مكتب

Homework Assignment

Week 10: Nested Loops and 2D Patterns

Total Points: 100 — Due: Next Class

Name: _____ **Date:** _____

Important Instructions

- Work on your own, but you may ask family members for hints only if stuck
- Write your code in Python *(using Thonny or any online IDE)*
- For setting up your code files, kindly see `instructions.pdf`
- For written answers, use the spaces provided on this sheet
- Show your work for full credit!

1 Part 1: Nested Loop Tracing (20 points)

Time estimate: 25 minutes

1.1 Exercise 1.1: Trace the Box Pattern (10 points)

Without running the code, carefully trace through this nested loop code:

```
rows = 3
cols = 4

for r in range(rows):
    print(f"Row {r}: ", end="")
    for c in range(cols):
        print(f"[{r},{c}]", end=" ")
    print()  # New line after each row
print("Pattern complete!")
```

Fill in the trace table below. Show EVERY iteration:

r value	c value	What prints	Notes (optional)
0	0	`[0,0]`	First box
0	1		
0	2		
0	3		End of row 0
— New line printed —			
1	0		
1	1		
1	2		
1	3		
— New line printed —			
2	0		
2	1		
2	2		
2	3		
After all loops: "Pattern complete!"			

1.2 Exercise 1.2: Predict the Number Grid (10 points)

Following code generates a number grid using nested loops. Predict the output without running it:

```
for i in range(3):
    for j in range(4):
        result = i * 10 + j
        print(result, end=" ")
    print()
print("Grid complete!")
```

Fill in what gets printed in each position:

_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

How many times does the inner loop run in total? _____

What is the largest value printed? _____

2 Part 2: Code Completion - Building Patterns (18 points)

Time estimate: 25 minutes

2.1 Exercise 2.1: Rectangle of Symbols (9 points)

Open week10_ex2_1.py and complete the code to create a customizable rectangle pattern. The user should be able to specify the symbol, width, and height of the rectangle.

```
# Custom Rectangle Builder
print("=== Rectangle Pattern Generator ===")
symbol = input("Enter a symbol to use: ")
```

```
width = int(input("Enter width: "))
height = int(input("Enter height: "))

print("\nYour rectangle:")

# Use nested loops to create the rectangle

for row in range(_______):  # How many rows?
    for col in range(_______):  # How many columns?
        print(_______, end="")  # Print symbol without newline
    _______  # What goes here to start a new row?
print("\nRectangle complete!")

# Number of symbols on the border (perimeter):
total_border = 2 * _______ + 2 * (_______) - 4  # Why subtract 4?
print(f"Border symbols: {total_border}")
```

Sample Output:

```
=== Rectangle Pattern Generator ===
Enter a symbol to use: *
Enter width: 5
Enter height: 3

Your rectangle:
*****
*****
*****

Rectangle complete!
Border symbols: 12
```

2.2 Exercise 2.2: Coordinate Grid System (9 points)

Open week10_ex2_2.py and complete the code to create a coordinate grid for a game board. The board should be 8x8, with alternating symbols for a checkerboard pattern.

```
# Game Board Coordinate System
print("=== Game Board Coordinates ===")
size = 8  # 8x8 game board

# Print column headers (1-8)
print("  ", end="")  # Initial spacing
for col in range(size):
    print(f" {_______}", end="")  # What number to show?
print()  # New line after headers

# Create the board with row numbers
for row in range(size):
    # Print row number (8 at top, 1 at bottom - like chess)
    row_num = _______ - _______
    print(f"{row_num} ", end="")

    # Print each square in the row
    for col in range(_______):
        # Alternate between . and * for checkerboard pattern
        if (row + col) % 2 == 0:
            square = "."
        else:
```

```
            square = "*"
        print(f" {_______}", end="")

    print()  # End of row

print("\nExample: Knight at position (3, 6)")

# Challenge: Find what symbol is at a specific position
target_row = 3  # Row 3 from bottom
target_col = 5  # Column 6 (remember: 0-indexed)

# Convert to array indices
board_row = _______ - _______
board_col = _______

# Determine the symbol at that position
if (board_row + board_col) % 2 == 0:
    symbol = "."
else:
    symbol = "*"

print(f"Position ({_______}, {_______}) has symbol: {symbol}")
```

