Lesson 1: The Bell Schedule Hack (Phase 1)

Paragraph 1: A System Out of Sync (*Syntax Error*)

Noah Anderson didn't mind fixing code. In fact, he enjoyed it. Whether it was a broken program or a miswritten command, he saw every error as a puzzle to solve. But when the school's bell schedule malfunctioned, it wasn't a simple fix—it was chaos. Bells rang at the wrong times, some didn't ring at all, and students wandered the halls confused. Teachers blamed a system glitch, but Noah suspected something deeper. He accessed the scheduling code and immediately spotted a syntax error—a misplaced colon in the timing function. It was a small mistake, but one that could have thrown the entire schedule off balance. With a few keystrokes, he corrected it, assuming it was just a careless programming mistake. What he didn't know was that someone had placed it there on purpose.

Paragraph 2: A Timed Disruption (*Integer*)

Across the school, Damian Dalton smirked as he watched the confusion unfold. His plan had been simple—manipulate the bell system's timing by adjusting a few key values. Instead of using the usual integer values for minutes, he had altered them just enough to throw everything off. An integer, a whole number like 10 or 15, dictated when the bells should ring. By shifting these values slightly, he caused classes to start and end at unpredictable moments. But his favorite change? He made lunch last twenty extra minutes by increasing the integer value for that period. Students celebrated, teachers were frustrated, and administrators scrambled for answers. It wasn't enough to break the system entirely—just enough to make people second-guess the schedule. When he saw Noah head toward the office, Damian knew the game had begun.

Paragraph 3: Tracing the Problem (*Float*)

Noah sat at the front office computer, scanning through the bell schedule's logs. Something didn't add up. The system had specific integer values for when the bells should ring, but mixed in with them were unexpected float values—numbers with decimal points. Instead of ringing exactly on time, some bells were set to 11.47 or 2.03, causing small but noticeable delays. A float was typically used for more precise calculations, not something as straightforward as a school schedule. Noah frowned. This wasn't just a simple syntax error—someone had deliberately tampered with the settings. But why? And more importantly, how?

Paragraph 4: The Cover-Up (*Argument*)

Noah adjusted his glasses and began restoring the correct bell times. As he worked, he noticed that some of the changes weren't random—whoever had altered the system had used specific arguments to modify the code controlling the bells. An argument is a value passed into a command to control how it behaves, like telling a robot how many steps to take. In this case, someone had entered new arguments that changed the timing of certain periods. Lunch had been extended by exactly 20 minutes, and some class transitions were shortened to 90 seconds instead of the usual five minutes. Noah quickly replaced the altered arguments with the correct

ones, unaware that someone was watching his every move, waiting to see how long it would take him to fix the mess.

Paragraph 5: Glitch's Takeaway (*Logic Error*)

Damian leaned back in his chair, watching the school day unfold exactly as he had planned. The bell schedule had caused just enough confusion to make students question the system, yet not enough to raise major alarms. But as he reviewed the system logs from his hidden access point, he noticed something unexpected—Noah had fixed everything faster than he anticipated. That wasn't supposed to happen. Damian had introduced what should have looked like a natural logic error—an issue where the code runs but produces unexpected results. It was like writing a program to calculate a student's average grade but mistakenly dividing by the wrong number, leading to an incorrect final score. By adjusting the bell times slightly and making lunch longer, the system technically still functioned, but in a way that disrupted the schedule. Yet Noah had identified the problem too quickly. That meant one thing: someone was watching just as closely as he was.

Paragraph 6: A Hidden Signature (*Dot Notation*)

Noah wasn't satisfied with just fixing the problem—he wanted to know why it had happened. As he scrolled through the bell schedule's settings, something unusual caught his eye. One of the modified values had been updated using dot notation, a method programmers use to access specific properties of an object in a program. Instead of a straightforward time entry, the system showed an override like `schedule.lunchTime = 12.45`. This wasn't how the school normally programmed its schedule. Someone had directly accessed the system's attributes, adjusting them manually instead of using the standard input forms. That was when it clicked—this wasn't just a glitch. Someone had deliberately changed the code.

Paragraph 7: A Pattern Emerges (*Indentation*)

Noah leaned in closer, scanning the modified code. Something else felt off—the indentation wasn't standard. In programming, indentation helps organize code into clear blocks, ensuring that functions and loops execute correctly. Every school system update followed a strict format, with neatly aligned lines of code. But here, the spacing was inconsistent—some lines were indented too much, others too little. It reminded Noah of beginner programmers who didn't yet understand how indentation controlled a program's flow. Whoever had done this wasn't just altering values—they were manually coding inside the system. That meant this wasn't an accident. Someone had hacked in.

Paragraph 8: The First Signature (*Final Reveal for Lesson 1*)

Noah was about to close the system when something unusual caught his eye—a stray line of code buried deep in the scheduling program. It didn't affect the bell times or cause an error, but it stood out because it served no real purpose. Curious, he clicked on it. The line read:

```python
# Glitch was here
```

Noah’s stomach tightened. This wasn’t just a mistake or a system bug—someone had left a signature. Whoever had done this wanted to be noticed. His mind raced through possibilities. No student should have been able to access the bell schedule, and even if someone had tried, they wouldn’t have thought to leave a message in the code. This was intentional.

He exhaled slowly, his fingers hovering over the keyboard. Was this some kind of joke? A harmless prank? Or was someone testing the system—seeing what they could change without being caught? If they could alter the bell schedule, what else could they do? Noah didn’t know who was behind it yet, but one thing was certain: this was just the beginning.

At the bottom of the screen, the message remained.

```python
# Glitch was here
```