Documentation

This solution converts an integer to its English word representation by breaking the problem into smaller, manageable parts, focusing on chunks of thousands. Given that the English language has specific terms for certain numbers (like "Hundred," "Thousand," "Million," and "Billion"), the approach here leverages these specific terms to break down any number up to billions into readable words. It handles numbers up to (2^{31} - 1) (2,147,483,647), making it versatile within typical integer bounds.

The solution begins by defining arrays for commonly used words, including numbers from 0 to 19, tens multiples (20, 30, etc.), and large place values like "Thousand," "Million," and "Billion." These arrays serve as a dictionary, allowing the program to match each number with its corresponding word directly. This structure simplifies the conversion process, as it avoids the need for conditional logic to generate words for numbers within these ranges repeatedly.

To translate any given number, the solution uses a helper function that converts segments of the number (up to three digits, or thousands) into words. This function handles different scenarios: numbers below 20, directly mapped; numbers below 100, which require accessing the "tens" array and possibly a remainder; and numbers in the hundreds, which use "Hundred" as a separator. By isolating the task of converting small number segments into words, this helper function ensures that the code remains organized and the logic is clear.

The primary conversion logic iterates over each thousand segments of the number, from lowest to highest. It breaks the number down into chunks of three digits (e.g., converting millions, then thousands, then the remainder), calling the helper function to convert each segment into words. Each segment is then concatenated with the appropriate place value (like "Thousand" or "Million") before being added to the final result. This stepwise approach allows the program to handle large numbers without confusion and ensures that each section of the number is correctly placed within the overall English word representation.

Finally, the solution assembles and trims the full English word representation to produce a readable sentence-like format, removing extra spaces and ensuring that the wording follows standard English conventions. This approach is efficient and modular, with each function handling specific tasks, which improves readability and makes it easy to update or modify parts of the program if needed. By organizing the solution this way, the code not only performs the conversion correctly but also remains adaptable for future improvements or expansions in scope.