Section A: Theory (10 Questions)

1. Explain difference between discard() and remove()?

Remove: her remove method is used to remove a specified item from the set. If the item is not found in the set, the method raises a KeyError exception. The syntax for using the remove method is: ``` set.remove(item) ```

```
s={1,2,3}
s.remove(5)
```

output:

KeyError: 5

Discard: The discard method is similar to the remove method, but it does not raise a KeyError exception when the item to be removed is not found in the set. Instead, it simply does nothing. The syntax for using the discard method is: ``` set.discard(item) ```

```
s={1,2,3}
s.discard(5)
```

output: nothing

2. How can you update dictionary in python, Explain the method with example.

The update() method in Python dictionaries merges the contents of one dictionary into another, updating the value of keys that exist in both dictionaries and adding any keys from the second dictionary that are not in the first.

You can also pass iterable having key value pair in dictionary

```
d={1:10,2:20}
d.update({(3,90)})
print(d)
```

output:

{1:10,2:20,3:90}

```
d={1:10,2:20}
d.update({4:400})
print(d)
```

{1: 10, 2: 20, 3: 90, 4: 400}

3. How can you perform set operations like union, intersection and difference in python, provide the example.

To perform set operation in python we already have union,intersection and difference function

```
# 1. Find the union of two sets.
# s={1,2,3,4,8,9}
# s2={2,3,8,0,90}
# print(s.union(s2))
# output:-
# {0, 1, 2, 3, 4, 8, 9, 90}

# 2. Find the intersection of two sets.
# s={1,2,3,4,8,9}
# s2={2,3,8,0,90}
# print(s.intersection(s2))
# output:
# {8,2,3}
```

```
# 5. Check if two sets have any elements in common.
# s={1,2,3,4,8,9}
# s2={2,3,8,0,90}
# print(s.intersection(s2))
# output:{8,2,3}
```

4. Describe the difference between the map(), filter(), and reduce() functions in Python with examples

Map: **The map () function returns a map object(which is an iterator)** of the results after applying the given function to each item of a given iterable (list, tuple, etc.).

***Syntax**: map(fun, iter)*

Parameters:

- ***fun:** It is a function to which map passes each element of given iterable.*
- ***iter:** iterable object to be mapped.*

```
    return n * 2

# Using map to double all numbers
numbers = [5, 6, 7, 8]
result = map(double, numbers)
print(list(result))
```

- *Output: [10,12,14,16]*

Filter: **The filter() method filters the given sequence with the help of a function that tests each element in the sequence to be true or not if element satisfy condition then is return that element in new list otherwise it doesn't add the element in list**

Syntax: filter(function, sequence)

Parameters:

- ***function: function that tests if each element of a sequence is true or not.***
- ***sequence: sequence which needs to be filtered, it can be sets, lists, tuples, or containers of any iterators.***

```python
# Define a function to check if a number is even
def is_even(n):
    return n % 2 == 0

# Define a list of numbers
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Use filter to filter out even numbers
even_numbers = filter(is_even, numbers)
print("Even numbers:", list(even_numbers))
```

Output: [2,4,6,8,10]

Reduce: The reduce() function is exceptionally useful when it comes to cumulative calculations and aggregations as it provides a concise and efficient way to reduce it to a single value rather than using iterative loops or explicit functions.

Syntax:

from functools import reduce

```
reduce(function, sequence, initial=None)
```

→ function: The function to be applied to the elements of the sequence. It should take two arguments and return a single value.

→ sequence: The sequence of elements on which the reduction operation will be performed.

→initial (optional): An initial value to be used as the first argument in the reduction operation. If not provided, the first two elements of the sequence will be used as the initial arguments.

How Reduce Works:-

1. The syntax, reduce(function, sequence) is called, where function is the specified function and sequence is the sequence of elements.
2. The first two elements of the sequence are passed as **arguments** to the function. These two elements can be either the first two elements of the sequence or an initial value provided as the initial parameter.
3. The function returns a result based on the two arguments.
4. The result from function becomes the first argument for the following function call, and the next element of the sequence becomes the second argument.
5. Steps 3 and 4 are repeated until all elements in the sequence have been processed.
6. The final output is the accumulated result, which represents the reduction of the sequence to a single value.

