مكتب

Lab Activity

Week 10: Nested Loops and 2D Patterns

Duration: 60 minutes — Work in pairs!

Name: _____ **Partner:** _____

Lab Goals

By the end of this lab, you will be able to:

- Write loops inside other loops (nested loops)
- Understand how inner and outer loops work together
- Create simple 2D patterns using nested loops
- Trace through nested loop execution step by step

Getting Started

1. Extract the given `week10_lab.zip` file to your desktop. This folder contains all the files you need for this lab.

1 Exercise 1: From Single to Nested Loops (8 minutes)

Let's see why we need nested loops by comparing two approaches:
Type this code exactly as shown and run it:

1.1 Exercise 1: Single loop

Using a single loop *(limited approach)*. This can only print the same thing each time.

Open `week10_ex1_1.py` and type the following:

```
for i in range(4):
    print("* * *")
print()
```

Run it and show it to your teacher.

1.2 Exercise 2: Nested loop

Using nested loops (flexible approach!). Here we can control both rows and columns independently.

Open `week10_ex1_2.py` and type the following:

```
for row in range(4):            # Outer loop: handles rows
    for col in range(3):        # Inner loop: handles columns
        print("*", end=" ")     # Print star, stay on same line
    print()                     # Move to next line after each row
```

Now modify the above code:

- Change it to make a 5×5 grid instead of 4×3
- Add row numbers: print `f"Row {row}: "` before the inner loop
- What happens if you forget the `print()` after the inner loop? Try it!

Tip: The outer loop controls rows (vertical), the inner loop controls columns (horizontal)!

Checkpoint: Both methods produce the same output? Show your teacher!

2 Exercise 2: Complete the Star Box (7 minutes)

Open week10_ex2.py and complete the code to create a star box pattern:

```
print("=== Star Box Pattern ===")
# Ask for dimensions
width = int(input("Enter box width: "))
height = int(input("Enter box height: "))

print("Your star box:")
# Create the box using nested loops
for i in range(_____):          # How many rows?
    for j in range(_____):      # How many stars per row?
        print("*", end="")      # Print star
    _______                     # What goes here to start new row?
print("Box complete!")

# Count total stars
total_stars = _____ * _____     # How to calculate total?
print(f"Total stars used: {total_stars}")
```

Test values:

- When `width = 6` and `height = 3`, you should see 18 stars total
- When `width = 4` and `height = 4`, you should see 16 stars total

Checkpoint: Box displays correctly with proper dimensions? Great job!

Pair Programming: Switch who types every 5 minutes!

3 Exercise 3: Explore Loop Execution Order (10 minutes)

Let's explore how nested loops execute by adding debug prints:

Step 1: Open week10_ex3.py and start with the following code. First try running it as is and observe the output:

```
# Exercise 3: Understanding execution order
print("=== Loop Execution Tracker ===")

for outer in range(2):
```

```
    print(f"\nOUTER loop: outer = {outer}")

    for inner in range(3):
        print(f"  INNER loop: inner = {inner}")
        print(f"    Position: ({outer}, {inner})")
```

Step 2: Try these experiments (run after each change):

1. What happens if you swap the ranges: outer gets `range(3)` and inner gets `range(2)`?
   Result: _____
2. What happens if you change inner `range(3)` to `range(2)`?
   Result: _____
3. Can you make it print only when `outer == inner`? Add an if statement in the inner loop:
   Your code: _____
   _____

Tip: The inner loop completes ALL its iterations before the outer loop moves to the next value!

4 Exercise 4: Multiplication Table Explorer (10 minutes)

Your Challenge: Create a multiplication table that helps young students learn their times tables! The program should display a neat table and highlight special numbers.

