مكتب

Homework Assignment

Week 9: Introduction to `for` Loops

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: Loop Tracing Practice (15 points)

Time estimate: 20 minutes

1.1 Exercise 1.1: Trace the Temperature Converter (8 points)

For this part you are not supposed to run the code on computer. Read this code carefully and trace through each iteration, filling in the following table:

```
for celsius in range(4):
    fahrenheit = (celsius * 9/5) + 32
    print(f"{celsius} C = {fahrenheit} F")
```

Fill in the trace table below:

Iteration	celsius	Calculation	fahrenheit	What prints
1	0	`(0 * 9/5) + 32`		
2				
3				
4				

1.2 Exercise 1.2: Predict the Output (7 points)

Without running the code, predict what this prints:

```
message = "Go "
for count in range(3):
    message = message + "Team! "
print(message)
print("Loop ran", count + 1, "times")
```

Your prediction:

Line 1 output: _____

Line 2 output: _____

Now run the code to check your prediction:

Actual output:

Line 1 output: _____

Line 2 output: _____

2 Part 2: Code Completion - Loop Basics (18 points)

Time estimate: 25 minutes

2.1 Exercise 2.1: Countdown Timer (10 points)

Open week9_ex2_1.py and complete the countdown timer program. This will help you practice using loops to count down from a number.

```
# Eid Countdown Timer
print("=== Days Until Eid ===")
days_left = 10

# Count down from 10 to 1
for day in _______:     # Use range() to go from 0 to 9
    actual_day = _______ - _______  # Calculate countdown
    print(f"{actual_day} days until Eid!")

print("Eid Mubarak!")

# Now create an exciting countdown for the last 3 seconds
print("\nFinal countdown:")
for second in range(_______):    # How many iterations for 3,2,1?
    countdown = _______ - _______
    print(f"{countdown}...")

print("CELEBRATION TIME!")
```

2.2 Exercise 2.2: Savings Calculator (8 points)

Open week9_ex2_2.py and complete the savings tracker program. This will help you practice using loops to accumulate savings over weeks.

```
# Weekly Savings Tracker
print("=== Savings Goal Tracker ===")
weekly_amount = int(input("How much do you save each week (Rs.)? "))
weeks = int(input("For how many weeks? "))
```

```
5
6  total_saved = 0    # Initialize accumulator
7  print()
8  print("Week by week progress:")
9
10 # Use a for loop to calculate savings
11 for week_num in range(weeks):
12     # Add this week's savings
13     total_saved = total_saved + weekly_amount
14     # Display progress (week numbers should start at 1, not 0)
15     print(f"Week {week_num+1}: Rs. {total_saved} saved so far")
16
17 print()
18 print(f"Congratulations! You'll save Rs. {total_saved} in total!")
19
20 # Check if goal is met
21 print()
22 goal = int(input("What was your savings goal? Rs. "))
23 if total_saved >= goal:
24     print("Great job! You'll meet your goal!")
25 else:
26     shortfall = goal - total_saved
27     print(f"You'll be Rs. {shortfall} short. Save a bit more each week!")
```

3 Part 3: Loop Analysis and Comparison (15 points)

Time estimate: 20 minutes

3.1 Exercise 3.1: Compare These Loops (8 points)

Without running the code on a computer, study these three code snippets that all print numbers, but differently:

Code A:

```
1 for x in range(5):
2     print(x)
```

Code B:

```
1 for x in range(5):
2     print(x + 1)
```

Code C:

```
1 for x in range(5):
2     print(10 - x)
```

Fill in what each code prints:

Code A prints:	Code B prints:	Code C prints:
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Which code *(A, B or C)* would you use to:

- Print player numbers 1-5 on jerseys? Code _____
- Create a countdown from 10 to 6? Code _____
- Index items starting from 0? Code _____

3.2 Exercise 3.2: Fix the Logic (7 points)

This program should print a staircase of hash symbols, but it's not working correctly:

```
# Broken Staircase Builder
print("Building a staircase:")
stairs = "#"
for level in range(5):
    print(stairs)
```

Something is missing within the for loop to make stairs grow!

Current output:

```
#
#
#
#
#
```

Desired output:

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

Open week9_ex3_2.py and fix the code by adding ONE line in the loop to make the staircase grow correctly.

```
# Staircase Builder
print("Building a staircase:")
stairs = "#"
for level in range(5):
    print(stairs)
    ------------------------  # Add your line here
```

4 Part 4: Problem Solving with Loops (18 points)

Time estimate: 25 minutes

4.1 Exercise 4.1: Cricket Score Calculator (18 points)

Scenario: Cricket Match Scoring

You're creating a simple cricket scoring system. The program should track runs scored in each over and calculate the total score and run rate.

Open week9_ex4_1.py and complete the cricket score tracker program. Use the accumulator pattern to keep track of total runs.

```python
# Cricket Score Tracker

print("=== Cricket Match Scorer ===")
overs = int(input("How many overs to track? "))

total_runs = 0
print("\nEnter runs for each over:")

# Loop through each over
for over_num in range(overs):
    # Get runs for this over (remember: display over 1, not over 0)
    runs_this_over = int(input(f"Over {_______}: "))
    # Add to total using accumulator pattern
    _______ = _______ + _______
    # Calculate current run rate (runs per over)
    current_run_rate = _______ / (_______)
    print(f"Total after {_______} over(s): {_______} runs")
    print(f"Current run rate: {current_run_rate:.2f}")
    print()

# Final statistics
print("=" * 30)
print(f"FINAL SCORE: {_______} runs in {_______} overs")
print(f"Average runs per over: {_______}")

# Bonus calculation: project score for 20 overs
if overs < 20:
    projected = int(_______ * (20 / _______))
    print(f"Projected 20-over score: {projected} runs")
```

5 Part 5: Full Program Development (22 points)

Time estimate: 35 minutes

In this exercise, you will create a complete program that helps users practice multiplication tables.

