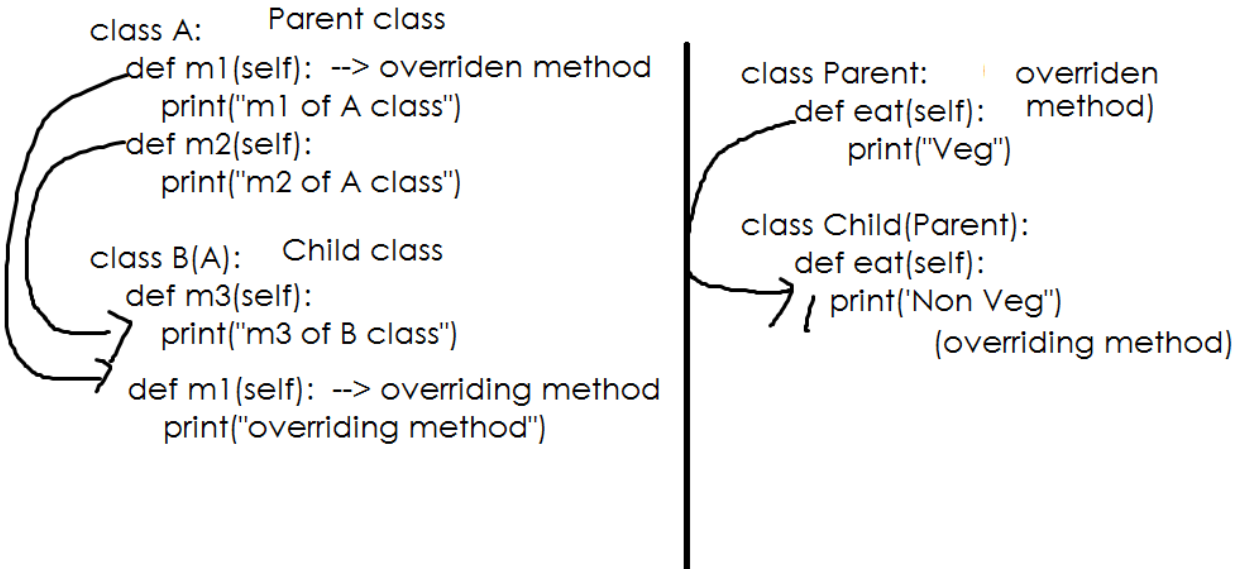
Method Overriding

Redefining of base class object level method within derived class is called method overriding.

Defining method in derived class with same name as method exists in base class is called method overriding.

Method overriding allows modifying or extending functionality of base class method within derived class.



Example:

```
class A:
    def m1(self):
        print("m1 of A")
    def m2(self):
        print("m2 of A")
class B(A):
    def m3(self):
        print("m3 of B")
    def m1(self): # overridig method
        print("overriding method")

def main():
    objb=B()
    objb.m1()
    objb.m2()
    objb.m3()
```

```
main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest38.py =======
overriding method
m2 of A
m3 of B
```

Example:

```
class Employee:
    def __init__(self):
        self.__empno=None
        self.__ename=None
    def read(self):
        self.__empno=int(input("EmployeeNo"))
        self.__ename=input("EmployeeName")
    def print_emp(self):
        print(f'EmployeeNo {self.__empno}')
        print(f'EmployeeName {self.__ename}')
class SalaryEmployee(Employee):
    def __init__(self):
        super().__init__()
        self.__salary=None
    def read(self): # overriding method
        super().read()
        self.__salary=float(input("Salary"))
    def print_emp(self): # overriding method
        super().print_emp()
        print(f'Salary {self.__salary}')
def main():
    emp1=SalaryEmployee()
    emp1.read()
    emp1.print_emp()

main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest39.py =======
EmployeeNo101
```

EmployeeNameNaresh
Salary5000
EmployeeNo 101
EmployeeName Naresh
Salary 5000.0

What is polymorphism is python?

The literal meaning of polymorphism is the condition of occurrence in different forms. Polymorphism is a very important concept in programming. It refers to the use of a single type entity (method, operator or object) to represent different types in different scenario.

