



LAB REPORT 04

SUBMITTED TO : MR. MUBASHIR IQBAL

SUBMITTED BY: AHMED SALEEM RANA

REG ID: 24-CYS-023

SECTION : B

COURSE TITLE: ARTIFACIAL INTEELLIGENCE LAB

HITEC UNIVERSITY, TAXILA CANTT

TASK 01

Employee Management:

- Create a Person class with attributes/variables “name” and “age”.
- Create a Job class with attributes/variables “designation” and “salary”.
- Create an Employee class that inherits the above both classes
- Implement a method to display employee details

INPUT:

```
main.py  [ ] [ ] [ ] Share Run

1 class Person:
2     def __init__(self, name, age):
3         self.name = name
4         self.age = age
5 class Job:
6     def __init__(self, designation, salary):
7         self.designation = designation
8         self.salary = salary
9 class Employee(Person, Job):
10    def __init__(self, name, age, designation, salary):
11        Person.__init__(self, name, age)
12        Job.__init__(self, designation, salary)
13    def display_details(self):
14        print(f"Employee Name: {self.name}")
15        print(f"Age: {self.age}")
16        print(f"Designation: {self.designation}")
17        print(f"Salary: ${self.salary:,.2f}")
18    if __name__ == "__main__":
19        emp1 = Employee("Ahmed Saleem", 20, "Software Analyst", 50,000)
20        emp1.display_details()
```

OUTPUT:

```
Output  Clear

Employee Name: Ahmed Saleem
Age: 20
Designation: Software Analyst
Salary: $50,000.00

=== Code Execution Successful ===
```

EXPLANATION:

This program demonstrates a simple example of combining data from multiple classes. The Person class stores personal details like name and age, and the Job class stores job-related information such as designation and salary. The Employee class combines both sets of data and displays them using the show_details() method. The program creates a person named “Ali,” aged 25, who works as a “Manager” with a salary of 50,000, and then prints all these details together.

TASK 02

Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

INPUT:

main.py

```
1 import math
2 class Circle:
3     def __init__(self, radius):
4         if radius <= 0:
5             raise ValueError("Radius must be a positive number.")
6         self.radius = radius
7     def area(self):
8         """Returns the area of the circle."""
9         return math.pi * self.radius ** 2
10    def perimeter(self):
11        """Returns the perimeter (circumference) of the circle."""
12        return 2 * math.pi * self.radius
13    def diameter(self):
14        """Returns the diameter of the circle."""
15        return 2 * self.radius
16    def display_details(self):
17        print(f"Circle with Radius: {self.radius}")
18        print(f"Diameter: {self.diameter():.2f}")
19        print(f"Area: {self.area():.2f}")
20        print(f"Perimeter: {self.perimeter():.2f}")
```

OUTPUT:

```
Output Clear  
Enter the radius of the circle: 5  
Circle with Radius: 5.0  
Diameter: 10.00  
Area: 78.54  
Perimeter: 31.42  
  
=== Code Execution Successful ===
```

EXPLANATION:

This program defines a Circle class to calculate the area and perimeter of a circle. The class has two methods: `area(r)` returns the area using the formula $3.14 * r * r$, and `perimeter(r)` returns the perimeter using $2 * 3.14 * r$. The user enters the radius, and the program calls both methods to display the calculated area and perimeter of the circle.

TASK 03

Write a Python class to implement `pow(x, n)`. Do not use built-in `math.pow()` function for this code. Also handle this code for all positive as well as negative numbers.

INPUT:

```
main.py ⌵ ☰ ☼ 🔗 Share Run  
1 - class Power:  
2 - def my_pow(self, x: float, n: int) -> float:  
3 - if n == 0:  
4 - return 1.0  
5 - if x == 0:  
6 - if n > 0:  
7 - return 0.0  
8 - else:  
9 - raise ValueError("Cannot raise 0 to a negative power")  
10 - if n < 0:  
11 - x = 1 / x  
12 - n = -n  
13 - result = 1.0  
14 - while n > 0:  
15 - if n % 2 == 1:  
16 - result *= x  
17 - x *= x  
18 - n //= 2  
19 - return result  
20 - power_obj = Power()  
21 - x = float(input("Enter the base (x): "))
```

OUTPUT:

```
Output Clear  
Enter the base (x): 3  
Enter the exponent (n): 4  
3.0 raised to power 4 is 81.0  
  
=== Code Execution Successful ===
```

EXPLANATION:

This Python program finds the power of a number without using `math.pow()`. The Power class has a method `calculate(x, n)` that multiplies the base `x` by itself `n` times. If `n` is negative, it returns the reciprocal, and if `n` is zero, it returns 1. The user enters the base and exponent, and the program shows the result.

TASK 04

Write a Python code for the following figure

INPUT:

```
main.py ⌵ ☀ 🔗 Share Run  
1 - class Item:  
2 - def setItem(self, name, description, price):  
3 - self.name = name  
4 - self.description = description  
5 - self.price = price  
6 - def viewFullDescription(self):  
7 - print("Item:", self.name)  
8 - print("Description:", self.description)  
9 - print("Price:", self.price)  
10 - def addToShoppingBasket(self):  
11 - print(self.name, "added to cart.")  
12 - def removeFromShoppingBasket(self):  
13 - print(self.name, "removed from cart.")  
14 - class MP3(Item):  
15 - def set_mp3(self, artist, duration):  
16 - self.artist = artist  
17 - self.duration = duration  
18 - def play(self):  
19 - print("Music by", self.artist)  
20 - def download(self):
```

```
main.py  [Icons] [Share] [Run]

21 print("Song duration", self.duration)
22 class DVD(Item):
23     def set_dvd(self, rating, actors):
24         self.rating = rating
25         self.actors = actors
26     def viewTrailer(self):
27         print("Viewing trailer with actors:", self.actors)
28 class Book(Item):
29     def set_book(self, author, numberOfPages, genre):
30         self.author = author
31         self.numberOfPages = numberOfPages
32         self.genre = genre
33     def previewContent(self):
34         print("Previewing content of book by", self.author)
35         print("Book genre:", self.genre)
36 Item1 = Item()
37 Item1.setItem("DVD", "HORROR", 1800)
38 Item1.viewFullDescription()
39 Item1.addToShoppingBasket()
40 Item1.removeFromShoppingBasket()

41 MP3_1 = MP3()
42 MP3_1.set_mp3("ZAIN", 2)
43 MP3_1.play()
44 MP3_1.download()
45 DVD1 = DVD()
46 DVD1.set_dvd("PG-13", "MAJID")
47 DVD1.viewTrailer()
48 Book1 = Book()
49 Book1.set_book("FAIZAN", 689, "THRILLER")
50 Book1.previewContent()
```

OUTPUT:

```
Output [Clear]

Item: DVD
Description: HORROR
Price: 1800
DVD added to cart.
DVD removed from cart.
Music by ZAIN
Song duration 2
Viewing trailer with actors: MAJID
Previewing content of book by FAIZAN
Book genre: THRILLER

=== Code Execution Successful ===
```

EXPLANATION:

This program shows inheritance using classes. The base class Item holds common details like name, description, and price, while the subclasses MP3, DVD, and Book add their own attributes and functions such as play, view trailer, and preview content. It demonstrates how subclasses can extend the features of a parent class to create specific item types.

CONCLUSION:

In conclusion, these programs show how Python can be used to solve simple problems using classes, functions, loops, and conditions. They help understand basic concepts like inheritance, calculations, and data handling in an easy and clear way.

GITHUB REPOSITORY LINK: