مكتب

Homework Assignment

Week 11: Smart 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: Tracing Smart Loops (20 points)

Time estimate: 25 minutes

1.1 Exercise 1.1: Filter and Count Pattern (10 points)

Without running the code, trace this code that counts specific numbers:

```
# Counting multiples of 3 between 10 and 25
print("Finding multiples of 3...")
count = 0
sum_total = 0

for num in range(10, 21):
    if num % 3 == 0:
        print(f"Found: {num}")
        count = count + 1
        sum_total = sum_total + num

print(f"\nTotal found: {count}")
print(f"Sum of multiples: {sum_total}")
```

Fill in this trace table showing ALL iterations:

num	num % 3	num % 3 == 0?	count	sum_total	Output
10	1	False	0	0	*(nothing prints)*
11	___	______	___	___	_____
12	___	______	___	___	_____
13	___	______	___	___	_____
14	___	______	___	___	_____
15	___	______	___	___	_____
16	___	______	___	___	_____
17	___	______	___	___	_____
18	___	______	___	___	_____
19	___	______	___	___	_____
20	___	______	___	___	_____

List the first 4 multiples of 7 found:

1. _____

2. _____

3. _____

4. _____

Final count value: _____

Final sum_total value: _____

Now run the code in Thonny or an online IDE. Did you get the same results? _____

1.2 Exercise 1.2: Search with Early Exit (10 points)

Without running the code, trace this code that searches for a perfect cube:

```
# Finding first perfect cube greater than 100
print("Searching for perfect cube > 100...")
found = False
cube_root = 0

for n in range(1, 20):
    cube = n * n * n
    print(f"Checking: {n}^3 = {cube}")

    if cube > 100:
```

```
        print(f"Found it! {cube} is the first cube > 100")
        cube_root = n
        found = True
        break
    else:
        print("Too small, continuing...")

if found:
    print(f"The cube root is {cube_root}")
else:
    print("No cube found in range")
```

How many values of n are actually checked? _____

What is the first cube greater than 100? _____

Which values of n (1-19) are never checked due to break? _____

2 Part 2: Code Completion - Smart Loop Patterns (15 points)

Time estimate: 20 minutes

2.1 Exercise 2.1: Number Range Analyzer (8 points)

Open week11_ex2_1.py and complete the following program that analyzes numbers in different ranges:

```
# Number Range Analyzer
print("=== Number Range Analyzer ===")
print("Enter 10 numbers between 1 and 100")

# Initialize counters for different ranges
low_count = 0       # 1-33
medium_count = 0    # 34-66
high_count = 0      # 67-100
out_of_range = 0    # Outside 1-100

# Get 10 numbers and analyze
for i in range(10):
    num = int(input(f"Number {i+1}: "))

    # Check which range the number falls into
    if num < 1 or num > 100:
        _______ = _______ + 1
        print("  Out of range!")
    elif num <= 33:
        _______ = _______ + _______
        print("  Low range")
    elif _______ <= _______:  # Check for medium range
        medium_count = medium_count + 1
        print("  Medium range")
    else:
        _______ = _______ + _______
        print("  High range")

# Display results
print("\n=== Analysis Results ===")
print(f"Low (1-33): {low_count}")
print(f"Medium (34-66): {_______}")
print(f"High (67-100): {high_count}")
print(f"Out of range: {out_of_range}")
```

```
35
36 # Find most common range
37 if low_count > medium_count and low_count > high_count:
38     print("\nMost numbers were in the LOW range")
39 elif _______ > _______ and _______ > _______:
40     print("\nMost numbers were in the MEDIUM range")
41 elif high_count > low_count and high_count > medium_count:
42     print("\nMost numbers were in the HIGH range")
43 else:
44     print("\nNumbers were evenly distributed")
```

2.2 Exercise 2.2: Smart Sum Calculator (7 points)

Open week11_ex2_2.py and complete the following program that sums numbers with conditions:

```
1  # Smart Sum with Early Exit
2  print("=== Smart Sum Calculator ===")
3  print("Enter positive numbers to sum (negative to stop)")
4  print("Sum stops if total exceeds 1000")
5
6  total = 0
7  count = 0
8  exceeded_limit = False
9
10 # Allow up to 50 numbers
11 for i in range(50):
12     num = int(input(f"Number {i+1}: "))
13
14     # Check for stop condition
15     if num < 0:
16         print("Negative number entered - stopping!")
17         _______  # Exit the loop
18
19     # Add to total
20     total = _______ + _______
21     count = count + 1
22
23     # Check if exceeded limit
24     if _______ > _______:  # Check against 1000
25         print("Sum exceeded 1000 - stopping!")
26         exceeded_limit = True
27         _______  # Exit the loop
28
29 # Display results
30 print(f"\nNumbers entered: {count}")
31 print(f"Total sum: {_______}")
32
33 if exceeded_limit:
34     print("Stopped due to: Limit exceeded")
35 elif count == 50:
36     print("Stopped due to: Maximum numbers reached")
37 else:
38     print("Stopped due: User choice (negative number)")
```

