#Assignment-8.2

#halltickettnumber:2303A51545

#batch-08

#Task 1 – Test-Driven Development for Even/Odd Number Validator

```
def is_even(n):
    if not isinstance(n, int):
        raise TypeError("Input must be an integer")
    return n % 2 == 0


# Tests
assert is_even(2) is True
assert is_even(7) is False
assert is_even(0) is True
assert is_even(-4) is True
assert is_even(9) is False

print("Task 1 Passed ")
print("is_even(2) =", is_even(2))
```

```
Task 1 Passed
is_even(2) = True
```

#Task 2 – Test-Driven Development for String Case Converter

```
def to_uppercase(text):
    if not isinstance(text, str):
        raise TypeError("Input must be a string")
    return text.upper()


def to_lowercase(text):
    if not isinstance(text, str):
        raise TypeError("Input must be a string")
    return text.lower()


# Tests
assert to_uppercase("ai coding") == "AI CODING"
assert to_lowercase("TEST") == "test"
assert to_uppercase("") == ""
assert to_lowercase("PyThOn") == "python"

print("Task 2 Passed ")
print("to_uppercase('ai coding') =", to_uppercase("ai coding"))
```

```
Task 2 Passed
to_uppercase('ai coding') = AI CODING
```

#Task 3 – Test-Driven Development for List Sum Calculator

```python
def sum_list(numbers):
    if not isinstance(numbers, list):
        raise TypeError("Input must be a list")

    total = 0
    for item in numbers:
        if isinstance(item, bool):
            continue
        if isinstance(item, (int, float)):
            total += item
    return total


# Tests
assert sum_list([1, 2, 3]) == 6
assert sum_list([]) == 0
assert sum_list([-1, 5, -4]) == 0
assert sum_list([2, "a", 3]) == 5

print("Task 3 Passed ")
print("sum_list([1,2,3]) =", sum_list([1, 2, 3]))
```

```
Task 3 Passed
sum_list([1,2,3]) = 6
```

#Task 4 – Test Cases for Student Result Class

```python
class StudentResult:
    def __init__(self, marks):
        if not isinstance(marks, list) or len(marks) == 0:
            raise ValueError("Invalid marks list")

        for m in marks:
            if not isinstance(m, (int, float)) or m < 0 or m > 100:
                raise ValueError("Marks must be 0–100")

        self.marks = marks

    def average(self):
        return sum(self.marks) / len(self.marks)

    def result(self):
        return "Pass" if self.average() >= 40 else "Fail"
```

```
# Tests
s1 = StudentResult([60, 70, 80])
s2 = StudentResult([30, 35, 40])

assert s1.average() == 70
assert s1.result() == "Pass"
assert s2.average() == 35
assert s2.result() == "Fail"

print("Task 4 Passed ")
```

```
Task 4 Passed
```

#Task 5 – Test-Driven Development for Username Validator

```
def is_valid_username(username):
    if not isinstance(username, str):
        raise TypeError("Username must be a string")

    if len(username) < 5:
        return False
    if " " in username:
        return False
    if not username.isalnum():
        return False

    return True


# Tests
assert is_valid_username("user01") is True
assert is_valid_username("ai") is False
assert is_valid_username("user name") is False
assert is_valid_username("user@123") is False

print("Task 5 Passed ")
print("is_valid_username('user01') =", is_valid_username("user01"))
```

```
Task 5 Passed
is_valid_username('user01') = True
```