Applied Python: From Fundamentals to Portfolio

Student Guide - v1.0 (regenerated) - 2025-12-18

Audience	Self-paced / instructor-led (beginner-to-intermediate)
Duration	6 stages (suggested 6-10 weeks), plus optional extensions
Prerequisites	Basic computer literacy; no prior programming required
Deliverable	A staged portfolio with a capstone project and GitHub packaging

Learning Outcomes

- Write clean Python code using functions, modules, and standard library tooling.
- Model real problems with data structures, control flow, and algorithmic thinking.
- Build reliable programs with testing, debugging, and logging.
- Work with files, APIs, and data formats (CSV/JSON) in a secure and maintainable way.
- Use Git/GitHub for version control, documentation, packaging, and release artifacts.
- Deliver a portfolio of staged projects mapped to NICE cybersecurity work roles (optional).

How to Use This Course

- Each stage has topic notes, exercises, and a project deliverable.
- Complete the 'Checkpoint' items before moving to the next stage.
- Publish each stage to GitHub using the provided repo structure and templates.

Stage Roadmap

Stage 1 - Foundations and Setup

Suggested pacing: Week 1

Topics

- Installing Python, VS Code, and an isolated environment (venv)

- Running scripts and using the interactive REPL
- Variables, types, strings, numbers, booleans
- Branching (if/elif/else) and loops (for/while)
- Functions and basic error handling (try/except)

Deliverables

- Hello Toolkit: a small command-line program with multiple utilities (calculator, unit converter, text formatter)
- Coding standards: formatting, naming, docstrings, and basic logging

Checkpoints

- Write 5 small functions and call them from a single menu-driven script
- Handle invalid input without crashing

Stage 2 - Data Structures and Problem Solving

Suggested pacing: Week 2-3

Topics

- Lists, tuples, sets, dictionaries
- Iteration patterns, comprehensions, slicing
- Sorting, searching, and basic algorithmic complexity
- Functions that accept/return collections
- Intro to type hints

Deliverables

- Data Toolkit: analyze a small dataset (provided CSV/JSON samples) to compute summaries
- Challenge set: 20 small exercises (arrays, strings, dictionary counting, etc.)

Checkpoints

- Implement clean input parsing and validation
- Write reusable helpers (parse_int, parse_date, normalize_text)

Stage 3 - Files, Automation, and Robust CLI Programs

Suggested pacing: Week 4

Topics

- Reading/writing files safely (pathlib)
- CSV and JSON workflows
- Argument parsing with argparse
- Logging best practices (levels, handlers, rotating files)
- Defensive programming and safe defaults

Deliverables

- Log Analyzer: parse a log file, produce a summary report, export JSON
- Backup/Sync helper: copy files by pattern with a dry-run mode

Checkpoints

- Program supports --help, --input, --output, --verbose flags
- Program exits with meaningful status codes

Stage 4 - APIs, Networking Concepts, and Data Integration

Suggested pacing: Week 5-6

Topics

- HTTP fundamentals and REST patterns
- Using requests (or urllib) with timeouts and retries
- Secrets hygiene: environment variables and .env (no secrets in code)
- Rate limiting and pagination
- Basic data visualization (optional)

Deliverables

- API Client: pull data from a public API (weather, currency, etc.), cache results, and output a report
- Integration mini-project: merge API data with local CSV/JSON

Checkpoints

- All network calls include timeouts and error handling
- No API keys committed to Git

Stage 5 - Testing, Packaging, CI, and Documentation

Suggested pacing: Week 7

Topics

- Unit testing with pytest (or unittest)
- Mocking and testing external I/O
- Packaging with pyproject.toml (setuptools)
- Linting/formatting (ruff/black) (optional)
- GitHub Actions CI pipeline (tests on push/PR)

Deliverables

- Convert one prior stage project into an installable package
- CI pipeline that runs tests and produces artifacts

Checkpoints

- >= 10 unit tests with meaningful assertions
- README includes install, usage, examples, and troubleshooting

Stage 6 - Capstone Portfolio Project

Suggested pacing: Week 8-10

Topics

- Project planning, scope control, and acceptance criteria
- Threat modeling (lightweight) and secure coding review
- Performance and reliability improvements
- Release management: versioning and changelog

Deliverables

- Capstone: 'OpsOps' - a configurable automation + reporting tool (CLI) that ingests logs/data, enriches via API, outputs reports
- Portfolio release: tagged GitHub release with a packaged build and docs

Checkpoints

- Meets acceptance criteria and includes test suite
- Includes user guide, developer guide, and demo run transcript

Assessment Rubric

Category	What 'Good' Looks Like	Weight
Correctness	Program meets requirements; accurate outputs; handles edge cases.	30%
Code Quality	Readable structure; functions; naming; docstrings; minimal duplication.	20%
Reliability	Validation, error handling, logging, and safe defaults.	15%
Testing	Meaningful unit tests; coverage of core logic; mocks for I/O/network.	15%
Documentation	Clear README, install/run steps, examples, assumptions, limitations.	10%
Professionalism	Git hygiene, commits, versioning, and release notes.	10%

NICE Workforce Framework Mapping (Optional)

This mapping is provided as a portfolio-positioning aid. Tailor it to your target roles and job postings.

Stage	Primary Focus	NICE-aligned area (high-level)	Evidence artifacts
Stage 1	Core Python & tooling	Software Development; Data Analysis (foundational)	Code hygiene, environment setup, basic scripts
Stage 2	Data structures & problem solving	Data Analysis; Securely Provision (foundational)	Input validation, handling edge cases
Stage 3	Files, CSV/JSON, automation	Systems Administration; Cyber Defense (support)	Log parsing, backups, automation
Stage 4	APIs, networking basics, CLI tools	Cyber Defense Analysis; Network Services	API clients, HTTP, rate limits, secrets hygiene
Stage 5	Testing, packaging, CI, docs	Software Development; Systems Security	Unit tests, dependency pinning, secure defaults
Stage 6	Capstone & portfolio	Any (tailored)	End-to-end deliverable with README, tests, and release

GitHub Packaging Checklist

1. Create a repository per stage (or a monorepo with /stages).
2. Include: README.md, LICENSE, requirements.txt (or pyproject.toml), and /src or /app folder.
3. Add a /tests folder for Stage 5+ and any project you want to showcase.
4. Tag releases for major milestones (v1.0.0, v1.1.0).
5. Never commit secrets. Use environment variables and an example .env.example file.

Beginner-Friendly SSH Connection Steps (for GitHub)

Windows (PowerShell):

1) ssh-keygen -t ed25519 -C "your_email@example.com"

2) Get public key: type $env:USERPROFILE\.ssh\id_ed25519.pub

3) Add key to GitHub -> Settings -> SSH and GPG keys.

4) Test: ssh -T git@github.com

macOS/Linux (Terminal):

1) ssh-keygen -t ed25519 -C "your_email@example.com"

2) Get public key: cat ~/.ssh/id_ed25519.pub

3) Add key in GitHub settings, then test: ssh -T git@github.com