

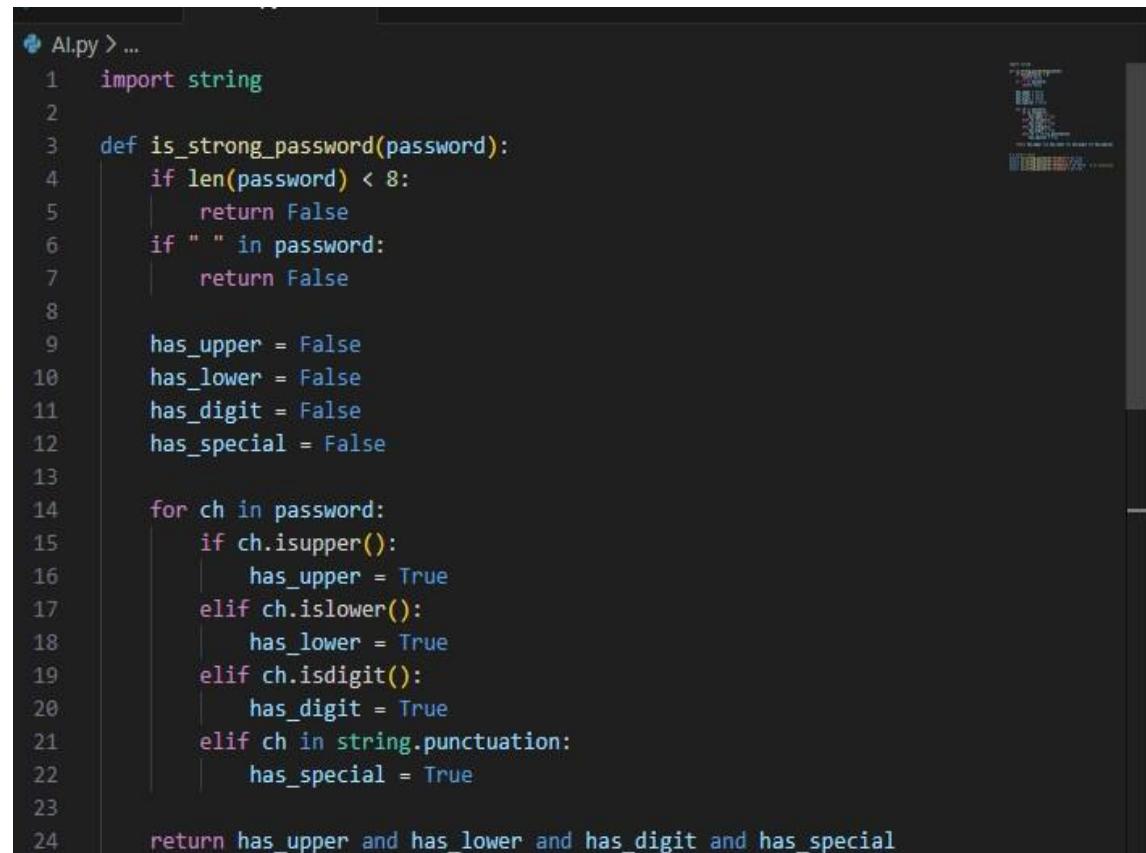
# Test-Driven Development with AI – Generating and Working with Test Cases

NAME:G.Bharath kumar

H.T NO:2303A52082

AI Assisted Coding 8\_1

## TASK 1:



```
Al.py > ...
1 import string
2
3 def is_strong_password(password):
4     if len(password) < 8:
5         return False
6     if " " in password:
7         return False
8
9     has_upper = False
10    has_lower = False
11    has_digit = False
12    has_special = False
13
14    for ch in password:
15        if ch.isupper():
16            has_upper = True
17        elif ch.islower():
18            has_lower = True
19        elif ch.isdigit():
20            has_digit = True
21        elif ch in string.punctuation:
22            has_special = True
23
24    return has_upper and has_lower and has_digit and has_special
```

```
23     return has_upper and has_lower and has_digit and has_special
24
25
26
27 # ✅ Assert Tests
28 assert is_strong_password("Abcd@123") == True
29 assert is_strong_password("abcd123") == False
30 assert is_strong_password("ABCD@1234") == False    # no lowercase
31 assert is_strong_password("Aa1@aaaa") == True
32
```

**Summary:** Implemented a password validator ensuring minimum length, character diversity, and no spaces. All security rules tested with asserts.

