مكتب

Homework Assignment

Week 12: Debugging Workshop

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: Error Type Detective (12 points)

Time estimate: 20 minutes

1.1 Exercise 1.1: Classify the Errors (12 points)

Without running the code, for each code snippet below, identify the error type (Syntax, Runtime, or Logic) and briefly explain why. If the code runs but gives wrong output, show what it produces vs. what it should produce.

Code Snippet	Error Type	Explanation
`# Calculate discount price` `price = 1000` `discount = 20` `final = price - discount / 100` `print(f"Pay: Rs.{final}")`		Expected: Rs.800 Actually prints:

Continued on next page

Table 1 – continued from previous page

Code Snippet	Error Type	Explanation

`# Sum even numbers`
`total = 0`
`for i in range(1, 11):`
`    if i % 2 = 0:`
`        total += i`
`print(f"Sum: {total}")`		

`# Print pattern`
`for i in range(3)`
`    print("*" * i)`		

`# Calculate average`
`sum = 0`
`count = 5`
`for i in range(count):`
`    num = int(input("Number: "))`
`    sum += num`
`average = sum / Count`
`print(f"Average: {average}")`		

2 Part 2: Error Message Decoder (15 points)

Time estimate: 25 minutes

2.1 Exercise 2.1: Match Error Messages to Causes (8 points)

Match each error message to its most likely cause and provide a simple fix:
Causes to match:

A. Dividing by a variable that equals zero

B. Mixing strings and numbers in addition

C. Inconsistent spacing after if/for/while

D. Converting decimal string to integer directly

Error Message	Cause	Quick Fix
`TypeError: can only concatenate str (not "int") to str`		
`IndentationError: unindent does not match any outer indentation level`		
`ValueError: invalid literal for int() with base 10: '12.5'`		
`ZeroDivisionError: division by zero`		

2.2 Exercise 2.2: Predict the Error (7 points)

Without running on a computer, for each code snippet, predict what error message Python will show (be specific!):

```python
# Code 1
total = 100
count = 0
average = total / count
```

Error message: ______________________________

```python
# Code 2
age = "15"
if age < 18:
    print("Too young")
```

Error message: ______________________________

```python
# Code 3
score = input("Enter score: ")
bonus = 10
total = score + bonus
```

Error message: ______________________________

3 Part 3: Systematic Debugging Practice (20 points)

Time estimate: 30 minutes

3.1 Exercise 3.1: Grade Calculator Debug (10 points)

This grade calculator has multiple bugs. Use various debugging techniques to find and fix all issues. Followng table shows grade assignment rules:

Grade	Percentage Range	Status
A+	90% and above	Pass
A	80% – 89%	Pass
B	70% – 79%	Pass
C	60% – 69%	Pass
F	Below 60%	Fail

Table 2: Grade Distribution Rules

The following code is already provided in the `week11_ex3_1.py` file.

```python
# Buggy Grade Calculator
print("=== Grade Calculator ===")

# Get marks for 5 subjects
total = 0
for i in range(5):
    subject = input(f"Enter marks for subject {i}: ")
    total = total + subject

# Calculate percentage
percentage = total / 500 * 100
```

```python

# Determine grade
if percentage >= 90:
    grade = "A+"
elif percentage >= 80
    grade = "A"
elif percentage > 70:
    grade = "B"
elif percentage > 60:
    grade = "C"
else:
    grade = "F"

print(f"Total: {total}/500")
print(f"Percentage: {percentage}%")
print(f"Grade: {grade}")

# Check if passed
if grade = "F":
    print("Failed - Study harder!")
else:
    print("Passed - Well done!")
```

