Lab Assignment 6.2 2303A52490

Task 1: Student Class for Pass/Fail

Prompt used: Generate a Python Student class with name, rollno, marks attributes. Include **init** and is_pass method (pass if marks >=40).

Generated Code:

```
class Student:
    def __init__(self, name, rollno, marks):
        self.name = name
        self.rollno = rollno
        self.marks = marks

    def is_pass(self):
        if self.marks >= 40:
            return True
        else:
            return False

# Test
s1 = Student("John Doe", "CS001", 45)
print(f"{s1.name} ({s1.rollno}): {'Pass' if s1.is_pass() else 'Fail'}")

s2 = Student("Jane Smith", "CS002", 35)
print(f"{s2.name} ({s2.rollno}): {'Pass' if s2.is_pass() else 'Fail'}")
```

Output:

```
John Doe (CS001): Pass
Jane Smith (CS002): Fail
```

Analysis: Constructor initializes attributes correctly. is_pass method simple and accurate. Logic clear, no errors. Good class design for validation.

Task 2: Star Pattern with For Loop

Prompt used: Write function print_triangle_for(n) to print right-angled triangle of stars using for loop.

Generated Code:

```
def print_triangle_for(n):
    for i in range(n):
        for j in range(i+1):
            print("*", end="")
        print()

print_triangle_for(5)
```

Output:

```
*
**
***
****
*****
```

Prompt used: Same pattern using while loop.

Generated Code:

```
def print_triangle_while(n):
    i = 0
    while i < n:
        j = 0
        while j <= i:
            print("*", end="")
            j += 1
        print()
        i += 1

print_triangle_while(5)
```

Output:

```
*
**
***
****
*****
```

Analysis: Both loops produce correct pattern. For loop more concise, while loop explicit. Conditions proper, no infinite loops.

Task 3: Number Classification

Prompt used: Function check_number(num) returns 'positive', 'negative', or 'zero' using if-elif-else. Test with 10, -5, 0.

Generated Code:

```
def check_number(num):
    if num > 0:
        return "positive"
    elif num < 0:
        return "negative"
    else:
        return "zero"

print(check_number(10))
print(check_number(-5))
print(check_number(0))
print(check_number(3.14))
```

Output:

```
positive
negative
zero
positive
```

Analysis: Handles all cases perfectly. Elif structure efficient. Works for integers and floats.

Task 4: Nested Conditionals for Discount

Prompt used: Function check_discount(age, is_member) for discount: age>=60 senior 10%, member extra 5%. Use nested if.

Generated Code:

```
def check_discount(age, is_member):
    discount = 0
    if age >= 60:
        discount += 10
        print("Senior discount applied")
    if is_member:
        discount += 5
        print("Member discount applied")
    return discount

print(f"Discount: {check_discount(65, True)}%")
print(f"Discount: {check_discount(45, True)}%")
print(f"Discount: {check_discount(70, False)}%")
```

Output:

```
Senior discount applied
Member discount applied
Discount: 15%
Member discount applied
Discount: 5%
Senior discount applied
Discount: 10%
```

Analysis: Nested logic clear - senior first, then member. Accumulates discounts correctly. Decision flow easy to follow.

Task 5: Circle Class Math Operations

Prompt used: Circle class with radius, methods area() and circumference(). Use pi=3.14159.

Generated Code:

```
import math
```

```
class Circle:
    def __init__(self, radius):
        self.radius = radius

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

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

c = Circle(7)
print(f"Area: {c.area():.2f}")
print(f"Circumference: {c.circumference():.2f}")
```

Output:

```
Area: 153.94
Circumference: 43.98
```

Analysis: Formulas correct ($A=\pi r^2$, $C=2\pi r$). Uses math.pi for precision. Methods well-structured, output formatted nicely.