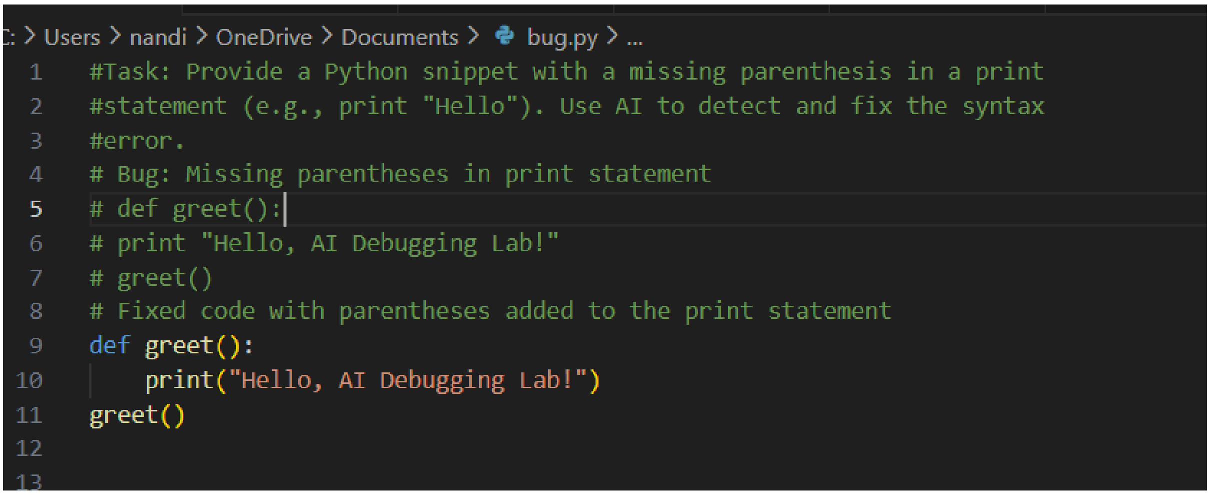
ASSIGNMENT-7.1

NAME: Syeda Hafsa Fathima

HTNO:2303A52381

BATCH:39

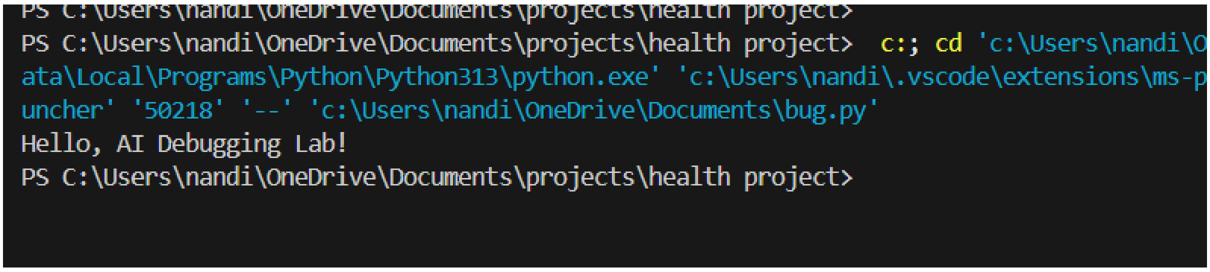
TASK-1:



: > Users > nandi > OneDrive > Documents > bug.py > ...

