

A4Q
Selenium 4
Tester Foundation

Alliance for Qualification

A4Q Selenium Tester Foundation Workshop Exercise and solution (Python)

Released 2025

A4Q Alliance for Qualification

Copyright Notice

Revision History

Version	Date	Remark
Alpha	05.06.2024	Initial Layout
Beta	03.07.2024	Beta version
Release	06.01.2025	Release version

Table of Contents

Acknowledgments

This document was formally released by A4Q on 30.01.2025

Setting up Selenium with Python

There are following steps to setup Selenium using Python:

- Download and Install Python on Windows
- Install Selenium Libraries in Python
- Download and Install PyCharm
- Create a new project using PyCharm

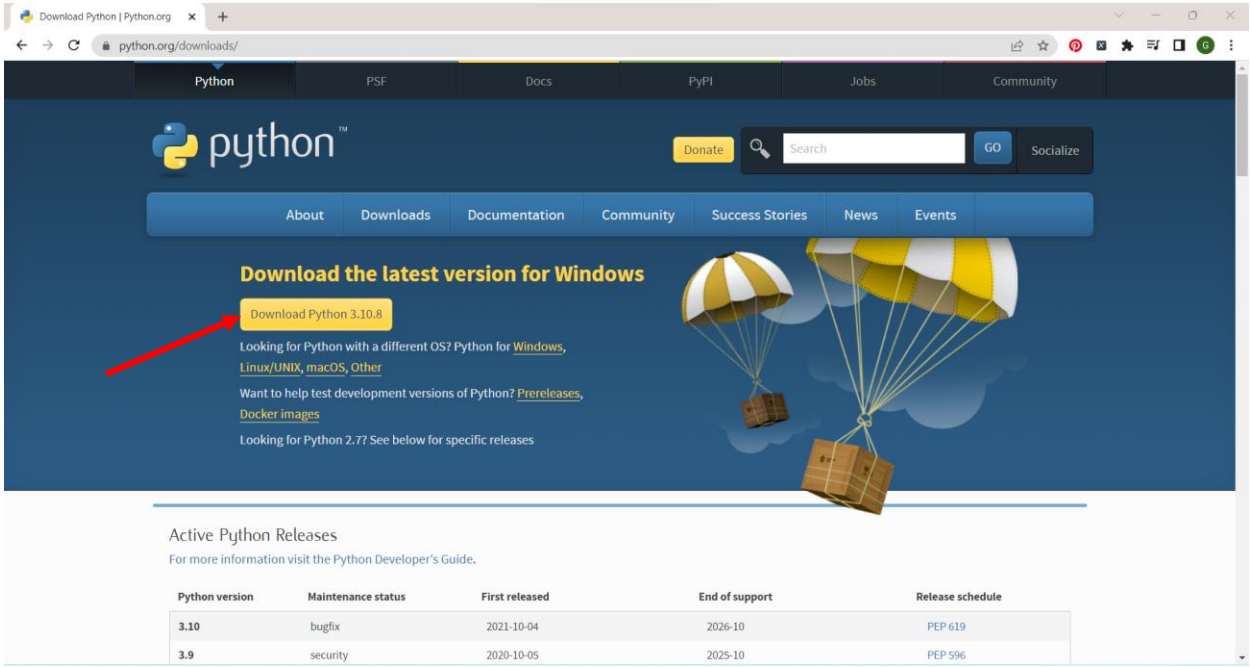
Note that installation steps are provided for Windows platform. Similar steps may apply to other platforms like iOS, Linux & Ubuntu.

Step 1 – Download and install Python on Windows

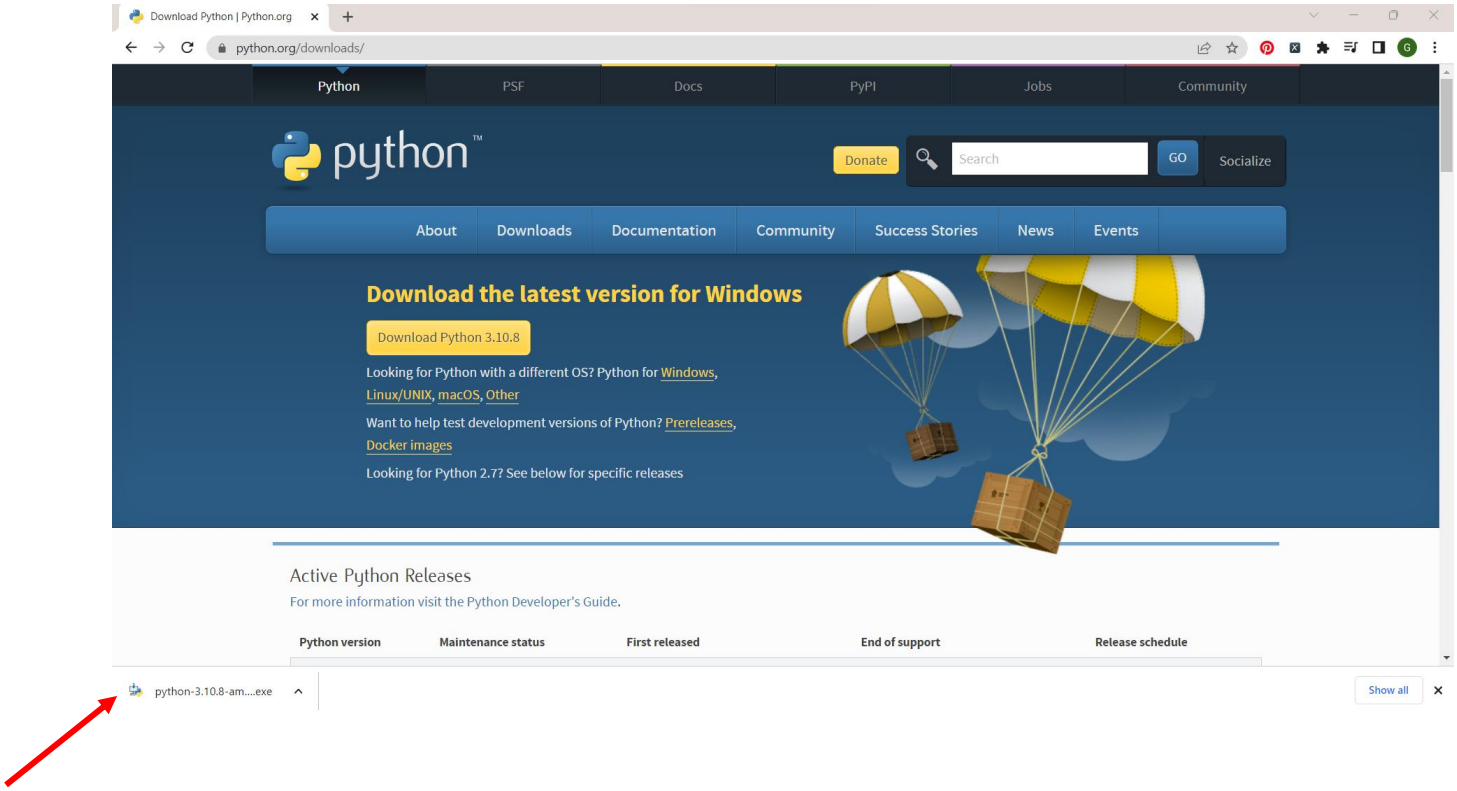
Navigate to the link below to download the most recent version of Python for Windows platforms:

https://www.Python.org/downloads/

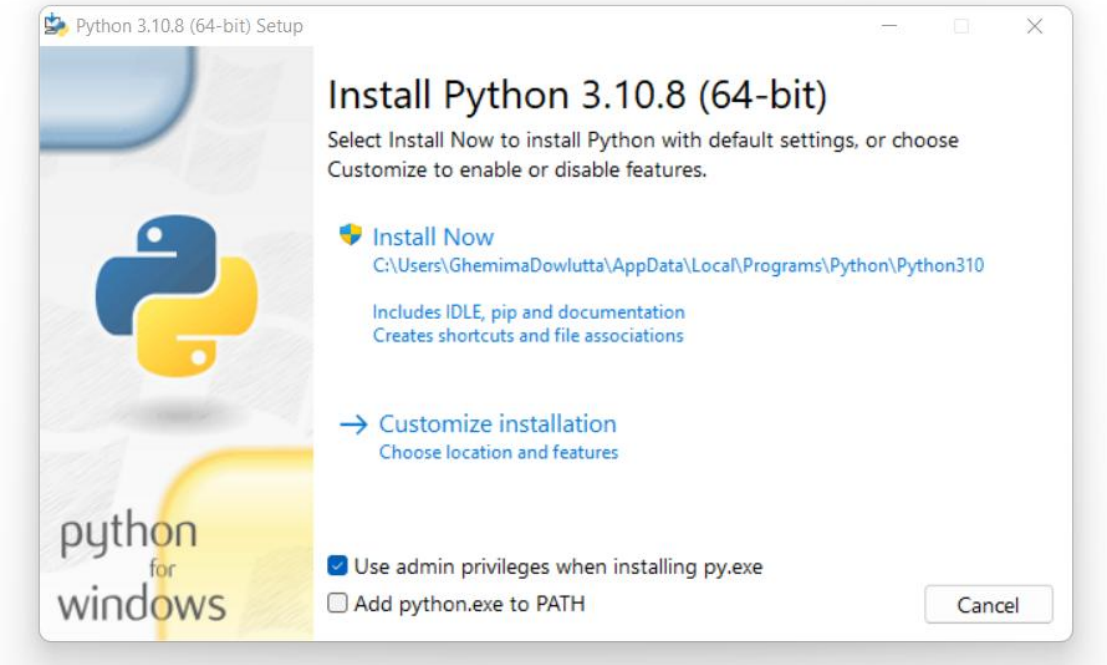
Click on **Download Python 3.10.8** button (Depend on latest version available).



