Playwright Curriculum — Syllabus (just topics & exercises)

Module A — Foundations (must know)

- Python basics refresher (context managers, async/await)
- HTTP fundamentals (methods, headers, status codes)
- HTML / DOM / CSS selectors
- Browser DevTools quick actions (Elements, Network, Console)
- Git basics (commit / branch / PR)
- Playwright vs Selenium vs Scrapy vs Puppeteer comparison

Module B — Playwright Core API (python)

- Install & setup (playwright, playwright-python)
- sync vs async API (`playwright.sync_api`, `playwright.async_api`)
- `async_playwright()` lifecycle
- Browser types: Chromium / Firefox / WebKit
- BrowserContext vs Page
- Locators: CSS, XPath, text, role selectors
- Actions: click, fill, type, hover, dblclick
- Navigation & routing: `page.goto()`, `page.wait_for_load_state()`
- Waits: explicit waits, `locator.wait_for()`, `page.wait_for_selector()`
- Tracing, screenshots, PDFs, video recording
- Emulation: viewport, device, user-agent, geolocation
- File upload & download handling
- Frames & iframes, child frames
- Shadow DOM access
- Handling dialogs, alerts, confirm, prompt

Module C — Network & Data extraction

- Intercepting requests/responses (`page.route`, `route.continue()`)
- Capturing XHR / fetch / GraphQL responses
- Parsing JSON responses vs scraping DOM
- Observing WebSocket / SSE (events)
- Saving assets (images, PDFs)
- Headless vs headful tradeoffs
- HAR capture & replay

Module D — Automation reliability & debugging

- Robust selectors and fallback strategies
- Retry patterns and exponential backoff
- Flaky test handling patterns and auto-retry decorators
- Timeouts & cancellation strategies
- Debugging techniques: `playwright.open_browser`, slowMo, devtools
- Using Playwright inspector and trace viewer
- Logging best practices

Module E — Stealth, Anti-bot & Ethics

- Browser context isolation (separate contexts)
- User-agent rotation strategies
- Proxy integration (HTTP, SOCKS, rotating proxies)
- Rate limiting and politeness (respect robots.txt & legal limits)
- Captcha detection patterns (integration points only)
- Ethical/legal checklist (scrape policy awareness)

Module F — Performance & Concurrency

- Running multiple contexts/pages in one browser instance
- Semaphore & concurrency control with `asyncio`
- Batching, pooling, and worker patterns
- CPU / memory footprint optimization
- Headless browser scaling considerations

Module G — Integration with data stack

- Export formats: CSV, JSONL, Parquet
- Direct DB writes (Postgres) and bulk upserts
- Streaming results to Kafka / Redis queues
- Integrating with Pandas for cleaning
- Vectorization pipeline hooks (preparing text for embeddings)

Module H — Testing, CI, Containerization

- Unit testing Playwright flows (pytest & fixtures)
- End-to-end test patterns
- Playwright in Docker (browsers in containers)
- CI pipelines: GitHub Actions / GitLab (headless runs, artifacts)
- Secrets management for tokens/credentials

Module I — Advanced targets & patterns

- Infinite scroll / lazy-loaded content strategies
- Pagination patterns (cursor, offset)
- Simulating human-like interactions (typing delays, mouse paths)
- Multi-step authenticated sessions (login flows, 2FA handling patterns)
- Interacting with SPA frameworks (React/Vue/Angular)
- Extracting from sites that render via WebAssembly

Module J — Anti-fraud & resilience

- Detecting bot blocks (status codes, response body, redirect loops)
- Circuit-breaker patterns and ban recovery
- IP/proxy rotation orchestration
- Request fingerprinting minimization

Module K — Production orchestration & scale

- Architecture patterns: orchestrator + workers + result store
- Task queues: RabbitMQ / Redis Queue / Celery / RQ
- Monitoring & alerting (success rates, latency, error budgets)
- Autoscaling worker pools (Kubernetes hints)
- Cost estimation and budgeting for large crawls

Module L — Playwright as an Agent Tool (AI agents)

- Exposing Playwright actions as callable tools/functions
- Designing sandboxed action interfaces for agents
- Safe I/O boundaries and prompt-to-action mappings
- Observability hooks for agent decisions

Hands-on Projects (progressive)

1. Project 1 — Single-page scraper: fetch title + top 10 links → save CSV.
2. Project 2 — Login-protected scraper: authenticate + scrape profile data.
3. Project 3 — Infinite-scroll scraper: scroll, capture items, dedupe, export JSONL.
4. Project 4 — XHR-intercept scraper: capture API responses and store raw JSON.
5. Project 5 — Multi-user crawler: run 20 concurrent contexts, write to Postgres.
6. Project 6 — Playwright-agent connector: build simple tool that an LLM can call to fetch page text and screenshots for RAG ingestion.
7. Capstone — Distributed crawler: Dockerized workers, queue-based orchestration, proxy rotation, resilience, and end-to-end pipeline to Vector DB.

Security & Compliance checks (final checklist)

- Credential vaulting used (no hardcoded secrets)

- Respect robots.txt and site TOS where applicable
- Rate limits configured and enforced
- Audit logs enabled for scraping runs

Recommended tooling / quick refs (names only)

- Playwright Python docs
- Browser DevTools Protocol
- Proxy providers (rotating proxy)
- Headless browser CI images
- pytest, Docker, PostgreSQL, Redis, RabbitMQ, ChromaDB/Pinecone

Suggested study sequence (order only)

1. Module A → 2. Module B → 3. Module C → 4. Module E → 5. Module D → 6. Projects 1–3 → 7. Module F/G → 8. Projects 4–6 → 9. Module H/I/J/K → 10. Capstone