**RIPHAH INTERNATIONAL UNIVERSITY, ISLAMABAD**

****

**Lab 5**

**Bachelors of Computer science – 6th semester**

**Subject:** Artificial Intelligence Lab

**Submitted to:** Ma’am Ayesha

**Submitted by:** Areeba Sadaqat

**Sap Id:** 47633

**Date:** 4h March, 2025

**Task 1:** **Define a function that accepts roll number and returns whether the student is present or absent.**

1. **def** is\_present(roll\_number, present\_students):
2. return "Present" if roll\_number in present\_students else "Absent"
3. *# Example usage:*
4. present\_students = {101, 102, 103}  *# Set of present students*
5. print(is\_present(101, present\_students))
6. print(is\_present(105, present\_students))

def is\_present(roll\_number, present\_students):

return "Present" if roll\_number in present\_students else "Absent"

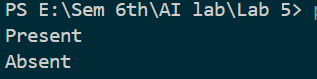
# Example usage:

present\_students = {101, 102, 103} # Set of present students

print(is\_present(101, present\_students))

print(is\_present(105, present\_students))

**Output:**

****

**Task 2: Define a class and create multiple object of class, access attributes and assign new values.**

Class with multiple objects

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

# Creating objects

p1 = Person("Alice", 25)

p2 = Person("Bob", 30)

# Accessing and modifying attributes

print(p1.name, p1.age)

p1.age = 26

print(p1.age)

*# 2. Class with multiple objects*

**class** Person:

**def** \_\_init\_\_(self, name, age):

        self.name = name

        self.age = age

*# Creating objects*

p1 = Person("Alice", 25)

p2 = Person("Bob", 30)

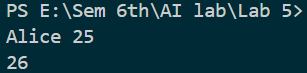
*# Accessing and modifying attributes*

print(p1.name, p1.age)

p1.age = 26

print(p1.age)

**Output:**

****

**Task 3:** **Create a student class with attributes name, age, and grades (list). Add a method average grade that calculates and returns the average of the grades.**

Student class with average grade method

class Student:

def \_\_init\_\_(self, name, age, grades):

self.name = name

self.age = age

self.grades = grades

def average\_grade(self):

return sum(self.grades) / len(self.grades)

# Example usage:

s1 = Student("John", 20, [85, 90, 78])

print(s1.average\_grade())

*# 3. Student class with average grade method*

**class** Student:

**def** \_\_init\_\_(self, name, age, grades):

        self.name = name

        self.age = age

        self.grades = grades

**def** average\_grade(self):

        return sum(self.grades) / len(self.grades)

*# Example usage:*

s1 = Student("John", 20, [85, 90, 78])

print(s1.average\_grade())

**Output:**

****

**Task 4:** Create a base class **Employee** with:

* name
* salary
* Method **display\_details()** to show employee info.

Create two subclasses:

1. **Manager** (inherits Employee) and has an additional attribute **department**
2. **Developer** (inherits Employee) and has an additional attribute **programming\_language**

Override the **display\_details()** method in both subclasses to include their specific attributes.

Employee, Manager, and Developer classes

class Employee:

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

def display\_details(self):

print(f"Name: {self.name}, Salary: {self.salary}")

class Manager(Employee):

def \_\_init\_\_(self, name, salary, department):

super().\_\_init\_\_(name, salary)

self.department = department

def display\_details(self):

print(f"Name: {self.name}, Salary: {self.salary}, Department: {self.department}")

class Developer(Employee):

def \_\_init\_\_(self, name, salary, programming\_language):

super().\_\_init\_\_(name, salary)

self.programming\_language = programming\_language

def display\_details(self):

print(f"Name: {self.name}, Salary: {self.salary}, Programming Language: {self.programming\_language}")

# Example usage:

m = Manager("Alice", 60000, "HR")

d = Developer("Bob", 70000, "Python")

m.display\_details()

d.display\_details()

*# 4. Employee, Manager, and Developer classes*

**class** Employee:

**def** \_\_init\_\_(self, name, salary):

        self.name = name

        self.salary = salary

**def** display\_details(self):

        print(**f**"Name: {self.name}, Salary: {self.salary}")

**class** Manager(Employee):

**def** \_\_init\_\_(self, name, salary, department):

        super().\_\_init\_\_(name, salary)

        self.department = department

**def** display\_details(self):

        print(**f**"Name: {self.name}, Salary: {self.salary}, Department: {self.department}")

**class** Developer(Employee):

**def** \_\_init\_\_(self, name, salary, programming\_language):

        super().\_\_init\_\_(name, salary)

        self.programming\_language = programming\_language

**def** display\_details(self):

        print(**f**"Name: {self.name}, Salary: {self.salary}, Programming Language: {self.programming\_language}")

*# Example usage:*

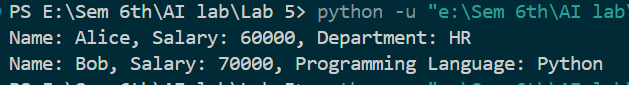
m = Manager("Alice", 60000, "HR")

d = Developer("Bob", 70000, "Python")

m.display\_details()

d.display\_details()

**Output:**

****

1. **Task 5:** Create a base class **Shape** with a method **area()**.

Create the following subclasses:

* **Circle** (takes radius and implements area() as π \* r²)
* **Rectangle** (takes length and width and implements area() as length × width)
* **Triangle** (takes base and height and implements area() as 0.5 × base × height)

Use **polymorphism** to calculate the area of different shapes.

Shape and its subclasses

import math

class Shape:

def area(self):

pass

class Circle(Shape):

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return math.pi \* self.radius \*\* 2

class Rectangle(Shape):

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

def area(self):

return self.length \* self.width

class Triangle(Shape):

def \_\_init\_\_(self, base, height):

self.base = base

self.height = height

def area(self):

return 0.5 \* self.base \* self.height

# Example usage:

shapes = [Circle(5), Rectangle(4, 6), Triangle(3, 7)]

for shape in shapes:

print(f"Area: {shape.area()}")

import math

**class** Shape:

**def** area(self):

        pass

**class** Circle(Shape):

**def** \_\_init\_\_(self, radius):

        self.radius = radius

**def** area(self):

        return math.pi \* self.radius \*\* 2

**class** Rectangle(Shape):

**def** \_\_init\_\_(self, length, width):

        self.length = length

        self.width = width

**def** area(self):

        return self.length \* self.width

**class** Triangle(Shape):

**def** \_\_init\_\_(self, base, height):

        self.base = base

        self.height = height

**def** area(self):

        return 0.5 \* self.base \* self.height

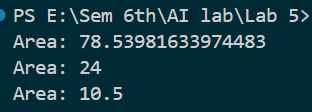
*# Example usage:*

shapes = [Circle(5), Rectangle(4, 6), Triangle(3, 7)]

for shape in shapes:

    print(**f**"Area: {shape.area()}")

**Output:**

****