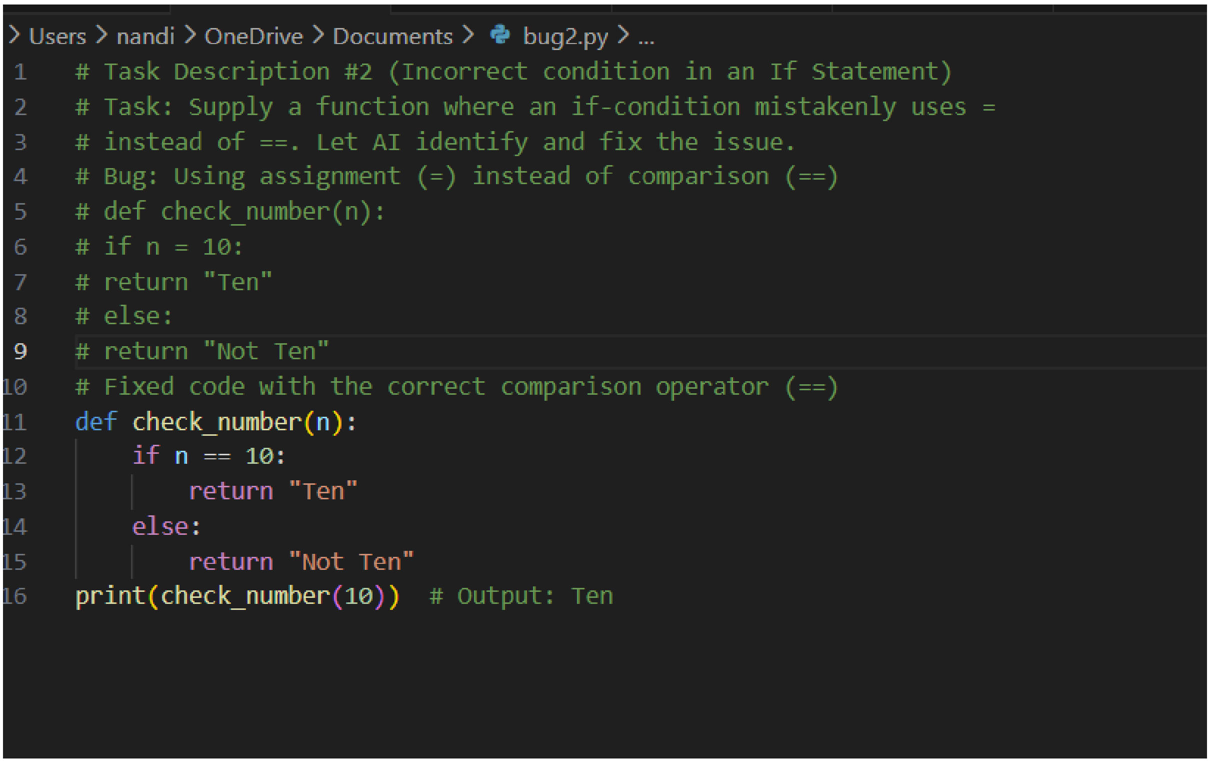
```python
#Task: Provide a Python snippet with a missing parenthesis in a print
#statement (e.g., print "Hello"). Use AI to detect and fix the syntax
#error.
# Bug: Missing parentheses in print statement
# def greet():
# print "Hello, AI Debugging Lab!"
# greet()
# Fixed code with parentheses added to the print statement
def greet():
    print("Hello, AI Debugging Lab!")
greet()


```

Output:



```
PS C:\Users\nandi\OneDrive\Documents\projects\health project>
PS C:\Users\nandi\OneDrive\Documents\projects\health project>  c:; cd 'c:\Users\nandi\O
ata\Local\Programs\Python\Python313\python.exe' 'c:\Users\nandi\.vscode\extensions\ms-p
uncher' '50218' '--' 'c:\Users\nandi\OneDrive\Documents\bug.py'
Hello, AI Debugging Lab!
PS C:\Users\nandi\OneDrive\Documents\projects\health project>
```

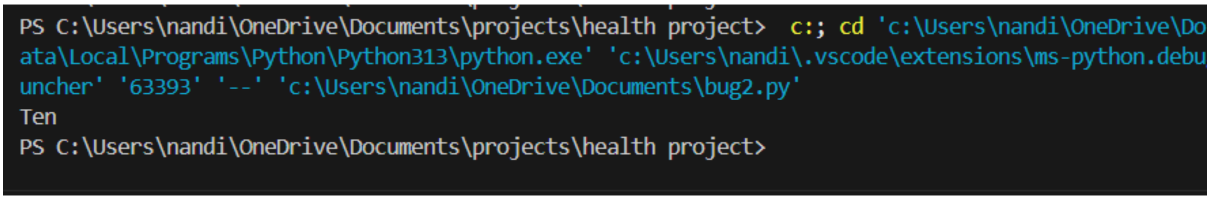
TASK 2:



> Users > nandi > OneDrive > Documents > bug2.py > ...