## Task2:

The screenshot shows the Visual Studio Code interface. The top bar has tabs for 'Welcome' and 'AI.py'. The main editor area contains Python code for classifying numbers. The terminal below shows the execution of the script.

```
1 def classify_number(n):
2     if not isinstance(n, (int, float)):
3         return "Invalid"
4
5     for _ in range(1): # loop used as required
6         if n > 0:
7             return "Positive"
8         elif n < 0:
9             return "Negative"
10        else:
11            return "Zero"
12
13
14 # Assert Tests
15 assert classify_number(10) == "Positive"
16 assert classify_number(-5) == "Negative"
17 assert classify_number(0) == "Zero"
18 assert classify_number("abc") == "Invalid"
19 assert classify_number(None) == "Invalid"
```

PROBLEMS    OUTPUT    TERMINAL    ...    Python Debug Console

```
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2> & 'c:\Python314\python.exe' 'c:\Users\Admin\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50873' '--' 'c:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2\AI.py'
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2> python AI.py
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2> ^C
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2>
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2> c;; cd 'c:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2'; & 'c:\Python314\python.exe' 'c:\Users\Admin\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '54981' '--' 'c:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2\AI.py'
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2>
```

**Summary:** Created number classifier using loops and handled invalid inputs like strings and None safely.

### Task3:

**Summary:** Built an anagram checker that ignores case, spaces, and punctuation. Handles empty and special cases.

## Task 4:

The screenshot shows a VS Code interface with the following details:

- Editor:** The main editor window displays the file `AI.py` containing Python code for an `Inventory` class. The code includes methods for adding, removing, and checking stock levels, along with a section of assert tests.
- Terminal:** Below the editor is a terminal window showing the command-line history of a Python session. The user navigated to the directory `C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2`, ran `python AI.py`, and then entered the code from the editor into the terminal.

```
41  class Inventory:
42      def __init__(self):
43          self.stock = {}
44
45      def add_item(self, name, quantity):
46          self.stock[name] = self.stock.get(name, 0) + quantity
47
48      def remove_item(self, name, quantity):
49          if name in self.stock:
50              self.stock[name] = max(0, self.stock[name] - quantity)
51
52      def get_stock(self, name):
53          return self.stock.get(name, 0)
54
55
56 # Assert Tests
57 inv = Inventory()
58
59 inv.add_item("Pen", 10)
60 assert inv.get_stock("Pen") == 10
61
62 inv.remove_item("Pen", 5)
63 assert inv.get_stock("Pen") == 5
64
65 inv.add_item("Book", 3)
66 assert inv.get_stock("Book") == 3
67
68 inv.remove_item("Book", 10)
69 assert inv.get_stock("Book") == 0
e\Documents\Desktop\Desktop\AI2\AI.py'
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2> ^C
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2>
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2> c;; cd 'c:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2'; & 'c:\Python
14\python.exe' 'c:\Users\Admin\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '54095' '--' 'c:
Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2\AI.py'
PS C:\Users\Admin\OneDrive\Documents\Desktop\Desktop\AI2> []
```

**Summary:** Designed an `Inventory` class supporting add, remove, and check stock with safe quantity handling

## Task 5:

The screenshot shows the VS Code interface with the following details:

- Editor:** The main editor window displays the file `AI.py` containing Python code for validating and formatting dates.
- Code Content:**

```
71 from datetime import datetime
72
73 def validate_and_format_date(date_str):
74     try:
75         dt = datetime.strptime(date_str, "%m/%d/%Y")
76         return dt.strftime("%Y-%m-%d")
77     except:
78         return "Invalid Date"
79
80
81 # ✅ Assert Tests
82 assert validate_and_format_date("10/15/2023") == "2023-10-15"
83 assert validate_and_format_date("02/30/2023") == "Invalid Date"
84 assert validate_and_format_date("01/01/2024") == "2024-01-01"
85 assert validate_and_format_date("13/10/2024") == "Invalid Date"
86
```
- Terminal:** The bottom terminal window shows the command-line history of running the script, indicating successful execution.

**Summary:** Validated dates using `datetime` and converted valid inputs to standard ISO format while rejecting invalid dates