Eg- 2

```
>>> from functools import reduce
>>> numbers = [10, 50, 40, 60, 70]
>>> max_number = reduce(lambda x, y: x if x > y else y, numbers)
>>> print("Maximum number:", max_number)
```

Output: 70

Eg-1

```
>>> from functools import reduce
>>> def combine_values(x, y):
>>>     return x + y
>>> my_list = [1, 2, 3, 4, 5]
>>> result = reduce(combine_values, my_list)
>>> print(result)
```

Output: 15

5. What is the purpose of a constructor in Python classes? Provide an example of a class with a constructor that initializes a student's name and age.

In Python, a constructor is a special method that is called automatically when an object is created from a class. Its main role is to initialize the object by setting up its attributes or state.

__init__ Method

This method initializes the newly created instance and is commonly used as a constructor in Python. It is called immediately after the object is created

Types of Constructors

Constructors can be of two types.

1. Default Constructor

A **default constructor** does not take any parameters other than **self**. It initializes the object with default attribute values.

2. Parameterized Constructor

A **parameterized constructor** accepts arguments to initialize the object's attributes with specific values.

it is not mandatory. If no constructor is defined, Python provides a default constructor that takes no arguments and does nothing.

```
class student:
    def __init__(self,name,age):
     self.name=name
     self.age=age
s=student('lavanya',24)
print(s.name,s.age)
```

6. What is the filter() function in Python? Provide an example where you filter even numbers from a list using filter().

Filter: **The filter() method filters the given sequence with the help of a function that tests each element in the sequence to be true or not if element satisfy condition then is return that element in new list otherwise it doesn't add the element in list**

Syntax: filter(function, sequence)

Parameters:

- ***function: function that tests if each element of a sequence is true or not.***
- ***sequence: sequence which needs to be filtered, it can be sets, lists, tuples, or containers of any iterators.***

```
# Define a function to check if a number is even
def is_even(n):
    return n % 2 == 0

# Define a list of numbers
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Use filter to filter out even numbers
even_numbers = filter(is_even, numbers)
print("Even numbers:", list(even_numbers))
```

Output:[2,4,6,8,10]

7. Explain the concept of encapsulation in OOP with an example.

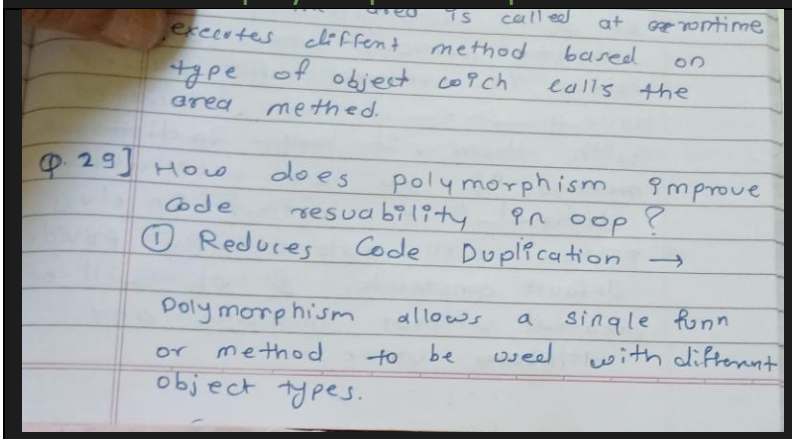
Encapsulation is a technique for hiding a class' internal operations and exposing only the information required for external interactions. Encapsulation restricts access to a class's data to its methods and keeps the class's variables private.

