Certainly! I'll provide a step-by-step guide for completing this automation assignment using Python with Selenium WebDriver. This guide will cover all the required steps and best practices.

1. Setup and Environment Preparation:

First, install the necessary tools:

Copy

```
pip install selenium
pip install webdriver-manager
```

2. Script Structure:

Here's a basic structure for the automation script:

python

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.action_chains import ActionChains
from webdriver_manager.chrome import ChromeDriverManager


class FitPeoAutomation:
    def __init__(self):
        service = Service(ChromeDriverManager().install())
        self.driver = webdriver.Chrome(service=service)
        self.wait = WebDriverWait(self.driver, 10)


    def run_automation(self):
        try:
            self.navigate_to_homepage()
            self.navigate_to_revenue_calculator()
            self.adjust_slider(820)
            self.update_text_field(560)
            self.validate_slider_value(560)
            self.select_cpt_codes()

            self.validate_total_reimbursement("$110700")

        except Exception as e:

            print(f"An error occurred: {str(e)}")

        finally:

            self.driver.quit()


    # Implement other methods here


if __name__ == "__main__":

    automation = FitPeoAutomation()

    automation.run_automation()
```

3. Implementing the Required Steps:

```
python

def navigate_to_homepage(self):

    self.driver.get("https://fitpeo.com")  # Replace with actual URL


def navigate_to_revenue_calculator(self):

    calculator_link = self.wait.until(EC.element_to_be_clickable((By.LINK_TEXT, "Revenue Calculator")))

    calculator_link.click()


def adjust_slider(self, value):

    slider = self.wait.until(EC.presence_of_element_located((By.ID, "slider-id")))  # Replace with actual 
ID

    action = ActionChains(self.driver)

    action.click_and_hold(slider).move_by_offset(value, 0).release().perform()


    text_field = self.driver.find_element(By.ID, "text-field-id")  # Replace with actual ID

    assert text_field.get_attribute("value") == str(value), f"Text field value does not match: 
{text_field.get_attribute('value')}"


def update_text_field(self, value):
    text_field = self.wait.until(EC.element_to_be_clickable((By.ID, "text-field-id")))  # Replace with actual ID
    text_field.clear()
    text_field.send_keys(str(value))


def validate_slider_value(self, expected_value):
    slider = self.wait.until(EC.presence_of_element_located((By.ID, "slider-id")))  # Replace with actual ID
    actual_value = slider.get_attribute("value")
    assert int(actual_value) == expected_value, f"Slider value does not match: expected {expected_value}, got {actual_value}"


def select_cpt_codes(self):
    cpt_codes = ["CPT-99091", "CPT-99453", "CPT-99454", "CPT-99474"]
    for code in cpt_codes:
        checkbox = self.wait.until(EC.element_to_be_clickable((By.XPATH, f"//input[@type='checkbox' and @value='{code}']")))
        checkbox.click()


def validate_total_reimbursement(self, expected_value):
    total_reimbursement = self.wait.until(EC.presence_of_element_located((By.XPATH, "//h2[contains(text(), 'Total Recurring Reimbursement for all Patients Per Month:')]")))
    actual_value = total_reimbursement.text.split(":")[1].strip()
    assert actual_value == expected_value, f"Total reimbursement does not match: expected {expected_value}, got {actual_value}"
```

4. Exception Handling:

Add this method to handle exceptions:

python

```
def handle_exception(self, e):
    if isinstance(e, NoSuchElementException):
        print("An element was not found on the page.")
    elif isinstance(e, TimeoutException):
        print("The page took too long to load an element.")
    elif isinstance(e, AssertionError):
        print(f"A validation failed: {str(e)}")
    else:
        print(f"An unexpected error occurred: {str(e)}")
```

5. Setup and Run Documentation:

Create a README.md file with the following content:

markdown

```
# FitPeo Automation Test

This script automates the testing of FitPeo's Revenue Calculator.

## Prerequisites

- Python 3.7+
- Chrome browser

## Setup

1. Clone the repository:
   git clone https://github.com/yourusername/fitpeo-automation.git

2. Navigate to the project directory:
   cd fitpeo-automation

3. Install required packages:
   pip install -r requirements.txt

## Running the Tests

1. Open a terminal in the project directory
2. Run the following command:
```

```
python fitpeo_automation.py
```

## Troubleshooting

- If you encounter element locator issues, check the HTML structure of the FitPeo website and update the locators in the script accordingly.

- Ensure your internet connection is stable for reliable test execution.

6. Best Practices and Maintainability:

- Use meaningful variable and function names
- Add comments to explain complex logic
- Use configuration files for easily changeable values (e.g., URLs, expected values)
- Implement logging for better debugging

7. Handling Dynamic Web Elements:

```
python
def wait_for_element(self, locator):
    return self.wait.until(EC.presence_of_element_located(locator))


def wait_for_clickable(self, locator):
    return self.wait.until(EC.element_to_be_clickable(locator))
```

Remember to replace placeholder IDs and XPaths with actual values from the FitPeo website. Test the script thoroughly and handle any site-specific behaviors or edge cases. Good luck with your assignment!

NAME=Ashutosh giri

REG No-39110093

Email-ashugiri199@gmail.com

Phone no.=7879103075