()
print(obj.x)
print(obj._y)
print(obj.get())
obj.set(50)
print(obj.get())
```

Output:-

```
10
20
30
50
```

Note:-

Public can be accessed from one module to another module

Protected can be accessed from within the class,outside the class or within the package.

Private can be accessed within the class

Public and protected can be easily accessible .

Private can be accessible by using getter method and setter method.

38. Explain the difference between object method and class method?

Object method	Classmethod
1. Object method is called called by using object name 2. By using object we can access and modify the specific properties 3. It carries one mandatory argument that is self 4. Self carries the address of instance object	1. Class method can be created by using @classmethod decorator 2. Class method will be called by using both class name and object name 3. Class method can be access and modify the generic properties 4. Class method will one mandatory argument cls 5. Cls carries the class address

39. Explain the difference between class method and static method?

Classmethod	Static method
1. Class method can be created by using @classmethod decorator 2. Class method will be called by using both class name and object name 3. Class method can be access and modify the generic properties 4. Class method will one mandatory argument cls 5. Cls carries the class address	1. Static method can be decorated by using @static method decorator 2. Static method can be called by using class name and object name 3. Static method cannot access and modify the class members(generic properties) or object members(specific properties) 4. Static method doesn’t take any argument

40. What is Encapsulation?

It is used for binding the states and behaviours of an object under one roof

42. What is inheritance and explain types of inheritance?

Inheritance is the process of creating a class from another class

Types of inheritance

1.single level inheritance

2.multiple inheritance

3.hierarical inheritance

4.multilevel inheritance

5.hybrid inheritance

1. **Single level inheritance**

It is process of inheriting the properties from single parent class to single child class

Ex:

```
class A:
    def __init__(self):
        print('init of a')
class B(A):
    pass
ob=B()
```

Output

```
init of a
```

2. **Multiple inheritance**

It is a process of inheriting the properties of multiple parent class to single child class.

Ex:

```
class A:
    def __init__(self):
        print('init of a')
class B():
    def __init__(self):
        print('init of b')
class C(A,B):
    pass
ob=C()
```

Output:

```
init of a
```

3. **Hierarical inheritance**

It is the process of inheriting the properties from single parent class to multiple child class.

Ex:

```
class A:
    def __init__(self):
        print('init of a')
class B(A):
    def __init__(self):
```

```
        print('init of b')
class C(A):
    pass
ob=C()
```

Output

init of a

4. **Multilevel inheritance**

It is a process of inheriting the properties from one child class to another child class.

Ex:

```
class A:
    def __init__(self):
        print('init of a')
class B(A):
    def __init__(self):
        print('init of b')
class C(B):
    pass
ob=C()
```

Output

init of b

5. **Hybrid inheritance**

It is the combination of more than one type of inheritance or It is a process of acquiring level of inheritance

```
class A:
    def __init__(self):
        print('init of a')
class B(A):
    def __init__(self):
        print('init of b')
class C(A):
    def __init__(self):
        print('init of c')
class D(B,C):
     def __init__(self):
        print('init of d')
obj=D()
```

Output:-

init of d

43. Explain Super method?

Super method is a method which is used in case of inheritance only

Syntax

super().methodname(arguments) or super(). __init__(arguments)

44. How to implement parent class method along with child class method in case of multilevel inheritance?

```
class A:
    def __init__(self):
```

```
        print('init of a')
class B(A):
    def __init__(self):
        print('init of b')
class C(B):
    pass
ob=C()
```

Output

init of b

45. What is polymorphism and how to achieve it in python? Is overloading possible in python and why?

Polymorphism is process of using one method with different functionalities

We can achieve the polymorphism by using method overriding

Method overriding

It is a process of changing the implementation of superclass(parent class) in subclass(child class)

When we use method overriding the code redundancy is more using chaining process along with method overriding we can achieve that.

Ex:-

```
Class BankV1( ) :
        bank_name=’sbi’
        bank_ifsc=1234
        def __init__(self,n,a,bal):
                self.name=n
                self.age=a
                self.balance=bal
Class Bank_v2( ) :
```

```
    def __init__(self,n,a,bal,ac,mo):
        self.name=n
        self.age=a
        self.balance=bal
        self.account=ac
        self.mobile=mo
```

Method overloading

It is process of creating a method multiple times in a single class with different arguments

Method overloading is not possible in python

Method overloading is not possible because it will take the latest defined method

Ex:-

```
Class Bank( ) :
    def __init__(self,n,a,bal):
        self.name=n
        self.age=a
        self.balance=bal
    def __init__(self,n,a,bal,ac,mo):
        self.name=n
        self.age=a
        self.balance=bal
        self.account=ac
        self.mobile=mo
```

If we use method overriding the code redundancy is more by using chaining process along with method overriding we can achieve the method overriding.