**Experiment No. 9**

**Aim:** To implement different types of inheritance

**Theory:** Inheritance in Python is a mechanism that allows a new class (subclass or derived class) to inherit attributes and behaviors from an existing class (superclass or base class). This promotes code reuse, extensibility, and the creation of a hierarchical structure among classes. The class that is inherited from is known as the parent class, while the class that inherits is called the child class.

**Types of inheritance in python are:**

**Single Inheritance:** In single inheritance, a class can inherit attributes and behaviors from only one parent class.

**Multilevel Inheritance:** Multilevel inheritance involves a chain of inheritance where a class inherits from another class, and then another class inherits from the derived class.

**Syntax:**

class BaseClass: # Base class members

class DerivedClass1(BaseClass): # Derived class 1 members

class DerivedClass2(DerivedClass1): # Derived class 2 members

**Hierarchical Inheritance:** Hierarchical inheritance occurs when multiple classes inherit from a single parent class.

**Syntax:**

class BaseClass: # Base class members

class DerivedClass1(BaseClass): # Derived class 1 members

class DerivedClass2(BaseClass): # Derived class 2 members

**Hybrid Inheritance:** Hybrid inheritance is a combination of two or more types of inheritance mentioned above.

**Syntax:**

class BaseClass1: # Base class 1 members

class BaseClass2: # Base class 2 members

class DerivedClass1(BaseClass1): # Derived class 1 members

class DerivedClass2(DerivedClass1, BaseClass2): # Derived class 2 members

class DerivedClass3(BaseClass1): # Derived class 3 members

class DerivedClass4(DerivedClass3, BaseClass2): # Derived class 4 members

**Cyclic Inheritance:** Cyclic inheritance occurs when a class is derived from itself directly or indirectly. **Syntax:**

class CyclicClass(CyclicClass): # Cyclic inheritance

**Program:**

**class Professor:**

**def \_\_init\_\_(self, name, department):**

**self.name = name**

**self.department = department**

**def display\_info(self):**

**print("Name:", self.name)**

**print("Department:", self.department)**

**class Teacher(Professor):**

**def \_\_init\_\_(self, name, department, subject):**

**super().\_\_init\_\_(name, department)**

**self.subject = subject**

**def display\_info(self):**

**super().display\_info()**

**print("Subject:", self.subject)**

**class AssistantProfessor(Professor):**

**def \_\_init\_\_(self, name, department, research\_area):**

**super().\_\_init\_\_(name, department)**

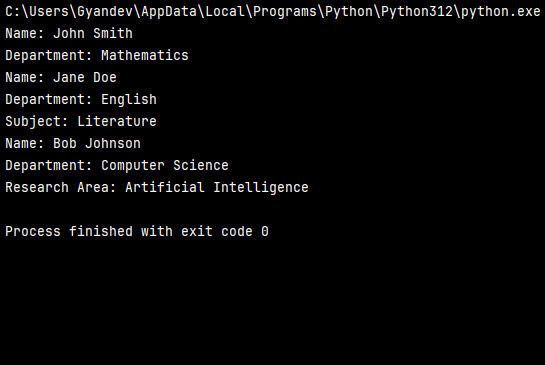
**self.research\_area = research\_area**

**def display\_info(self):**

**super().display\_info()**

**print("Research Area:", self.research\_area)**

**OutPut:**

****

**Conclusion:** Hence we have successfully implemented Inheritance in python.