LESSON 106

Developing the Computer Model and Simulations

200 minutes (5 periods) | Coursework Project Preparation | Outcome 1.22 | Outcome 2.6 | Outcome 2.7 | Outcome 2.20 | Outcome 2.21

WHAT THIS LESSON DOES

Please note: this module follows the 2026 project brief, whose theme was forests. The State Examinations Commission publishes a new brief each year, with its own theme and its own dates. These lessons will be updated once this year's brief is published. Check the current brief on the SEC website before you begin. In this lesson, you will build a complete Python computer model for forest disaster risk, incorporating real data from your embedded system and additional datasets. You will create two what-if scenario simulations, implement an adaptive feedback mechanism, and visualise results with graphs to meet all advanced requirements.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.22	read, write, test, and modify computer programs
2.6	construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement
2.7	implement algorithms using a programming language to solve a range of problems
2.20	identify and fix/debug warnings and errors in computer code and modify as required
2.21	critically reflect on and identify limitations in completed code and suggest possible improvements

WHAT STUDENTS SHOULD BE ABLE TO DO

- Develop a comprehensive Python-based computer model for simulating forest-related disaster risks, incorporating real and supplementary data sources.
- Design and implement two distinct 'what-if' scenario simulations to evaluate model behaviour under varied environmental conditions.
- Integrate an adaptive feedback mechanism to enable dynamic system responses, either virtually or physically.
- Analyse and visualise simulation outcomes through tables and graphs to compare baseline and scenario results effectively.
- Ensure the model fully complies with specified advanced requirements, producing documented evidence of functionality and adaptability.

BEFORE THE LESSON

- Confirm pandas, matplotlib and numpy install and import on the room machines before class, filters and version clashes often block them here.
- Make sure each student can reach the CSV file their embedded system produced earlier, and have a fallback copy for anyone who lost theirs.
- Have the 2026 brief page 5 and 6 open or printed so students can check the exact advanced requirements and citation rules.
- If any student plans physical feedback via serial, cable up and test one micro:bit link yourself first so you know it works in this room.

Developing the Computer Model and Simulations

HOW THE LESSON RUNS

- Objectives
- Exact Advanced Requirements You Must Meet
- Data You Must Use
- Required Python Libraries
- Core Structure of Your Model File
- Two Types of Adaptive Feedback (Choose One)
- Practical Coding Time: Step-By-Step Tasks
- Example Output You Should See
- Evidence to Save
- Final Checklist, Advanced Requirements Met

TEACHING MOVES

- 2. Exact Advanced Requirements You Must Meet. Read the three advanced requirements aloud and have students tick which of the three their current file already touches. This turns a vague brief into a concrete to-do list before anyone writes code.
- 3. Data You Must Use. Stress the two-source rule: real embedded data plus at least one more dataset. Point at the citation requirement on page 6 now, not later, so sources go into code comments as students add them.
- 4. Required Python Libraries. Have everyone run just the four import lines first and fix any that fail before writing logic. A missing matplotlib caught now saves an hour lost later.
- 5. Core Structure of Your Model File. Get students to paste these section headings in as comments before coding, giving an empty scaffold. It stops them building scenarios before the risk function exists.
- 6. Two Types of Adaptive Feedback (Choose One). Make students commit to Option A or B out loud before coding time. Steer anyone unsure toward the virtual option so feedback is guaranteed working evidence rather than hardware that may not trigger.
- 7. Practical Coding Time: Step-By-Step Tasks. Enforce the order: data loads before the risk function, function before scenarios, scenarios before the graph. Circulate and check each student has real data displaying before they move on.
- 8. Example Output You Should See. Show the sample output so students know what success looks like, but warn that their numbers will differ. The point is three comparable risk values and a triggered response, not matching these figures.
- 9. Evidence to Save. Have students capture the console screenshot and graph the moment their model runs cleanly, before they tweak anything. Evidence is easiest to gather at the working state, not reconstructed later.

WHAT TO WATCH FOR

- Students think two scenarios means running the model twice with the same values, without actually changing a variable. Require each scenario to name the exact variable it changes and by how much, temperature plus three or rainfall minus fifty percent, and show it differs from baseline in the output.
- Students believe a model that produces a graph is therefore correct, without checking the risk numbers make sense. Ask them to predict whether hotter and drier should raise or lower risk, then confirm the numbers move that way. A graph can look right while the logic is inverted.
- Students skip try-except and edge cases, so zero rainfall or a missing CSV row crashes the run during their recording. Have them deliberately test zero rainfall and a missing value now, and wrap the risk calculation so it logs errors instead of stopping.

DIFFERENTIATION

- Emerging: Give the student a working CSV load and a stub risk function so they start from a running file and focus only on adding one scenario. Pair them with a student whose data loads, so environment problems do not eat their whole session.
- Developing: Have them explain in words what their risk function does and why each scenario should change it, before adding the second dataset. Point them to Option A virtual feedback so the adaptive requirement is achievable without hardware dependencies.
- Proficient: Push them toward Option B physical feedback, writing a file or serial command that triggers the embedded system, as stronger project evidence. Ask them to justify their additional dataset choice and citation in comments to Higher Level standard, ready to defend it for the Coursework Project.