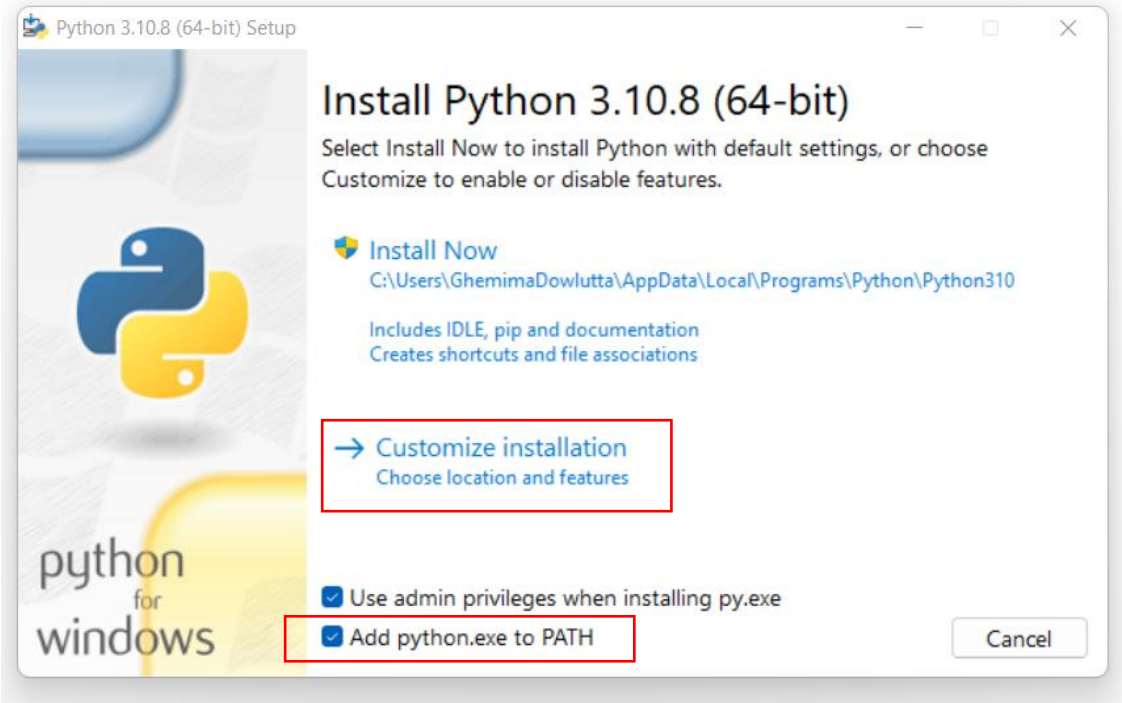
Once downloaded, double-click on the downloaded executable file.



The **Python 3.10.8(64-bit)** setup window will appear on screen.

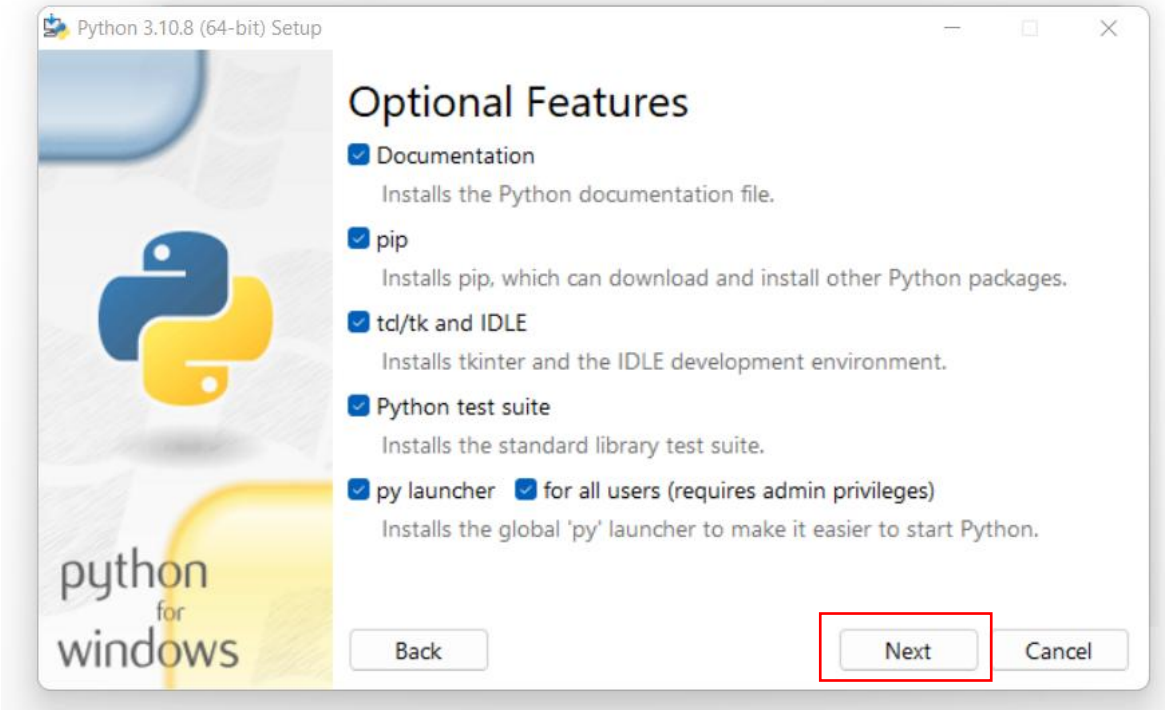


Click on the **Add python.exe to PATH** checkbox.



After, click on **Customize installation**, the **Optional Features** will appear on the screen, where we can select and deselect the features according to our requirements.

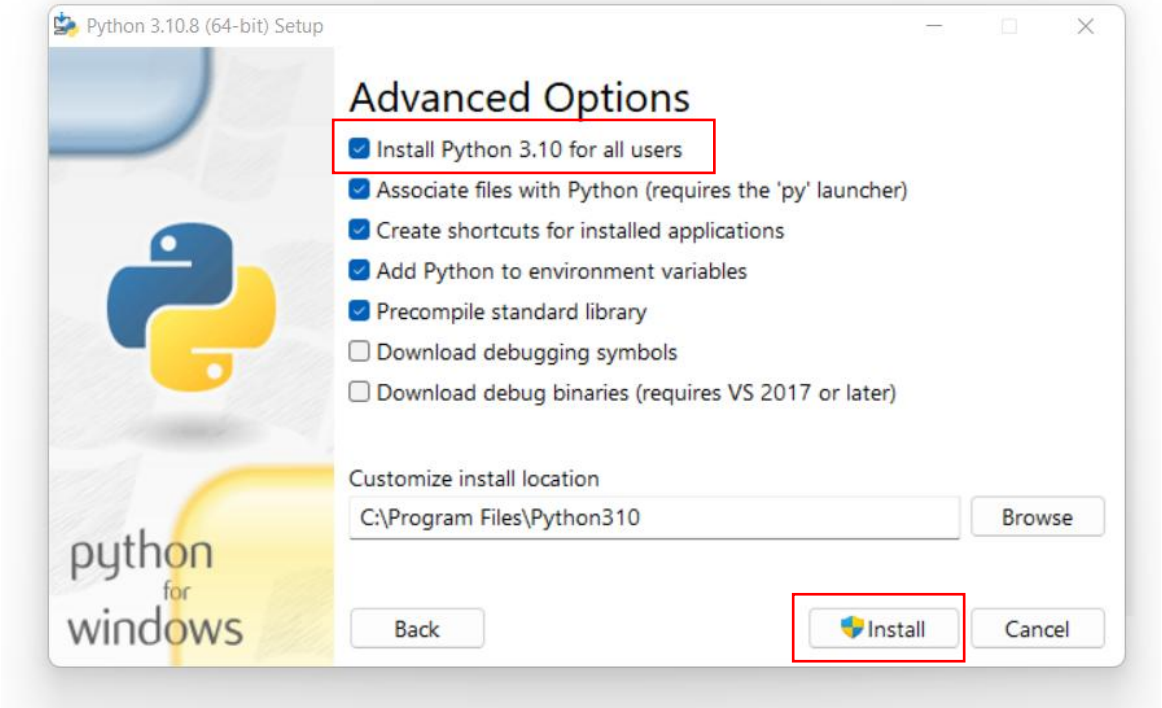
Then, click on the **Next** button.



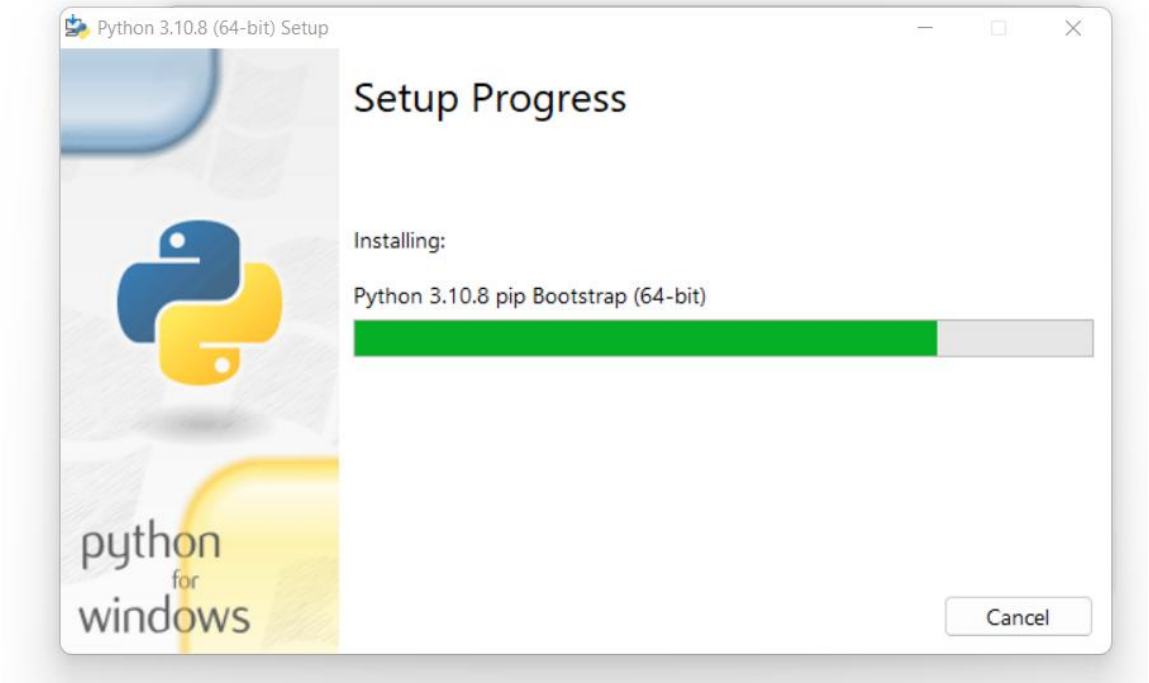
Advanced Options will appear, where we can select the options based on our needs.

Make sure that the **Install Python 3.10 for all users** checkbox is checked. We can also customize the install location according to our convenience by clicking on the **Browse** button.