Sample Output:

```
=== Game Board Coordinates ===
   1 2 3 4 5 6 7 8
8  . * . * . * . *
7  * . * . * . * .
6  . * . * . * . *
5  * . * . * . * .
4  . * . * . * . *
3  * . * . * . * .
2  . * . * . * . *
1  * . * . * . * .

Example: Knight at position (3, 6)
Position (3, 6) has symbol: .
```

3 Part 3: Pattern Recognition (15 points)

Time estimate: 20 minutes

3.1 Exercise 3.1: Analyze These Patterns (8 points)

Look at each pattern and identify what the nested loops must be doing:

Pattern A:

```
1 2 3 4
1 2 3 4
1 2 3 4
```

Pattern B:

```
1 1 1 1
2 2 2 2
3 3 3 3
```

Pattern C:

```
1 2 3
2 3 4
3 4 5
```

For each pattern, explain what values are being used:

- **Pattern A tests:** Inner loop uses _____ value, regardless of _____

- **Pattern B tests:** Inner loop uses _____ value, repeated _____

- **Pattern C tests:** Inner loop adds values _____ and _____ together

3.2 Exercise 3.2: Create the Diagonal Pattern (7 points)

Write code to produce this diagonal line pattern:

```
X . . . .
. X . . .
. . X . .
. . . X .
. . . . X
```

```python
size = 5

for row in range(size):
    for col in range(size):
        # Your condition here to decide when to print X
        if ________________________:
            print("X", end=" ")
        else:
            print(".", end=" ")
    print()
```

Save this code as `week10_ex3_2.py`

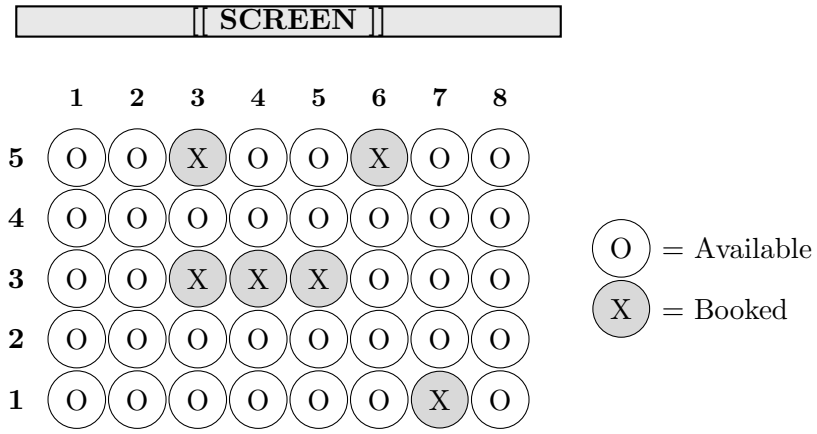
4 Part 4: Problem Solving with Nested Loops (15 points)

Time estimate: 20 minutes

4.1 Exercise 4.1: Cinema Seat Booking Display (15 points)

Scenario: Cinema Seat Map

You're creating a seat display system for a small cinema. The cinema has 5 rows (numbered 5 at the top to 1 at the bottom) with 8 seats each. For this show, every 3rd seat in row 5 is booked, the middle 3 seats in row 3 are booked, and seat 7 in row 1 is booked. Your program should display the seating arrangement and calculate availability.



