**Prerequisites:**

1. **Python Installed:** Make sure you have Python installed on your system.

* https://www.python.org/

1. **Visual Studio Code (VSCode):** Download and install VSCode.

* https://code.visualstudio.com/

**How to implement TDD**

**1. Write a Test:**

* Define What You Want to Build: Before writing any code, you start by thinking about what you want your code to do. This could be as simple as, "I want a button to change color when clicked."
* Write a Test for That Functionality: You then write a test (in code) that checks if the button changes color when clicked. However, at this point, your test will fail because you haven't written the code to change the button color yet.

**2. Write the Code:**

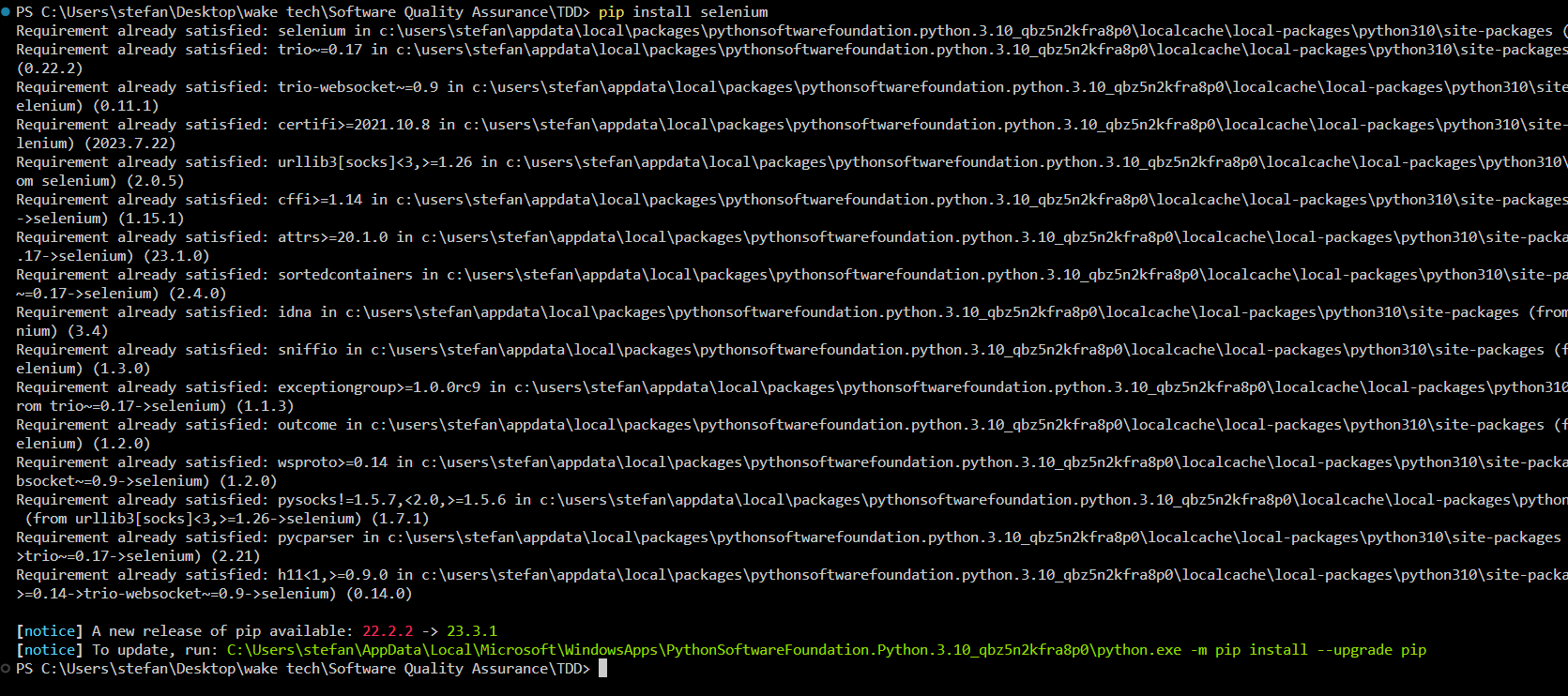
* Make the Test Pass: Once you've written the failing test, you start writing the code that changes the button's color when it's clicked. The goal here is to write just enough code to make your failing test pass.

**3. Refactor:**

* Improve Your Code Without Changing the Functionality: After your test passes, you can improve your code. This might mean making it cleaner, more efficient, or more readable, but you should avoid changing the behavior that was specified by the test.

**Open VS code and make a directory you want to work on  
**

Make sure you open up a terminal and enter the command -> pip install selenium



Now we have are environment setup we will be making creating TDD for our lab.