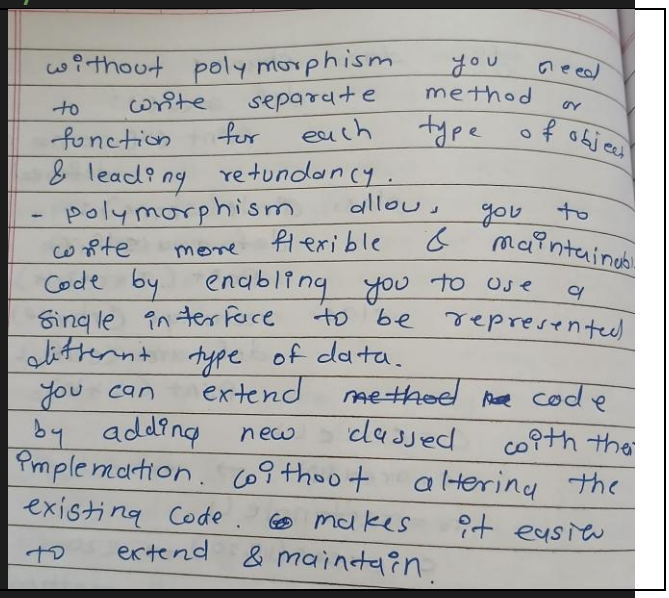
With help of getter and setters methods we can access and update the private data of class

```python
class A:
    def __init__(self):
        self.__a='Lava@2203'
    def get_a(self):
        return self.__a
    def set_a(self,new_a):
        self.__a=new_a
# a=A()
# print(a.get_a())
# a.set_a('Sara@2203')
# print('updated private varible',a.get_a())
# output:
# Lava@2203
# updated private varible Sara@2203
```

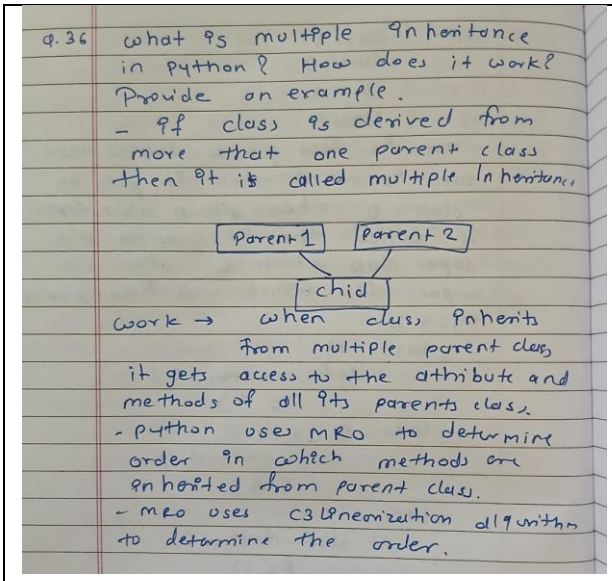
If you try to do print(a._a) this will throw error

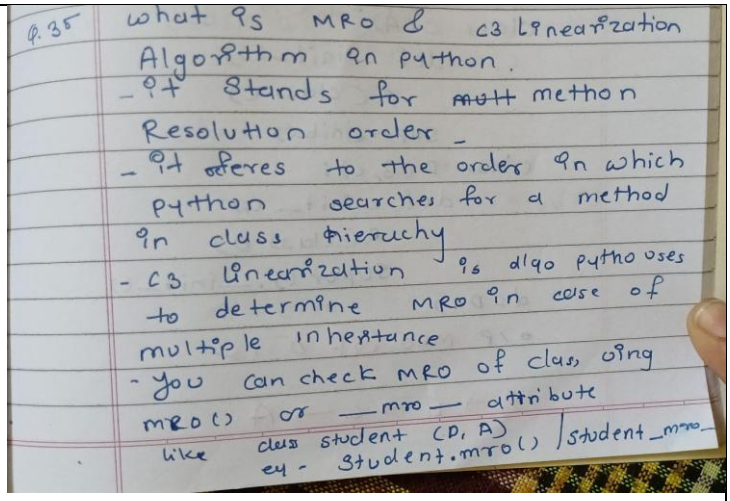
8. How does polymorphism improve code reusability in OOP?

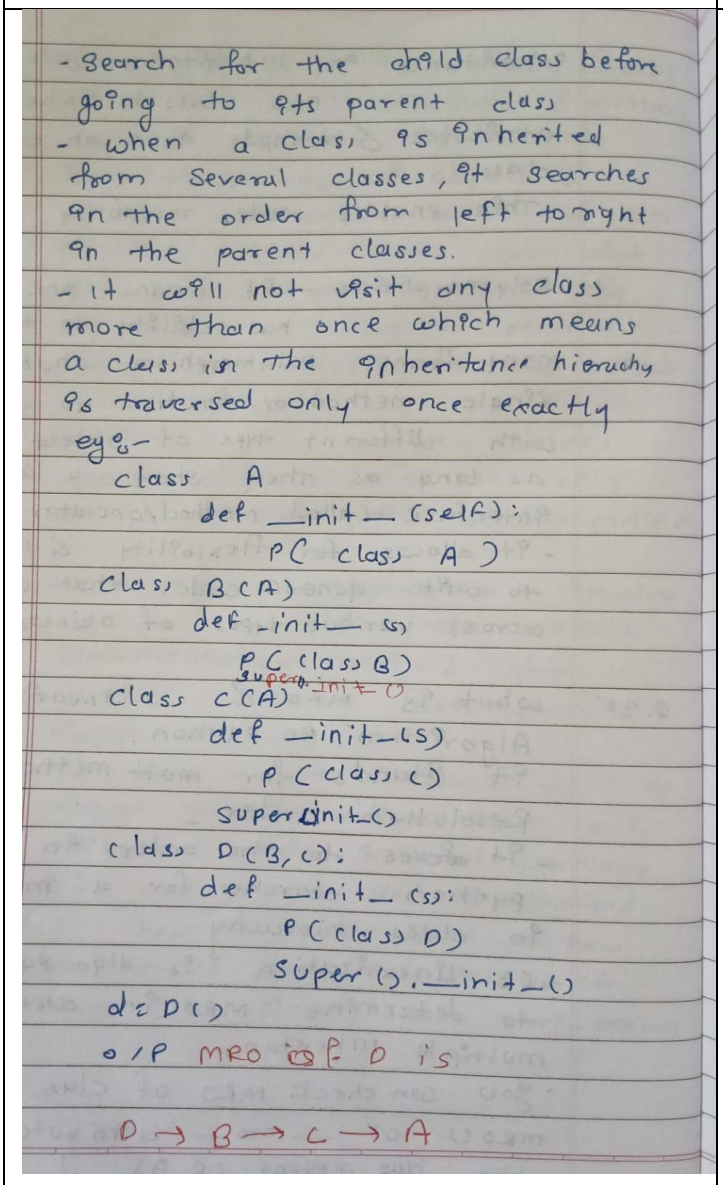


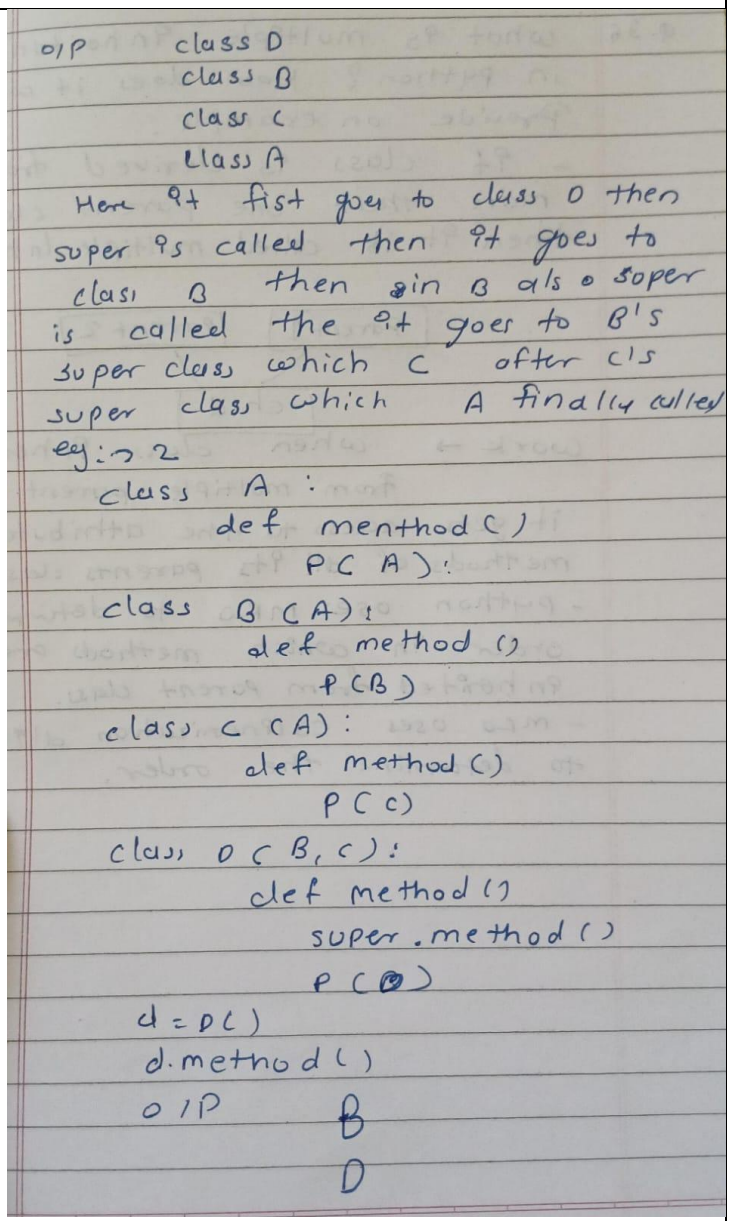


9. What is multiple inheritance in Python? How does it work? Provide an example.

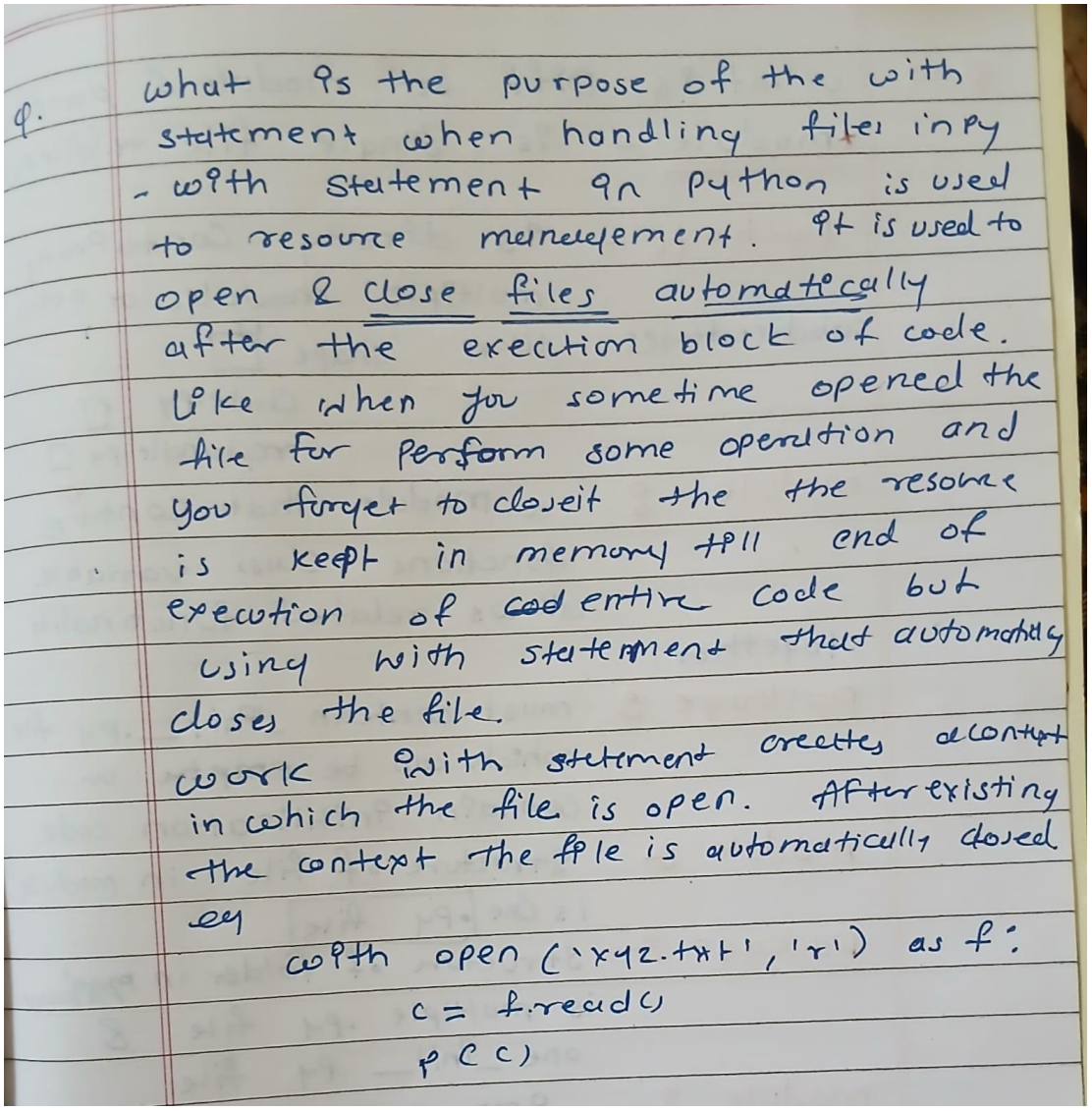








10. What is the purpose of the with statement when handling files in Python?



Q. What is the purpose of the with statement when handling files in py

- with statement in Python is used to resource management. It is used to open & close files automatically after the execution block of code. Like when you sometime opened the file for perform some operation and you forget to close it the the resource is kept in memory till end of execution of entire code but using with statement that automatically closes the file.

work with statement creates a context in which the file is open. After existing the context the file is automatically closed

eg

```
with open ('xyz.txt', 'r') as f:
    c = f.read()
    p(c)
```

Section 3: Theory

```
