Lab Experiment – 04

AIM: To implement Classes, Objects and Inheritance

THEORY:

Classes: A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

Objects: An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with actual values. It's not an idea anymore, it's an actual dog, like a dog of breed pug who's seven years old. You can have many dogs to create many different instances, but without the class as a guide, you would be lost, not knowing what information is required.

Inheritance: Inheritance is the capability of one class to derive or inherit the properties from another class. It represents real-world relationships well.It provides the **reusability** of a code. We don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.

IMPLEMENTATION:

1. Explain Classes and Objects with suitable examples.

```
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
    def printname(self):
        print(self.firstname,self.lastname)
p1=Person('Vidhi','Kansara')
p1.printname()
```

```python
# init constructor
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Study\Academics\python\lab4> python -u "c:\Study\Academics\python\lab4\objclass.py"
Vidhi Kansara
PS C:\Study\Academics\python\lab4>
```

2. Explain use of __init__() function in class with suitable example.

__init__(): It is known as a constructor. The __init__ method can be called when an object is created from the class, and access is required to initialize the attributes of the class.

```python
# print(p1.a)

class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname

p1=Person('Vidhi','Kansara')
print(p1.firstname,p1.lastname)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS C:\Users\djsce.student\Desktop\pythonlab4> python objclass.py
Vidhi Kansara
PS C:\Users\djsce.student\Desktop\pythonlab4>
```

3. Code for Object Methods, Modifying Object Properties and Deleting Objects.

```
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
    def printname(self):
        print(self.firstname,self.lastname)
p1=Person('Vidhi','Kansara')
p1.printname()
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
    def printname(self):
        print(self.firstname,self.lastname)
p1=Person('Vidhi','Kansara')
print("Modification")
p1.firstname=('Aarti')
p1.printname()
print("Deletion:")
del(p1)
p1.printname()
```

```
PROBLEMS    OUTPUT    TERMINAL    JUPYTER    DEBUG CONSOLE

PS C:\Study\Academics\python\lab4> python -u "c:\Study\Academics\python\lab4\objclass.py"
Vidhi Kansara
Modification
Aarti Kansara
Deletion:
Traceback (most recent call last):
  File "c:\Study\Academics\python\lab4\objclass.py", line 33, in <module>
    p1.printname()
NameError: name 'p1' is not defined
PS C:\Study\Academics\python\lab4>
```

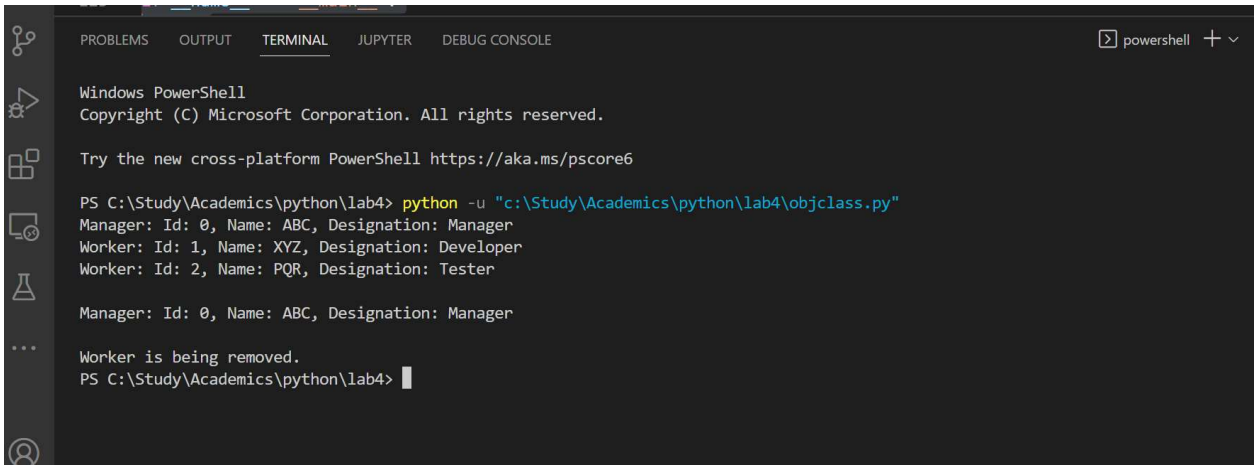
4. Python Inheritance with suitable examples.

```
class Person:
    def __init__(self,fname,lname):
        self.firstname=fname
        self.lastname=lname
    def printname(self):
        print(self.firstname,self.lastname)
class Student(Person):
    def __init__(self,fname,lname):
        super().__init__(fname,lname)
a=Student('Vidhi','Kansara')
print("Derieving from parent class")
a.printname()
```

5. Write python code to implement the following inheritance example:Classes: Employee, Developer, Tester, ManagerDeveloper, tester, Manager inherit EmployeeManager handles Developer, testerManager class : implement functions to add Developer/Tester and Remove Developer/ TesterDisplay .. to see the list of employees he manages

```
from typing import Union, Sequence

class Employee:

    _id: int = 0

    name: str = ""

    designation: str = ""

    def __init__(self, **kwargs):

        self._id = kwargs["_id"]

        self.name = kwargs["name"]

        self.designation = kwargs["designation"]

    def __str__(self):

        return f"Id: {self._id}, Name: {self.name}, Designation:
{self.designation}\n"


class Developer(Employee):

    def __init__(self, **kwargs):

        super().__init__(**kwargs)


class Tester(Employee):

    def __init__(self, **kwargs):

        super().__init__(**kwargs)


Worker = Union[Developer, Tester]


class Manager(Employee):

    _developers: Sequence[Worker] = []

    def __init__(self, **kwargs):

        super().__init__(**kwargs)
```

```
    def add_worker(self, worker: Worker):
        self._developers.append(worker)
    def remove_worker(self, worker_id: int) -> bool:
        for worker in self._developers:
            if worker._id == worker_id:
                self._developers.remove(worker)
                return True
        return False
    def __str__(self):
        details = f"Manager: {super().__str__()}"
        for worker in self._developers:
            details += f"Worker: {worker.__str__()}"
        return details

def main():
    manager = Manager(_id=0, name="ABC", designation="Manager")
    worker1 = Developer(_id=1, name="XYZ", designation="Developer")
    worker2 = Tester(_id=2, name="PQR", designation="Tester")
    manager.add_worker(worker1)
    manager.add_worker(worker2)
    print(manager)
    manager.remove_worker(worker1._id)
    manager.remove_worker(worker2._id)
    print(manager)
    print("Worker is being removed.")

if __name__ == "__main__":
 main()
```



CONCLUSION: In this experiment we learnt about classes, object and inheritance in python, the interrelation of classes and objects that class is a collection of objects and understood inheritance and thus, implemented it using an example.