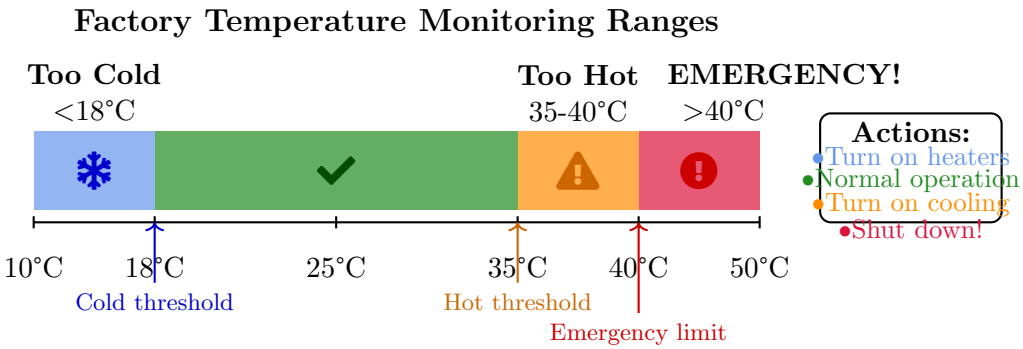
3 Part 3: Problem Solving - Temperature Monitor (18 points)

Time estimate: 25 minutes

Scenario: Factory Temperature Monitor

A factory needs to monitor temperatures throughout the day. The system should:

- Take 12 hourly readings (8 AM to 7 PM)
- Alert if temperature goes above 35°C (too hot) or below 18°C (too cold)
- Count how many hours were in the safe range (18-35°C)
- Calculate average temperature for the day
- Stop monitoring if temperature exceeds 40°C (emergency)



Open week11_ex3_1.py and complete the following program:

```python
# Factory Temperature Monitor
print("=== Factory Temperature Monitor ===")
print("Enter hourly temperatures (8 AM - 7 PM)")

# Initialize tracking variables
safe_hours = 0
total_temp = 0
emergency = False
hottest_temp = -100  # Very low starting value
coldest_temp = 100   # Very high starting value
hottest_hour = 0
coldest_hour = 0

# Monitor 12 hours
for hour in range(12):
    # Calculate actual time (8 AM + hour)
    current_time = 8 + hour
    if current_time > 12:
        display_time = f"{current_time - 12} PM"
    elif current_time == 12:
        display_time = "12 PM"
    else:
        display_time = f"{current_time} AM"

    temp = float(input(f"Temperature at {display_time}: "))
    total_temp = _______ + _______

    # Check for emergency
    if temp > 40:
        print("*** EMERGENCY! Temperature too high! ***")
        print("Shutting down monitoring system...")
        emergency = True
        _______  # Exit immediately
```

```
    # Track hottest and coldest

    if temp > _______:
        hottest_temp = temp
        hottest_hour = current_time

    if _______ < _______:
        coldest_temp = temp
        coldest_hour = _______

    # Check temperature range
    if temp < 18:
        print("  WARNING: Too cold!")
    elif temp > 35:
        print("  WARNING: Too hot!")
    else:
        print("  Temperature OK")
        _______ = _______ + 1

# Calculate statistics
if emergency:
    hours_monitored = hour + 1  # How many before emergency
else:
    hours_monitored = 12

average_temp = _______ / _______

# Display report
print("\n=== Daily Temperature Report ===")
print(f"Hours monitored: {hours_monitored}")
print(f"Average temperature: {average_temp} C")
print(f"Hours in safe range: {safe_hours}")
print(f"Hottest: {hottest_temp} C at ", end="")
if hottest_hour > 12:
    print(f"{hottest_hour - 12} PM")
elif hottest_hour == 12:
    print("12 PM")
else:
    print(f"{hottest_hour} AM")

print(f"Coldest: {coldest_temp} C at ", end="")
# Similar time display for coldest_hour
# Your code here...

if emergency:
    print("\n*** Emergency shutdown occurred ***")
else:
    safety_rate = _______ / 12 * 100
    print(f"\nSafety rate: {safety_rate}%")
```

4 Part 4: Debug Detective - Smart Loop Errors (12 points)

Time estimate: 20 minutes
Each code snippet has errors related to loops with conditions.

