Practical-15

May 19, 2025

0.1 Practical 15 :-

Name: Sarthak Sanay
Enrollment No: 230031101611051

0.1.1 Problem Statement 1:-

Create a base class Shape with methods to calculate area and perimeter. Then create subclasses Rectangle and Circle that inherit from Shape and implement their specific area and perimeter calculation methods.

[1]:
```
import math

# Base class
class Shape:
    def area(self):
        return 0

    def perimeter(self):
        return 0

# Rectangle subclass
class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

    def perimeter(self):
        return 2 * (self.width + self.height)

# Circle subclass
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
```

```
        return math.pi * self.radius * self.radius

    def perimter(self):
        return 2 * math.pi * self.radius

print("Choose shape:-")
print("Enter 1 for Rectangle, or 2 for Circle.")
choice = int(input("Enter choice: "))

if choice == 1:
    w = float(input("Enter width: "))
    h = float(input("Enter height: "))
    rect = Rectangle(w, h)
    print(f"Rectangle Area: {rect.area()}")
    print(f"Rectangle Perimeter: {rect.perimeter()}")

elif choice == '2':
    r = float(input("Enter radius: "))
    circ = Circle(r)
    print(f"Circle Area: {circ.area():.2f}")
    print(f"Circle Perimeter: {circ.perimeter():.2f}")

else:
    print("Invalid choice.")
```

```
Choose shape:-
Enter 1 for Rectangle, or 2 for Circle.

Enter choice:  1
Enter width:  15
Enter height:  14

Rectangle Area: 210.0
Rectangle Perimeter: 58.0
```

0.1.2 Problem Statement 2:-

Create a base class Vehicle with attributes like make, model, and year, and a method to display vehicle information. Then create subclasses of Car and Motorcycle that are inherited from the Vehicle and add their specific attributes like seating capacity or type of engine.

[1]:
```
# Base class
class Vehicle:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def display_info(self):
```

```
        print(f"Make: {self.make}")
        print(f"Model: {self.model}")
        print(f"Year: {self.year}")

# Car subclass
class Car(Vehicle):
    def __init__(self, make, model, year, seating_capacity):
        super().__init__(make, model, year)
        self.seating_capacity = seating_capacity

    def display_info(self):
        super().display_info()
        print(f"Seating Capacity: {self.seating_capacity}")

# Motorcycle subclass
class Motorcycle(Vehicle):
    def __init__(self, make, model, year, engine_type):
        super().__init__(make, model, year)
        self.engine_type = engine_type

    def display_info(self):
        super().display_info()
        print(f"Engine Type: {self.engine_type}")

print("Choose Vehicle Type:-")
print("Enter 1 for Car, or 2 for Motorcycle.")
choice = int(input("Enter choice: "))

if choice == 1:
    make = input("Enter make: ")
    model = input("Enter model: ")
    year = input("Enter year: ")
    capacity = input("Enter seating capacity: ")
    car = Car(make, model, year, capacity)
    print("\nCar Details:")
    car.display_info()

elif choice == 2:
    make = input("Enter make: ")
    model = input("Enter model: ")
    year = input("Enter year: ")
    engine = input("Enter engine type: ")
    bike = Motorcycle(make, model, year, engine)
    print("\nMotorcycle Details:")
    bike.display_info()

else:
```

```
    print("Invalid choice.")
```

```
Choose Vehicle Type:-
Enter 1 for Car, or 2 for Motorcycle.

Enter choice:  1
Enter make:  Tata
Enter model:  Safari XZ Plus
Enter year:  2024
Enter seating capacity:  7


Car Details:
Make: Tata
Model: Safari XZ Plus
Year: 2024
Seating Capacity: 7
```

0.1.3 Problem Statement 3:-

Create a base class Employee with attributes like name, ID, and salary, and methods to calculate and display details. Then create subclasses Manager and Developer that inherit from Employee and add attributes like department or programming language.

[3]:
```python
# Base class
class Employee:
    def __init__(self, name, emp_id, salary):
        self.name = name
        self.emp_id = emp_id
        self.salary = salary

    def display_details(self):
        print(f"Name     : {self.name}")
        print(f"ID       : {self.emp_id}")
        print(f"Salary   :  {self.salary}")

# Subclass: Manager
class Manager(Employee):
    def __init__(self, name, emp_id, salary, department):
        super().__init__(name, emp_id, salary)
        self.department = department

    def display_details(self):
        super().display_details()
        print(f"Department: {self.department}")

# Subclass: Developer
class Developer(Employee):
    def __init__(self, name, emp_id, salary, language):
```

```
        super().__init__(name, emp_id, salary)
        self.language = language

    def display_details(self):
        super().display_details()
        print(f"Programming Language: {self.language}")

# Sample usage
print("Select Employee Type:")
print("1. Manager")
print("2. Developer")
choice = input("Enter 1 or 2: ")

name = input("Enter name: ")
emp_id = input("Enter ID: ")
salary = float(input("Enter salary: "))

if choice == '1':
    dept = input("Enter department: ")
    m = Manager(name, emp_id, salary, dept)
    print("\nManager Details:")
    m.display_details()

elif choice == '2':
    lang = input("Enter programming language: ")
    d = Developer(name, emp_id, salary, lang)
    print("\nDeveloper Details:")
    d.display_details()

else:
    print("Invalid choice.")
```

```
Select Employee Type:
1. Manager
2. Developer

Enter 1 or 2:  2
Enter name:  Sarthak
Enter ID:  DEV51
Enter salary:  2165000
Enter programming language:  Python, Java, C, C++, Kotlin, Dart, Bash


Developer Details:
Name     : Sarthak
ID       : DEV51
Salary   : 2165000.0
Programming Language: Python, Java, C, C++, Kotlin, Dart, Bash
```

