SDET Intern Pre-Onboarding Preparation Tasks

Welcome Message

Congratulations on being selected as SDET interns at Cyware!

As your Tech Ambassador, I'm excited to welcome you to our Cyware team. To ensure you hit the ground running during your onboarding, please complete the following preparation tasks. These will build your foundation in Python, Pytest, Selenium, and automation testing.

Why This Preparation Matters: At Cyware, we protect organizations from cyber threats through robust Products. As SDET professionals, your automated tests will be the frontline safeguard for detecting bugs before they reach production. This preparation ensures you're ready to contribute meaningfully from day one.

Pre-Onboarding Tasks Checklist

Note: Each task has a one-day deadline for completion.

Task 1: Environment Setup and Installation

Objective: Set up your development environment.

Requirements to Install:

1. Install Python 3.9 or higher from python.org
2. Install Visual Studio Code or Pycharm with Python extensions
3. Create a project directory called "sdet_preparation"
4. Set up a Python virtual environment in your project directory
5. Install the following Python packages: pytest, selenium, requests, pandas, openpyxl

Deliverable:

- Create a verification script that imports all installed packages and prints their versions
- Take a screenshot showing successful execution
- Document any installation issues you encountered and how you resolved them

---

Task 2: Python Fundamentals for Testing

Objective: Demonstrate Python programming skills relevant to test automation.

What to Create: Create a Python file with the following functions and classes:

Part A: Basic Functions

1. **create_test_user_data()**: Create a function that returns a dictionary containing sample user data (name, email, password, age, role)
2. **validate_email_format()**: Write a function that checks if an email contains "@" and "." symbols
3. **read_test_data_from_json()**: Create a function that reads data from a JSON file with proper error handling

Part B: Object-Oriented Programming

1. **TestUser Class**: Create a class to represent a test user with:
    - Constructor that accepts name, email, password
    - Method to validate if user data is complete and valid
    - Class method to generate a random test user
2. **TestDataManager Class**: Create a class to manage test data with:
    - Constructor that accepts a file path
    - Method to load test users from JSON file
    - Method to save test results to a file

Part C: Error Handling

1. **safe_web_request()**: Create a function that makes HTTP requests with proper error handling for connection errors, timeouts, and invalid URLs

Supporting Files:

- Create a JSON file with sample valid and invalid user data for testing

Deliverable:

- Complete Python file with all functions and classes implemented
- JSON test data file
- Documentation explaining your implementation choices

Task 3: Pytest Framework Implementation

Objective: Create basic test suites using pytest framework.

What to Create:

Part A: Basic Test Implementation

1. Create a test class for user validation functionality
2. Write test methods to verify valid email formats work correctly
3. Write test methods to verify invalid email formats are rejected properly
4. Use pytest parametrize decorator to test multiple email scenarios in one test

Part B: Fixtures and Configuration

1. Create pytest fixtures to provide sample test data
2. Set up a pytest configuration file (pytest.ini) with proper settings
3. Implement test markers for categorizing tests (smoke, regression, readonly)

Part C: Advanced Testing Scenarios

1. Create tests that use your TestDataManager class
2. Write tests that combine multiple functions/classes
3. Implement both positive and negative test scenarios

Test Categories to Cover:

- Smoke tests for critical functionality
- Regression tests for detailed scenarios
- Parametrized tests for data-driven testing

Deliverable:

- Complete test file with multiple test classes and methods
- Pytest configuration file
- Test execution report showing all tests passing
- Documentation of test coverage and approach

Task 4: Selenium Web Automation Basics and Learning about Xpaths

Objective: Create basic web automation scripts using Selenium WebDriver.

What to Create:

Part A: Browser Management

1. **BrowserManager Class**: Create a utility class that can:
    - Initialize Chrome WebDriver with chrome options
    - Handle browser startup and shutdown safely
    - Navigate to URLs with error handling
    - Set appropriate timeouts and window settings

Part B: Element Interaction Helper

1. **WebElementHelper Class**: Create a helper class that can:
    - Find web elements safely using explicit waits
    - Enter text into input fields with clearing existing text
    - Click elements with wait conditions
    - Handle common web element interactions

Learning Focus:

- Understanding WebDriver initialization
- Implementing explicit waits vs implicit waits
- Handling dynamic web elements
- Error handling in web automation
- Browser compatibility considerations

Deliverable:

- Complete automation script with both classes implemented
- Demonstration of successful automation on a test website
- Documentation of challenges faced and solutions implemented

---

Task 5: Page Object Model Design

Objective: Implement the Page Object Model design pattern for maintainable test automation.