3.2 Exercise 3.2: Number Pattern Analyzer Debug (10 points)

Open the file week11_ex3_2.py and fix the bugs in this number pattern analyzer program. It has various issues that prevent it from working correctly.

```python
# Number Pattern Analyzer - Has logic bugs!
print("=== Number Pattern Analyzer ===")
print("Enter 5 numbers to analyze patterns")

# Get 5 numbers and track patterns
sum_all = 0
even_count = 0
positive_count = 0

for i in range(5):
    num = input(f"Enter number {i+1}: ")
    sum_all = sum_all + num

    # Check if even
    if num % 2 == 1:
        even_count = even_count + 1

    # Check if positive
    if num > 0
        positive_count = positive_count + 1

# Calculate average
average = sum_all / 4

# Analyze the pattern
print(f"\nAnalysis Results:")
print(f"Sum: {sum_all}")
print(f"Average: {average}")
print(f"Even numbers: {even_count}")
print(f"Positive numbers: {positive_count}")

# Determine pattern type
```

```python
if even_count = 5:
    print("Pattern: All even!")
elif positive_count == 5:
    print("Pattern: All positive!")
elif even_count > 3 or positive_count > 3:
    print("Pattern: Mostly even and positive!")
else
    print("Pattern: Mixed numbers")

# Bonus calculation
if average > 50:
    bonus = average * 1
    print(f"Bonus points: {bonus}")
elif average < 0:
    bonus = 5
    print(f"Bonus points: {bonus}")
else:
    print("No bonus - average too low!")
```

Testing checklist:

- ☐ Test with: 2, 4, 6, 8, 10 - should show ”All even!”
- ☐ Test with: 1, 3, 5, 7, 9 - should show correct even count
- ☐ Test with: -5, -2, 0, 3, 4 - should show correct positive count
- ☐ Average calculation works correctly

Sample Output After Fixing:

```
Enter 5 numbers to analyze patterns
Enter number 1: 2
Enter number 2: 3
Enter number 3: 4
Enter number 4: 5
Enter number 5: 6

Analysis Results:
Sum: 20
Average: 4.0
Even numbers: 3
Positive numbers: 5
Pattern: All positive!
Bonus points: 5
```

4 Part 4: Code Tracing Challenge (15 points)

Time estimate: 20 minutes

4.1 Exercise 4.1: Loop Logic Trace (15 points)

This program has a subtle logic error. Trace through it step by step to find the bug:

```python
# Find all numbers divisible by both 3 and 5
print("Numbers divisible by both 3 and 5:")
count = 0

```

```python
for num in range(1, 16):
    # Check if divisible by 3
    if num % 3 == 0:
        # Check if also divisible by 5
        if num % 5 == 0:
            print(num, end=" ")
        count += 1

print(f"\nFound {count} numbers")
```

Fill in the trace table for key iterations:

num	**num % 3**	**num % 5**	**Prints?**	**count**	**Notes**
1	1	1	No	0	Not divisible by 3
2					
3					
5					
6					
9					
10					
12					
15					Should print!

What does the program actually count? ______________________________

How would you fix it? ______________________________

5 Part 5: Real-World Debugging Challenge (18 points)

Time estimate: 25 minutes
Your friend wrote a program to calculate the cost of a school trip, but it's full of bugs. They asked for your help to fix it. Use systematic debugging to make it work correctly!

5.1 Exercise 5.1: Fix the School Trip Calculator (18 points)

Your friend wrote a program to calculate the cost per student for a school trip, but it's full of bugs! The program should:

- Calculate the per-student cost for bus, lunch, and entry tickets
- Apply group discounts (10% for 20+ students, 15% for 30+ students)
- Show a cost breakdown and check if it's within the Rs.500 budget