After that, click on the **Install** button, to install Python.

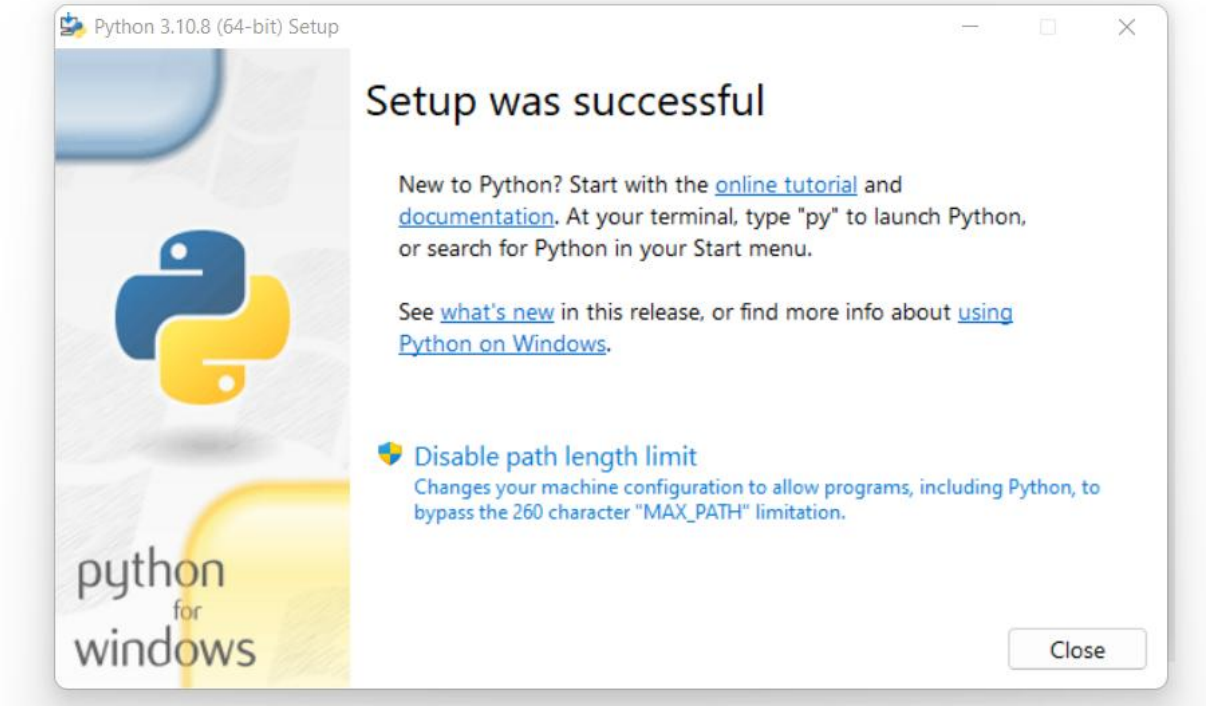


The **Setup Progress** window will appear.

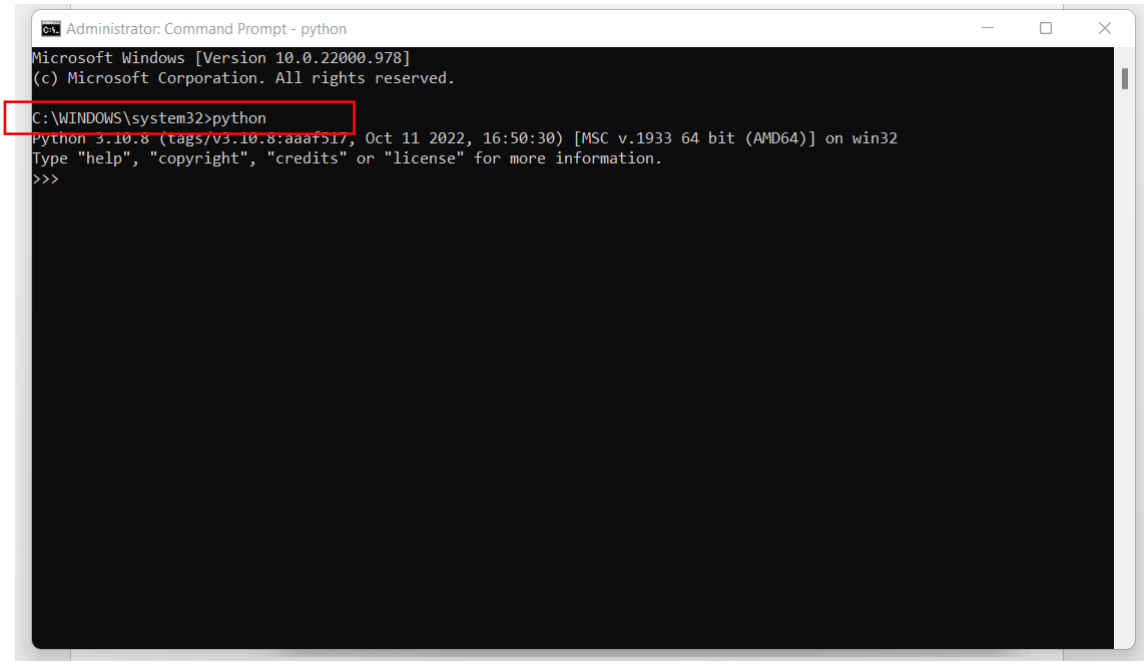


Once the installation is done, **Setup was successful** message will appear, which means that the Python is installed successfully for the Windows operating system.

Then, click on the **Close** button to close the setup window.



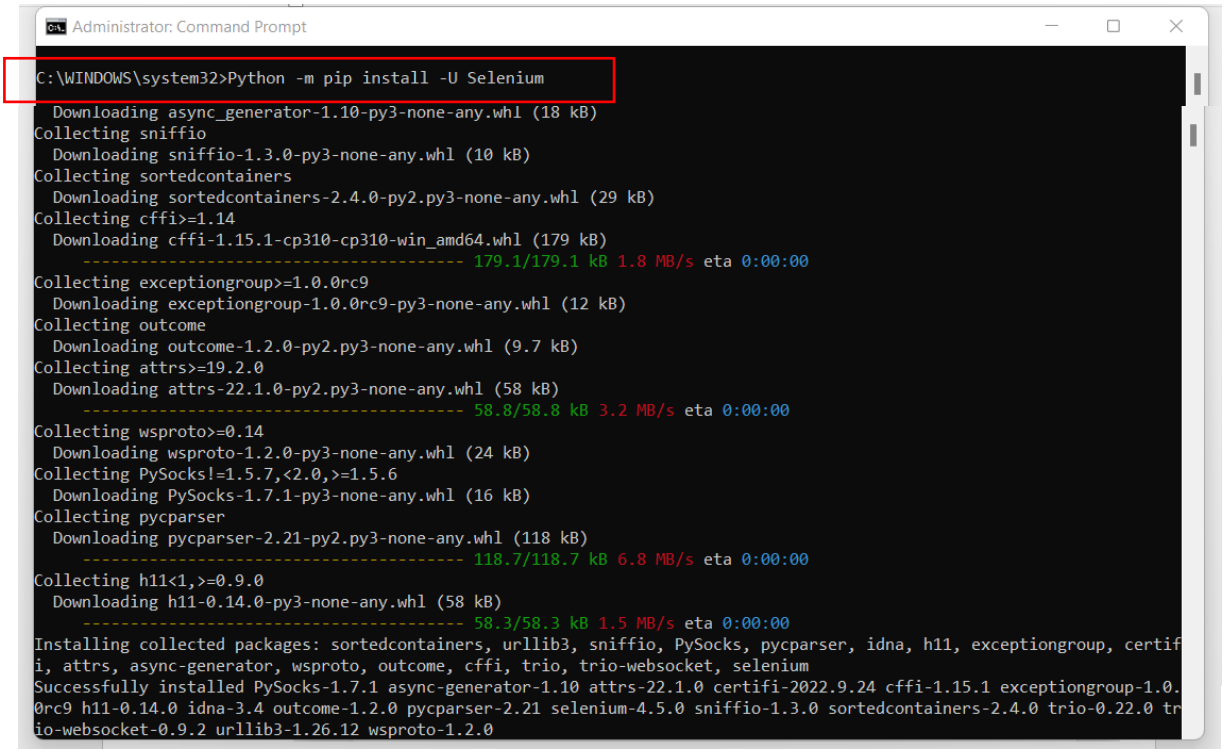
To check if Python was successfully installed:
Open the **Command Prompt**, type the command **python** and press the **Enter** key. It will open the **Python interpreter shell** where we can implement the Python program.



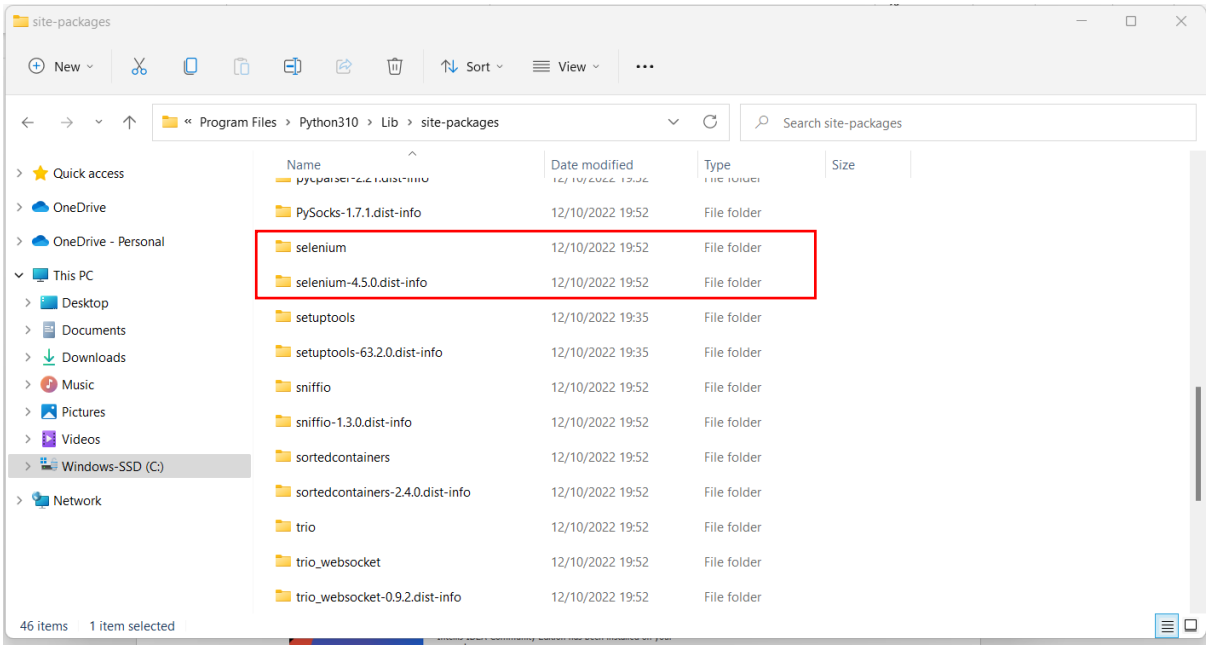
Step 2 - Install Selenium Libraries in Python

Open the **Command Prompt**, and execute the following command:

Python -m pip install -U Selenium



After successful installation of the command, the **selenium libraries** will automatically be created.