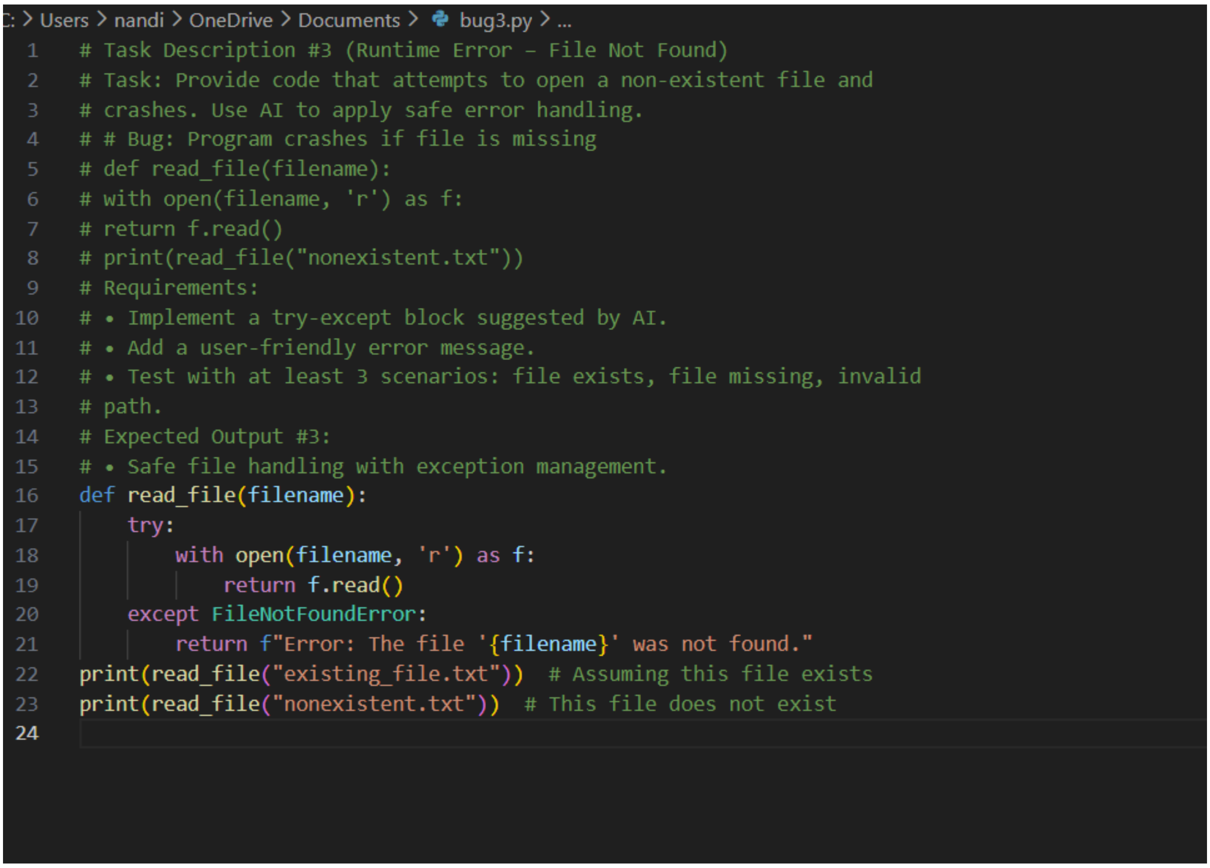
```python
# Task Description #2 (Incorrect condition in an If Statement)
# Task: Supply a function where an if-condition mistakenly uses =
# instead of ==. Let AI identify and fix the issue.
# Bug: Using assignment (=) instead of comparison (==)
# def check_number(n):
# if n = 10:
# return "Ten"
# else:
# return "Not Ten"
# Fixed code with the correct comparison operator (==)
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"
print(check_number(10))  # Output: Ten
```

