

ASSIGNMENT-6.2

Name: Akshaya Nemalipuri


Batch: 37

Roll No: 2303A2441

Lab: 06

Task Description-1 (Classes – Data Validation) :

- Prompt AI to generate a Student class with attributes: name, roll_no, and marks. Add a method is_pass() that returns whether the student has passed (marks \geq 40).



```
[1] ✓ Os
class Student:
    def __init__(self, name, roll_no, marks):
        self.name = name
        self.roll_no = roll_no
        self.marks = marks

    def is_pass(self):
        return self.marks >= 40

Now, let's create a couple of student objects and test the is_pass() method:

[2] ✓ Os
student1 = Student("Alice", "A001", 75)
student2 = Student("Bob", "B002", 35)

print(f"student1.name {student1.roll_no} scored {student1.marks} marks. Passed: {student1.is_pass()}")
print(f"student2.name {student2.roll_no} scored {student2.marks} marks. Passed: {student2.is_pass()}")

Alice (Roll No: A001) scored 75 marks. Passed: True
Bob (Roll No: B002) scored 35 marks. Passed: False
```

Explanation :

- The `__init__` constructor initializes student details.
- The `is_pass()` method checks if marks are greater than or equal to 40.

- Returns True for pass and False for fail.

Analysis

- Code is simple, readable, and correct.
- Validation logic is clearly implemented.
- AI-generated class design follows Python best practices.

Task Description-2 (Loops – Pattern Generation)

- Ask AI to generate a function that prints a right-angled triangle star pattern using a for loop. Then regenerate the same pattern using a while loop.

```
Let's define two functions to print a right-angled triangle star pattern. One will use a for loop, and the other will use a while loop.

def print_star_triangle_for_loop(rows):
    print(f"\nRight-angled triangle using for loop (rows={rows}):")
    for i in range(1, rows + 1):
        print("*" * i)

def print_star_triangle_while_loop(rows):
    print(f"\nRight-angled triangle using while loop (rows={rows}):")
    i = 1
    while i <= rows:
        print("*" * i)
        i += 1

Now, let's test these functions with an example. We'll use 5 rows for both patterns.

num_rows = 5

print_star_triangle_for_loop(num_rows)
print_star_triangle_while_loop(num_rows)

Right-angled triangle using for loop (rows=5):
*
**
***
****
*****

Right-angled triangle using while loop (rows=5):
*
**
***
****
*****
```

Explanation

- The loop counter controls the number of stars printed.
- Both loops produce identical output with different logic structures.

Analysis

- AI correctly used loop conditions.

- Logical flow is easy to understand.
- Demonstrates equivalent solutions using different looping constructs.

Task Description-3 (Conditional Statements – Number Analysis) :

- Ask AI to write a function that checks whether a given number is positive, negative, or zero using if-elif-else. Test the function with multiple inputs.

```
Let's define a function to check if a number is positive, negative, or zero using if-elif-else.
```

```
def check_number_sign(number):  
    if number > 0:  
        return "Positive"  
    elif number < 0:  
        return "Negative"  
    else:  
        return "Zero"
```

```
Now, let's test this function with a few numbers.
```

```
print(f"Number 10 is: {check_number_sign(10)}")  
print(f"Number -5 is: {check_number_sign(-5)}")  
print(f"Number 0 is: {check_number_sign(0)}")  
print(f"Number 3.14 is: {check_number_sign(3.14)}")  
print(f"Number -0.001 is: {check_number_sign(-0.001)}")
```

```
Number 10 is: Positive  
Number -5 is: Negative  
Number 0 is: Zero  
Number 3.14 is: Positive  
Number -0.001 is: Negative
```

Explanation

- Uses if, elif, and else to classify numbers.
- Covers all possible conditions.

Analysis

- Decision logic is complete and correct.
- AI handled edge case (zero) properly.
- Code is concise and readable.

Task Description-4 (Nested Conditionals)

- Generate a function `check_discount(age, is_member)` that determines discount eligibility: • Age $\geq 60 \rightarrow$ Senior discount • Member \rightarrow Additional discount

Use nested if statements.

```
[7]
✓ Os
def check_discount(age, is_member):
    if age >= 60:
        # Senior discount path
        if is_member:
            return "Senior discount + Additional member discount"
        else:
            return "Senior discount"
    else:
        # Not a senior
        if is_member:
            return "Member discount"
        else:
            return "No discount"
```

Now, let's test the `check_discount` function with different combinations of age and membership status.

```
[8]
✓ Os
print(f"Age 65, Member True: {check_discount(65, True)}")
print(f"Age 60, Member False: {check_discount(60, False)}")
print(f"Age 45, Member True: {check_discount(45, True)}")
print(f"Age 30, Member False: {check_discount(30, False)}")
print(f"Age 70, Member False: {check_discount(70, False)}")
print(f"Age 59, Member True: {check_discount(59, True)}")
```

```
...
Age 65, Member True: Senior discount + Additional member discount
Age 60, Member False: Senior discount
Age 45, Member True: Member discount
Age 30, Member False: No discount
Age 70, Member False: Senior discount
Age 59, Member True: Member discount
```

Explanation

- Nested if statements check age first, then membership.
- Clear hierarchical decision flow.

Analysis

- AI correctly implemented nested conditionals.
- Logic is easy to trace and debug.
- Matches real-world discount scenarios.

Task Description-5 (Class – Mathematical Opera)

- Ask AI to create a Circle class with methods to calculate area () and circumference () given the radius.

```
Let's define a Circle class with methods to calculate its area and circumference given a radius.

import math

class Circle:
    def __init__(self, radius):
        if radius <= 0:
            raise ValueError("Radius must be a positive number.")
        self.radius = radius

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

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

Now, let's create a Circle object and test its calculate_area() and calculate_circumference() methods.

circle1 = Circle(5)
print(f"Circle with radius {circle1.radius}:")
print(f"  Area: {circle1.calculate_area():.2f}")
print(f"  Circumference: {circle1.calculate_circumference():.2f}")

circle2 = Circle(10.5)
print(f"\nCircle with radius {circle2.radius}:")
print(f"  Area: {circle2.calculate_area():.2f}")
print(f"  Circumference: {circle2.calculate_circumference():.2f}")

Circle with radius 5:
  Area: 78.54
  Circumference: 31.42

Circle with radius 10.5:
  Area: 346.36
  Circumference: 65.97
```

Explanation

- Uses math.pi for accurate calculations.
- Area and circumference formulas are correctly implemented.
- Class design is reusable.

Analysis

- Mathematical logic is accurate.
- Code follows object-oriented principles.
- AI-generated solution is efficient and well-structured.