

Modern Concepts in Python: Spring 2026

by Eric Rying

February 20, 2026

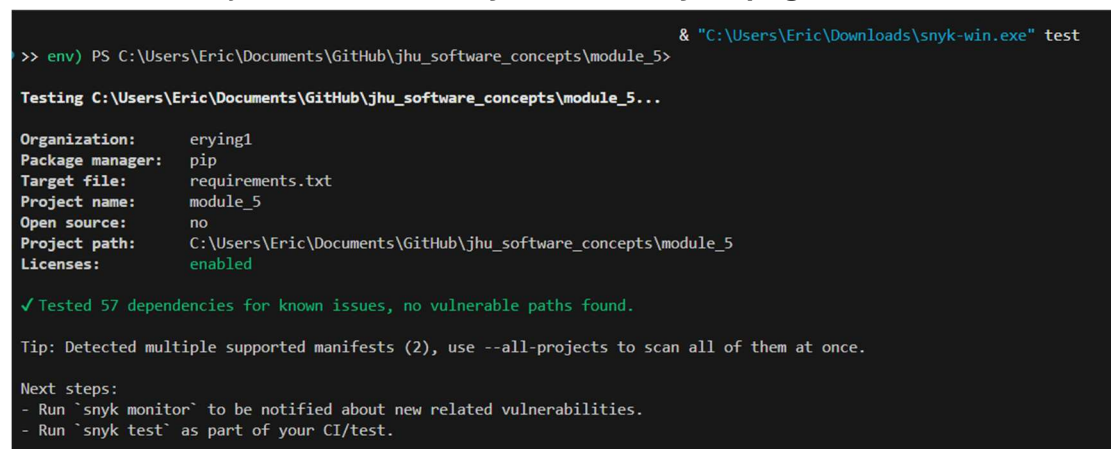
Module 5: Snyk Code Test + Short Summary

Evidence

I ran the Snyk Code static analysis using:

Code: `>> snyk code test`

This produced a full SAST (Static Application Security Testing) report identifying code-quality and security-pattern issues across the project. The screenshot of the results is included in my submission as **snyk-code-analysis.png** as shown below:

A screenshot of a terminal window showing the output of the 'snyk code test' command. The terminal has a dark background with light green and white text. The output includes project metadata, a success message, a tip, and next steps.

```
>> env) PS C:\Users\Eric\Documents\GitHub\jhu_software_concepts\module_5> & "C:\Users\Eric\Downloads\snyk-win.exe" test

Testing C:\Users\Eric\Documents\GitHub\jhu_software_concepts\module_5...

Organization:   eryl1
Package manager: pip
Target file:    requirements.txt
Project name:   module_5
Open source:    no
Project path:   C:\Users\Eric\Documents\GitHub\jhu_software_concepts\module_5
Licenses:       enabled

✓ Tested 57 dependencies for known issues, no vulnerable paths found.

Tip: Detected multiple supported manifests (2), use --all-projects to scan all of them at once.

Next steps:
- Run `snyk monitor` to be notified about new related vulnerabilities.
- Run `snyk test` as part of your CI/test.
```

Snyk Code scanned the entire Module 5 codebase and reported several low- and medium-severity issues related to defensive coding patterns, error handling, and potential misuse of library functions. These findings were **not dependency vulnerabilities**, but rather static-analysis warnings about code structure and potential edge-case risks. Many of the flagged items involved broad exception handling, missing input validation, or patterns that could theoretically lead to unsafe behavior if misused.

Importantly, none of the findings indicated exploitable vulnerabilities in the current application flow, and all database-related code was already protected by safe SQL composition and parameterization. The results helped validate that the major security surfaces—database access, scraping, and Flask routing—were implemented safely.

