ASSIGNMENT 7.3

D.Charmi

2303A51314

Batch – 05

TASK – 01 :

Prompt : Identify and fix the syntax error in a Python function add(a, b) where the function definition is incorrect. Correct the code and briefly explain what the error was and how it was fixed with example usage.

Code :

```
assignment7.3.py > ...
#Identify and fix the syntax error in a Python function add(a, b) where the function definition is incorrect.
#Correct the code and briefly explain what the error was and how it was fixed with example usage.
# Original code with syntax error
'''def add(a, b)
    return a + b'''
# Corrected code
def add(a, b):
    return a + b
# Explanation:
# The error in the original code was a missing colon (:) at the end of the function definition line.
# In Python, function definitions must end with a colon to indicate the start of the function body.
# By adding the colon, the syntax error is resolved, and the function can be executed correctly.
# Example usage
result = add(3, 5)
print(result)  # Output: 8
```

Output :

```
PS C:\Users\devis\OneDrive\Desktop\AI Assisted Coding> & C:/Users/devis/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/devis/OneDrive/Desktop/AI Assisted Coding/assignment7.3.py"
  File "c:\Users\devis\OneDrive\Desktop\AI Assisted Coding\assignment7.3.py", line 4
    def add(a, b)
               ^
SyntaxError: expected ':'
PS C:\Users\devis\OneDrive\Desktop\AI Assisted Coding> & C:/Users/devis/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/devis/OneDrive/Desktop/AI Assisted Coding/assignment7.3.py"
8
PS C:\Users\devis\OneDrive\Desktop\AI Assisted Coding>
```

Explanation :

The original function had a syntax error because the colon (:) was missing at the end of the function definition. In Python, the colon is required to start the function body. Adding the colon fixes the error, allowing the function to run correctly. The example shows the function successfully adding two numbers and printing the result.

TASK – 02:

Prompt : Find and fix a logical error in a Python loop that causes it to run endlessly. Correct the loop and briefly explain why the issue occurred and how it was resolved. Give both the original erroneous code and the corrected code.

Code :

```python
#Find and fix a logical error in a Python loop that causes it to run endlessly.
#Correct the loop and briefly explain why the issue occurred and how it was resolved.
#Give both the original erroneous code and the corrected code.
# Original erroneous code
count = 0
while count < 5:
    print("Count is:", count)
    # Missing increment of count variable
# Corrected code
count = 0
while count < 5:
    print("Count is:", count)
    count += 1  # Incrementing count to avoid infinite loop
# Explanation:
# The issue occurred because the loop condition (count < 5) was never updated within the loop.
# As a result, the loop would run indefinitely since 'count' remained 0.
```

Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Python

Count is: 0
Count is: 0
Count is: 0
Count is: 0
Count is: 0
Count is: 0
Count is: 0
Count is: 0
Count is: 0
Traceback (most recent call last):
  File "c:\Users\devis\OneDrive\Desktop\AI Assisted Coding\assignment7.3.py", line 7, in <module>
    print("Count is:", count)
    ~~~~^^^^^^^^^^^^^^^^^^^^
KeyboardInterrupt
PS C:\Users\devis\OneDrive\Desktop\AI Assisted Coding> & C:/Users/devis/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/devis/OneDrive/Desktop/AI Assisted Coding/ass
ignment7.3.py"
Count is: 0
Count is: 1
Count is: 2
Count is: 3
Count is: 4
PS C:\Users\devis\OneDrive\Desktop\AI Assisted Coding>
```

Explanation :

The loop ran endlessly because the count variable was never updated inside the loop. Since the condition count < 5 always remained true, the loop never stopped. Adding an increment to count inside the loop updates the condition and allows the loop to terminate correctly.

TASK 3 :

Prompt : Identify the runtime error in a Python function that performs division. Fix the issue using proper error handling and explain how the solution prevents the program from crashing.

Code

```python
#Identify the runtime error in a Python function that performs division.
#Fix the issue using proper error handling and explain how the solution prevents the program from
# Erroneous Code Snippet
def divide_numbers(num1, num2):
    result = num1 / num2
    return result
# Corrected Code Snippet
def divide_numbers(num1, num2):
    try:
        result = num1 / num2
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."
    except TypeError:
        return "Error: Invalid input type. Please provide numbers."
    return result
# Example Usage
print(divide_numbers(10, 2))  # Output: 5.0
print(divide_numbers(10, 0))  # Output: Error: Division by zero is not allowed.
print(divide_numbers(10, 'a'))  # Output: Error: Invalid input type. Please provide numbers.
```

Output :