OUTPUT:



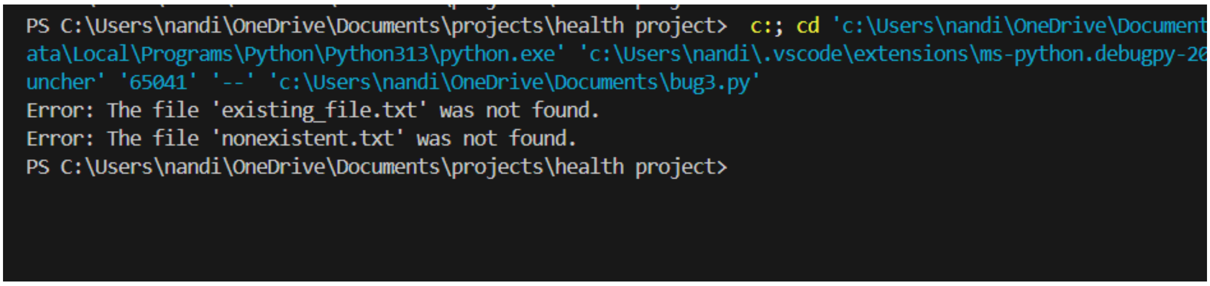
```
PS C:\Users\nandi\OneDrive\Documents\projects\health project>  c:; cd 'c:\Users\nandi\OneDrive\Do
ata\Local\Programs\Python\Python313\python.exe' 'c:\Users\nandi\.vscode\extensions\ms-python.debu
uncher' '63393' '--' 'c:\Users\nandi\OneDrive\Documents\bug2.py'
Ten
PS C:\Users\nandi\OneDrive\Documents\projects\health project>
```

TASK 3:



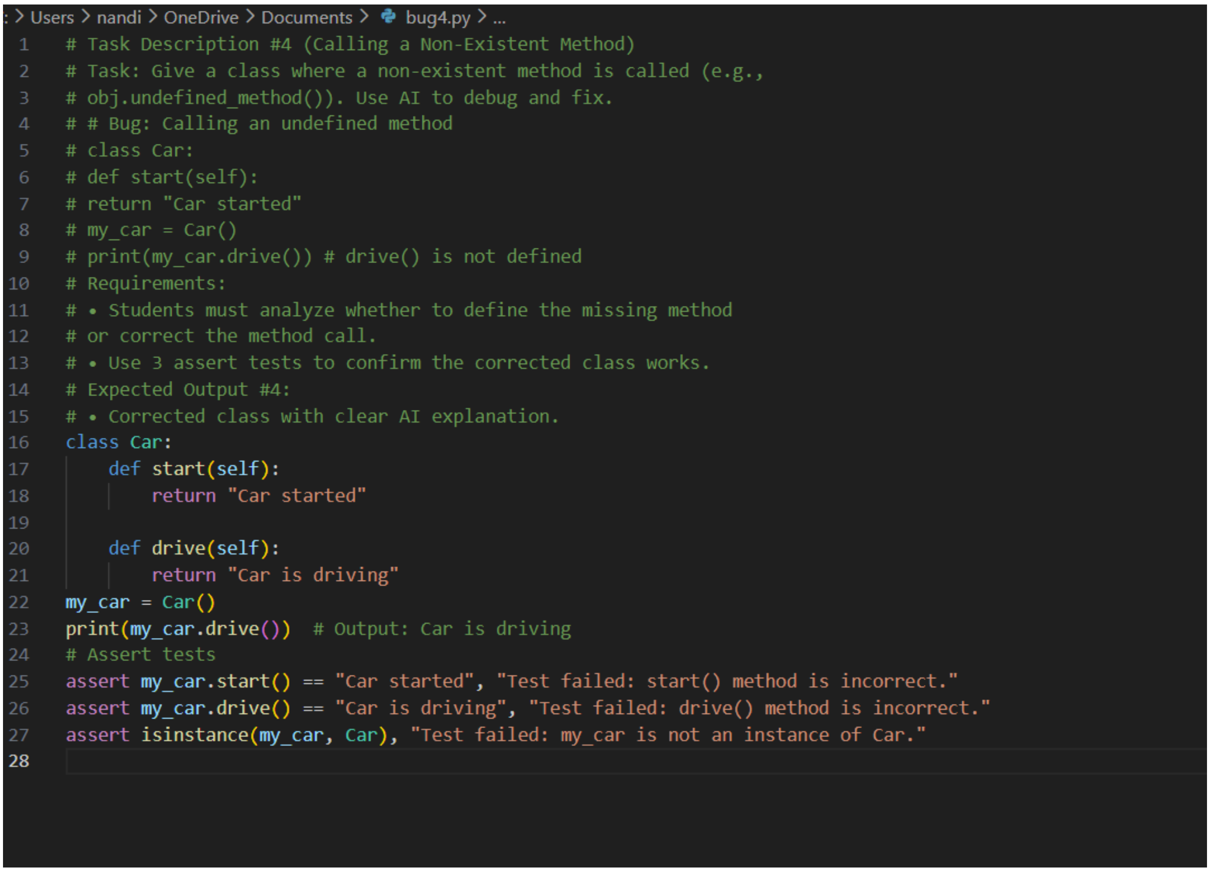
```
C: > Users > nandi > OneDrive > Documents > bug3.py > ...
# Task Description #3 (Runtime Error – File Not Found)
# Task: Provide code that attempts to open a non-existent file and
# crashes. Use AI to apply safe error handling.
# # Bug: Program crashes if file is missing
# def read_file(filename):
# with open(filename, 'r') as f:
# return f.read()
# print(read_file("nonexistent.txt"))
# Requirements:
# • Implement a try-except block suggested by AI.
# • Add a user-friendly error message.
# • Test with at least 3 scenarios: file exists, file missing, invalid
# path.
# Expected Output #3:
# • Safe file handling with exception management.
def read_file(filename):
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        return f"Error: The file '{filename}' was not found."
print(read_file("existing_file.txt"))  # Assuming this file exists
print(read_file("nonexistent.txt"))  # This file does not exist