In Python, Polymorphism lets us define methods in the child class that have the same name as the methods in the parent class.

The word “polymorphism” means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

In python polymorphism is achieved using method overriding and abstract classes and abstract methods

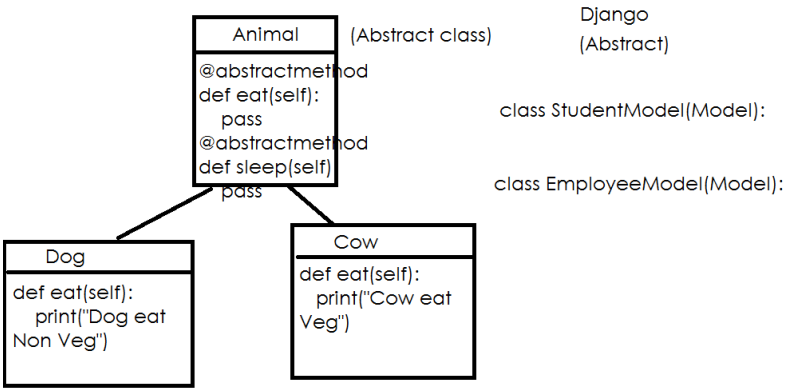
Abstract classes and abstract methods (abc module)

“abc” module, is a default module which comes with python software. using abc module we can implement abstract classes and abstract methods.

What is an abstract method?
Abstract method is an object level method without implementation in base class (OR) abstract method is empty method (OR) without body.
Abstract method defines specification which has to be implemented by every derived class.
Abstract method defines behavior of derived class without defining implementation details.
Abstract method defines a protocol.

Syntax:

```
@abc.abstractmethod
def <method-name>(self,arg1,arg2,..):
        pass
```

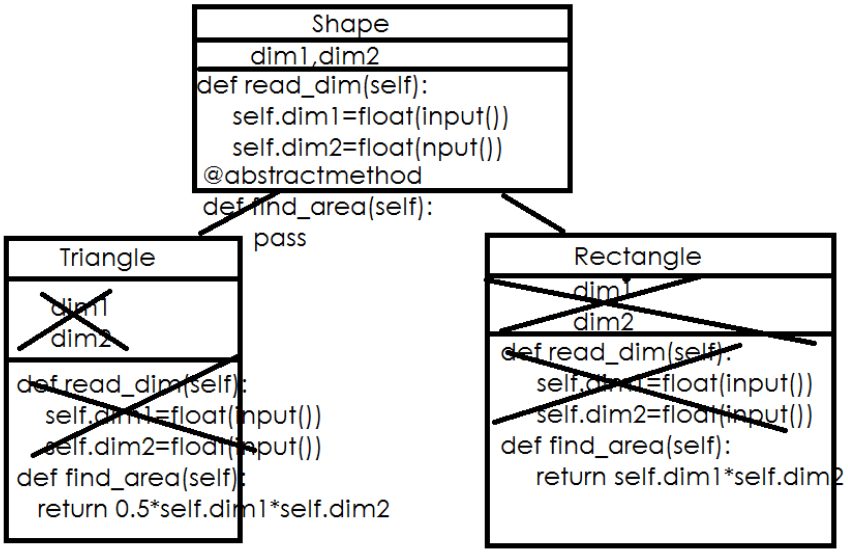


Abstract methods are defined inside abstract base class or abstract class.

Syntax:

```
class <class-name>(abc.ABC):
        abstract methods
        non abstract methods
```

abstract based class inherited from ABC class of abc module.
abstract class is inherited but not used for creating object.
Abstract class defines set of rules or regulation which to be followed every derived class. Abstract class is used to building similar type of classes.



