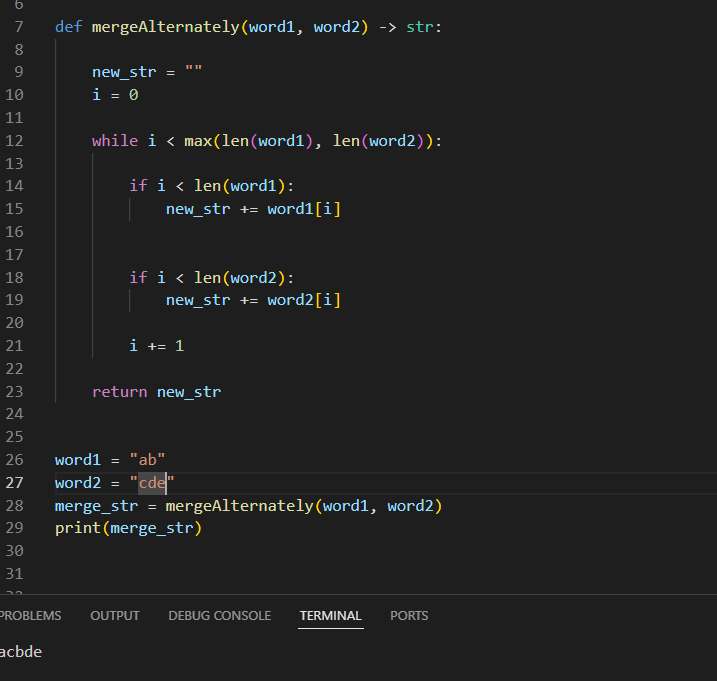
**Problem 1768 . Merge Strings Alternately**

Problem link

https://leetcode.com/problems/merge-strings-alternately/description/?envType=study-plan-v2&envId=leetcode-75

**Final code:**



**LIST AND SET:**

**Remember :**

Set is more faster then list I.e set has Faster Lookups (O(1) Time Complexity)

### 📌 Example:

If you're doing many lookups (like in this problem where you're checking every character), set will perform much faster than a list.

vowels = set('aeiouAEIOU')

Vowels = {‘a’, ‘e’ ….}

#### list:

1. A list in Python is an **ordered collection**.
2. When you do 'i' in my\_list, Python has to **check each element one by one** until it finds a match.
3. Worst-case: it checks **every item** → O(n) time.

#### set:

A set is an **unordered collection with unique elements**, implemented using a **hash table**.

When you do 'i' in my\_set, Python:

1. Computes a **hash** of 'i'.
2. Looks up that hash in constant time.
3. So it's typically **much faster**, especially when the set has many elements.

### Why use float('inf')?

To **initialize a variable with the largest possible value**, so we can find the **smallest** number during iteration.

### 💡 Analogy:

Think of float('inf') like saying:

Start with the biggest number imaginable, so the first real number I see will always be smaller.

Example:   
first = float('inf')

## What is the Fibonacci Series?

The **Fibonacci Series** is a sequence of numbers where:

**Each number is the sum of the two numbers before it.**

F(n) = F(n-1) + F(n-2)

# Dynamic Programming :

**Dynamic Programming (DP)** is a method for solving complex problems by breaking them down into **simpler subproblems** — and **saving the results** of those subproblems to avoid redundant work.

| **Concept** | **Description** |
| --- | --- |
| **Optimal Substructure** | A problem has optimal substructure if its solution can be built from the solutions of its subproblems. |
| **Overlapping Subproblems** | The problem repeatedly solves the same subproblems. |
| **Memoization (Top-Down)** | Solving problems recursively, but caching results to avoid duplicate work. |
| **Tabulation (Bottom-Up)** | Solving problems iteratively from the base case up, storing results in a table (usually an array). |