# 1. The following code contains an error. Identify and correct it:

# def my_func(x, y):
#       return x + y
# result = my_func(5)
# print(result)
# ANS
def my_func(x, y):
    return x + y
result = my_func(5,10)
print(result)

# 2. What will be the output of this code?

x = 10
for i in range(x):
    if i % 2 == 0:

        continue
    print(i)
# ANS
# 1 3 5 7 9


# 3. What will be the output of the following code?
def greet(name="User"):
    print("Hello, " + name)
greet("John")
greet()
# ANS
# Hello, John
# Hello, User


# 4. What will be the output of this code?
a = (1, 2, 3)
b = list(a)
b.append(4)
print(a)
# ANS (1,2,3)
```

```
# 5. Code:
# def greet(name):
#       print("Hello" + name)
# greet()
# ANS
# def greet(name):
#       print("Hello" + name)
# greet('lavanya')
# output: Hellolavanya
#    Error: Pass the required parameter to the function.

# 6. Correct the code to calculate the sum of numbers using reduce():
#    from functools import reduce
#    numbers = [1, 2, 3, 4]
#    result = reduce(lambda x, y: x + y, numbers)
#    print(result)
# ANS
from functools import reduce
numbers = [1, 2, 3, 4]
```

```python
result = reduce(lambda x,y: x + y, numbers)
print(result)


# no Error
# 7. Code:

#    class Animal:
#        def __init__(name):
#            self.name = name
#    Error: Correct the constructor's parameter list.
# ANS
class Animal:
      def __init__(self,name):
          self.name = name

# 8. The following code has an error. Correct it and explain the output:
#    def multiply(x, y):
#        return x * y
#    result = multiply(2, 3, 4)
#    print(result)
# ANS
def multiply(x, y):
      return x * y
result = multiply(2, 3)
print(result)


# 10. Code:

#     class Car:
#         def __init__(self, make, model):
#             make = make
#             model = model
#  Error: Fix the instance variable assignment issue.
# ANS
class Car:
       def __init__(self, make, model):
           self.make = make
           self.model = model
c=Car('o', 'iio')
print(c.make)
```