**To follow up do the following command in the terneal ->** git clone https://github.com/CSC-256-Group-Project-9/Group-9-TDD-Lab.git

* **We will be testing the ui of are app** Tests the functionality of adding contacts to the contact book and validates that the contents are displayed correctly.

1. **Create a file for the test call it test\_contact\_book\_ui.py**
2. **Find what you need to test in the file in Example we are going to use the flowing testcases.**
   1. **test\_CLI\_initialization:**
   * **Ensures the proper initialization of a CLI instance by checking if it is an instance of the CLI class.**
   1. **test\_display\_menu:**
   * **Verifies the correct representation of the display menu against an expected menu string.**
   1. **test\_invalid\_menu\_option:**
   * **Tests the behavior when an invalid option is input for the display menu. It checks for an appropriate error message display.**
   1. **test\_successful\_add\_person\_prompt:**
   * **Checks if the appropriate message is displayed when adding a valid person to the ContactBook.**
   * **Verifies that the contact list is updated after adding a person and contains the correct information.**
   1. **test\_invalid\_add\_person\_prompt:**
   * **Ensures that an error message is displayed when an invalid person name is input for addition to the ContactBook.**
   * **Verifies that the contact list remains empty after attempting to add an invalid person.**
   1. **test\_add\_existing\_person\_prompt:**
   * **Tests the scenario when attempting to add a person that already exists in the ContactBook.**
   * **Verifies that the appropriate error message is displayed, and duplicate addition to the contact list is prevented.**
   1. **test\_successful\_remove\_contact:**
   * **Ensures the correct message is displayed when removing a valid person from the ContactBook.**
   * **Verifies that the contact list is updated after successful removal.**
   1. **test\_remove\_nonexistent\_contact:**
   * **Checks the behavior when trying to remove a contact that does not exist in the ContactBook.**
   * **Verifies that the appropriate error message is displayed in such cases.**
   1. **test\_empty\_contact\_book\_display:**
   * **Verifies that the correct message is displayed when the contact book is empty.**
   1. **test\_list\_contacts:**
   * **Tests the functionality of adding contacts to the contact book and validates that the contents are displayed correctly.**
3. **Now we got to make some requirements for each test**

**1. test\_CLI\_initialization:**

* **Conditions: Empty constructor of CLI to ensure the instance is initialized without errors.**
* **Expectation: The test should pass if the CLI instance is successfully initialized and is an instance of the CLI class.**

**2. test\_display\_menu:**

* **Conditions: An instance of CLI with a valid implementation of the display\_menu method.**
* **Expectation: The displayed menu should match the expected string representation defined in the test.**

**3. test\_invalid\_menu\_option:**

* **Conditions: An instance of CLI that handles input for the display menu with invalid options.**
* **Expectation: The test should check if an appropriate error message is displayed when an invalid menu option is entered.**

**4. test\_successful\_add\_person\_prompt:**

* **Conditions: An instance of CLI capable of adding a person to the ContactBook.**
* **Expectation: Upon adding a valid person: The contact list should be updated. The added contact's information should be correct in the contact list.**

**5. test\_invalid\_add\_person\_prompt:**

* **Conditions: CLI instance that correctly identifies and handles an invalid person name input.**
* **Expectation: An error message should be displayed when attempting to add an invalid person to the ContactBook, and the contact list should remain unchanged.**

**6. test\_add\_existing\_person\_prompt:**

* **Conditions: CLI capable of handling duplicate contact entries.**
* **Expectation: The test should verify that attempting to add a person already present in the ContactBook results in an appropriate error message, and the contact list remains unchanged.**

**7. test\_successful\_remove\_contact:**

* **Conditions: CLI capable of removing a person from the ContactBook.**
* **Expectation: Upon successful removal: The contact list should be updated, and the removed person should not exist in the list.**

**8. test\_remove\_nonexistent\_contact:**

* **Conditions: CLI managing the scenario where a non-existent contact is being removed.**
* **Expectation: An appropriate error message should be displayed when attempting to remove a contact that does not exist in the ContactBook.**

**9. test\_empty\_contact\_book\_display:**

* **Conditions: A condition where the contact book is empty.**
* **Expectation: The test should check if the correct message is displayed when the contact book is empty.**

**10. test\_list\_contacts:**

* **Conditions: CLI capable of adding contacts to the ContactBook.**
* **Expectation: The test should ensure that the contact details are displayed correctly when contacts are added to the contact book.**

1. **Now when we run this tests it should fail right away and make these test Passes. Make sure you rerun the test file to insure it works all together.**
2. **Now refactor the code by Improve the structure or eliminate redundancies without altering the functionality.**