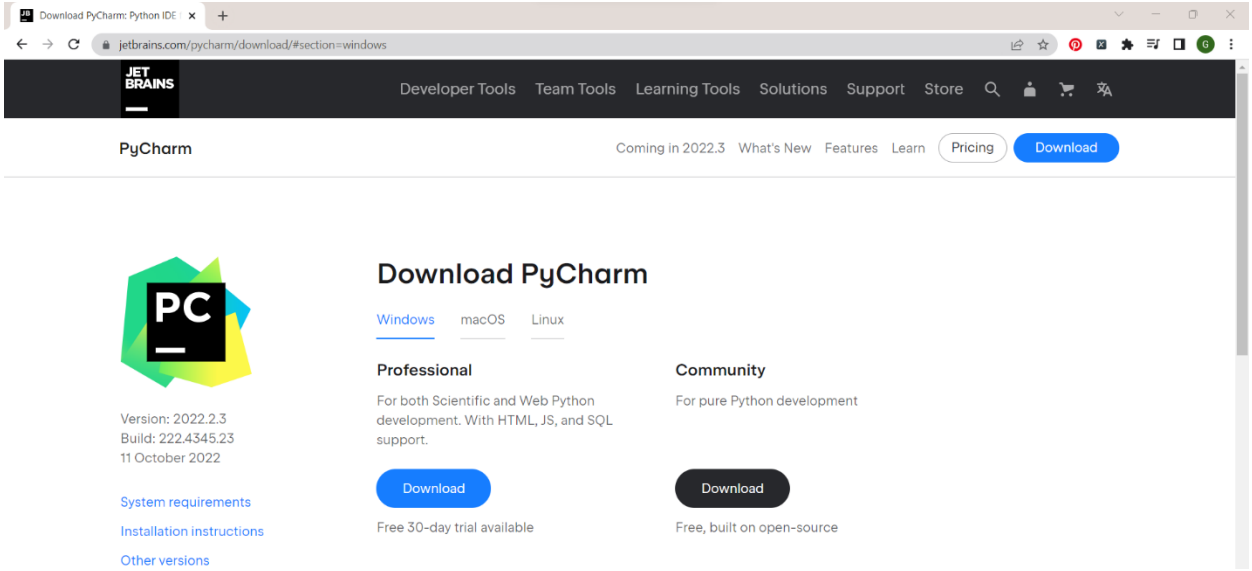


Step 3 - Download and Install PyCharm

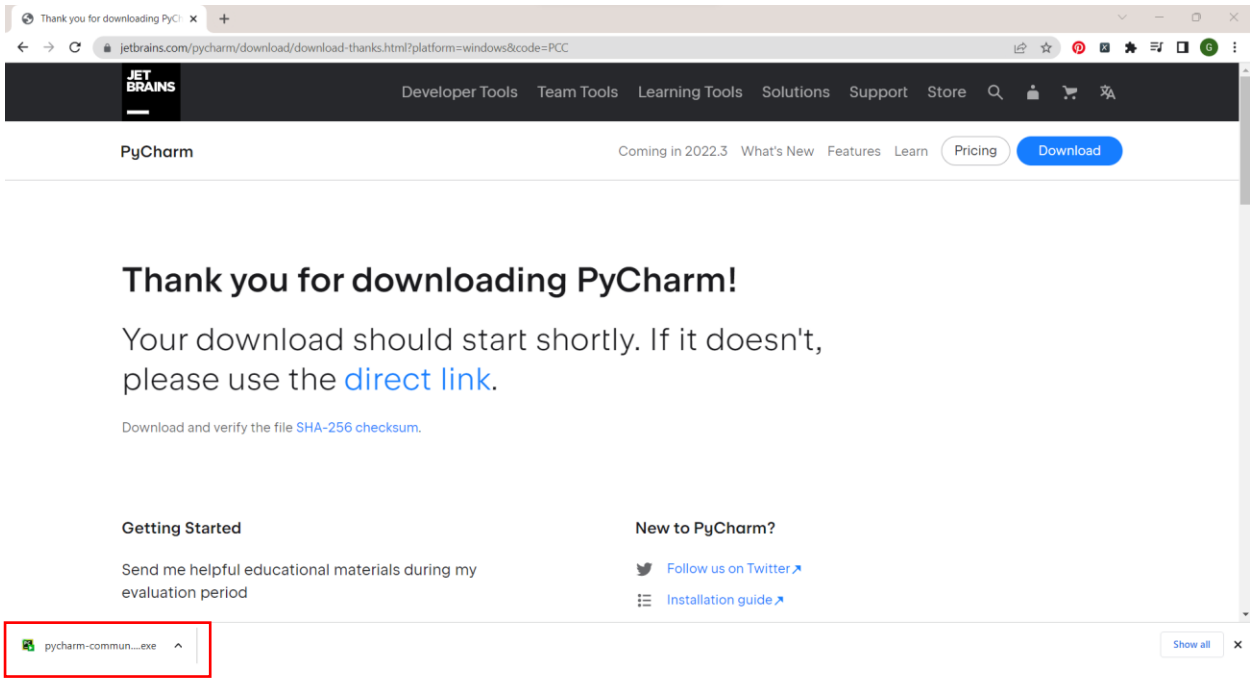
To download PyCharm, navigate to the link below:

https://www.jetbrains.com/pycharm/download/#section=windows

Click on the **Download** button under the **Community** section for the **Windows**

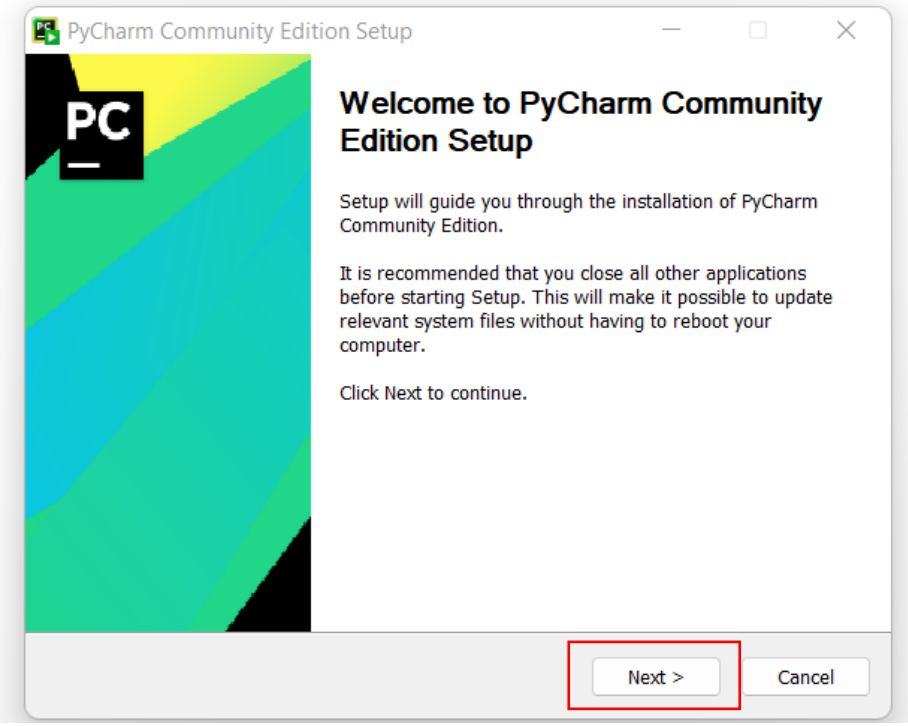


Once downloaded, double-click on the executable file to install the PyCharm

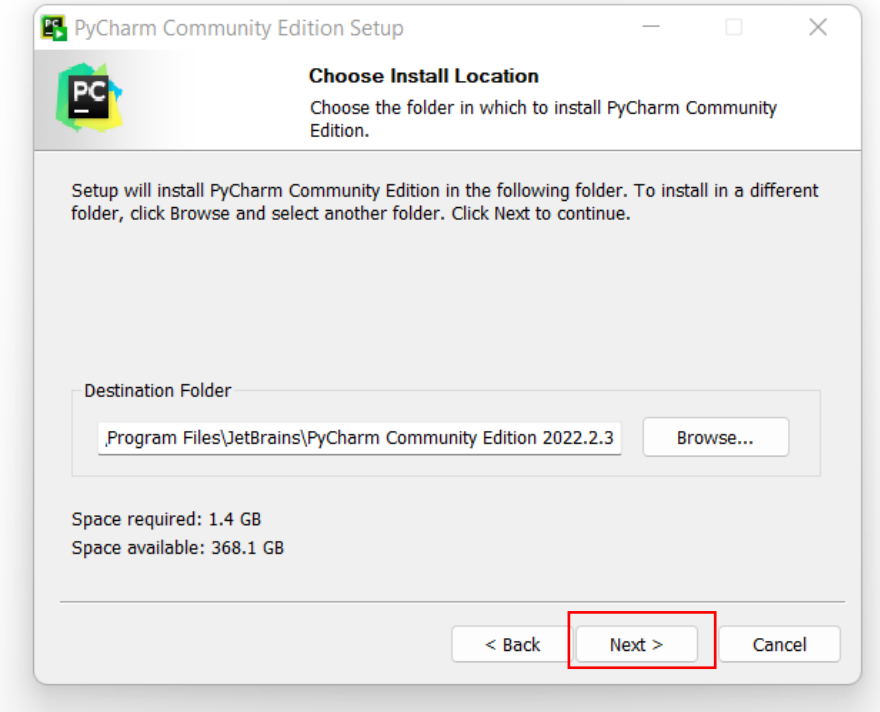


The **PyCharm Community Edition Setup** window will appear on the screen.