4.1 Snippet 1: Prime Checker (6 points)

Open week11_ex4_1.py and fix the errors in this prime number checker:

```
num = int(input("Enter a number to check: "))
is_prime = True

for divisor in range(2, num):
    if num % divisor = 0:
        is_prime == False
        break
        print("Found a divisor!")

if is_prime:
    print(f"{num} is prime!")
else:
    print(f"{num} is not prime!")
```

4.2 Snippet 2: Password Validator (6 points)

Open week11_ex4_2.py and fix the errors in this password validation code:

```
attempts = 0
valid = False

for attempt in range(3)
    password = input("Create password (8+ chars): ")
    attempts = attempts + 1

    if len(password) >= 8:
        print("Password accepted!")
        valid = True
    break
    else:
        print(f"Too short! {3 - attempts} tries left")

if valid = False:
    print("Account locked after 3 failed attempts")
```

5 Part 5: Program Development - Quiz Game System (30 points)

Program Requirements

Your program should:

- Ask the user how many questions they want (max 10)
- Generate random-style math problems using a pattern
- For each question:
  - Show a math problem (use question number to generate it)
  - Give the user 3 attempts to answer correctly
  - Use break when they get it right
  - Track score
- Stop the quiz if user types -999 as an answer
- Show different feedback based on attempts needed
- Display final score and performance message

Example Run:

```
=== Math Quiz Game ===
How many questions do you want? (1-10): 3

Question 1: What is 7 × 8?
Your answer: 54
Incorrect! Try again (2 attempts left)
Your answer: 56
Correct! Got it on attempt 2!

Question 2: What is 9 × 6?
Your answer: 54
Correct! Got it on the first try - Excellent!

Question 3: What is 5 × 12?
Your answer: -999
Quiz terminated early!

=== Quiz Results ===
Questions answered: 2 out of 3
Correct answers: 2
Score: 66.7%
Performance: Good job!
```

Open week11_ex5.py and write the complete program based on the requirements above.

6 Part 6: Reflection Questions (5 points)

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

1. The break statement helps us write more efficient programs. In the quiz game, why is it important to break out of the attempts loop when the answer is correct? What would happen if we didn't use break there?

2. We learned to combine loops with multiple conditions. Think about the temperature monitor program - how did using both "too hot" and "too cold" checks help make the program more useful? Can you think of another real-world monitoring situation that needs multiple condition checks?

3. Look back at your debugging exercises. What type of error with loops and conditions do you find most confusing? What strategy helps you spot these errors?

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

Only attempt if you've finished everything else!

7.1 Challenge 1: Fibonacci Finder (5 points)

Create a program that finds Fibonacci numbers with conditions: - Generate Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, ...) - Stop when you find the first Fibonacci number greater than 100 - Count how many even Fibonacci numbers you found - Each number is the sum of the previous two

Save this code as ***week11_ex7_1.py***

7.2 Challenge 2: Digital Root Calculator (5 points)

Write a program that calculates the digital root of numbers:

- The digital root is found by repeatedly summing a number's digits until you get a single digit
- Example: 38 → 3+8=11 → 1+1=2 (digital root is 2)
- Process multiple numbers until user enters 0 to quit
- Find the first number whose digital root equals a target value
- Count how many steps each calculation takes

The program should:

- Ask for a target digital root (1-9)
- Test numbers starting from 10
- Stop when finding a number with the target digital root
- Show the calculation steps
- Track which number required the most steps

Example run:

```
Enter target digital root (1-9): 5
Searching for a number with digital root 5...

Testing 10: 1+0=1 (1 step) - Not a match
Testing 11: 1+1=2 (1 step) - Not a match
Testing 12: 1+2=3 (1 step) - Not a match
Testing 13: 1+3=4 (1 step) - Not a match
Testing 14: 1+4=5 (1 step) - Found it!

First number with digital root 5 is: 14
Maximum steps needed so far: 1
```

Save this code as ***week11_ex7_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 thoughtfully
- ☐ Attempted bonus section (optional)
- ☐ Compressed all your code files into a single zip file named `week11_hw.zip`

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

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

Fantastic work mastering smart loops!
Next week: Systematic debugging strategies