Example:

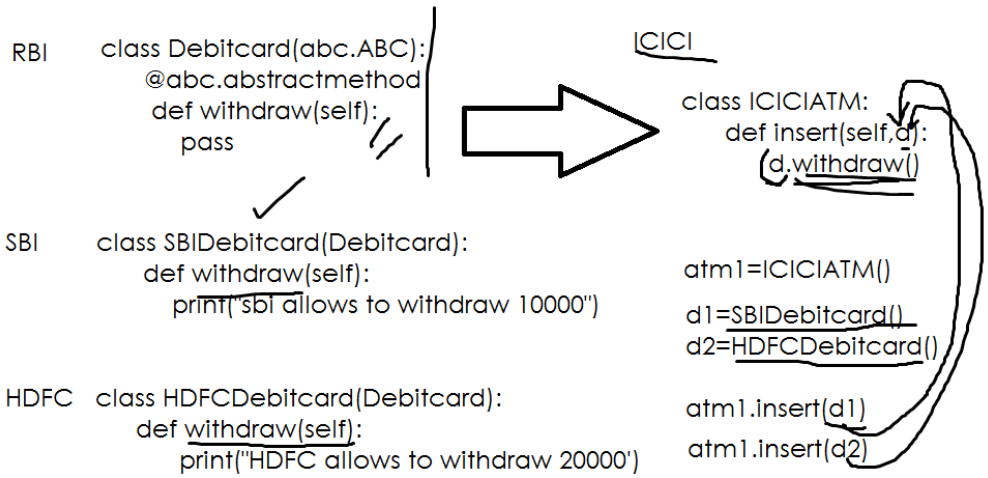
```
import abc
class Shape(abc.ABC): # abstract class or ADT
    def __init__(self):
        self._dim1=None
        self._dim2=None
    def read_dim(self):
        self._dim1=float(input("Dim1"))
        self._dim2=float(input("Dim2"))
    @abc.abstractmethod
    def find_area(self):
        pass

class Triangle(Shape):
    def __init__(self):
        super().__init__()
    def find_area(self): # overriding method
        return self._dim1*self._dim2*0.5
class Rectangle(Shape):
    def __init__(self):
        super().__init__()
    def find_area(self): # overriding method
        return self._dim1*self._dim2

def main():
    t1=Triangle()
```

```
    r1=Rectangle()
    t1.read_dim()
    r1.read_dim()
    area1=t1.find_area()
    area2=r1.find_area()
    print(f'area of triangle {area1:.2f}')
    print(f'area of rectangle {area2:.2f}')
main()
```

Output:

```
Dim11.2
Dim21.3
Dim11.5
Dim21.6
area of triangle 0.78
area of rectangle 2.40
```

Abstraction is hiding implementation details by giving only specifications.



Example:

```
import abc
class Debitcard(abc.ABC):
    @abc.abstractmethod
    def withdraw(self):
        pass

class SBIDebitcard(Debitcard):
    def withdraw(self): # overriding method
```

```
        print("withdraw of SBI")

class HDFCDebitcard(Debitcard):
    def withdraw(self): # overriding method
        print("withdraw of HDFC")
class ICICIATM:
    def insert(self,d):
        d.withdraw()
def main():
    atm1=ICICIATM()
    card1=SBIDebitcard()
    card2=HDFCDebitcard()
    atm1.insert(card1)
    atm1.insert(card2)
main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest41.py ========
withdraw of SBI
withdraw of HDFC
```

What is runtime polymorphism?

An ability of a reference variable change its behavior based on the type of object assigned is called runtime polymorphism.

This allows developing loosely coupled code. the code which work with any type is called loosely coupled code.

What is duck typing?

A programming style which does not look at an object's type to determine if it has the right interface; instead, the method or attribute is simply called or used ("If it looks like a duck and quacks like a duck, it must be a duck.")

Example:

```
class A:
    def m1(self):
        print("m1 of A class")


class B:
```

```
    def m1(self):
        print("m1 of B class")

def call(obj):
    obj.m1()


def main():
    obja=A()
    objb=B()
    call(obja)
    call(objb)

main()
```

Output:

```
======== RESTART: F:/python6pmaug/ooptest42.py ========
m1 of A class
m1 of B class
```

What is MRO?

MRO stands for Method Resolution Order