Find index of the first occurence

```
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        if haystack == "":
            return -1

        if needle == "":
            return 0

        search_index = 0
        first_occurence = []



        while search_index < len(needle) - 1:
            for i in haystack:
                if i == needle[search_index]:
                    first_occurence.append(haystack.index(i))
                    search_index += 1
                else:
                    search_index = 0
                    first_occurence = []
        return first_occurence[0]
```

My initial code of the solution, which is still incorrectly implemented, this traverses through the list very slowly and is meant to return the first instance of the first occurence

```
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        # If needle is empty, return 0
        if needle == "":
            return 0

        # If haystack is empty, return -1
        if haystack == "":
            return -1

        # Length of haystack and needle
        h_len = len(haystack)
        n_len = len(needle)

        # Iterate through each starting index in haystack
        for i in range(h_len - n_len + 1):
            # Check if the substring of haystack starting at i r
            if haystack[i:i + n_len] == needle:
                return i

        # If no match is found, return -1
        return -1
```

This would be more optimal as it iterates through range of (h_len - n_len + 1) so say for example

‘sadbutsad’ is your haystack

‘sad’ is your needle

i would only have to range from 0 - 5, this will make sense later as

the if statement splices haystack from [i:i + nlen], its 0 indexed so i = 0

so if i == 0

it would start from 0 to index 3, which covers exactly the correct amount of words

this strongly uses splicing

```
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        for i in range(len(haystack)-len(needle)+1):
            if haystack[i:len(needle)+i] == needle :
                return i
        return -1
```