

AI Assistant Coding

Assignment-6.2

Name of Student	: Ch Pushkar
Enrollment No.	: 2303A52362
Batch	: 38

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 ($\text{marks} \geq 40$).**

Expected Output-1

- **Python class with constructor and validation logic**
- **Clear method implementation returning pass/fail status**
- **Students analyze correctness and clarity of class design**

Prompt:

generate a Student class with attributes: name, roll_no, and marks. Add a method is_pass() that returns whether the student has passed ($\text{marks} \geq 40$)

Code:

```

class Student:
    """Represents a student with name, roll number, and marks."""

    def __init__(self, name: str, roll_no: int, marks: float):
        self.name = name
        self.roll_no = roll_no
        self.marks = marks

    def is_pass(self) -> bool:
        """Return True if the student's marks are >= 40, else False."""
        return self.marks >= 40

    def __repr__(self) -> str:
        status = "Pass" if self.is_pass() else "Fail"
        return f"Student(name={self.name!r}, roll_no={self.roll_no}, marks={self.marks}, status={status})"

if __name__ == "__main__":
    # Demo: create a few students and print whether they passed
    students = [
        Student("Alice", 1, 85),
        Student("Bob", 2, 39.5),
        Student("Charlie", 3, 40),
    ]

    for s in students:
        print(f"{s.name} (roll {s.roll_no}): {'Pass' if s.is_pass() else 'Fail'}")

```

Output:

```

Alice (roll 1): Pass
Bob (roll 2): Fail
Charlie (roll 3): Pass

```

Explanation:

Defines a Student class with attributes name, roll_no, and marks, plus a readable __repr__. Provides is_pass() (returns True when marks ≥ 40) that prints each student's pass/fail.

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.

Expected Output-2

- Correct pattern output using both loop types
- Logical loop structure with proper conditions

Prompt:

generate a function that prints a right angled triangle star pattern

1. using a for loop
2. using a while loop

Code:

```
def print_triangle_for(n: int) -> None:  
    """Print a right-angled triangle of height `n` using a for loop."""  
    for i in range(1, n + 1):  
        print("*" * i)  
  
def print_triangle_while(n: int) -> None:  
    """Print a right-angled triangle of height `n` using a while loop."""  
    i = 1  
    while i <= n:  
        print("*" * i)  
        i += 1  
  
if __name__ == "__main__":  
    # Demo: print triangle patterns then create a few students and print pass/fail  
    print("Triangle using for loop:")  
    print_triangle_for(5)  
    print()  
    print("Triangle using while loop:")  
    print_triangle_while(5)  
    print()
```

Output:

```
Triangle using for loop:  
*  
**  
***  
****  
*****
```

```
Triangle using while loop:  
*  
**  
***  
****  
*****
```

Explanation:

generated print_triangle_for(n) and print_triangle_while(n) to print a right-angled star triangle of height n using a for loop and a while loop respectively.

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.

Expected Output-3

- Function correctly classifies numbers
- Proper handling of all conditions
- Students analyze decision logic

Prompt:

Write a function that checks whether a given number is positive, negative, or zero using conditional statements. Test the function with multiple inputs..

Code:

```
def check_sign(n: float) -> str:  
    """Return 'positive', 'negative', or 'zero' for the given number using conditionals."""  
    if n > 0:  
        return "positive"  
    elif n < 0:  
        return "negative"  
    else:  
        return "zero"
```

Output:

```
Sign check tests:  
10: positive  
-5: negative  
0: zero  
3.14: positive  
-0.001: negative
```

Explanation:

This function simply checks whether the number is greater than, less than, or equal to zero. Based on that comparison, it returns "positive", "negative", or "zero" accordingly.

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.

Expected Output-4

- Python code using nested conditionals
- Clear explanation of decision flow

Prompt:

Write a function `check_discount(age, is_member)` that uses **nested if statements** to decide discounts.

If age is **60 or above**, give a **Senior discount**.

If the person is also a **member**, give an **Additional discount**.

Code:

```
def check_discount(age: int, is_member: bool) -> str:  
  
    if age >= 60:  
        discount = "Senior discount"  
        if is_member:  
            discount += " + Additional discount"  
        return discount  
    else:  
        return "No discount"  
  
  
# Tests for check_discount()  
print()  
print("Discount tests:")  
tests = [  
    (65, True),  
    (70, False),  
    (60, True),  
    (59, True),  
    (30, True),  
]  
for age, member in tests:  
    print(f"age={age}, member={member}: {check_discount(age, member)}")
```

Output:

```
Discount tests:  
age=65, member=True: Senior discount + Additional discount  
age=70, member=False: Senior discount  
age=60, member=True: Senior discount + Additional discount  
age=59, member=True: No discount  
age=30, member=True: No discount
```

Explanation:

The function first checks if the age is 60 or above to apply a senior discount. Inside that, it uses another if to add an extra discount if the person is also a member; otherwise, no discount is given.

Task Description-5 (Class – Mathematical Opera)

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

Expected Output-5

- Correct mathematical computation
- Well-structured class with methods
- Code explanation provided

Prompt:

Create a Circle class that takes a radius as input and has two methods: one to calculate the **area** and another to calculate the **circumference** of the circle. Ensure the calculations are correct

Code:

```
import math

class Circle:
    """Circle with radius; compute area and circumference accurately."""

    def __init__(self, radius: float):
        if radius < 0:
            raise ValueError("radius must be non-negative")
        self.radius = float(radius)

    def area(self) -> float:
        """Return the area: pi * r^2."""
        return math.pi * (self.radius ** 2)

    def circumference(self) -> float:
        """Return the circumference: 2 * pi * r."""
        return 2 * math.pi * self.radius

    def __repr__(self) -> str:
        return f"Circle(radius={self.radius})"

# Circle tests
print("Circle tests:")
cases = [1, 2.5, 0]
for r in cases:
    c = Circle(r)
    print(f"radius={r}: area={c.area():.6f}, circumference={c.circumference():.6f}")
```

OUTPUT:

```
Circle tests:
radius=1: area=3.141593, circumference=6.283185
radius=2.5: area=19.634954, circumference=15.707963
radius=0: area=0.000000, circumference=0.000000
```

Explanation:

This class stores the radius of a circle and provides methods to calculate its area and circumference using standard formulas.

It also validates the radius and uses Python's `math.pi` to ensure accurate mathematical results.