**Logic:**

This algorithm iterates through the string backwards because it's more efficient for finding the last word. By starting at the end, we can update the lastWordLength as soon as we encounter a space or the beginning, guaranteeing we capture the most recent word. Checking for non-space characters and incrementing currentWordLength helps us count the characters within each word.Resetting currentWordLength ensures we start fresh for the next word. Finally, returning lastWordLength provides the desired result.This code effectively utilizes two variables, iterates efficiently, and avoids unnecessary loops, making it a concise and optimized solution for finding the length of the last word in a string.

**Algorithm:**

1) Initialization:

Define two integer variables: lastWordLength and currentWordLength.

Set both variables to 0.

Prompt the user for input and store their string in a variable s.

2) Iterate backwards:

Loop through the string s from the end (index len(s) - 1) to the beginning (index -1) with a decreasing step of 1.

3) Check character:

Inside the loop, check each character at the current index i:

If the character is not a space (not ' '):

Increment currentWordLength by 1. This means we're currently in a word and counting its length.

If the character is a space or the loop reached the beginning (index i == -1):

Update the lastWordLength to the maximum between its current value and the currentWordLength. This ensures we keep the longest word encountered so far.

Reset the currentWordLength back to 0. This signifies the end of a word and prepares to count the next one.

4) Return result:

After iterating through the entire string, return the final value of lastWordLength. This will be the length of the last word encountered.