Student Name: ____________________ **Roll No:** ______________ **Section:** ____________

Lab No. 07

Lab 07 – Introduction to Abstract Classes

Objectives:

- Introduction to Abstract Classes
- Why use Abstract Base Classes
- What is Abstract classes in python
- Abstract and Concrete Methods

1. Introduction to Abstract Classes

A class is called an Abstract class if it contains one or more abstract methods. An abstract method is a method that is declared, but contains no implementation. Abstract classes may not be instantiated, and its abstract methods must be implemented by its subclasses.

2. Abstract Classes in Python

An abstract class can be considered as a blueprint for other classes. It allows you to create a set of methods that must be created within any child classes built from the abstract class. A class which contains one or more abstract methods is called an abstract class. An abstract method is a method that has a declaration but does not have an implementation. While we are designing large functional units we use an abstract class. When we want to provide a common interface for different implementations of a component, we use an abstract class.

Example:

People do not think of a car as a set of thousands of individual parts. Instead they see it as a well-defined object with its own unique behavior. This abstraction allows people to use a car to drive without knowing the complexity of the parts that form the car. They can ignore the details of how the engine transmission, and braking systems work. Instead, they are free to utilize the object as a whole

- Abstract classes are classes that contain one or more abstract methods.
- An abstract method is a method that is declared, but contains no implementation.
- Abstract classes cannot be instantiated, and require subclasses to provide implementations for the abstract methods.
- Abstract classes may not be instantiated, and its abstract methods must be implemented by its subclasses

3. Abstract Base Class in Python

1. An abstract class can be considered as a blueprint for other classes.
2. It allows you to create a set of methods that must be created within any child classes built

Student Name: ___________________________ Roll No: ________________ Section: ______________

from the abstract class.

3. A class which contains one or more abstract methods is called an abstract class.
4. An abstract method is a method that has a declaration but does not have an implementation.
5. While we are designing large functional units we use an abstract class.
6. When we want to provide a common interface for different implementations of a component, we use an abstract class.

4. Why to use ABC?

A class is called an Abstract class if it contains one or more abstract methods. An abstract method is a method that is declared, but contains no implementation

An abstract base class is a class from which we want to create subclasses, but the base class is not something we can instantiate (create). One canonical example is a vehicle. A vehicle is an abstraction, whereas cars and motorcycles are specific examples of vehicles. We would never want to create a "vehicle" in our code, but we certainly might want to create cars, motorcycles, bicycles, trucks, and other kinds of vehicles.

5. How Abstract Base classes work :

By default, Python does not provide abstract classes. Python comes with a module which provides the base for defining Abstract Base classes(ABC) and that module name is ABC. ABC works by decorating methods of the base class as abstract and then registering concrete classes as implementations of the abstract base. A method becomes abstract when decorated with the keyword @abstractmethod

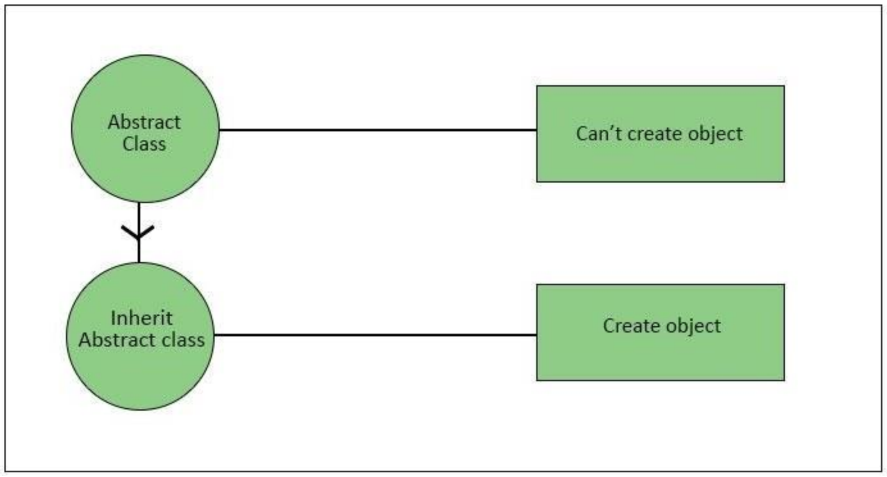
Abstract Classes In Python

```
Syntax

from abc import ABC

Class ClassName(ABC):
```

Student Name: ____________________________ **Roll No:** _________________ **Section:** _______________



Exercise 1: Write a program Create an ABC class of Polygon

```
from abc import ABC, abstractmethod

class Polygon(ABC):

     # abstract method
     def noofsides(self):
          pass

class Triangle(Polygon):

     # overriding abstract method
     def noofsides(self):
          print("I have 3 sides")
class Pentagon(Polygon):

     # overriding abstract method
     def noofsides(self):
          print("I have 5 sides")

class Hexagon(Polygon):

     # overriding abstract method
     def noofsides(self):
          print("I have 6 sides")

class Quadrilateral(Polygon):
# overriding abstract method
     def noofsides(self):
          print("I have 4 sides")


# Driver code
R = Triangle()
R.noofsides()

K = Quadrilateral()
K.noofsides()

R = Pentagon()
R.noofsides()

K = Hexagon()
K.noofsides()
```

Output:

Exercise 2: Write a code that will create Animal and add some abstract methods

```
from abc import ABC, abstractmethod
class Animal(ABC):

    def move(self):
        pass


class Snake(Animal):

    def move(self):
        print("I can crawl")

class Dog(Animal):

    def move(self):
```

Student Name: ____________________________ Roll No: _________________ Section: _______________

```
        print("I can bark")

class Lion(Animal):

    def move(self):
        print("I can roar")

# Driver code


K = Snake()
K.move()

R = Dog()
R.move()

K = Lion()
K.move()
```

Output:

6. Concrete Methods in Abstract Base Classes

Concrete classes contain only concrete (normal)methods whereas abstract classes may contains both concrete methods and abstract methods. Concrete class provide an implementation of abstract methods, the abstract base class can also provide an implementation by invoking the methods via super().
Let look over the example to invoke the method using super():

Student Name: ____________________________ **Roll No:** _________________ **Section:** _______________

Exercise 3 : Write a code that will create abstract class access the methods by using super().

```
# Python program invoking a
# method using super()

from abc import ABC, abstractmethod
class Person(ABC):
    def ShowName(self):
        print("Abstract Base Class")

class Student(Person):
    def ShowName(self):
        super().ShowName()
        print("subclass ")

# Driver code
S1 = Student()
S1.ShowName()
```

Output:

Student Name: ____________________________ **Roll No:** _________________ **Section:** _______________

Programming Exercise (Python)

Task 1: Discuss in detail

- What is Abstraction in Python?
- How can we achieve Abstraction in Python?
- Mention the name of the module to be imported for an abstract class

Task 2:

Create an ABC class of Bank and add some abstract method AccountName, rate of interest, deposit, withdraw,

Now add some classes in which you will implement Bank abstract class and its methods.

Task 3:

Find out one real world example of abstract class and abstract method and implement it by using python code