

AI Assisted Coding

2403A52100

Batch: 04

➤ Task 1:

```
"""Task Description #1:
• Introduce a buggy Python function that calculates the factorial of a number using recursion.
Use Copilot or Cursor AI to detect and fix the logical or syntax errors

"""

PROMPT:
Write a Python function that calculates the factorial of a number using recursion,
but intentionally include a logical or syntax error. Then, use Copilot or Cursor AI to
detect and fix the error in the function
"""

# Buggy factorial function (intentional error: missing base case for n==0)
def factorial_buggy(n):
    return n * factorial_buggy(n-1)

# Fixed factorial function
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n-1)

# Example usage
if __name__ == "__main__":
    print("Buggy factorial(5):")
    try:
        print(factorial_buggy(5))
    except RecursionError as e:
        print("Error:", e)
    print("Fixed factorial(5):", factorial(5))
```

TASK 2

"""

Task Description 2:

Provide a list sorting function that fails due to a type error

(e.g., sorting list with mixed integers and strings).

Prompt AI to detect the issue and fix the code for consistent sorting.

"""

"""

PROMPT:

Prompt: Write a Python function that attempts to sort a

list containing both integers and strings, causing a type error.

Then, prompt AI to detect the issue and fix the code so that the list is sorted consistently.

"""

Buggy sorting function (will fail with TypeError)

```
def sort_mixed_list_buggy(lst):  
    return sorted(lst)
```

Fixed sorting function: convert all elements to strings before sorting

```
def sort_mixed_list_fixed(lst):  
    return sorted(lst, key=str)
```

Example usage

```
if __name__ == "__main__":  
    mixed = [3, "2", 1, "10", 5]  
    print("Buggy sort:")  
    try:  
        print(sort_mixed_list_buggy(mixed))  
    except TypeError as e:  
        print("Error:", e)  
    print("Fixed sort:", sort_mixed_list_fixed(mixed))
```

Name: Md.shanawaz

Enroll: 2403A52100

Batch: 04

TASK 3:

```
"""
Task 3: • Write a Python snippet for file handling that opens a file but forgets to close it.
Ask Copilot or Cursor AI to improve it using the best practice (e.g., with open() block).
"""
"""
Prompt: Write a Python snippet for file handling that opens a file but forgets to close it. Then,
ask Copilot or Cursor AI
to improve the code using best practices, such as using a with open() block to ensure the file is
properly closed.
"""

# Buggy file handling (forgets to close the file)
def read_file_buggy(filename):
    f = open(filename, 'r')
    data = f.read()
    return data

# Fixed file handling using best practice (with open block)
def read_file_fixed(filename):
    with open(filename, 'r') as f:
        data = f.read()
    return data

# Example usage
if __name__ == "__main__":
    # Replace 'test.txt' with a valid filename to test
    try:
        print("Buggy file read:", read_file_buggy('test.txt'))
    except Exception as e:
        print("Error:", e)
    print("Fixed file read:", read_file_fixed('test.txt'))
```

Task 4:

```
"""
Task 4: Provide a piece of code with a ZeroDivisionError inside a loop.
Ask AI to add error handling using try-except and continue execution safely
"""

"""
Prompt: Write a Python code snippet with a loop that causes a ZeroDivisionError (e.g., dividing by
zero).
Then, ask AI to add error handling using try-except so the loop continues executing safely even when
an error occurs.
"""

# Buggy code: ZeroDivisionError inside a loop
def zero_division_buggy():
    for i in range(-2, 3):
        print(10 / i)

# Fixed code: error handling with try-except
def zero_division_fixed():
    for i in range(-2, 3):
        try:
            print(10 / i)
        except ZeroDivisionError:
            print(f"Cannot divide by zero for i={i}")

# Example usage
if __name__ == "__main__":
    print("Buggy ZeroDivisionError loop:")
    try:
        zero_division_buggy()
    except ZeroDivisionError as e:
        print("Error:", e)
    print("Fixed ZeroDivisionError loop:")
    zero_division_fixed()
```

TASK 5:

```
Task 5:
Include a buggy class definition with incorrect __init__
```

Name: Md.shanawaz
Enroll: 2403A52100
Batch: 04

parameters or attribute references. Ask AI to analyze and correct the constructor and attribute usage.

"""

"""

Prompt: Write a Python class definition with a buggy init method, such as incorrect parameters or wrong attribute references. Then, ask AI to analyze and correct the constructor and attribute usage so the class works as intended.

"""

Buggy class definition (incorrect __init__ parameters and attribute references)

class PersonBuggy:

def __init__(self, name, age):

self.nam = name # Typo in attribute name

self.agee = ag # Typo in parameter and attribute name

Fixed class definition

class Person:

def __init__(self, name, age):

self.name = name

self.age = age

Example usage

if __name__ == "__main__":

print("Buggy Person class:")

try:

p = PersonBuggy("Alice", 30)

print(p.name, p.age)

except Exception as e:

print("Error:", e)

print("Fixed Person class:")

p2 = Person("Bob", 25)

print(p2.name, p2.age)