Section C: Write Code For (10 Questions)

```
# 1. Write a Python program to create a dictionary that stores student names as keys and their scores as values. Write a function that returns the name of the student with the highest score.
d={'A':90,'B':89,'C':56,'D':40}
m=max(d.values())
for k,v in d.items():
    if v==m:
        print("winner",k)
        break

# 2. Write a Python program to sum all the numbers in a list. The program should use a loop to calculate the sum of the numbers in the list.
l=[10,20,30,4,5,6]
sum=0
for i in l:
    sum+=i
print("Sum:",sum)


# 3. Write a program that finds the factorial of a number using both a while loop and a for loop.
n=int(input("Enter no "))
fact=1
for i in range(1,n+1):
    fact*=i
print("Fact using for ",fact)
i=1
fact=1
while(i<=n):
    fact*=i
    i+=1

print("Fact using While",fact)
# output:
# Enter no 5
# Fact using for  120
# Fact using While 120
```

4. Write a function that uses map() to return a new list where each string in a list is reversed.

```python
l=['abc','efgh','ijkl','mnop']
print(list(map(lambda x: x[::-1],l)))
# output:['cba', 'hgfe', 'lkji', 'ponm']


# 5. Create a class Person with a constructor that accepts name and age. Add a method to check if the person is eligible to vote.
class Person:
    def __init__(self,name,age):
        self.name=name
        self.age=age
    def Eligible_check(self):
        if self.age>=18:
            print(self.name,"Is eligible for Vote")
        else:
            print(self.name,"not eligible for vote ")
# p=Person('lavanya',24)
# p1=Person('samarth',17)
# p.Eligible_check()
# p1.Eligible_check()
# output:
# lavanya Is eligible for Vote
# samarth not eligible for vote


# 6. Write a Python program that defines a class Car. The class should have a constructor that initializes the car's make, model, and year. Then, create an instance of the class and print the car's details.
class Car:
    def   __init__(self,make,model,year):
        self.make=make
        self.model=model
        self.year=year
    def details(self):
        print("Make:",self.make,"Model:",self.model,"Year:",self.year)
c=Car('TaTa','Toyoto',2009)
# c.details()
# output:Make: TaTa Model: Toyoto Year: 2009


# 7. Write a function 'sort_students_by_grade()' in the 'student_data.py' module that sorts the list of students by their grades in descending order. Print the sorted list of students.
import student_data
```

```
list=[10,40,89,90,9,8,78,76,30,20,1]
result=student_data.sort_students_by_grade(list)
print("Result:",result)
# output:
# Result: [90, 89, 78, 76, 40, 30, 20, 10, 9, 8, 1]


# 8. Nested Loops: Write a Python program using nested loops to print the following pattern:

#         A
#       A B A
#     A B C B A
#   A B C D C B A
# A B C D E D C B A

# 9. Write a Python program to demonstrate operator overloading in OOP.
class Vector:
    def __init__(self,a):
        self.a=a
        print(self.a)
    def __add__(self,another):
        return self.a+another.a
# v1=Vector(10)
# v2=Vector(20)
# print("Sum:",v1+v2)
# output:Sum: 30


# 10. Define a class BankAccount with the attributes balance (public), account_number
(private), and account_type (protected). Provide a method deposit() that adds to the balance,
and a method get_balance() to access the balance. Demonstrate usage by creating an instance
of BankAccount and performing deposits and balance retrieval.
class BankAccount:
    def __init__(self,balance,acc_num,acc_type):
        self.balance=balance
        self.__account_number=acc_num
        self._account_type=acc_type
        print(self.__account_number,self._account_type)
    def deposit(self,bal):

        self.balance+=bal
    def get_balance(self):
```

```
        print("Your Balance:",self.balance)
b=BankAccount(1000,39388090,'Saving account')
b.deposit(100000)
b.get_balance()
# output:
# 39388090 Saving account
# Your Balance: 101000
```