AI Assisted Coding

ASSIGNMENT: 8.1

Name: E.Sreeja

Batch:28

Ht.no:2303A1905

Test-Driven Development with AI

Task 1: Password Strength Validator

```
import re

def is_strong_password(password):

    if not isinstance(password, str):

        return False
```

```
if len(password) < 8:

    return False

if " " in password:

    return False

  if not re.search(r"[A-Z]", password):

 return False

if not re.search(r"[a-z]", password):

    return False

if not re.search(r"[0-9]", password):

    return False

if not re.search(r"[!@#$%^&*(),.?\":{}|<>]", password):

    return False
    return True

# AI Generated Test Cases

assert is_strong_password("Abcd@123") == True

assert is_strong_password("abcd123") == False

assert is_strong_password("ABCD@1234") == False

assert is_strong_password("Strong#Pass9") == True
```

Task 2: Number Classification

```
def classify_number(n):

    if not isinstance(n, (int, float)):

        return "Invalid Input"
    for _ in range(1):
        if n > 0:
            return "Positive"
  elif n < 0:
            return "Negative"
        else:
            return "Zero"



# AI Generated Test Cases

assert classify_number(10) == "Positive"

assert classify_number(-5) == "Negative"

assert classify_number(0) == "Zero"
assert classify_number("abc") == "Invalid Input"

assert classify_number(None) == "Invalid Input"
```

Task 3: Anagram Checker

```
import string

def is_anagram(str1, str2):

    if not isinstance(str1, str) or not isinstance(str2, str):

        return False

    translator = str.maketrans('', '', string.punctuation)

    s1 = str1.translate(translator).replace(" ", "").lower()

    s2 = str2.translate(translator).replace(" ", "").lower()

      return sorted(s1) == sorted(s2)
```

```
# AI Generated Test Cases

assert is_anagram("listen", "silent") == True

assert is_anagram("hello", "world") == False

assert is_anagram("Dormitory", "Dirty Room") == True

assert is_anagram("", "") == True

assert is_anagram("A gentleman", "Elegant man!") == True
```

Task 4: Inventory Class

```
class Inventory:

    def init(self):

        self.stock = {}

    def add_item(self, name, quantity):
```

```
    if name in self.stock:
        self.stock[name] += quantity
    else:
        self.stock[name] = quantity
def remove_item(self, name, quantity):
    if name in self.stock and self.stock[name] >= quantity:
        self.stock[name] -= quantity
    else:
        return "Insufficient stock or item not found"
def get_stock(self, name):
    return self.stock.get(name, 0)
```

```
# AI Generated Test Cases

inv = Inventory()

inv.add_item("Pen", 10)

assert inv.get_stock("Pen") == 10

inv.remove_item("Pen", 5)

assert inv.get_stock("Pen") == 5

inv.add_item("Book", 3)

assert inv.get_stock("Book") == 3
```

Task 5: Date Validation & Formatting

```
from datetime import datetime
```

```
def validate_and_format_date(date_str):

    try:

        date_obj = datetime.strptime(date_str, "%m/%d/%Y")

        return date_obj.strftime("%Y-%m-%d")

    except:

        return "Invalid Date"

# AI Generated Test Cases

assert validate_and_format_date("10/15/2023") == "2023-10-15"

assert validate_and_format_date("02/30/2023") == "Invalid Date"

assert validate_and_format_date("01/01/2024") == "2024-01-01"

assert validate_and_format_date("13/01/2023") == "Invalid Date"
```