Intro:

"Today, we’re tackling a classic interview puzzle in big O notation—different terms for inputs. This topic often confuses candidates, but understanding it can give you a significant edge."

A1:

"Let's revisit some code we've analyzed before: a sequence of two for loops, each originally running 'n' times, simplified to O(n) by dropping constants. But here’s where it gets tricky in interviews."

A2:

"We're going to modify our example by introducing two distinct variables for our loops—let's call them 'a' for the first loop and 'b' for the second. Typically, each loop would be considered O(n) individually. Adding them might misleadingly suggest O(2n), which simplifies back to O(n)."

A3:

"However, this simplification assumes 'a' and 'b' are essentially the same, which isn't always the case. What if 'a' is significantly smaller than 'b', such as one versus a million? Clearly, these loops don’t equate to the same workload."

A4:

"Correctly, we must treat these loops independently: the first loop as O(a) and the second as O(b). Their combined complexity, when they run sequentially and independently, is O(a + b). This expresses that the total time depends distinctly on each loop's workload."

A5:

"Further complicating matters, if these loops were nested instead of sequential, the complexity would multiply, resulting in O(a \* b). This distinction is crucial because it affects how we analyze and predict algorithm performance."

Outro:

"Understanding that we cannot universally replace different input variables with a single 'n' is key to mastering big O notation. This nuanced approach ensures accurate complexity analysis, particularly vital in scenarios where input sizes vary dramatically."