Open week10_ex4_1.py and complete the code to display the cinema seat map, count available seats, and calculate occupancy rate.

```
# Cinema Seat Display System
print("=== Gulistan Cinema - Seat Map ===")
rows = 5
seats_per_row = 8

# Display the screen
print("\n       [[ SCREEN ]]")
print("    ", end="")
for s in range(seats_per_row):
    print(f"{s+1:2}", end=" ")  # Seat numbers
print()

# Display seats with row letters
available_count = 0

for r in range(rows):
    # Show row number (5 down to 1)
    row_display = _______ - _______
    print(f"{row_display}  ", end="")

    # Check each seat in this row
    for s in range(_______):
        # Determine if this seat is booked based on the pattern
        is_booked = False

        # Row 5 (r=0): every 3rd seat (seats 3 and 6)
        if r == 0 and (s == 2 or s == 5):
            is_booked = True
        # Row 3 (r=2): middle 3 seats (seats 3, 4, 5)
        elif r == 2 and s >= _______ and s <= _______:
            is_booked = True
        # Row 1 (r=4): seat 7
        elif r == _______ and s == _______:
            is_booked = True

        # Display the seat
        if is_booked:
            print(" X", end=" ")  # X for booked
        else:
            print(" O", end=" ")  # O for available
            _______ += 1  # Count available seats

    print()  # End of row

```

```python
# Summary statistics
total_seats = _______ * _______
booked_count = total_seats - _______
occupancy_rate = (_______ / _______) * 100

print(f"\nTotal seats: {total_seats}")
print(f"Available: {available_count}")
print(f"Booked: {booked_count}")
print(f"Occupancy: {occupancy_rate:.1f}%")

# Show row occupancy
print("\nRow occupancy:")
for r in range(rows):
    row_num = 5 - r
    print(f"Row {row_num}: ", end="")
    # Your code to show a simple bar graph of occupied seats
```

5 Part 5: Pattern Maker Program (20 points)

Time: 35 minutes
Create a fun program that draws cool patterns with symbols!

What Your Program Should Do

1. Ask the user how many patterns they want to make (1-3)
2. Show a menu with 3 pattern choices
3. Let the user pick a pattern and size
4. Draw the pattern using loops
5. Count how many symbols were used
6. Say goodbye at the end!

Example of Your Program Running:

```
=== Pattern Maker ===
How many patterns do you want? 2

--- Pattern 1 ---
Choose a pattern:
1. Square
2. Triangle
3. Line Pattern

Your choice (1-3): 1
What size? 3
What symbol? *

Here's your pattern:
* * *
* * *
* * *
```

```
Used 9 symbols!

--- Pattern 2 ---
Choose a pattern:
1. Square
2. Triangle
3. Line Pattern

Your choice (1-3): 2
What size? 4
What symbol? #

Here's your pattern:
#
# #
# # #
# # # #

Used 10 symbols!

Thanks for using Pattern Maker!
You made 2 patterns today!
```

Starter Code:
Open up week10_ex5.py and complete the code to make your pattern maker program.

```python
print("=== Pattern Maker ===")

# Step 1: Ask how many patterns
num_patterns = int(input("How many patterns do you want? "))

# Step 2: Make that many patterns
for i in range(num_patterns):
    print(f"\n--- Pattern {i + 1} ---")

    # Show the menu
    print("Choose a pattern:")
    print("1. Square")
    print("2. Triangle")
    print("3. Line Pattern")

    # Get their choice
    choice = int(input("\nYour choice (1-3): "))
    size = int(input("What size? "))
    symbol = input("What symbol? ")

    print("\nHere's your pattern:")

    # Count symbols as we go
    symbol_count = 0

    if choice == 1:
        # Square pattern
        # HINT: Use two for loops
        # Count symbols used

    elif choice == 2:
        # Triangle pattern
```

```
        # HINT: First row has 1 symbol, second has 2, etc.
        # Count symbols used

    elif choice == 3:
        # Line pattern (you design this!)
        # Make it simple but fun
        # Example: * - * - * - *

    else:
        print("Oops! Pick 1, 2, or 3 next time!")

    print(f"\nUsed {symbol_count} symbols!")

# Step 3: Say goodbye
print("\nThanks for using Pattern Maker!")
print(f"You made {num_patterns} patterns today!")
```

Pattern Ideas:

Square Pattern (size 3)

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

Triangle Pattern (size 4)

```
#
# #
# # #
# # # #
```

Line Pattern Ideas

You can make:

- Alternating symbols: `* - * - * -`
- Repeated pattern: `@ @ @ @ @ @`
- Growing line: `. .. ... ....`

Be creative!

