Lab Practical Exercise: Python Abstract Classes and ABC Module (Step-by-Step)

Lab Title

Working with Abstract Classes and ABC in Python

Objective

By completing this lab, students will learn:

- What an abstract class is
- How to import and use the `abc` module
- How to create abstract methods
- How to inherit from an abstract class
- How to implement abstract methods in child classes
- How abstract classes improve program structure

Pre-Lab Requirements

Before starting, students should know:

- Python classes and objects
- Constructors `__init__()`
- Inheritance
- Functions and methods

Software Required

- Python installed on computer
- VS Code / PyCharm / IDLE / Jupyter Notebook

Important Theory

What is an Abstract Class?

An abstract class is a parent class used as a blueprint. It contains one or more methods that must be completed in child classes.

Why Use It?

Use abstract classes when multiple classes should follow the same structure.

Example:

- Animal classes must have `sound()`
- Shape classes must have `area()`
- Employee classes must have `salary()`

Step 1: Create Python File

Create a new Python file:

```
lab_abstract.py
```

Step 2: Import Required Module

At the top of the file, import:

```
ABC
abstractmethod
```

from Python `abc` module.

Step 3: Create Parent Abstract Class

Create a class named:

```
Animal
```

Make this class inherit from `ABC`.

Step 4: Create Abstract Method

Inside `Animal` class, create a method named:

```
sound()
```

Rules:

- Use `@abstractmethod`

- Keep method body empty using `pass`

Step 5: Create Child Class Dog

Create a new class:

```
Dog
```

This class must inherit from `Animal`.

Now override `sound()` method.

Inside method display:

```
Dog barks
```

Step 6: Create Child Class Cat

Create another child class:

```
Cat
```

This class also inherits from `Animal`.

Override `sound()` method.

Inside method display:

```
Cat meows
```

Step 7: Create Objects

Create objects of:

- Dog
- Cat

Store in variables.

Example:

```
d
c
```

Step 8: Call Methods

Call `sound()` method using both objects.

Expected behavior:

- Dog object prints barking sound
- Cat object prints meow sound

Step 9: Run Program

Run the file and observe output.

Step 10: Test Important Rule

Try creating object of parent abstract class:

```
Animal()
```

Observe the error message.

Write in notebook:

Why did the error occur?

Practical Exercise 2: Shape Program

Task

Create abstract class:

```
Shape
```

Create abstract method:

```
area()
```

Create Child Classes

Rectangle

Use constructor to receive:

- length
- width

Area formula:

```
length × width
```

Circle

Use constructor to receive:

- radius

Area formula:

Use `math.pi`

Final Steps

- Create object of Rectangle
- Create object of Circle
- Print both areas

Practical Exercise 3: Employee Salary System

Task

Create abstract class:

```
Employee
```

Create abstract method:

```
salary()
```

Create Child Classes

FullTimeEmployee

Return fixed salary.

PartTimeEmployee

Return smaller salary.

Final Steps

- Create both objects
- Print salaries

Challenge Exercise

Create abstract class:

```
Vehicle
```

Abstract method:

```
start()
```

Create child classes:

- Car
- Bike
- Bus

Each should print different start message.

Output Format Example

```
Dog barks
Cat meows
Rectangle Area: ...
Circle Area: ...
Full Time Salary: ...
```

Lab Report Questions

1. What is abstract class?
2. Why do we use `ABC`?
3. What is `@abstractmethod`?
4. Can we create object of abstract class?
5. What happens if child class does not define abstract method?

Submission Requirements

Students must submit:

- Python file
- Screenshot of output
- Answers of lab questions

Teacher Evaluation Rubric

Task	Marks
Correct use of ABC	3
Correct inheritance	3
Correct methods	4
Output working	5
Code formatting	2
Total	17