What to Create:

Part A: Base Page Foundation

1. **BasePage Class**: Create a base class with common functionality:
    - Constructor that accepts WebDriver instance
    - Methods for waiting for elements (present, clickable, visible)
    - Methods for common actions (click, enter text, get text)
    - Method to check if elements are displayed
    - Error handling for all interactions

Part B: Specific Page Objects

1. **LoginPage Class**: Create a page object for login functionality:
    - Define locators for all login page elements (username, password, login button, error messages)
    - Implement methods for each user action (enter username, enter password, click login)
    - Create a comprehensive login method that combines all actions
    - Add methods to retrieve error messages and verify page state

Part C: Page Object Organization

- Organize page objects in a separate directory/package
- Follow naming conventions and best practices
- Implement proper inheritance from BasePage
- Use appropriate locator strategies (ID, CSS, XPath)

Design Principles to Follow:

- Separation of test logic from page interaction logic
- Reusable methods for common actions
- Clear, descriptive method names
- Proper encapsulation of page elements
- Maintainable and scalable structure

Deliverable:

- Well-organized page object classes in separate files
- Demonstration of page object usage with sample tests
- Documentation explaining the Page Object Model benefits and implementation

---

Learning Goals Summary

By completing these tasks, you should demonstrate understanding of:

Python Programming:

- Object-oriented programming concepts
- Error handling and exception management
- File operations and data management

- Function design and implementation

Pytest Framework:

- Test organization and structure
- Fixtures and test data management
- Parametrized testing for data-driven tests
- Test markers and categorization
- Configuration and setup

Selenium Automation:

- WebDriver initialization and management
- Element location strategies and best practices
- Explicit waits and synchronization
- Browser automation workflows
- Error handling in web automation

Page Object Model:

- Design pattern implementation
- Code organization and maintainability
- Separation of concerns
- Reusable component design

Submission Requirements

What to Submit:

1. Code Files:

- Python practice file with all functions and classes
- Pytest test files with comprehensive test coverage
- Selenium automation scripts with browser management

- Page Object Model implementation with proper organization
- Integration test suite combining all concepts

2. Configuration and Data Files:

- Pytest configuration file
- JSON test data files
- Requirements.txt file listing all dependencies

3. Documentation:

- README.md file with setup instructions and project overview
- Individual task completion notes explaining your approach
- Screenshots of successful test executions
- Documentation of any challenges faced and solutions found

4. Evidence of Execution:

- Screenshots showing successful test runs
- Terminal output demonstrating working code
- Any generated reports or logs

How to Submit:

1. Create a GitHub repository named `sdet-preparation-[your-name]`
2. Organize all files in a clear directory structure
3. Include comprehensive README.md with setup and execution instructions
4. Send repository link via email to harsh.shukla@cyware.com

Learning Resources

Essential Documentation:

- Python Official Tutorial
- Pytest Documentation
- Selenium Python Documentation
- Page Object Model Best Practices

Recommended Practice Sites:

- Python.org Exercises - For Python fundamentals
- Real Python - For comprehensive Python learning

Video Resources:

- Search for "Python for beginners" tutorials
- Look for "Selenium WebDriver Python" tutorials
- Find "pytest framework" introduction videos
- Watch "Page Object Model" implementation examples

Tips for Success

Time Management:

- Start with Task 1 immediately (environment setup is crucial)
- Allocate 2-3 hours daily for preparation work
- Don't rush - focus on understanding concepts thoroughly
- Ask questions early if you encounter difficulties

Learning Approach:

- Read the requirements carefully before starting each task
- Start with simple implementations and gradually add complexity
- Test your code frequently as you develop
- Document your learning process and challenges

Common Pitfalls to Avoid:

- Skipping virtual environment setup
- Not handling errors properly in automation scripts
- Hardcoding values instead of using variables
- Poor code organization and naming conventions
- Not testing code thoroughly before submission

Getting Help:

- Use official documentation as your primary resource
- Search Stack Overflow for specific error messages
- Practice on simple examples before complex implementations
- Document problems and solutions for future reference

Quality Focus:

- Write clean, readable code with proper comments
- Follow consistent naming conventions
- Implement proper error handling throughout
- Create meaningful test cases and assertions
- Organize code in logical, maintainable structure

Final Message

These preparation tasks are designed to ensure you arrive ready to excel as SDET professionals at Cyware. Every concept you master and every line of code you write brings you closer to building the automated defenses that protect organizations from cyber threats.

Focus on understanding rather than just completing. The concepts you learn here will be the foundation of your entire SDET career. Take your time, ask questions, and enjoy the learning process.

We're excited to see your completed work and welcome you to the Cyware team. Your journey as an SDET professional starts with these preparation tasks - make them count!

Questions? Feel free to reach out. Good luck with your preparation!

Prepared by: , Harsh Shukla, Tech Ambassador, Cyware
Submission Deadline: Before 4th August
Contact: harsh.shukla@cyware.com for questions