TYPES OF INHERITANCE (PROGRAMS)

inherit1.py

```python
#inheritance in python
#single inheritance
class A:
    def show(self):
        print("Class A method")
class B(A):
    def show2(self):
        print("Class B method")
obj1=A()
obj1.show()
obj2=B()
obj2.show()
obj2.show2()
```

```
PS D:\pythonfolder> & C:/U
Class A method
Class B method
PS D:\pythonfolder>
```

inherit1.py

```python
#inheritance in python
#multiple inheritance
class A:
    def show1(self):
        print("Class A method")
class B:
    def show2(self):
        print("Class B method")
class C(A,B):
    def show3(self):
        print("Class C method")

obj=C()
obj.show1()
obj.show2()
obj.show3()

```

```
PROBLEMS    OUTPUT    DEBUG CON

PS D:\pythonfolder> & C:/U
Class A method
Class B method
Class C method
PS D:\pythonfolder>
```

```python
#inheritance in python
#multilevel inheritance
class A:
    def show1(self):
        print("Class A method")
class B(A):
    def show2(self):
        print("Class B method")
class C(B):
    def show3(self):
        print("Class C method")
class D(C):
    def show4(self):
        print("Class D method")


obj1=A()
obj1.show1()

obj2=B()
obj2.show1()
obj2.show2()

obj3=C()
obj3.show1()
obj3.show2()
obj3.show3()

obj4=D()
obj4.show1()
obj4.show2()
obj4.show3()
obj4.show4()
```

```
PS D:\pythonfolder> & C:/
Class A method
Class A method
Class B method
Class A method
Class B method
Class C method
Class A method
Class B method
Class C method
Class D method
PS D:\pythonfolder>
```

```python
#inheritance in python
#hierarchical inheritance
class A:
    def show1(self):
        print("Class A method")
class B(A):
    def show2(self):
        print("Class B method")
class C(A):
    def show3(self):
        print("Class C method")
class D(A):
    def show4(self):
        print("Class D method")

obj1=A()
obj1.show1()

obj2=B()
obj2.show1()
obj2.show2()

obj3=C()
obj3.show1()
obj3.show3()

obj4=D()
obj4.show1()
obj4.show4()
```

```
PS D:\pythonfolder> & C
Class A method
Class A method
Class B method
Class A method
Class C method
Class A method
Class D method
PS D:\pythonfolder>
```

```python
#inheritance in python
#hybrid inheritance
class A:
    def show1(self):
        print("Class A method")
class B(A):
    def show2(self):
        print("Class B method")
class C(A):
    def show3(self):
        print("Class C method")
class D(B,C):
    def show4(self):
        print("Class D method")

obj1=A()
obj1.show1()

obj2=B()
obj2.show1()
obj2.show2()

obj3=C()
obj3.show1()
obj3.show3()

obj4=D()
obj4.show2()
obj4.show3()
obj4.show4()
```

```
PS D:\pythonfolder> & C:/
Class A method
Class A method
Class B method
Class A method
Class C method
Class B method
Class C method
Class D method
PS D:\pythonfolder>
```

Abstraction in python

```python
#abstraction in python
from abc import ABC,abstractmethod
class A(ABC):
    def show1(self):  #normal method
        print("Abstract class method called")
    @abstractmethod
    def show2(self):   #abstract method
        pass
class B(A):
    def show2(self):
        print("Abstract class method is overridden")


obj1=B()
obj1.show1()
obj1.show2()
```

```
PS D:\pythonfolder> & C:/Users/maany
Abstract class method called
Abstract class method is overridden
```

```python
#abstraction in python
from abc import ABC,abstractmethod
class A(ABC):
    @abstractmethod
    def show1(self):   #abstract method
        print("Abstract method definition in abstract class")
class B(A):
    def show1(self):
        print("Abstract class method is overridden in derived class")
        super().show1()


obj1=B()
obj1.show1()
```

```
PS D:\pythonfolder> & C:/Users/maany/AppData/Local/Mi
Abstract class method is overridden in derived class
Abstract method definition in abstract class
PS D:\pythonfolder>
```

```python
#abstraction in python
from abc import ABC,abstractmethod
class A(ABC):
    @abstractmethod
    def show1(self):   #abstract method
        print("Abstract method definition in abstract class")
class B(A):
    def show1(self):
        print("Abstract class method is overridden in class B")
        super().show1()
class C(A):
    def show1(self):
        print("Abstract class method overridden in class C")


obj1=B()
obj1.show1()
obj2=C()
obj2.show1()
```

```
PS D:\pythonfolder> & C:/Users/maany/AppData/Loca
Abstract class method is overridden in class B
Abstract method definition in abstract class
Abstract class method overridden in class C
PS D:\pythonfolder>
```

Polymorphism in python

```python
#polymorphism in python
#method overridding(using same name of method in all classes and then calling it)
class A():
    def show1(self):
        print("Class A method")
class B(A):
    def show1(self):
        print("class B method")
        super().show1()
class C(B):
    def show1(self):
        print("Class C method")
obj1=C()
obj1.show1()
obj2=B()
obj2.show1()
```

```
PS D:\pythonfolder
Class C method
class B method
Class A method
```

```python
#polymorphism in python
#method overloading
def abc(x,y,z=2):
    return x+y+z
print(abc(1,2,3))
print(abc(1,3))
#calling method with different
#types of arguments
```

```
PS D:\py
6
6
```

Encapsulation in python

inherit1.py > B > __init__

```python
#encapsulation in python
#access specifiers
#a=10 public
#_a=10 protected
#__a=10 private
class A:
    def __init__(self):
        self.a=10 #public
        self._b=11  #protected
        self.__c=12 #private
class B(A):
    def __init__(self):
        A.__init__(self)
        print("Priniting public member of class A:",self.a) #accessed inside and outside the class
        print("Printing the protected member of class A:",self._b)#accessed inside and outside the class
        print("Printing the protected member of class A:",self.__c) #shows error because can be accessed only inside the class


obj=B()
print(obj._A__c) #this Terminal (Ctrl+`) access the private member also
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\pythonfolder> & C:/Users/maany/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/pythonfolder/inherit1.py
Priniting public member of class A: 10
Printing the protected member of class A: 11
Traceback (most recent call last):
  File "d:\pythonfolder\inherit1.py", line 19, in <module>
    obj=B()
        ^^^
  File "d:\pythonfolder\inherit1.py", line 16, in __init__
    print("Printing the protected member of class A:",self.__c) #shows error because can be accessed only inside the class
                                                      ^^^^^^^^
AttributeError: 'B' object has no attribute '_B__c'. Did you mean: '_A__c'?
PS D:\pythonfolder> & C:/Users/maany/AppData/Local/Microsoft/WindowsApps/python3.12.exe d:/pythonfolder/inherit1.py
Priniting public member of class A: 10
Printing the protected member of class A: 11
12
```