Pre-Class Reading: Nested Loops and Lists in Python

1. Learning Objectives

In this session, you will:

- Understand what nested loops are and how to use them.
- Learn how lists in Python can be used with loops, including nested loops.
- Explore practical uses for nested loops and lists in solving problems.
- Gain confidence in writing and reading code that involves multiple layers of loops.

2. Key Concepts or Vocabulary

- **Loop**: A sequence of instructions that repeats until a certain condition is met.
- **Nested Loop**: A loop inside another loop. The inner loop runs completely for every iteration of the outer loop.
- **List**: A collection in Python that holds multiple items, such as numbers or strings, in a specific order.
- **Iteration**: The process of repeating actions (or looping) for each item in a collection or sequence.
- **Index**: The position of an item in a list. Python uses zero-based indexing, meaning the first item is at index 0.

3. Real-World Examples

Imagine you are organizing a school with multiple classes, and you need to print the names of students in each class. To do this, you would go through each class (outer loop) and then list the students in that class (inner loop). This is exactly how a nested loop operates.

Similarly, think of a spreadsheet where each row has several columns. If you want to process each cell, you'd first go through the rows (outer loop) and then the columns within each row (inner loop).

4. Mini Code Snippets

Here’s a simple example of a nested loop that prints out the multiplication table from 1 to 3:

```
# Outer loop for the numbers 1 to 3
for i in range(1, 4):
    # Inner loop for the numbers 1 to 3
    for j in range(1, 4):
        # Print multiplication of i and j
        print(f"{i} * {j} = {i * j}")
    # Blank line to separate each table
    print()
```

Explanation:

- The outer loop (`for i in range(1, 4)`) goes through the numbers 1, 2, and 3.
- The inner loop (`for j in range(1, 4)`) does the same, but for each `i` , it runs completely before moving to the next `i` .
- The result is the multiplication table for 1 to 3.

Here's an example with a list of student names for different classes:

```
# List of classes, each class contains a list of students
classes = [
    ["John", "Alice", "Bob"],
    ["Sara", "Tom", "Linda"],
    ["Mike", "Jane", "Anna"]
]

# Outer loop to go through each class
for class_group in classes:
    # Inner loop to go through each student in the class
    for student in class_group:
        print(student)
    # Blank line to separate each class's list of students
    print()
```

Explanation:

- We have a list of lists (`classes`), where each inner list contains names of students in a class.
- The outer loop goes through each class, and the inner loop prints the students' names in that class.

5. Pre-Reading Questions or Simple Exercises

Try these questions to prepare for class:

1. What will be the output of the following code snippet?

```
for i in range(3):
    for j in range(2):
        print(f"i: {i}, j: {j}")
```

2. Can you write a nested loop to print the following pattern?

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

3. Create a list of three of your favorite foods, and use a nested loop to print each character in each food's name.

6. Common Mistakes or Misconceptions

- **Forgetting to fully understand the flow of loops**: Sometimes beginners get confused about how the inner loop completes all its iterations before the outer loop moves to the next step.
- **Confusing inner and outer loop variables**: It's easy to accidentally mix up the loop variables (e.g., using the same name for both loops).
- **Misunderstanding list indexing**: Python lists start at index 0, so the first item is `list[0]`, not `list[1]`.
- **Overcomplicating loops**: You might find yourself overthinking loop logic. Start simple and walk through each iteration manually if necessary.

7. A Teaser for the Lecture

Imagine you're designing a basic game where the player can move on a grid (like chess or checkers). How can we use nested loops to check each possible position on the board and determine if a move is valid? In our lecture, we'll explore how nested loops can be the key to solving problems like this in gaming, data analysis, and more!

This reading prepares you to dive into nested loops and lists. Don't worry if it feels tricky at first—working with loops becomes much easier with practice!