Skill Area	Description	Evidence from Units	Current Competence Level	Next Steps for Development
Object-Oriented Programming (OOP)	Understanding and applying OOP principles such as abstraction, encapsulation, inheritance, and polymorphism in Python.	Units 1–6: Implemented Python classes, abstract classes, inheritance, and polymorphism using examples from *Think Python* and class-based projects.	Intermediate	Continue to apply OOP principles in larger, multi-module projects and explore advanced patterns (e.g., dependency injection).
UML Design and System Modelling	Ability to represent system architecture and behaviour using UML diagrams (class, sequence, activity, and state diagrams).	Units 2–4: Created use case, class, and state machine diagrams; used UML to guide program design.	Intermediate	Use UML tools to design more complex systems, linking diagrams directly to code in design documentation.
Software Design Patterns	Understanding of reusable design solutions and how to apply multiple patterns together.	Units 11–12: Implemented Factory, Singleton, and Observer patterns; explored combining creational, structural, and behavioural patterns.	Developing	Practise identifying when to apply each pattern effectively in real-world projects. Study composite and MVC patterns.
Testing and Quality Assurance	Writing unit tests, using linters, and ensuring code style compliance with Python standards.	Units 9–10: Applied unittest, used *pylint* to check style, and followed PEP257 for documentation.	Intermediate	Explore automated testing with pytest and integrate continuous integration (CI) tools.
Debugging and Problem Solving	Identifying, diagnosing, and fixing code issues using Python tools and IDE debuggers.	Unit 7: Practised debugging with *PyCharm* and using linters for static analysis.	Proficient	Continue improving debugging efficiency through more complex projects and profiling tools.
Data Structures and Algorithms	Using and managing data structures (lists, dictionaries, sets) and implementing algorithms such as searches and recursion.	Units 7–8: Implemented data searches, recursion, and set operations; explored algorithmic problem-solving.	Intermediate	Study time complexity and experiment with sorting and graph algorithms to enhance efficiency.
Code Packaging and Documentation	Organising projects into packages and writing professional documentation.	Units 9–10: Packaged Python code, added __init__.py, and followed documentation standards.	Proficient	Create and distribute an open-source project using PyPI or GitHub to apply packaging skills.
Software Sustainability and Maintainability	Understanding efficient, sustainable, and maintainable code design.	Unit 11: Explored sustainable software development and memory management efficiency in Python.	Developing	Explore clean code principles, refactoring techniques, and green computing best practices.
Professional Reflection and Continuous Learning	Reflecting on learning outcomes, identifying strengths, and planning skill development.	Units 12: Reflected on overall progress and linked skills to future goals.	Proficient	Keep a regular learning journal.