Click on **Next** button to proceed.



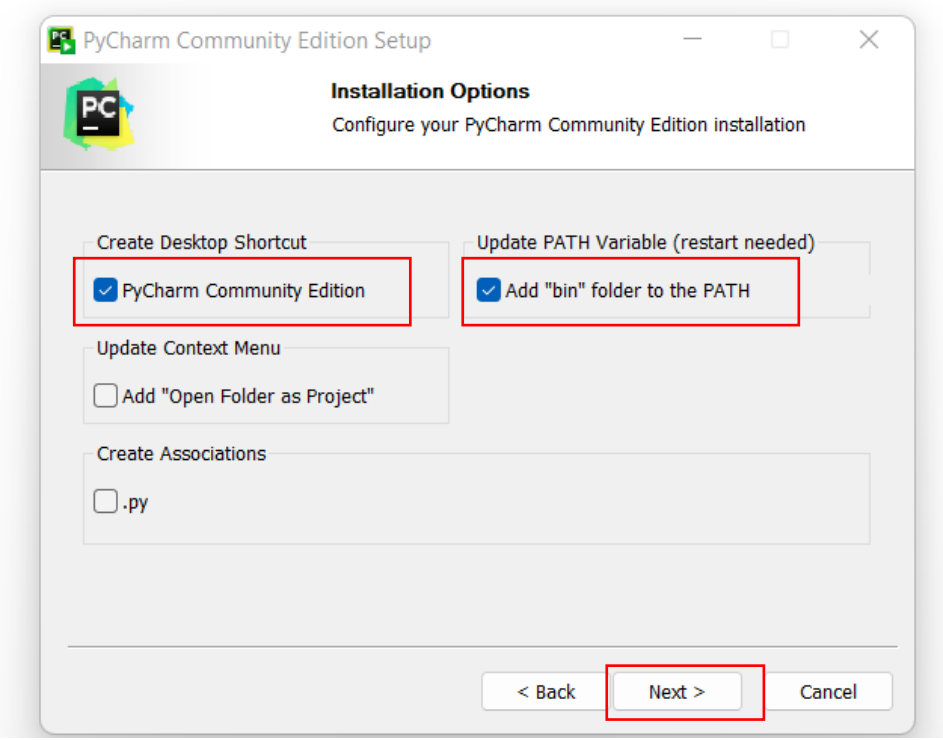
Click on **Next** button to proceed.



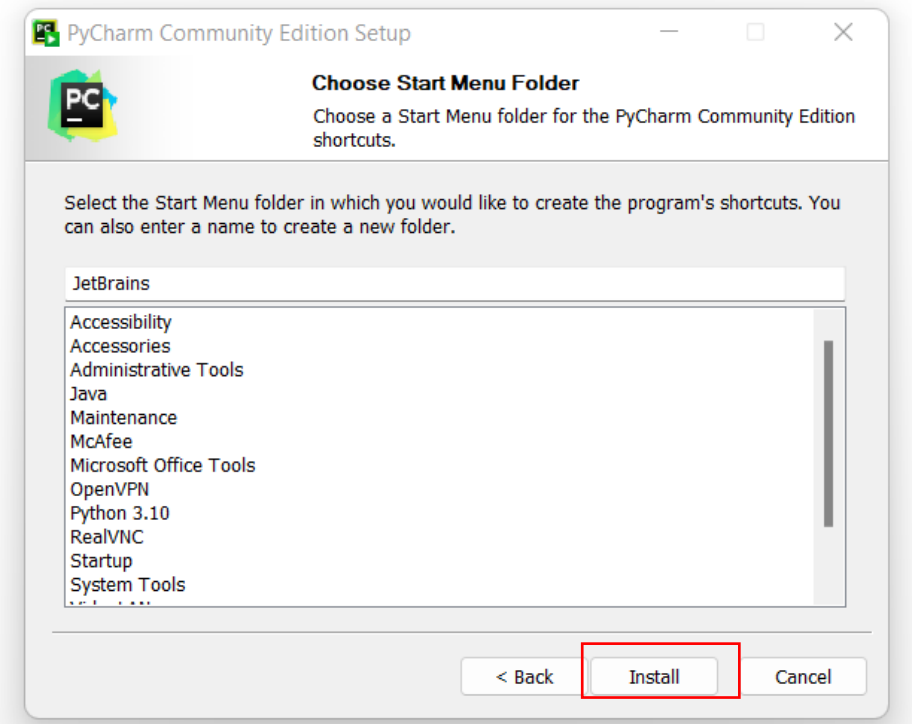
Click on **Pycharm Community Edition** checkbox to create desktop shortcut.

Click on **Add "bin" folder to the PATH** checkbox to update the PATH Variable

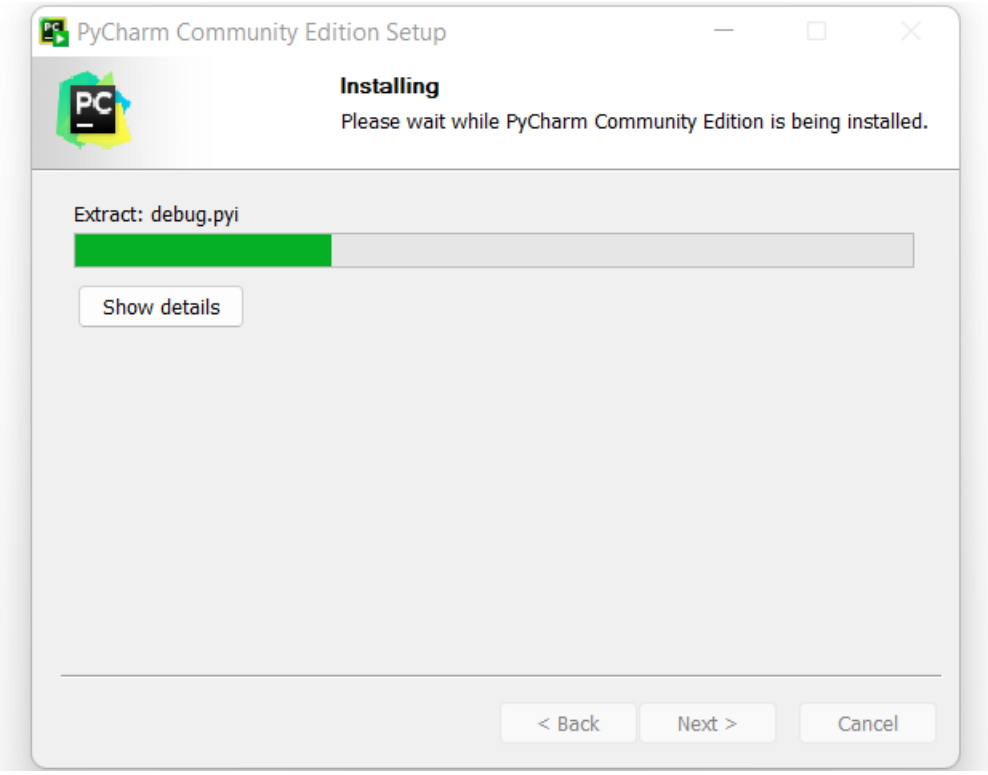
Click on **Next** button to proceed.



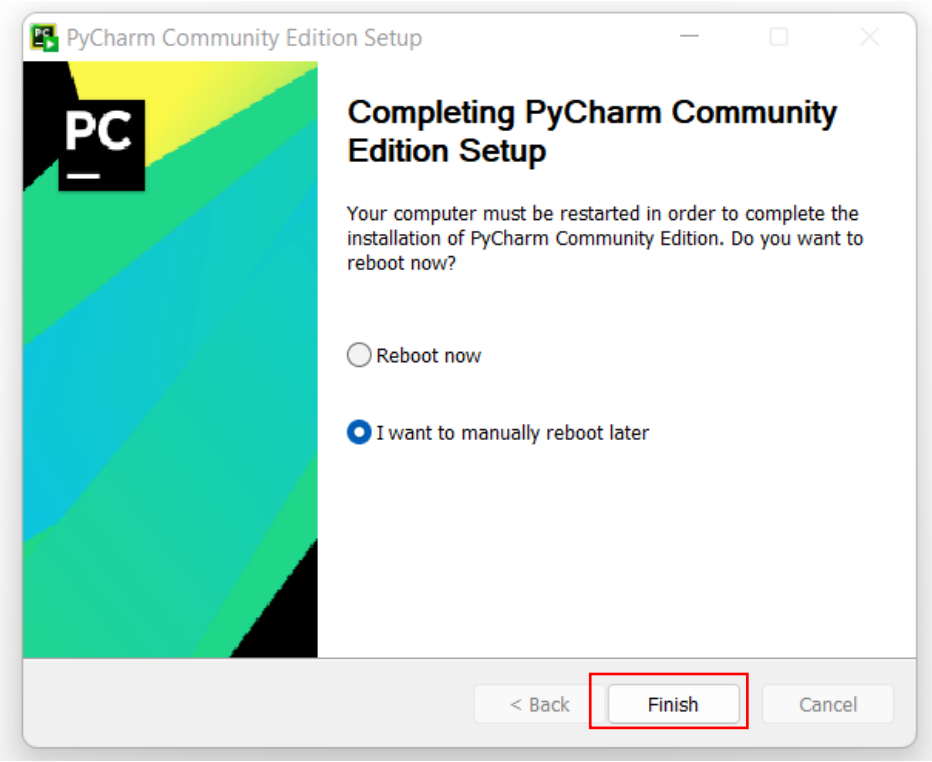
Click on the **Install** button to install PyCharm.



The installation process will start.