```

OUTPUT:



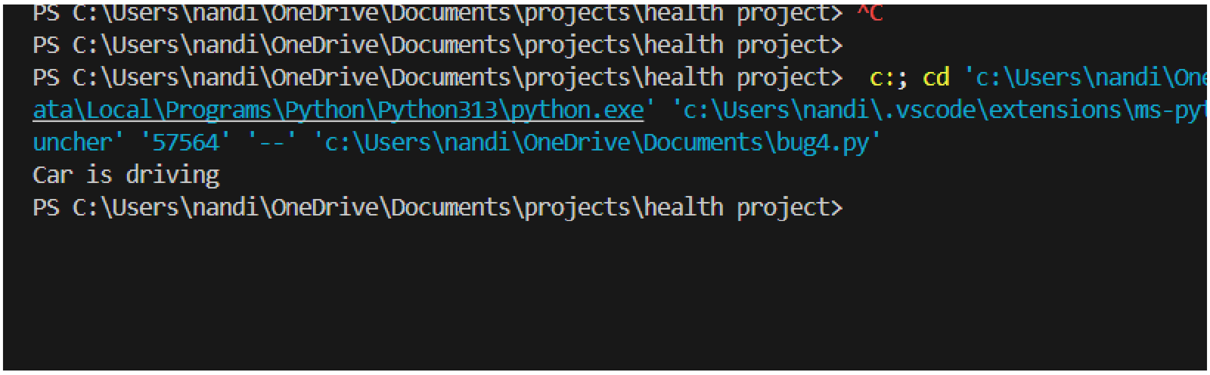
```
PS C:\Users\nandi\OneDrive\Documents\projects\health project>  c:; cd 'c:\Users\nandi\OneDrive\Document
ata\Local\Programs\Python\Python313\python.exe' 'c:\Users\nandi\.vscode\extensions\ms-python.debugpy-20
uncher' '65041' '--' 'c:\Users\nandi\OneDrive\Documents\bug3.py'
Error: The file 'existing_file.txt' was not found.
Error: The file 'nonexistent.txt' was not found.
PS C:\Users\nandi\OneDrive\Documents\projects\health project>
```

