



# PLAYWRIGHT/PYTEST + PYTHON/GITHUB

February 21, 2026

Presenter: Jordan Kim S. Jubilo





**DURATION: 2 WEEKS**

**SCHEDULE: 1 HOUR PER DAY (10 DAYS)**

**01** Install and configure Playwright with Python

**02** Write and run automated browser tests

**03** Use Pytest for structured test execution

**04** Implement Page Object Model (POM)

**05** Handle waits, locators, forms, assertions

**06** Integrate GitHub for collaboration and version control





## DAY 1 – INTRODUCTION & ENVIRONMENT SETUP

### GOAL: SET UP DEVELOPMENT ENVIRONMENT

#### Topics:

- What is Playwright?
- Why Playwright vs Selenium?
- Installing Python (3.10+)
- Creating virtual environment
- Installing Playwright
- Installing browsers

#### Assignment:

- Launch browser and open Google
- Take a screenshot

#### Hands-on:

```
bash
```

```
pip install playwright  
playwright install
```

Run a basic script that launches Chromium.





## DAY 2 – FIRST AUTOMATION SCRIPT

### GOAL: UNDERSTAND BROWSER, CONTEXT, AND PAGE

#### Topics:

- Browser vs Context vs Page
- Headed vs headless mode
- Basic navigation
- Taking screenshots

#### Hands-on:

- Navigate to URL
- Capture screenshot
- Close browser

#### Assignment:

Automate:

- Open <https://blisscoders.pythonanywhere.com/>
- Verify page title





## DAY 3 – LOCATORS & SELECTORS

### GOAL: LEARN HOW TO INTERACT WITH ELEMENTS

#### Topics:

- Locator strategies:
  - `get_by_text()`
  - `get_by_role()`
  - `get_by_label()`
  - CSS selectors
  - XPATH selectors
- Why Playwright locators are powerful

#### Hands-on:

- Click a button
- Fill a text field
- Select checkbox

#### Assignment:

Automate a login form (demo site)





## DAY 4 – ASSERTIONS & WAITING

### GOAL: VALIDATE RESULTS PROPERLY

#### Topics:

- Auto-waiting in Playwright
- expect() assertions
- Waiting for element visibility
- Waiting for navigation

#### Hands-on:

- Assert page title
- Assert element visible
- Assert text content

#### Assignment:

Verify login success message





## DAY 5 – PYTEST INTEGRATION

### GOAL: STRUCTURE TESTS PROFESSIONALLY

#### Topics:

- Install pytest
- Running tests with pytest
- Test naming conventions
- Using fixtures
- Running tests in parallel

#### Assignment:

- Create 2 test cases
- Run via pytest

#### Hands-on:

```
bash

pip install pytest
pytest -v
```

Convert script into proper test file.





## WEEK 2 – ADVANCED CONCEPTS + GITHUB COLLABORATION CONCEPTS

### DAY 6 – PAGE OBJECT MODEL (POM)

#### GOAL: BUILD MAINTAINABLE FRAMEWORK

#### Topics:

- What is POM?
- Creating page classes
- Separating test logic from page logic

#### Assignment:

Refactor login test into POM structure.

#### Structure:

```
project/  
|  
├── pages/  
|   └── login_page.py  
├── tests/  
|   └── test_login.py
```





## WEEK 2 – ADVANCED CONCEPTS + GITHUB COLLABORATION CONCEPTS

### DAY 7 – HANDLING ADVANCED SCENARIOS

### GOAL: HANDLE REAL-WORLD AUTOMATION CHALLENGES

#### Topics:

- Handling dropdowns
- File upload
- Handling alerts
- Multiple tabs
- Iframes

#### Structure:

- Upload file
- Switch tabs
- Handle iframe





## WEEK 2 – ADVANCED CONCEPTS + GITHUB COLLABORATION CONCEPTS

### DAY 8 – GIT & GITHUB COLLABORATION

#### GOAL: VERSION CONTROL AND TEAMWORK

#### Topics:

- What is Git?
- What is GitHub?
- Creating repository
- Cloning repo
- Branching
- Pull Requests
- Code reviews
- .gitignore for Python

#### Hands-on:

```
bash

git init
git add .
git commit -m "Initial commit"
git branch feature-login
git push origin main
```

#### GitHub Workflow:

1. Create repo
2. Clone
3. Create feature branch
4. Push changes
5. Open Pull Request
6. Review & merge

#### Assignment:

1. Push automation project to GitHub
2. Create feature branch
3. Make change
4. Submit PR





## WEEK 2 – ADVANCED CONCEPTS + GITHUB COLLABORATIONCONCEPTS

### DAY 9 – TEST REPORTS & CI (INTRO)

#### GOAL: PROFESSIONAL EXECUTION

#### Topics:

- Pytest HTML reports
- Running in headless mode
- Running tests from command line
- Intro to GitHub Actions (basic CI)

#### Hands-on:

```
bash

pip install pytest-html
pytest --html=report.html
```

Create simple GitHub Actions YAML file.





## WEEK 2 – ADVANCED CONCEPTS + GITHUB COLLABORATION CONCEPTS

### DAY 10 – FINAL MINI PROJECT GOAL: APPLY EVERYTHING LEARNED

#### Project:

Automate a demo e-commerce website:

#### Requirements:

- Search product
- Add to cart
- Verify cart count
- Logout
- Use POM
- Use pytest
- Push to GitHub

#### Bonus:

Add README.md with:

- Setup instructions
- How to run tests
- Project structure





## WEEK 2 – ADVANCED CONCEPTS + GITHUB COLLABORATION CONCEPTS

### FINAL PROJECT STRUCTURE

#### Tools Used

- Python
- Playwright
- Pytest
- Git
- GitHub
- VS Code

bash

```
playwright-python-framework/
```

```
|
```

```
├─ pages/
```

```
|
```

```
├─ base_page.py
```

```
|
```

```
├─ login_page.py
```

```
|
```

```
└─ cart_page.py
```

```
|
```

```
├─ tests/
```

```
|
```

```
├─ test_login.py
```

```
|
```

```
└─ test_cart.py
```

```
|
```

```
├─ requirements.txt
```

```
├─ pytest.ini
```

```
├─ README.md
```

```
└─ .github/workflows/ci.yml
```





BLISS CODERS



# Q&A

THANK YOU