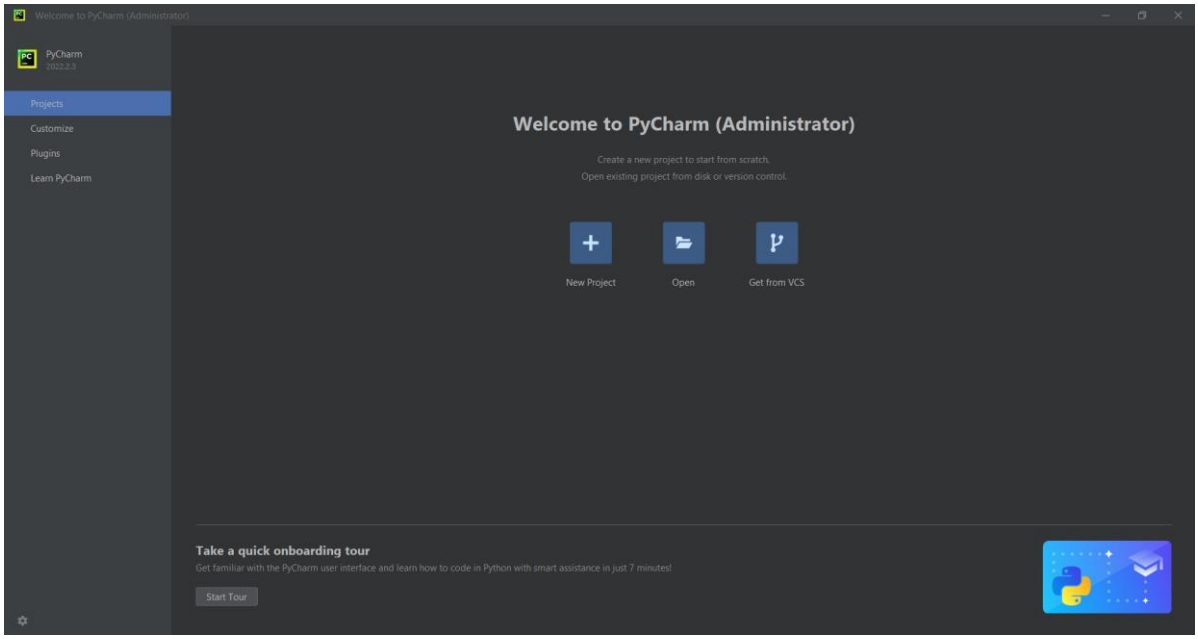
Once installation is complete, click on the **Finish** button.



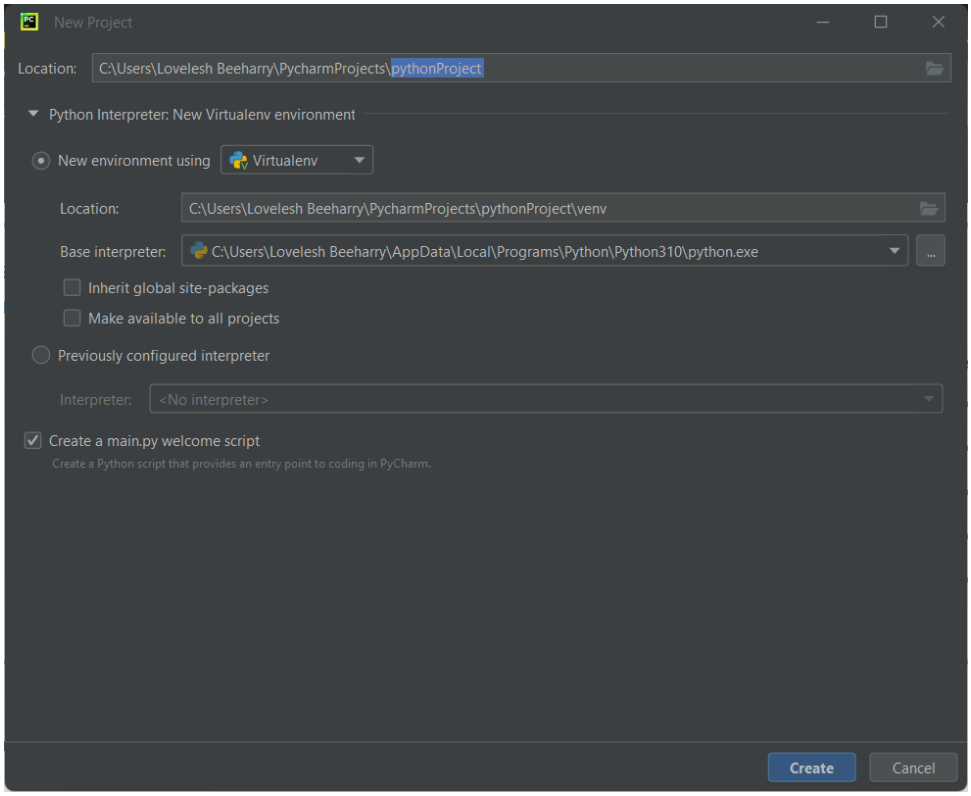
Step 4 - Create a new project using PyCharm

To create a new project in PyCharm:

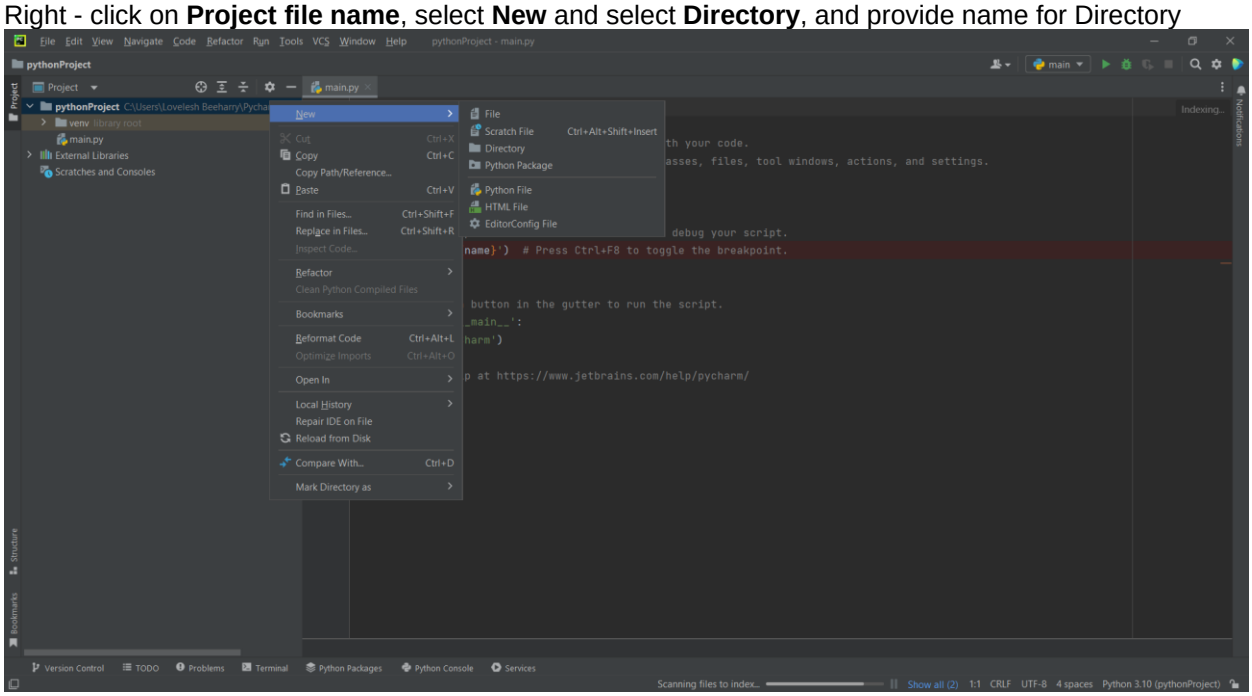
First, open the PyCharm by double-clicking on it, and click on **New Project**.



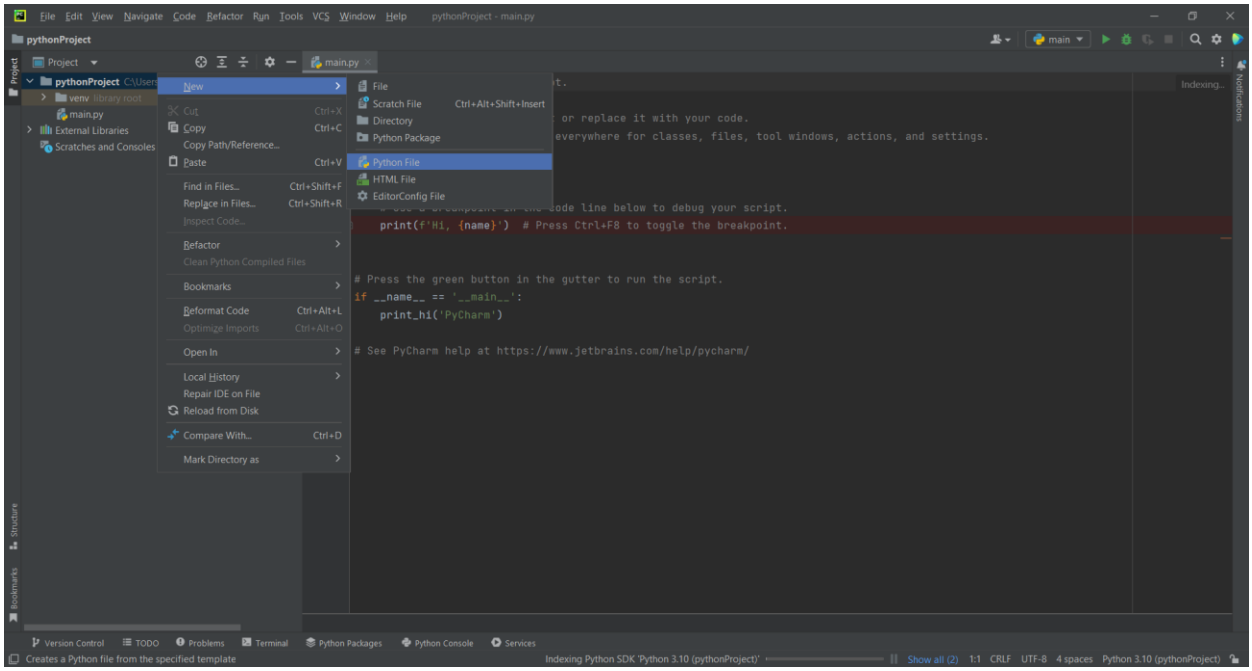
Provide the project name and click on the **Create** button.



Right - click on **Project file name**, select **New** and select **Directory**, and provide name for Directory



Right – click on **created Directory**, select **New** and select **Python File** and provide name for Python File.



Exercise 1

Write a Python program to print the text “Hello Word” in the console window.

Exercise 1 Solution

```
# Print "Hello World" to the console
print ("Hello World")
```


Exercise 2

Write a Python program using Selenium libraries that opens a Google Chrome browser, navigate to "https://www.saucedemo.com/" and closes the browser.