0.1.4 Problem Statement 4:-

Create a base class Animal with methods to make sound and display species. Then create subclasses Dog, Cat, and Bird that inherit from Animal and implement their specific sound methods.

[5]:
```
# Base class
class Animal:
    def __init__(self, species):
        self.species = species

    def make_sound(self):
        print("Some animal sound.")

    def display_species(self):
        print(f"Species: {self.species}")

# Dog subclass
class Dog(Animal):
    def __init__(self):
        super().__init__("Dog")

    def make_sound(self):
        print("Woof! Woof!")

# Cat subclass
class Cat(Animal):
    def __init__(self):
        super().__init__("Cat")

    def make_sound(self):
        print("Meow!")

# Bird subclass
class Bird(Animal):
    def __init__(self):
        super().__init__("Bird")

    def make_sound(self):
        print("Tweet! Tweet!")

# Sample usage
print("Choose Animal:")
print("Enter 1 for Dog, 2 for Cat, or 3 for Bird.")
choice = input("Enter choice: ")

if choice == '1':
    a = Dog()
elif choice == '2':
```

```
    a = Cat()
elif choice == '3':
    a = Bird()
else:
    print("Invalid choice.")
    exit()

print("\nAnimal Details:-")
a.display_species()
a.make_sound()
```

```
Choose Animal:
Enter 1 for Dog, 2 for Cat, or 3 for Bird.

Enter choice:  2


Animal Details:-
Species: Cat
Meow!
```

0.1.5 Problem Statement 5:-

Create a base class Account with methods for deposit, withdraw, and display balance. Then create subclasses SavingsAccount and CheckingAccount that inherit from Account and implement their specific interest calculation or overdraft protection methods.

[11]:
```
# Base class
class Account:
    def __init__(self, acc_number, holder_name, balance=0):
        self.acc_number = acc_number
        self.holder_name = holder_name
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f" {amount} deposited.")

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f" {amount} withdrawn.")
        else:
            print("Insufficient balance.")

    def display_balance(self):
        print(f"Account Holder: {self.holder_name}")
        print(f"Account Number: {self.acc_number}")
        print(f"Balance:  {self.balance}")
```

```
# SavingsAccount subclass
class SavingsAccount(Account):
    def __init__(self, acc_number, holder_name, balance=0, interest_rate=0.05):
        super().__init__(acc_number, holder_name, balance)
        self.interest_rate = interest_rate

    def apply_interest(self):
        interest = self.balance * self.interest_rate
        self.balance += interest
        print(f"Interest of  {interest:.2f} applied at {self.
 ↪interest_rate*100}% rate.")

# CheckingAccount subclass
class CheckingAccount(Account):
    def __init__(self, acc_number, holder_name, balance=0,␣
 ↪overdraft_limit=1000):
        super().__init__(acc_number, holder_name, balance)
        self.overdraft_limit = overdraft_limit

    def withdraw(self, amount):
        if amount <= self.balance + self.overdraft_limit:
            self.balance -= amount
            print(f" {amount} withdrawn (with overdraft if needed).")
        else:
            print("Withdrawal exceeds overdraft limit.")


print("Choose account type:-")
print("Enter 1. for Savings Account, or 2. for Checking Account.")
choice = int(input("Enter choice: "))

acc_number = input("Enter account number: ")
holder_name = input("Enter account holder name: ")
initial_balance = float(input("Enter initial balance: "))

if choice == 1:
    acc = SavingsAccount(acc_number, holder_name, initial_balance)
    acc.deposit(500)
    acc.apply_interest()
    acc.withdraw(200)
    print("\nAccount Summary:")
    acc.display_balance()

elif choice == 2:
    acc = CheckingAccount(acc_number, holder_name, initial_balance)
    acc.deposit(300)
```

```
    acc.withdraw(1500)  # Test overdraft
    print("\nAccount Summary:")
    acc.display_balance()

else:
    print("Invalid account type.")
```

```
Choose account type:-
Enter 1. for Savings Account, or 2. for Checking Account.

Enter choice:  2
Enter account number:  007
Enter account holder name:  James Bond
Enter initial balance:  700

 300 deposited.
 1500 withdrawn (with overdraft if needed).

Account Summary:
Account Holder: James Bond
Account Number: 007
Balance:  -500.0
```

[ ]: