**EXPERIMENT NO : 3C**

**Programs to implement Polymorphism with Method Overloading and Method Overriding.**

**NAME : AKASH RAMKRIT YADAV ID.NO: VU4F2122016**

**BATCH : A BRANCH : IT DIV : A**

**Aim :- Python Programs to Implement Polymorphism with Method**

**Overloading and Method Overriding.**

***THEORY:***

***OUTPUT:***

*Python 3.11.0a4 (main, Mar 1 2023, 10:57:32) [MSC v.1929 32 bit (Intel)] on win32*

*Type "help", "copyright", "credits" or "license()" for more information.*

*#AKASH YADAV ID.NO:VU4F2122016*   *EXP:3C DATE:1/3/2023*

***Python :Method Overloading & Method overriding***

***Method Overloading:***

*Method Overloading is an example of Compile time polymorphism.*

*In this, more than one method of the same class shares the same method name having different signatures.*

*Two or more methods have the same name but different numbers of parameters or different types of parameters, or both.*

*These methods are called overloaded methods and this is called method overloading.*

*The problem with method overloading in Python is that we may overload the methods but can only use the latest defined method.*

***EXAMPLE:***

***1]***

*#First MULTI method.*

*# Takes two argument and print their*

*# MULTI*

*def multi(a,b):*

*m=a\*b*

*print(m)*

*# Second product method*

*# Takes three argument and print their*

*# product*

*def multi(a,b,c):*

*m=a\*b\*c*

*print(m)*

***# Uncommenting the below line shows an error***

*#multi(5,6)*

*#This line will call the second product method*

*multi(3,4,5)*

***>>>*** *60*

*In the above code, we have defined two multi methods we can only use the second multi method, as python does not support method overloading.*

*We may define many methods of the same name and different arguments, but we can only use the latest defined method.*

*Calling the other method will produce an error. Like here calling product(5,6)*

*will produce an error as the latest defined multi method takes three arguments.*

*Thus, to overcome the above problem we can use different ways to achieve the method overloading.*

***2]***

***By Using Multiple Dispatch Decorator***

*Multiple Dispatch Decorator Can be installed by:*

***pip3*** *install multipledispatch*

*if you do not install pip then first following command in your* ***CMD****:*

***1] curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py***

***2] python get-pip.py***

***from multipledispatch import dispatch***

*# passing one parameter*

***@dispatch(int, int)***

*def multi(a1,a2):*

*r=a1\*a2*

*print(r)*

*# passing second parameter*

***@dispatch(int,int,int)***

*def multi(a1,a2,a3):*

*r=a1\*a2\*a3*

*print(r)*

*# passing Third parameter*

***@dispatch(float,float,float)***

*def multi(a1,a2,a3):*

*r=a1\*a2\*a3*

*print(r)*

*# calling multi method with 2 arguments*

*multi(4,5)*

*# calling multi method with 3 arguments but all int*

*multi(6,8,7)*

*# calling multi method with 3 arguments but all float*

*multi(5.5,5.7,8.5)*

***>>>****20*

*336*

*266.475*

***Method overriding***

*Method overriding is an example of run time polymorphism.*

*In this, the specific implementation of the method that is already provided by the parent class is provided by the child class.*

*It is used to change the behavior of existing methods and there is a need for at least two classes for method overriding.*

*In method overriding, inheritance always required as it is done between parent class(superclass) and child class(child class) methods.*

***EXAMPLE:***

*class* ***AKASH****:*

*def fun1(self):*

*print('feature\_1 of class AKASH')*

*def fun2(self):*

*print('feature\_2 of class AKASH')*

*class* ***YADAV****(AKASH):*

*# Modified function that is*

*# already exist in class AKASH*

*def fun1(self):*

*print('Modified feature\_1 of class AKASH by class YADAV')*

*def fun3(self):*

*print('feature\_3 of class YADAV')*

*# Create instance*

*obj = YADAV()*

*# Call the override function*

*obj.fun1()*

***>>>*** *Modified feature\_1 of class* ***AKASH*** *by class* ***YADAV***