THE SMART GREENHOUSE PILOT PROGRAM

Computer Science, Grown in the Greenhouse.

An impact report on the 2025 curriculum pilot for the Hazelwood School District with curriculum materials produced and delivered by Unity Foundation.

Students arrived with no programming experience. Seven cumulative courses later they were building complete data pipelines — reading a file, evaluating every record against a safety threshold, appending a computed status and an audit timestamp, and writing a result a facility manager could open in a spreadsheet.

THE RESULT

7 courses

Structured as a cumulative sequence, each course builds on the previous one by addressing specific challenges encountered in earlier lessons. The design and planned operations of the greenhouse provided the practical problems, while computer science supplied the analytical methods used to solve them.

WHAT THE PILOT DELIVERED

The pilot introduced students to programming using an agricultural context, with the premise that students engage more deeply with computing when the problems it solves are visible and relevant. Since the greenhouse was not yet built, instruction began in the classroom, where students walked through the plans for the growing space before moving to their machines.

Students worked in Python using professional development tools—PyCharm and Visual Studio Code—with the standard library on school-assigned Chromebooks or their own computers. The curriculum relied entirely on open-source, non-proprietary software, ensuring portability and skill transferability.

THE DEFINING CHARACTERISTIC OF THE PILOT

Students learned conditional logic by classifying water pH, iteration by surveying zones across a week, collections by managing sensor arrays, nested structures by mapping zones against observation hours, and file handling by building logs a facility manager could open in a spreadsheet. The computer science stayed rigorous and sequential, and the agricultural context supplied continuous, concrete motivation for concepts that are frequently taught in the abstract.

WHAT STUDENTS BUILT

1 **Fertilizer calculator**
Two floating-point inputs — area and application rate — and one computed total. The first program that fails visibly if input is not converted from text.

2 **Watering scheduler**
Soil moisture, elapsed days and weather weighed together, with rainfall overriding every other rule — an authentic introduction to decision precedence.

3 **Multi-zone survey**
Twelve hourly readings examined against an acceptable range, every exception reported and counted — the first work resembling real instrumentation software.

4 **Harvest tracker**
Parallel lists of daily yields and day names producing a weekly total, a daily average, and the best and worst days — a genuine multi-statistic summary.

5 **Modular data logger**
Composed functions under a main entry point, drawing on the standard library for timestamps, statistics and simulated sensor values.

6 **Data-processing pipeline**
Reads a data file, evaluates each row against a threshold, appends a status column and audit timestamp, and writes a validated new file.

7
courses forming a single dependency chain, each unworkable without its predecessors

7
skill domains documented for every course, from technical to computational thinking

6
working programs in each student portfolio, addressing genuine agricultural problems

2
professional IDEs supported — PyCharm and Visual Studio Code, both free

$0
in software licensing: open-source and freely available tooling throughout

THE SEVEN-COURSE SEQUENCE

Each course resolves a limitation students encountered in the one before it. Read as a whole, the sequence moves from single values to sequences, from sequences to grids, from linear scripts to composed functions, and from volatile memory to persistent storage.

COURSE	FOCUS	CAPABILITY ADDED
One	**Foundations of Computing and the Python Development Environment**	Build a working environment; write, run and debug a first program
Two	**Data Types, User Input, and Conditional Logic**	Accept external values and make decisions about them
Three	**Iteration and Loop Control**	Repeat a process automatically and summarize across repetitions
Four	**Lists and Sequential Data Management**	Manage a data set rather than individual values
Five	**Nested Lists and Two-Dimensional Data Structures**	Represent and operate on tabular, multi-zone data
Six	**Functions, Modular Design, and the Python Standard Library**	Decompose a program into reusable, composable units
Seven	**File Input, Output, and Data Persistence**	Produce records that outlast the program and open in analytical tools

WHAT THE PILOT PROVED

1 **Sequential integrity**
The dependency chain is genuine. Each course requires its predecessors, and the culminating project requires all seven.

2 **Authentic problem framing**
Scenarios describe situations a grower would recognize, carrying the constraints and trade-offs those situations actually hold.

3 **Professional tooling**
Students learned the safe file-handling pattern, the entry-point guard, standard naming and the standard library — not teaching substitutes.

4 **Hard concepts, taught**
Argument-passing semantics, variable scope and resource management addressed directly and concretely rather than deferred.

5 **Transferable structures**
Two-dimensional data demonstrated across academic records, inventories and statistics, so students recognize the pattern as general.

6 **Portability**
Runs on district-standard devices with open-source and freely available software, which keeps replication costs low for adopting districts.

SCOPE STATED PLAINLY

This was a programming curriculum, delivered in software. Physical instrumentation, microcontroller assembly and machine learning were outside the pilot's instructional scope; sensors, zones, irrigation and climate control appear as the subject matter of the problems students solved, using operator-supplied and generated data. That clarity is an asset — it identifies precisely where the next phase of the pathway begins, and it means every outcome stated here can be verified against the work students produced.

WHAT COMES NEXT

Students completing the pilot arrive at subsequent coursework with the software foundation already established. Future courses on physical sensors can concentrate on wiring, calibration, and signal interpretation.A course in data analysis can begin with real analytical technique. A course in automation and control can address actuators and safety interlocks — because students have already written threshold logic, alert generation and emergency-stop conditions in software. For the district, the pilot establishes a validated foundation on which a full Smart Agriculture pathway can be built.

THE UNITY MISSION

Unity Foundation builds community through open technology. The pilot materials close on that identity, and the curriculum honors it: freely available tooling throughout, no licensing cost to replicate, and no institutional access required for a student to keep working at home. Students who complete the sequence hold a portfolio of working software addressing genuine problems — evidence of capability for postsecondary admissions reviewers and prospective employers alike.

The foundation is laid. Build the pathway on it.

On the evidence of the materials reviewed, the pilot achieved what it set out to achieve: computer science can be taught to a high standard through agricultural application, and students will engage with demanding technical content when its purpose is visible.

WEB
unityfoundation.io

EMAIL
info@unityfoundation.io

PHONE
314-579-0066

EIN
501(c)(3) 86-2158056

Unity Foundation™