Unfortunately, the code has multiple errors of different types. Open the file week12_ex5_1.py and read through the code. Your task is to find and fix all the bugs.

```python
print("=== School Trip Cost Calculator ===")
print("Calculate cost per student for the trip")

# Get trip details
total_bus_cost = input("Total bus rental cost (Rs.): ")
lunch_per_student = input("Lunch cost per student (Rs.): ")
ticket_per_student = input("Entry ticket per student (Rs.): ")
num_students = input("Number of students going: ")

# Calculate per student costs
bus_per_student = total_bus_cost / num_students
total_per_student = bus_per_student + lunch_per_student + ticket_per_student

# Apply group discount
print("\nChecking for group discount...")
if num_students >= "30":
    discount = 15
    print(f"Large group! You get {discount}% discount")
elif num_students >= 20
    discount = 10
    print(f"Medium group! You get {discount}% discount")
else:
    discount = 0
    print("Small group - no discount available")

# Calculate final cost
discount_amount = bus_per_student * discount / 100
final_cost = total_per_student + discount_amount

# Print summary
print("\n----- Cost Breakdown -----")
print(f"Students going: {num_students}")
print(f"Bus cost per student: Rs.{bus_per_student}")
print(f"Lunch per student: Rs.{lunch_per_student}")
print(f"Ticket per student: Rs.{ticket_per_student}")
print(f"Subtotal per student: Rs.{total_per_student}")

if discount < 0:
    print(f"Discount: {discount}%")
    print(f"You save: Rs.{discount_amount}")

print(f"\nFinal cost per student: Rs.{final_cost}")
print("-------------------------")

# Calculate total collection needed
total_collection = total_per_student * num_students
print(f"\nTotal to collect: Rs.{total_collection}")

# Check if affordable
if Final_cost >= 500:
    print("Great! This is within budget.")
else:
    print("This exceeds the Rs.500 budget per student.")
```

Your debugging tasks:

1. Run the program and document each error you encounter in the table below

2. Fix the errors one by one using systematic debugging techniques
3. Test your fixed program with the example input provided
4. Ensure your output matches the expected output exactly

Test Input:

```
Total bus rental cost (Rs.): 12000
Lunch cost per student (Rs.): 150
Entry ticket per student (Rs.): 200
Number of students going: 25
```

Expected Output (after fixing all bugs):

```
=== School Trip Cost Calculator ===
Calculate cost per student for the trip
Total bus rental cost (Rs.): 12000
Lunch cost per student (Rs.): 150
Entry ticket per student (Rs.): 200
Number of students going: 25

Checking for group discount...
Medium group! You get 10% discount

----- Cost Breakdown -----
Students going: 25
Bus cost per student: Rs.480.0
Lunch per student: Rs.150.0
Ticket per student: Rs.200.0
Subtotal per student: Rs.830.0
Discount: 10%
You save: Rs.83.0

Final cost per student: Rs.747.0
-------------------------

Total to collect: Rs.18675.0

This exceeds the Rs.500 budget per student.
```

Debugging Documentation Table:
Complete this table as you find and fix each bug. Include syntax errors, type errors, logic errors, and any other issues you encounter.

Bug #	Line #	Error Type	Error Description	Fix Applied
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

6 Part 6: Reflection Questions (5 points)

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

1. Which type of error (syntax, runtime, or logic) do you find hardest to debug? Why? What strategy helps you most with that type?

2. Print debugging helped us "see inside" our programs. Describe a specific time this week when adding a print statement helped you solve a problem.

3. Error messages used to be scary, but now they're helpful clues. What's your process now when you see an error message? List your steps.

7 Bonus Section: Expert Debugger Challenge (Optional - 10 extra points)

Only attempt if you've finished everything else!

7.1 Challenge 1: The Mystery Bug (5 points)

This number guessing game works sometimes but fails mysteriously other times.
Open the file week12_ex7_1.py and find the bugs:

```python
import random

# Number guessing game with mystery bug
secret = random.randint(1, 10)
attempts = 3

print("Guess my number (1-10)!")
print(f"You have {attempts} attempts.")

for attempt in range(attempts):
    guess = input(f"\nAttempt {attempt + 1}: ")

    if guess == secret:  # Bug: comparing different types!
        print("Correct! You win!")
        break
    elif guess < secret:
        print("Too low!")
    else:
        print("Too high!")
else:  # This runs if loop completes without break
    print(f"\nGame over! The number was {secret}")
```

Hint: The bug appears when certain numbers are chosen. Why does it work sometimes?

7.2 Challenge 2: Debug by Design (5 points)

Create a "Debugging Puzzle" program that:

1. Contains exactly 5 different bugs (mix of all three types)
2. When all bugs are fixed, prints a secret message
3. Include comments hinting at each bug's location (but not the solution!)

Share your puzzle with a classmate next week!

Save your work as `week12_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 `week12_hw.zip`

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

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

Excellent debugging work! You're now equipped to tackle any Python error!
Next week: Celebrating your journey with creative Python Turtle Graphics!