Exercise 2 Solution

```
from selenium import webdriver

# Create a new instance of the Chrome driver
driver = webdriver.Chrome()

# Navigate to the webpage
driver.get("https://www.saucedemo.com/")

# Close the browser
driver.quit()
```


Exercise 3

Write a Python program using Selenium libraries that opens a Google Chrome browser, navigate to "https://www.saucedemo.com/", enter the username 'standard_user' in the username textbox identified using relative XPath locator, enter the password 'secret_sauce' in the password textbox which is identified using CSS selector and click on the login button using ID locator. The program should then verify that the user is successfully logged in by verifying if the element 'inventory_container' is displayed using the classname locator.

If the user is logged in the text 'Login successful!' should be displayed, if not, the text 'Login failed!'.

After this verification, the browser should then close.

Exercise 3 Solution

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Initialize the Chrome WebDriver
driver = webdriver.Chrome()

# Navigate to the URL
driver.get("https://www.saucedemo.com/")

# Find the username textbox using relative XPath locator and enter the username
username_input = driver.find_element(By.XPATH, "//input[@id='user-name']")
username_input.send_keys("standard_user")

# Find the password textbox using CSS selector and enter the password
password_input = driver.find_element(By.CSS_SELECTOR, "input#password")
password_input.send_keys("secret_sauce")

# Find the login button using ID locator and click on it
login_button = driver.find_element(By.ID, "login-button")
login_button.click()

# Wait for the inventory container to be displayed
wait = WebDriverWait(driver, 10)
inventory_container = wait.until(EC.visibility_of_element_located((By.CLASS_NAME,
"inventory_container")))

# Verify successful login and print the result
if inventory_container.is_displayed():
    print("Login successful!")
else:
    print("Login failed!")

# Close the browser
driver.quit()
```


Exercise 4

Same as exercise 4 with the implementation of a try catch construct. The automated test needs to be wrapped in a function named ‘login_to_saucedemo’ which will be called to execute the test.

Exercise 4 Solution

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Function to perform the automated tests
def login_to_saucedemo():
    # Initialize the Chrome WebDriver
    driver = webdriver.Chrome()

    try:
        # Navigate to the URL
        driver.get("https://www.saucedemo.com/")

        # Find the username textbox using relative XPath locator and enter the username
        username_input = driver.find_element(By.XPATH, "//input[@id='user-name']")
        username_input.send_keys("standard_user")

        # Find the password textbox using CSS selector and enter the password
        password_input = driver.find_element(By.CSS_SELECTOR, "input#password")
        password_input.send_keys("secret_sauce")

        # Find the login button using ID locator and click on it
        login_button = driver.find_element(By.ID, "login-button")
        login_button.click()

        # Wait for the inventory container to be displayed
        wait = WebDriverWait(driver, 10)
        inventory_container = wait.until(EC.visibility_of_element_located((By.CLASS_NAME,
"inventory_container")))

        # Verify successful login and print the result
        if inventory_container.is_displayed():
            print("Login successful!")
        else:
            print("Login failed!")

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

    finally:
        # Close the browser
        driver.quit()

# Call the function to perform the login process
login_to_saucedemo()
```


Exercise 5

Same as exercise 4 but now the password field will be located using friendly locator.

First the textbox for the username needs to be found using the id locator then the browser will find the input tag element just below the username textbox to enter the password.

Exercise 5 Solution

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.relative_locator import locate_with
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Function to perform the automated tests
def login_to_saucedemo():
    # Initialize the Chrome WebDriver
    driver = webdriver.Chrome()

    try:
        # Navigate to the URL
        driver.get("https://www.saucedemo.com/")

        # Find the username textbox using id locator and enter the username
        username_input = driver.find_element(By.ID, "user-name")
        username_input.send_keys("standard_user")

        # Find the password textbox relative to the username textbox
        password_input =
driver.find_element(locate_with(By.TAG_NAME,"input").below({By.ID:"user-name"}))
        password_input.send_keys("secret_sauce")

        # Find the login button using ID locator and click on it
        login_button = driver.find_element(By.ID, "login-button")
        login_button.click()

        # Wait for the inventory container to be displayed
        wait = WebDriverWait(driver, 10)
        inventory_container = wait.until(EC.visibility_of_element_located((By.CLASS_NAME,
"inventory_container")))

        # Verify successful login and print the result
        if inventory_container.is_displayed():
            print("Login successful!")
        else:
            print("Login failed!")

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

    finally:
        # Close the browser
        driver.quit()

# Call the function to perform the login process
login_to_saucedemo()
```


Exercise 6

Same as exercise 5 but now the automation solution need to wait for the page to completely load before starting to starting to enter the username and password. If the page does not load within 180s then the automation test should continue.

The page status need to be scanned to check if the page is ready or not.

Exercise 6 Solution

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.relative_locator import locate_with
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

# Function to perform the automated tests
def login_to_saucedemo():
    # Initialize the Chrome WebDriver
    driver = webdriver.Chrome()


    try:
        # Navigate to the URL
        driver.get("https://www.saucedemo.com/")

        # Wait for the page to load completely
        while_limit = 0
        while driver.execute_script("return document.readyState") != "complete" and
while_limit < 180:
            time.sleep(1)
            while_limit += 1

        # Find the username textbox using id locator and enter the username
        username_input = driver.find_element(By.ID, "user-name")
        username_input.send_keys("standard_user")

        # Find the password textbox
        password_input =
driver.find_element(locate_with(By.TAG_NAME,"input").below({By.ID: "user-name"}))
        password_input.send_keys("secret_sauce")

        # Find the login button using ID locator and click on it
        login_button = driver.find_element(By.ID, "login-button")
        login_button.click()

        # Wait for the inventory container to be displayed
        wait = WebDriverWait(driver, 10)
        inventory_container = wait.until(EC.visibility_of_element_located((By.CLASS_NAME,
"inventory_container")))

        # Verify successful login and print the result
        if inventory_container.is_displayed():
            print("Login successful!")
        else:
            print("Login failed!")

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

    finally:
        # Close the browser
```

```
    driver.quit()

# Call the function to perform the login process
login_to_saucedemo()
```


Exercise 7

Same as exercise 6 but now as the user is successfully logged in, a partial screen capture of the third item on the inventory list (with its product description and the add to card button) will be taken and saved to the main project directory under the filename inventory_3.png.

Example of the main project directory is C:\Users\<Your UserName>\PycharmProjects\<Your project name>\<your subfolder>

Exercise 7 Solution

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.relative_locator import locate_with
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

# Function to perform the automated tests
def login_to_saucedemo():
    # Initialize the Chrome WebDriver
    driver = webdriver.Chrome()


    try:
        # Navigate to the URL
        driver.get("https://www.saucedemo.com/")

        # Wait for the page to load completely
        while_limit = 0
        while driver.execute_script("return document.readyState") != "complete" and
while_limit < 180:
            time.sleep(1)
            while_limit += 1

        # Find the username textbox using id locator and enter the username
        username_input = driver.find_element(By.ID, "user-name")
        username_input.send_keys("standard_user")

        # Find the password textbox
        password_input =
driver.find_element(locate_with(By.TAG_NAME,"input").below({By.ID: "user-name"}))
        password_input.send_keys("secret_sauce")

        # Find the login button using ID locator and click on it
        login_button = driver.find_element(By.ID, "login-button")
        login_button.click()

        # Wait for the inventory container to be displayed
        wait = WebDriverWait(driver, 10)
        inventory_container = wait.until(EC.visibility_of_element_located((By.CLASS_NAME,
"inventory_container")))

        # Verify successful login and print the result
        if inventory_container.is_displayed():
            print("Login successful!")

            # Take a screenshot of the third inventory item
            inventory_item = driver.find_element(By.XPATH, ".//*[@class='inventory_item'][3]")
            screenshot = inventory_item.screenshot_as_png
            with open("inventory_3.png", "wb") as file:
                file.write(screenshot)

        else:
```

```
            print("Login failed!")

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

    finally:
        # Close the browser
        driver.quit()

# Call the function to perform the login process
login_to_saucedemo()
```


Exercise 8

Same as exercise 7 but now the browser will be launched in headless mode. After login, a new tab will be opened to navigate to 'https://www.saucedemo.com/cart.html'. After navigation, the context will switch back to the original tab and a partial screen capture of the third item on the inventory list (with its product description and the add to card button) will be taken and saved as inventory_three.png in the main project directory.

Example of the main project directory is C:\Users\<Your UserName>\PycharmProjects\<Your project name>\<your subfolder>

Exercise 8 Solution

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.relative_locator import locate_with
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
from selenium.webdriver.chrome.options import Options

# Function to perform the automated tests
def login_to_saucedemo():

    # Set up Chrome options
    chrome_options = Options()
    chrome_options.add_argument("--headless")

    # Initialize the Chrome WebDriver
    driver = webdriver.Chrome(options=chrome_options)

    try:
        # Navigate to the URL
        driver.get("https://www.saucedemo.com/")

        # Wait for the page to load completely
        while_limit = 0
        while driver.execute_script("return document.readyState") != "complete" and
while_limit < 180:
            time.sleep(1)
            while_limit += 1

        # Find the username textbox using id locator and enter the username
        username_input = driver.find_element(By.ID, "user-name")
        username_input.send_keys("standard_user")

        # Find the password textbox
        password_input =
driver.find_element(locate_with(By.TAG_NAME,"input").below({By.ID: "user-name"}))
        password_input.send_keys("secret_sauce")

        # Find the login button using ID locator and click on it
        login_button = driver.find_element(By.ID, "login-button")
        login_button.click()

        # Wait for the inventory container to be displayed
        wait = WebDriverWait(driver, 10)
        inventory_container = wait.until(EC.visibility_of_element_located((By.CLASS_NAME,
"inventory_container")))

        # Verify successful login and print the result
        if inventory_container.is_displayed():
            print("Login successful!")


          # Store the current window handle
```

```
            original_window = driver.current_window_handle

            # Open a new tab and navigate to the cart page
            driver.switch_to.new_window(WindowTypes.TAB)
            driver.get("https://www.saucedemo.com/cart.html")

            # Switch back to the original tab
            driver.switch_to.window(original_window)

             # Take a screenshot of the third inventory item
             inventory_item = driver.find_element(By.XPATH, ".//*[@class='inventory_item'][3]")
             screenshot = inventory_item.screenshot_as_png
             with open("inventory_three.png", "wb") as file:
                 file.write(screenshot)

        else:
            print("Login failed!")

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

    finally:
        # Close the browser
        driver.quit()

# Call the function to perform the login process
login_to_saucedemo()
```


Exercise 9

Same as exercise 7 (not in headless mode) but now a soft assertion will be made after the user is logged in to make sure that the product filter is displayed on the page.