```
PS C:\Users\devis\OneDrive\Desktop\AI Assisted Coding> & C:/Users/devis/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/devis/OneDrive/Desktop/AI Assisted Coding/assignment7.3.py"
5.0
Error: Division by zero is not allowed.
Error: Invalid input type. Please provide numbers.
PS C:\Users\devis\OneDrive\Desktop\AI Assisted Coding>
```

Explanation :

The erroneous code snippet does not handle the case where the second number (num2) is zero, which would raise a ZeroDivisionError and cause the program to crash. Additionally, if non-numeric types are passed as arguments, a TypeError would occur. The corrected code snippet includes a try-except block to catch these specific exceptions. This prevents the program from crashing by providing user-friendly error messages instead of terminating unexpectedly.

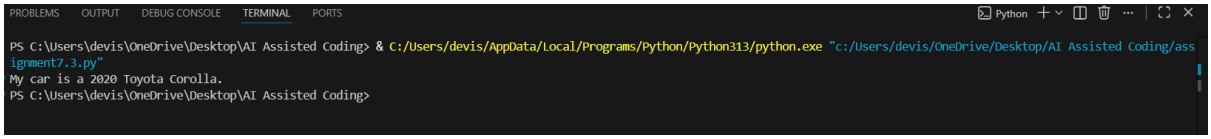
TASK – 04

Prompt : Find and correct the error in a Python class constructor where the required parameter is missing. Fix the class and briefly explain why this parameter is necessary. Give both the incorrect and corrected versions of the code.

Code :

```python
#Find and correct the error in a Python class constructor where the required parameter is missing.
#Fix the class and briefly explain why this parameter is necessary.
#Give both the incorrect and corrected versions of the code.
# Incorrect version
class Car:
    def __init__(self, make, model):
        self.make = make
        self.model = model
# Corrected version
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
# Explanation:
# The 'year' parameter is necessary to fully define the Car object.
# Without it, we cannot capture all essential attributes of a car, which may be important for various operations or comparisons involving Car o
# Example usage:
my_car = Car("Toyota", "Corolla", 2020)
print(f"My car is a {my_car.year} {my_car.make} {my_car.model}.")
```

Output :



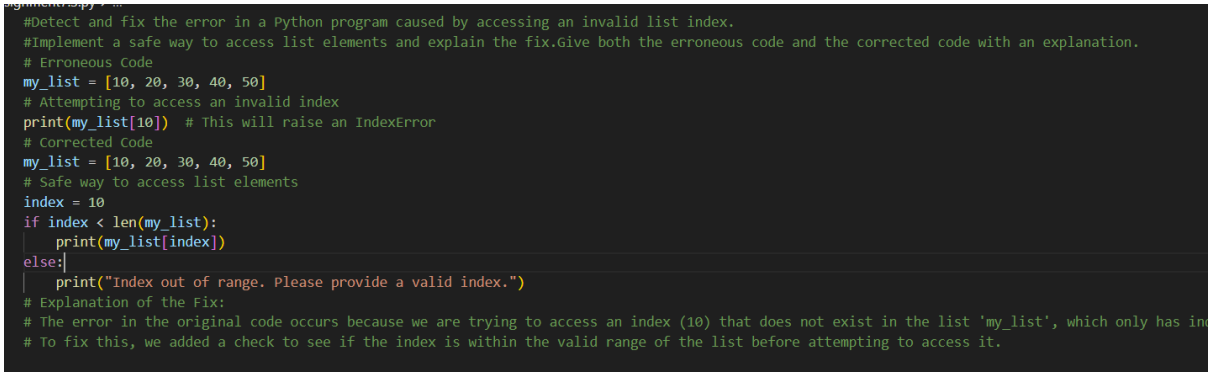
Explanation :

The incorrect version of the class was missing the year parameter, so the Car object did not contain complete information. The corrected constructor includes the year parameter to properly initialize all essential attributes of a car. This allows each object to store full details and be used correctly in the program.

TASK – 05

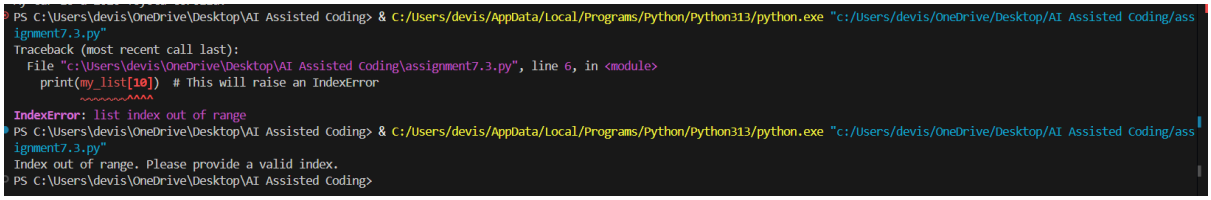
Prompt : Detect and fix the error in a Python program caused by accessing an invalid list index. Implement a safe way to access list elements and explain the fix.Give both the erroneous code and the corrected code with an explanation**.**

Code :



```python
#Detect and fix the error in a Python program caused by accessing an invalid list index.
#Implement a safe way to access list elements and explain the fix.Give both the erroneous code and the corrected code with an explanation.
# Erroneous Code
my_list = [10, 20, 30, 40, 50]
# Attempting to access an invalid index
print(my_list[10])  # This will raise an IndexError
# Corrected Code
my_list = [10, 20, 30, 40, 50]
# Safe way to access list elements
index = 10
if index < len(my_list):
    print(my_list[index])
else:
    print("Index out of range. Please provide a valid index.")
# Explanation of the Fix:
# The error in the original code occurs because we are trying to access an index (10) that does not exist in the list 'my_list', which only has in
# To fix this, we added a check to see if the index is within the valid range of the list before attempting to access it.
```

Output :



Explanation :

The error occurred because the program tried to access an index that does not exist in the list. In the corrected code, a length check is added before accessing the list element. This ensures the index is within the valid range and prevents the program from crashing due to an IndexError.