TASK 4:



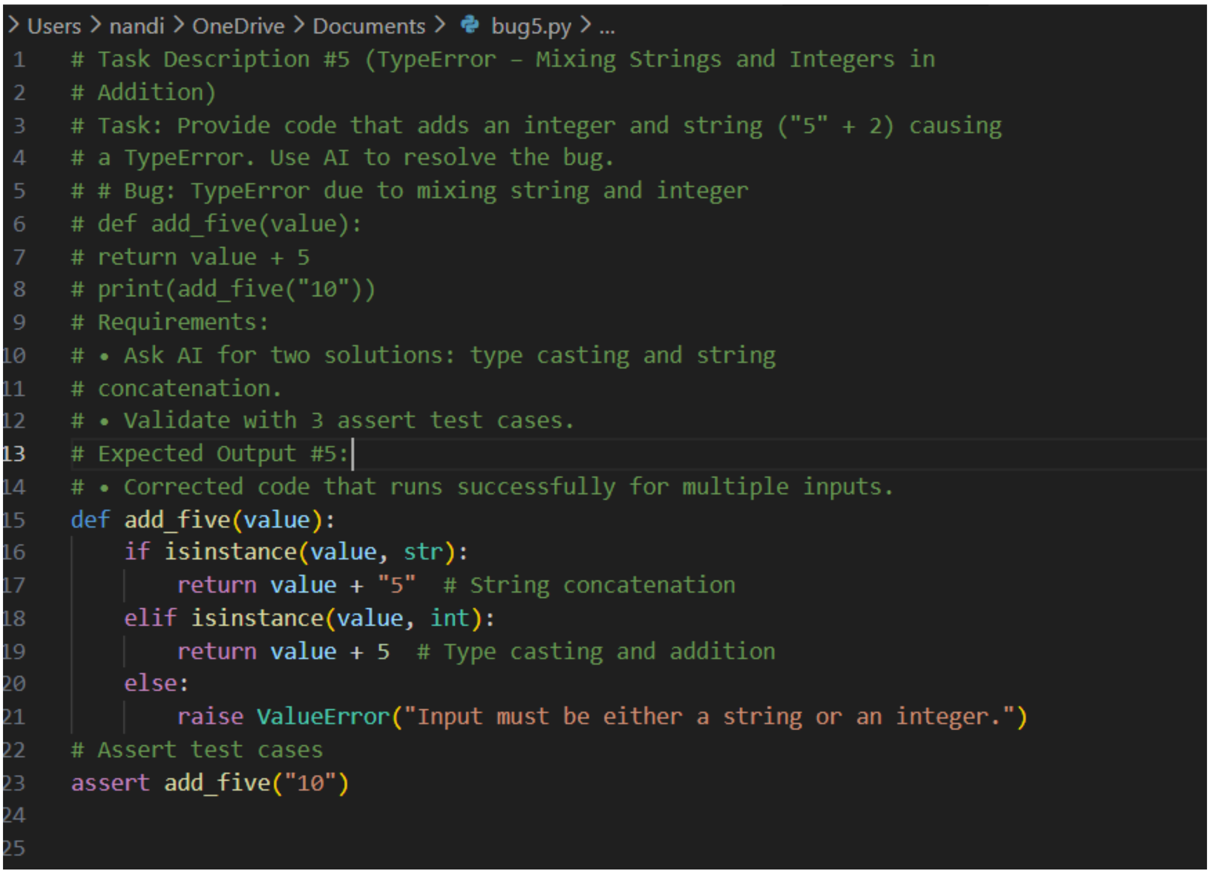
```python
# Task Description #4 (Calling a Non-Existent Method)
# Task: Give a class where a non-existent method is called (e.g.,
# obj.undefined_method()). Use AI to debug and fix.
# # Bug: Calling an undefined method
# class Car:
# def start(self):
# return "Car started"
# my_car = Car()
# print(my_car.drive()) # drive() is not defined
# Requirements:
# • Students must analyze whether to define the missing method
# or correct the method call.
# • Use 3 assert tests to confirm the corrected class works.
# Expected Output #4:
# • Corrected class with clear AI explanation.
class Car:
    def start(self):
        return "Car started"

    def drive(self):
        return "Car is driving"
my_car = Car()
print(my_car.drive())  # Output: Car is driving
# Assert tests
assert my_car.start() == "Car started", "Test failed: start() method is incorrect."
assert my_car.drive() == "Car is driving", "Test failed: drive() method is incorrect."
assert isinstance(my_car, Car), "Test failed: my_car is not an instance of Car."
```