The product filter should be found using classname locator and when the assertion passes, the text “Assertion Pass: Product filter is displayed” should be printed on the console window.

Exercise 9 Solution

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.relative_locator import locate_with
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

# Function to perform the automated tests
def login_to_saucedemo():
    # Initialize the Chrome WebDriver
    driver = webdriver.Chrome()

    try:
        # Navigate to the URL
        driver.get("https://www.saucedemo.com/")

        # Wait for the page to load completely
        while_limit = 0
        while driver.execute_script("return document.readyState") != "complete" and
while_limit < 180:
            time.sleep(1)
            while_limit += 1

        # Hard assertion to verify the page title before login
        page_title = driver.title
        assert page_title == "Swag Labs", f"Assertion Fail: Title is not Swag Labs, it is
{page_title}"
        print("Assertion Pass: Title is Swag Labs")

        # Find the username textbox using ID locator and enter the username
        username_input = driver.find_element(By.ID, "user-name")
        username_input.send_keys("standard_user")

        # Find the password textbox using relative locator and enter the password
        password_input = driver.find_element(locate_with(By.TAG_NAME,
"input").below({By.ID: "user-name"}))
        password_input.send_keys("secret_sauce")

        # Find the login button using ID locator and click on it
        login_button = driver.find_element(By.ID, "login-button")
        login_button.click()

        # Wait for the inventory container to be displayed
        wait = WebDriverWait(driver, 10)
        inventory_container = wait.until(EC.visibility_of_element_located((By.CLASS_NAME,
"inventory_container")))

        # Verify successful login and print the result
        if inventory_container.is_displayed():
            print("Login successful!")

            # Soft assertion for the product filter
            try:
```

```
            product_filter = driver.find_element(By.CLASS_NAME, "product_sort_container")
            assert product_filter.is_displayed(), "Product filter is not displayed"
            print("Assertion Pass: Product filter is displayed")
        except AssertionError as e:
            print(str(e))

        # Take a screenshot of the third inventory item
        inventory_item = driver.find_element(By.XPATH,
"(.//*[@class='inventory_item'])[3]")
        screenshot = inventory_item.screenshot_as_png
        with open("inventory_3.png", "wb") as file:
            file.write(screenshot)

    else:
        print("Login failed!")

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

finally:
    # Close the browser
    driver.quit()

# Call the function to perform the login process
login_to_saucedemo()
```


Exercise 10

Same as exercise 9 but now a hard assertion will be made on the page title before the user logs in.

The page title should be “Swag Labs”. If the title matches the text “Assertion Pass: Title is Swag Labs" should be displayed.

Exercise 10 Solution

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.relative_locator import locate_with
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

# Function to perform the automated tests
def login_to_saucedemo():
    # Initialize the Chrome WebDriver
    driver = webdriver.Chrome()

    try:
        # Navigate to the URL
        driver.get("https://www.saucedemo.com/")

        # Wait for the page to load completely
        while_limit = 0
        while driver.execute_script("return document.readyState") != "complete" and
while_limit < 180:
            time.sleep(1)
            while_limit += 1

        # Hard assertion to verify the page title before login
        page_title = driver.title
        assert page_title == "Swag Labs", f"Assertion Fail: Title is not Swag Labs, it is
{page_title}"
        print("Assertion Pass: Title is Swag Labs")

        # Find the username textbox using ID locator and enter the username
        username_input = driver.find_element(By.ID, "user-name")
        username_input.send_keys("standard_user")

        # Find the password textbox using relative locator and enter the password
        password_input = driver.find_element(locate_with(By.TAG_NAME,
"input").below({By.ID: "user-name"}))
        password_input.send_keys("secret_sauce")

        # Find the login button using ID locator and click on it
        login_button = driver.find_element(By.ID, "login-button")
        login_button.click()

        # Wait for the inventory container to be displayed
        wait = WebDriverWait(driver, 10)
        inventory_container = wait.until(EC.visibility_of_element_located((By.CLASS_NAME,
"inventory_container")))

        # Verify successful login and print the result
        if inventory_container.is_displayed():
            print("Login successful!")

            # Soft assertion for the product filter
            try:
```

```
            product_filter = driver.find_element(By.CLASS_NAME, "product_sort_container")
            assert product_filter.is_displayed(), "Product filter is not displayed"
            print("Assertion Pass: Product filter is displayed")
        except AssertionError as e:
            print(str(e))

        # Take a screenshot of the third inventory item
        inventory_item = driver.find_element(By.XPATH,
"(.//*[@class='inventory_item'])[3]")
        screenshot = inventory_item.screenshot_as_png
        with open("inventory_3.png", "wb") as file:
            file.write(screenshot)

    else:
        print("Login failed!")

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

finally:
    # Close the browser
    driver.quit()

# Call the function to perform the login process
login_to_saucedemo()
```


Exercise 11

Same as exercise 10 except that when the test is completed, the time taken for the test to be executed is displayed in milliseconds.

Exercise 11 Solution

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.relative_locator import locate_with
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

# Function to perform the automated tests
def login_to_saucedemo():
    # Initialize the Chrome WebDriver
    driver = webdriver.Chrome()

    # Start time for the test
    start_time = time.time()

    try:
        # Navigate to the URL
        driver.get("https://www.saucedemo.com/")

        # Wait for the page to load completely
        while_limit = 0
        while driver.execute_script("return document.readyState") != "complete" and
while_limit < 180:
            time.sleep(1)
            while_limit += 1

        # Hard assertion to verify the page title before login
        page_title = driver.title
        assert page_title == "Swag Labs", f"Assertion Fail: Title is not Swag Labs, it is
{page_title}"
        print("Assertion Pass: Title is Swag Labs")

        # Find the username textbox using ID locator and enter the username
        username_input = driver.find_element(By.ID, "user-name")
        username_input.send_keys("standard_user")

        # Find the password textbox using relative locator and enter the password
        password_input = driver.find_element(locate_with(By.TAG_NAME,
"input").below({By.ID: "user-name"}))
        password_input.send_keys("secret_sauce")

        # Find the login button using ID locator and click on it
        login_button = driver.find_element(By.ID, "login-button")
        login_button.click()

        # Wait for the inventory container to be displayed
        wait = WebDriverWait(driver, 10)
        inventory_container = wait.until(EC.visibility_of_element_located((By.CLASS_NAME,
"inventory_container")))

        # Verify successful login and print the result
        if inventory_container.is_displayed():
            print("Login successful!")
```

```
            # Soft assertion for the product filter
            try:
                product_filter = driver.find_element(By.CLASS_NAME, "product_sort_container")
                assert product_filter.is_displayed(), "Product filter is not displayed"
                print("Assertion Pass: Product filter is displayed")
            except AssertionError as e:
                print(str(e))

            # Take a screenshot of the third inventory item
            inventory_item = driver.find_element(By.XPATH,
"(.//*[@class='inventory_item'])[3]")
            screenshot = inventory_item.screenshot_as_png
            with open("inventory_3.png", "wb") as file:
                file.write(screenshot)

        else:
            print("Login failed!")

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

    finally:
        # End time for the test
        end_time = time.time()

        # Calculate and display the test execution time in milliseconds
        execution_time = (end_time - start_time) * 1000  # Convert to milliseconds
        print(f"Test completed in {execution_time:.2f} ms")

        # Close the browser
        driver.quit()

# Call the function to perform the login process
login_to_saucedemo()
```


Exercise 12

Reporting functionalities will now be added to the result of exercise 11.
pytest-html will be used for test logging and reporting.

The tests to be logged onto the report are as below:

1) Navigate to Website
2) Verify the page title
3) Login on Website
4) Verify Product Filter

An HTML report should be produced with the main test case being “test_login_to_saucedemo”. Under this test case, there should then be the individual test steps of navigation, page title verification, login, and product filter verification.

Pytest fixtures are to be used for the report. One fixture is to setup the report configuration and the second fixture is for the web driver configuration.

Note:

1) Run pip install pytest-html in PyCharm Terminal to install the libraries needed for pytest-html in the project.
2) To generate the html report, run the command below in the PyCharm Terminal in the correct folder:
   pytest <script name> --html=report.html

   For example: in the directory
   “C:\Users\LoveleshBeeharry\PycharmProjects\pyTestAutomation\Python Exercises>” in the PyCharm Terminal, run the below command:
   pytest Exercises.py --html=report.html

Exercise 12 Solution

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.relative_locator import locate_with
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
import pytest

@pytest.fixture(scope='session', autouse=True)
def setup_reporting():
    config = {
        'report_title': 'Test Report for Saucedemo',
        'output_file': 'report.html'
    }
    return config

@pytest.fixture(scope='function')
def driver():
    driver = webdriver.Chrome()
    yield driver

def test_login_to_saucedemo(setup_reporting, driver):

    start_time = time.time()

    try:
        # Step 1: Navigate to Website
        driver.get("https://www.saucedemo.com/")
        assert driver.current_url == "https://www.saucedemo.com/", "URL did not match."
        print("\nStep 1: Navigate to Website - Pass")

        # Wait for the page to load completely
        while_limit = 0
        while driver.execute_script("return document.readyState") != "complete" and
while_limit < 180:
            time.sleep(1)
            while_limit += 1

        # Step 2: Page Title Verification
        page_title = driver.title
        assert page_title == "Swag Labs", f"Assertion Fail: Title is not Swag Labs, it is
{page_title}"
        print("Step 2: Page Title Verification - Pass")

        # Step 3: Login on Website
        username_input = driver.find_element(By.ID, "user-name")
        username_input.send_keys("standard_user")
        password_input = driver.find_element(locate_with(By.TAG_NAME,
"input").below({By.ID: "user-name"}))
        password_input.send_keys("secret_sauce")
        login_button = driver.find_element(By.ID, "login-button")
        login_button.click()
```

```
        wait = WebDriverWait(driver, 10)
        inventory_container = wait.until(EC.visibility_of_element_located((By.CLASS_NAME,
"inventory_container")))
        assert inventory_container.is_displayed(), "Login failed, inventory container not
displayed."
        print("Step 3: Login on Website - Pass")

        # Step 4: Verify Product Filter
        product_filter = driver.find_element(By.CLASS_NAME, "product_sort_container")
        assert product_filter.is_displayed(), "Product filter is not displayed"
        print("Step 4: Verify Product Filter - Pass")

    except Exception as e:
        print(f"An error occurred: {str(e)}")
    finally:
        end_time = time.time()
        execution_time = (end_time - start_time) * 1000  # Convert to milliseconds
        print(f"Test completed in {execution_time:.2f} ms")
        driver.quit()
```