Open week10_ex4.py and complete this code:

```
# Exercise 4: Multiplication Table with Highlights
print("=== Times Table Helper ===")
size = 5  # We'll make a 5x5 table

# Print column headers (1 to 5)
print("   ", end="")  # Space for row labels
for num in range(1, size + 1):
    print(f" {num} ", end="")
print("\n   " + "-" * 15)  # Divider line

# Create the multiplication table
perfect_squares = 0  # Count numbers like 1, 4, 9, 16, 25

for row in range(1, size + 1):
    # Print row number
    print(f"{row} |", end="")

    for col in range(1, _______):  # What should go here?
        result = _______ * _______   # Calculate the product

        # Check if it's a perfect square (same number multiplied by itself)
        if row == col:
            print(f" {result} ", end="")  # Print normally
            perfect_squares = _______ + 1  # Increment counter
        else:
            print(f" {_______} ", end="")  # Print the result

    print()  # New line after each row

print(f"\nFound {perfect_squares} perfect squares on the diagonal!")
print("Can you spot them? They're at positions where row = column.")
```

Expected Output:

```
=== Times Table Helper ===
    1  2   3   4   5
   -----------------
1 | 1  2   3   4   5
2 | 2  4   6   8   10
3 | 3  6   9   12  15
4 | 4  8   12  16  20
5 | 5  10  15  20  25

Found 5 perfect squares on the diagonal!
Can you spot them? They're at positions where row = column.
```

Tip: The diagonal shows perfect squares: 1×1=1, 2×2=4, 3×3=9, etc!

Checkpoint: Table displays correctly and counts 5 perfect squares? Math made visual!

5 Exercise 5: What Will It Print? (5 minutes)

Before running this code, predict what it will output:

```
for i in range(2):
    for j in range(4):
        if j < 2:
            print("A", end="")
        else:
            print("B", end="")
    print()
```

Your prediction:

Line 1: _____

Line 2: _____

Now run it.

Were you correct? ○ Yes ○ No

If not, can you trace through and see why?

_____

Pair Programming: Switch who types every 5 minutes!

6 Exercise 6: Mini-Challenge - Number Pyramid (15 minutes)

Create a program that builds a number pyramid!

1. Ask the user for the pyramid height (1-9)
2. Print a pyramid where each row shows numbers from 1 up to the row number
3. Make it look neat and centered

Example Output:

```
=== Number Pyramid Builder ===
Enter pyramid height (1-9): 5

Your pyramid:
    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5

Pyramid complete!
Total numbers printed: 15
```

Starter Help:
Open week10_ex6.py and complete the following code:

```python
# Mini-Challenge: Number Pyramid Builder
print("=== Number Pyramid Builder ===")
height = int(input("Enter pyramid height (1-9): "))

print("\nYour pyramid:")
total_numbers = 0

# Build the pyramid
for row in range(1, _____):          # 1) range should go from 1 up to height

# Step 1: Print spaces for centering
    spaces = _____ - row             # 2) how many spaces?
    print(" " * spaces, end="")

    # Step 2: Print numbers for this row
    for num in range(1, _____):      # 3) numbers 1 through current row
        print(num, end=" ")
        total_numbers = total_numbers + _____  # 4) how many numbers did we
                                               #    just print?

    # Step 3: Move to next line
    print()

print("\nPyramid complete!")
print(f"Total numbers printed: {_____}")  # 5) what variable holds the count
```

Bonus challenges:

- Can you make it print stars (*) instead of numbers?
- Can you make an upside-down pyramid?

Checkpoint: Pyramid displays correctly? Excellent work with nested loops!

7 Lab Summary

☐ Completed all exercises with your partner

☐ Understood how inner loops complete before outer loops continue

- ☐ Traced through loop execution order
- ☐ Built a number pyramid
- ☐ Shown all checkpoints to your teacher

Reflection (2 minutes)

Rate your understanding of today's concepts:

Concept	Need Help	Getting It	Got It!
Nested loop structure	○	○	○
Inner vs outer loops	○	○	○
Creating 2D patterns	○	○	○
Loop execution order	○	○	○

One thing I created today that I'm proud of:

_____

The trickiest part about nested loops is:

_____

Great work exploring nested loops today!