OUTPUT:



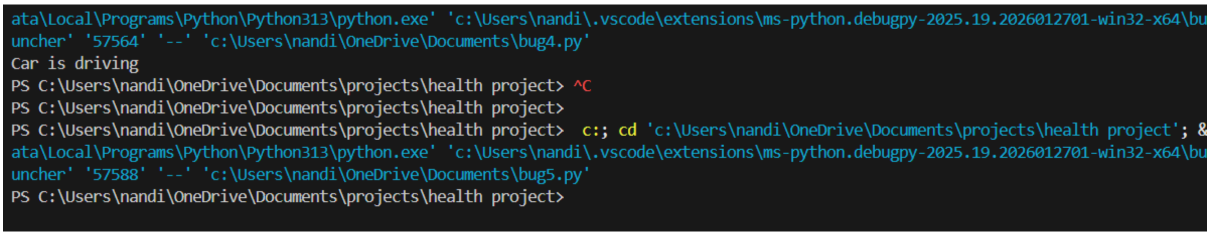
```
PS C:\Users\nandi\OneDrive\Documents\projects\health project> ^C
PS C:\Users\nandi\OneDrive\Documents\projects\health project>
PS C:\Users\nandi\OneDrive\Documents\projects\health project>  c:; cd 'c:\Users\nandi\One
ata\Local\Programs\Python\Python313\python.exe' 'c:\Users\nandi\.vscode\extensions\ms-pyt
uncher' '57564' '--' 'c:\Users\nandi\OneDrive\Documents\bug4.py'
Car is driving
PS C:\Users\nandi\OneDrive\Documents\projects\health project>
```

TASK 5:



```python
# Task Description #5 (TypeError – Mixing Strings and Integers in
# Addition)
# Task: Provide code that adds an integer and string ("5" + 2) causing
# a TypeError. Use AI to resolve the bug.
# # Bug: TypeError due to mixing string and integer
# def add_five(value):
# return value + 5
# print(add_five("10"))
# Requirements:
# • Ask AI for two solutions: type casting and string
# concatenation.
# • Validate with 3 assert test cases.
# Expected Output #5:
# • Corrected code that runs successfully for multiple inputs.
def add_five(value):
    if isinstance(value, str):
        return value + "5"  # String concatenation
    elif isinstance(value, int):
        return value + 5  # Type casting and addition
    else:
        raise ValueError("Input must be either a string or an integer.")
# Assert test cases
assert add_five("10")
```

OUTPUT:



```
ata\Local\Programs\Python\Python313\python.exe' 'c:\Users\nandi\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bu
uncher' '57564' '--' 'c:\Users\nandi\OneDrive\Documents\bug4.py'
Car is driving
PS C:\Users\nandi\OneDrive\Documents\projects\health project> ^C
PS C:\Users\nandi\OneDrive\Documents\projects\health project>
PS C:\Users\nandi\OneDrive\Documents\projects\health project>  c:; cd 'c:\Users\nandi\OneDrive\Documents\projects\health project'; &
ata\Local\Programs\Python\Python313\python.exe' 'c:\Users\nandi\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bu
uncher' '57588' '--' 'c:\Users\nandi\OneDrive\Documents\bug5.py'
PS C:\Users\nandi\OneDrive\Documents\projects\health project>
```