Program Requirements

Your program should:

- Ask which multiplication table to practice (2-12)
- Ask how many problems to generate
- Use a for loop to generate problems in order
- Keep track of correct answers
- Calculate and display the score as a percentage
- Give encouraging feedback based on performance
- Handle incorrect answers gracefully

Sample output:

```
=== Multiplication Table Practice ===
Which table do you want to practice? (2-12): 7
How many problems do you want? 5

Let's practice the 7 times table!

Problem 1: 7 x 1 = ? 7
Correct!

Problem 2: 7 x 2 = ? 14
Correct!

Problem 3: 7 x 3 = ? 20
Oops! 7 x 3 = 21

Problem 4: 7 x 4 = ? 28
Correct!

Problem 5: 7 x 5 = ? 35
Correct!

--- Practice Complete ---
You got 4 out of 5 correct!
Score: 80%

Great job! Keep practicing!
```

Starter Structure:

Open week9_ex5.py and complete the multiplication practice program. This will help students practice their multiplication tables in a fun way.

```python
print("=== Multiplication Table Practice ===")

# Step 1: Get user input
table = int(input("Which table do you want to practice? (2-12): "))
num_problems = int(input("How many problems do you want? "))

```

```
print(f"\nLet's practice the {table} times table!\n")

# Step 2: Initialize score counter
correct = 0

# Step 3: Generate problems using a for loop
for i in range(num_problems):
    # Create the problem (use i+1 so problems start at x1, not x0)

    # Get user's answer

    # Check if correct and update score

    # Give feedback

# Step 4: Calculate and display final score
print("\n--- Practice Complete ---")
# Calculate percentage

# Step 5: Give encouraging feedback based on score
# 90-100%: "Excellent! Perfect or near perfect!"
# 70-89%: "Great job! Keep practicing!"
# 50-69%: "Good effort! Practice makes perfect!"
# Below 50%: "Keep trying! You'll get better with practice!"
```

Testing Checklist:

- ☐ Program runs without errors
- ☐ Problems display in correct order (1, 2, 3... not 0, 1, 2...)
- ☐ Correctly identifies right and wrong answers
- ☐ Score calculation is accurate
- ☐ Appropriate feedback for different score ranges
- ☐ Handles all inputs properly

6 Part 6: Debug Detective (12 points)

Time estimate: 20 minutes
Each code snippet has issues. Find and fix them all:

6.1 Snippet 1: Rupee Counter (4 points)

Open week9_ex6_1.py and fix the code that counts money in rupees. It should print Rs. 0, Rs. 100, Rs. 200, etc.

```
print("Counting money:")
For amount in range(5)
    rupees = amount * 100
    print(f"Rs. {rupees}")
```

6.2 Snippet 2: Star Printer (4 points)

Open week9_ex6_2.py and fix the code that prints a pattern of stars. It should print 5 rows of stars, each with 3 stars.

```
print("Star Pattern:")
for row in range(5)
print("*" * 3)
print(f"Drew {row} rows of stars")
```

6.3 Snippet 3: Attendance Counter (4 points)

Open week9_ex6_3.py and fix the code that counts students present. It should count all 25 students, not just one.

```
# This code has a logic error - the count is wrong!
students_present = 0
for student in range(25):
    students_present = 1
print(f"Total students present: {students_present}")
# Should show 25 students, but doesn't. Fix it!
```

7 Part 7: Reflection Questions (5 points)

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

1. The `range(5)` function creates 0,1,2,3,4 - not 1,2,3,4,5. Why do you think Python designers chose to start at 0? How does this connect to what you might learn about lists later?

2. Look at the three loop patterns from class (building, accumulating, processing). Which pattern did you use most in this homework? Give a specific example.

3. When you had to fix the loop errors in Part 6, what was the most common mistake? How will you remember to avoid this error in your own code?

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

Only attempt if you've finished everything else!

8.1 Challenge 1: Number Pyramid (5 points)

Create a program that prints this number pyramid:

```
    1
   222
  33333
 4444444
555555555
```

Hints:

- You'll need to figure out spacing
- Each row has an odd number of digits (1, 3, 5, 7, 9)
- Row number determines which digit to print

Save this code as `week9_ex8_1.py`

8.2 Challenge 2: Fibonacci Sequence (5 points)

Create a program that generates the Fibonacci sequence using loops:

- First two numbers are 0 and 1
- Each next number is the sum of the previous two
  - *Example:* 0, 1, 1, 2, 3, 5, 8, 13, 21...
- Ask user how many Fibonacci numbers to generate
- Use a `for` loop to calculate and display them

Save this code as `week9_ex8_2.py`

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
- ☐ Attempted bonus section (optional)
- ☐ Compressed all your code files into a single zip file named `week9_hw.zip`

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

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

Fantastic work with your first loops!
Next week: Advanced loops with different range() options