Comprehensive Lesson Plan – Python Logical Operators & Introduction to Loops

🧠 Recap & Diagnostic Quiz (10 min)

- **Goal:** Reactivate understanding of if/else and compound conditions.
- **Analogy:** Think of planning a trip: you can go out only if the weather is sunny **and** your friend is free (AND), or you can stay home if it rains **or** your friend is busy (OR).
- **Gamified Challenge:** Present a scoreboard with stars. Ask quick true/false questions (e.g., "In Python, `True and False` evaluates to True?"). Each correct answer earns a star; three stars unlock a fun fact about programming.
- **Visual Aids:** Draw a truth table for `and`/`or` (True/True, True/False, etc.). Use colored stickers to track conditions that succeed.

🚦 Chunk 1 – Deepening Compound Conditions (20 min)

- **Objective:** Ensure mastery of `and`, `or`, and `not` with multi-step logic.

- **Analogy:** A scholarship eligibility rule: a student qualifies **if** (average ≥ 70 **and** attendance ≥ 80) **or** they are a sports champion. Use this scenario to discuss how `and` and `or` combine.

- **Gamified Challenge:** Provide each learner with cards that say "Condition A", "Condition B". Call out statements (e.g., "A student scored 75, attended 82 %"). They must show whether `A and B` or `A or B` evaluate to True.

- **Visual Aids:** Venn diagrams showing A ∧ B vs A ∨ B.

- **Coding Examples:**

```
age = 16
has_invitation = True
if age >= 18 and has_invitation:
   print("Entry granted")
else:
   print("Entry denied")
```

  Demonstrate with varying `age`/`has_invitation` values.

- **Practice Questions:**

  - Easy: Write a program that checks whether a number is between 10 and 20 (inclusive).
  - Medium: Input three scores; print "Excellent" if all are ≥ 80, "Good" if any score ≥ 80, otherwise "Needs Improvement".
  - Exam-level: Read age, income, and membership status. Print eligibility for a discount using compound conditions.

🔁 Chunk 2 – Introduction to Loops (25 min)

- **Objective:** Introduce `for` and `while` loops, building on selection.

- **Analogy:** Repetition is like a merry-go-round: you continue to go in circles until you decide to stop.

- **Gamified Challenge:** “Counting Game”: ask the student to list numbers from 1 to 5 out loud, then show the equivalent Python code using a `for` loop. Award stars for correct sequences.

- **Visual Aids:** Flowchart showing initialization, condition check, loop body, update.

- **Coding Examples:**

    - **For loop to sum numbers:**

      ```
      total = 0
      for i in range(1, 6):
          total += i
      print("Sum:", total)
      ```

    - **While loop to validate input:**

      ```
      n = int(input("Enter a positive number: "))
      while n <= 0:
          print("Invalid input.")
          n = int(input("Try again: "))
      ```

- **Practice Questions:**

    - Easy: Use a `for` loop to print all even numbers between 1 and 20.
    - Medium: Write a program that repeatedly asks for marks until the user enters −1, then prints the average.
    - Exam-level: Implement a login system where the user has three attempts; break the loop upon correct password or after three failures.

🌳 Chunk 3 – Combining Conditions with Loops (20 min)

- **Objective:** Solve problems that require both loops and compound conditions.

- **Analogy:** Checking exam results: for each mark in a list, determine if it is a pass and if it’s an A grade (≥ 75 and ≥ 40).

- **Coding Examples:**

    - **Filter data:**

      ```
      marks = [34, 67, 82, 49, 73]
      passes = []
      for m in marks:
          if m >= 40 and m <= 100:
              passes.append(m)
      print("Passed:", passes)
      ```

- **Practice Questions:**

    - Easy: Count how many numbers in a list are multiples of 3 **and** 5.
    - Medium: Ask the user to input daily temperatures for seven days; print a warning if any day meets the “heat wave” condition (≥ 35 °C **or** humidity ≥ 80 %).

- Exam-level: Develop a program that records product data (name, price, quantity) in a list of dictionaries, then uses a loop with compound conditions to find products priced under Rs.1000 **and** stock below 10.

📝 Past-Paper Style Practice

MCQ Examples

1. In Python, what does the following evaluate to?

   ```
   True or False and False
   ```

   a. True
   b. False
   c. Error
   d. None
2. How many iterations will this code run?

   ```
   for i in range(2, 10, 2):
     print(i)
   ```

   a. 4
   b. 5
   c. 8
   d. 16

Structured Questions

1. **Code Analysis:**

   ```
   scores = [90, 45, 67, 82]
   passed = []
   for s in scores:
     if s >= 50 or s % 2 == 0:
       passed.append(s)
   print(passed)
   ```

   a. What does this program print?
   b. Explain why an odd score of 67 passes.
2. **Error Correction:** The following program is intended to print multiples of 5 from a list, but it contains logical errors:

   ```
   nums = [10, 12, 15, 23, 25]
   for n in nums:
     if n % 5 == 1 and n % 5 == 0:
       print(n)
   ```

   a. Identify the error.
   b. Rewrite it correctly.

Essay Question

- **Explain, with a flowchart and code, how a while loop can be used to read numbers until the user enters a negative number, at which point the program stops and prints the total and average.** Describe how the loop condition controls repetition and how sentinel values are used to

terminate loops.

🎯 Final Mini-Project & Reflection

Project – Simple Voting System:

- Prompt for the number of voters.
- Use a loop to collect each voter's age.
- For each age, check if the person can vote (age ≥ 18) **and** is on the list of registered voters (simulate with a list).
- Count and display the total number of eligible voters.
- Allow the user to re-run the process (Yes/No) using another loop.

Reflection:

- Ask her to reflect on which analogies helped her grasp `and`/`or` operators.
- Have her explain (in Sinhala) how loops can simplify repetitive tasks.
- Discuss any difficulties she encountered and plan a quick review of those areas in the next session.