D.PAVAN
2403A52L07
BATCH-50

Lab Ass-7.5

Task 1: (Mutable Default Argument – Function Bug)

Bug: Mutable default argument

```
def add_item(item, items=[]):
   items.append(item)
   return items
print(add_item(1))
print(add_item(2))
```

Fixed code:

```
#Task 1:(Mutable Default Argument – Function Bug)
def add_items(items, item=[]):
    item.append(items)
    return item
print(add_items(1))
print(add_items(2))
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
[1]
[1, 2]
```

Task 2: **2 (Floating-Point Precision Error)**

Bug: Floating point precision issue

```
def check_sum():
   return (0.1 + 0.2) == 0.3
print(check_sum())
```

Fixed code:

```
#Task 2:(Floating-Point Precision Error)
def check_sum():
    return 0.1 + 1.0 == 1.1
print(check_sum())
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
False
PS D:\collage\AI-AC> python ass-7.5.py
True
```

Task 3: **(Recursion Error – Missing Base Case)**

Bug: No base case

def countdown(n):

  print(n)

  return countdown(n-1)

countdown(5)

Fixed code:

```
#Task 3 (Recursion Error – Missing Base Case)
def countdown(n):
    if n <= 0:
        print("Done!")
        return
    print(n)
    countdown(n - 1)

countdown(10)
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
10
9
8
7
6
5
4
3
2
1
Done!
```

Task 4: **(Dictionary Key Error)**

Bug: Accessing non-existing key

```
def get_value():
    data = {"a": 1, "b": 2}
    return data["c"]
print(get_value())
```

Fixed code:

```
#Task 4 (Dictionary Key Error)
def get_value():
    data = {"a": 1, "b": 2}
    return data.get("c", None)
print(get_value())
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
None
```

Task 5: **(Infinite Loop – Wrong Condition)**

Bug: Infinite loop

```
def loop_example():
    i = 0
    while i < 5:
        print(i)
```

Fixed code:

```
#Task 5 (Infinite Loop – Wrong Condition
# Bug: Infinite loop
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1

loop_example()
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
0
1
2
3
4
```

Task 6: **(Unpacking Error – Wrong Variables)**

Bug: Wrong unpacking

a, b = (1, 2, 3)

Fixed code:

```
#Task 6 (Unpacking Error – Wrong Variables)
a, b, c = (1, 2, 3)
print(a, b, c)
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
1 2 3
```

Task 7: **(Mixed Indentation – Tabs vs Spaces)**

Bug: Mixed indentation

def func():

  x = 5

    y = 10

  return x+y

Fixed code:

```
#Task 7 (Mixed Indentation – Tabs vs Spaces)
def func():
    x = 5
    y = 10
    return x + y
print(func())
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
15
```

Task 8: **(Import Error – Wrong Module Usage)**

Bug: Wrong import

import maths

print(maths.sqrt(16))

Fixed code:

```python
#Task 8 (Import Error - Wrong Module Usage)
# Bug: Wrong import
import math
print(math.sqrt(16))
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
4.0
```

Task 9: **(Unreachable Code – Return Inside Loop)**

Bug: Early return inside loop

```
def total(numbers):
    for n in numbers:
        return n
print(total([1,2,3]))
```

```python
#Task 9 (Unreachable Code - Return Inside Loop)
def total(numbers):
    sum_value = 0
    for n in numbers:
        sum_value += n
    return sum_value

print(total([1, 2, 3]))
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
6
```

Task 10: **(Name Error – Undefined Variable)**

Bug: Using undefined variable

```
def calculate_area():
    return length * width
print(calculate_area())
```

Fixed code:

```
#Task 10 (Name Error – Undefined Variable)

def calculate_area(length, width):
    return length * width

# Test cases
assert calculate_area(5, 10) == 50
assert calculate_area(3, 7) == 21
assert calculate_area(0, 10) == 0

print("All tests passed successfully!")
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
All tests passed successfully!
```

Task 11: **(Type Error – Mixing Data Types Incorrectly)**

 # Bug: Adding integer and string

def add_values():

return 5 + "10"

print(add_values())

Fixed code:

```
#Task 11 (Type Error – Mixing Data Types Incorrectly)
def add_values():
    return 5 + int("10")

# Test cases
assert add_values() == 15
assert 7 + int("3") == 10
assert int("0") + 5 == 5

print("All tests passed successfully!")
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
All tests passed successfully!
```

Task 12: **(Type Error – String + List Concatenation)**

Bug: Adding string and list

def combine():

  return "Numbers: " + [1, 2, 3]

print(combine())

Fixed code:

```
#Task 12 (Type Error - String + List Concatenation)
def combine():
    return "Numbers: " + str([1, 2, 3])

print(combine())

assert combine() == "Numbers: [1, 2, 3]"
assert "Numbers: " + str([4]) == "Numbers: [4]"
assert "Numbers: " + str([]) == "Numbers: []"

print("All tests passed successfully!")
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
Numbers: [1, 2, 3]
All tests passed successfully!
```

Task 13: (Type Error – Multiplying String by Float)

 # Bug: Multiplying string by float

def repeat_text():

  return "Hello" * 2.5

print(repeat_text())

Fixed code:

```
#Task 13 (Type Error - Multiplying String by Float)
def repeat_text():
    return "Hello" * int(2.5)

print(repeat_text())
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
HelloHello
```

Task 14: (Type Error – Adding None to Integer)

 # Bug: Adding None and integer

def compute():

value = None

return value + 10

print(compute())

Fixed code:

```
#Task 14 (Type Error – Adding None to Integer)
def compute():
    value = None
    if value is None:
        value = 0
    return value + 10
print(compute())
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
10
```

Task 15: **(Type Error – Input Treated as String Instead of Number)**

Bug: Input remains string

def sum_two_numbers():

a = input("Enter first number: ")

b = input("Enter second number: ")

return a + b

print(sum_two_numbers())

Fixed code:

```python
#Task 15 (Type Error - Input Treated as String Instead of Number)
def sum_two_numbers(a, b):
    return int(a) + int(b)

# Test cases
assert sum_two_numbers(5, 10) == 15
assert sum_two_numbers("3", "7") == 10
assert sum_two_numbers(0, 0) == 0

print("All tests passed successfully!")
print(sum_two_numbers("5", 52))
```

Output:

```
PS D:\collage\AI-AC> python ass-7.5.py
All tests passed successfully!
57
```