Helpful Hints:

- The `end=" "` in print keeps things on the same line
- `print()` by itself makes a new line
- `range(3)` gives you 0, 1, 2
- Don't forget to count your symbols!

Check Your Work:

☐ Program asks for number of patterns

☐ Menu shows up for each pattern

☐ Square pattern works

- ☐ Triangle pattern works
- ☐ Your creative line pattern works
- ☐ Symbol counter is correct
- ☐ Program says goodbye with pattern count

Bonus Challenge (if you finish early):
Add a 4th pattern choice - maybe a hollow square or a pyramid!

6 Part 6: Debug Detective (7 points)

Time estimate: 15 minutes
Each code snippet has issues. Find and fix ALL errors:

6.1 Snippet 1: Table Headers (3 points)

Open up week10_ex6_1.py and fix the code to print a 3x3 table with headers.

```
print("  A B C")
for row in range(3):
    print(row+1, end=" ")
    for col in range(3)
        print(".", end=" ")
print()
```

6.2 Snippet 2: Multiplication Grid (4 points)

Open up week10_ex6_2.py and fix the code to print a 4x4 multiplication grid and find the largest value.

```
# This should show a 4x4 multiplication grid - fix all issues!
size = 4
for i in range(size):
    for j in range(size):
        product = i * j
        print(product, end=" ")
        print()

# Should print which entry is largest
print(f"Largest value: {i * j}")  # Logic error!
```

Expected Output:

```
1  2  3  4
2  4  6  8
3  6  9  12
4  8  12 16
Largest value: 16
```

7 Part 7: Reflection Questions (5 points)

Time estimate: 10 minutes
Answer these questions in complete sentences:

1. Nested loops can be confusing at first. What mental model or visualization helps you understand how the inner and outer loops work together? Draw or describe it. *(Hint: Think about how a clock's minute and hour hands move differently)*

2. What real-world situations (besides those in this homework) would need nested loops? Give two specific examples and explain why nested loops are necessary.

8 Bonus Section: Extra Challenges (Optional - 10 extra points)

Only attempt if you've finished everything else!

8.1 Challenge 1: Hollow Diamond Pattern (5 points)

Create a program that prints a hollow diamond pattern:

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

Hints:

- First half: spaces decrease, gap between stars increases
- Second half: reverse the pattern
- Only print stars at edges of each row (except single star rows)

Open `week10_ex8_1.py` and write the code to generate this pattern.

8.2 Challenge 2: Number Waterfall (5 points)

Create a program that generates this cascading number pattern:

```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

Hints:

- First loop goes up to 5
- Second loop goes back down
- Inner loop counts backwards from current number

Open `week10_ex8_2.py` and write the code to generate this pattern.

Submission Checklist

Before submitting, make sure you have:

- ☐ Completed all required sections
- ☐ Saved each code exercise in its own file
- ☐ Tested each code file individually to ensure it runs without errors
- ☐ Written your name at the top of this paper
- ☐ Answered all reflection questions thoughtfully
- ☐ Attempted bonus section (optional)
- ☐ Compressed all your code files into a single zip file named `week10_hw.zip`

Time Tracking:
How long did this homework take you? _____ hours

Parent/Guardian Signature: _____
(Confirming student completed work independently)

Excellent work mastering nested loops!
Next week: Combining loops with conditions for smart iterations