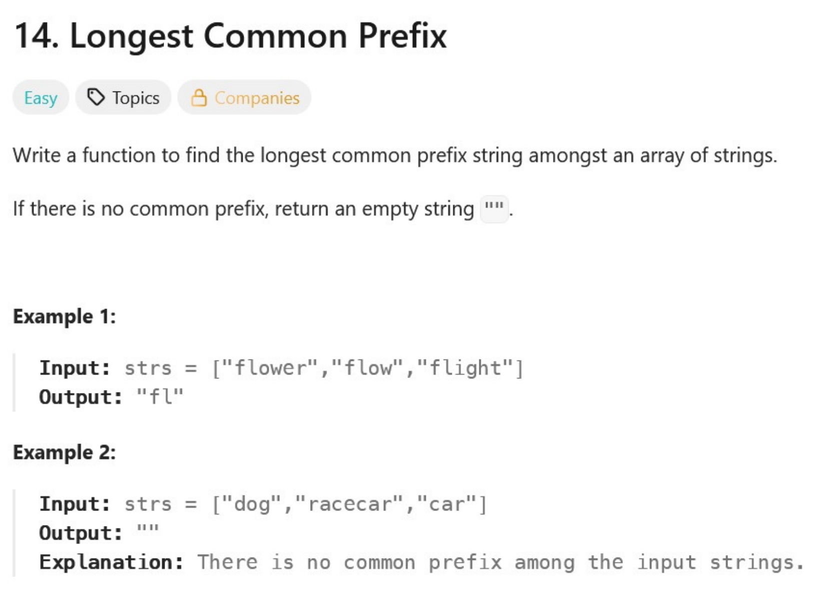


# 14. Longest Common Prefix

Write a function to find the longest common prefix string amongst an array of strings.

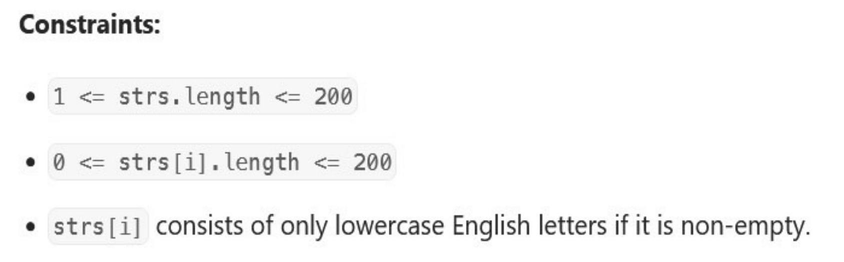
If there is no common prefix, return an empty string `""`.

**Example 1:**

```
Input: strs = ["flower","flow","flight"]
Output: "fl"
```

**Example 2:**

```
Input: strs = ["dog","racecar","car"]
Output: ""
Explanation: There is no common prefix among the input strings.
```

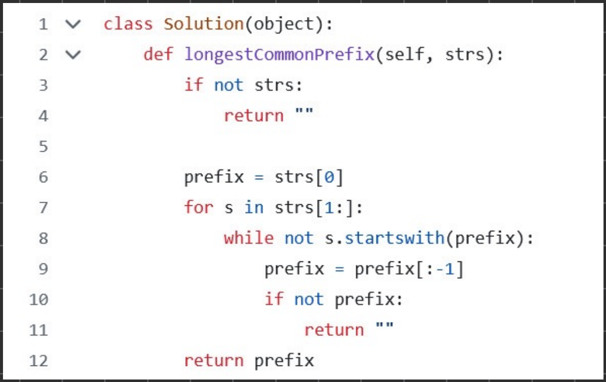


**Constraints:**

- `1 <= strs.length <= 200`
- `0 <= strs[i].length <= 200`
- `strs[i]` consists of only lowercase English letters if it is non-empty.

↳ Mencari awalan yang sama (common)
dari sebuah array of strings

## → CODE



```python
class Solution(object):
    def longestCommonPrefix(self, strs):
        if not strs:
            return ""

        prefix = strs[0]
        for s in strs[1:]:
            while not s.startswith(prefix):
                prefix = prefix[:-1]
                if not prefix:
                    return ""
        return prefix
```

## → Steps:

↱ array of strings

1.) def longestCommonPrefix (self, strs)

nama Fungsi

↓ menunjukkan Object instance dari class Solution {OOP}

2.) if not strs: return " "

↳ cek apakah String kosong ; jika iya return " "

ex: [ ] → return " "

3.) prefix : strs[0]

↳ prefix mulai dari string pertama (index : 0)

ex: ["Flower", "Flow", "Flight"] → prefix : "Flower"

Kenapa ambil yang pertama?

↳ Prefix tidak akan lebih panjang dari string pertama

Nanti akan dipotong jika tidak cocok

↳ paling awal akses aja, prefix pasti pendek even tidak lebih panjang dari String manapun

4.) for s in strs[1:]:

↳ Looping semua string mulai dari String ke-2

Tujuannya untuk membandingkan prefix dengan String-string berikutnya

5.) while not s.startswith(prefix):

↳ Cek apakah string s tidak di awali dengan prefix saat ini

↳ Jika tidak, potong prefix dari belakang satu huruf

6.) prefix = prefix[:-1]

↳ potong 1 karakter dari belakang

7.) if not prefix: return " "

↳ Jika prefix habis → return " "

8.) return prefix

↳ kembalikan hasil akhir prefix

# TRACING

Strs : [ "Flower", "Flow", "Flight" ]

⇓

Prefix : "Flower"

bandingkan "Flower" dengan "Flow"

↓ strs [0] ↓ strs [1]

Flower vs Flow → bukan prefix

→ potong jadi "Flow"

↳ cocok

Bandingkan "Flow" vs "Flight"

"Flow" vs "Flight" → tidak cocok

→ potong jadi "Flo"

→ tidak cocok

→ potong jadi "Fl"

→ cocok

→ hasil : "Fl"

Prefix : 'Flower' {strs [0]}

(1) s : 'Flow'

('Flow').startswith('Flower') → False

prefix : 'Flower' → potong 1 huruf

↳ prefix : 'Flowe'

('Flow').Startswit('Flowe') → False

↳ prefix : 'Flow'

('Flow').startswit('Flow') → True

↳ hasil sementara : Prefix = 'Flow'

(2) s : 'Flight'

→ ('Flight').startswith('Flow') → FALSE

Prefix : 'Flo'

→ ('Flight').Startswith('Flo') → FALSE

Prefix : 'Fl'

→ ('Flight').startswith('Fl') → TRUE

⇓

Hasil akhir : prefix = 'Fl'

Output : 'Fl'