Index of first occurrence

Given two strings `needle` and `haystack`, return the index of the first occurrence of `needle` in `haystack`, or `-1` if `needle` is not part of `haystack`.

Example 1:

```
Input: haystack = "sadbutsad", needle = "sad"
Output: 0
Explanation: "sad" occurs at index 0 and 6.
The first occurrence is at index 0, so we return 0.
```

Example 2:

```
Input: haystack = "leetcode", needle = "leeto"
Output: -1
Explanation: "leeto" did not occur in "leetcode", so we return -1.
```

We will use the two pointer approach for this like we have with most of our easy problems so our two pointers are given to us in the question `needle` and `haystack` so lets define them as `h` and `n`.

```
# h is the index of the haystack
    # n is the index of the needle
    h = 0
    n = 0
```

next we will define a while loop to keep running whilst our pointer `h` is less than our `haystack` array.

```
# iterate through the haystack
        while h < len(haystack):
```

next our if statement just has a condition if to say if haystack and needle are the same we will enter a while loop with three conditions.

1. n is less than the length of the needle
2. h and n is less than the length of the length of the haystack
3. haystack of h and n = needle[n]

then we will increase n

```
if haystack[h] == needle[0]:
                # iterate through the needle
                # while n is less than the length of needle
                # and h+n is less than the length of haystack
                # and the current element of haystack is equal to the current element o
                # increment n by one
                while n < len(needle) and h + n < len(haystack) and haystack[h + n] ==
                    n += 1
```

we will then have another condition to say if n is equal to the length of the needle than return h

```
# if n is equal to the length of needle
            if n == len(needle):
                return h
```

than in the initial while loop we are always increasing h until we reach the end of the hay stack

```
# increment h by one
            h += 1
```

Complete solution

```
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        # Check if needle is empty
        if not needle:
            return 0

        # Check if needle is longer than haystack
        if needle not in haystack:
            return -1

        # h is the index of the haystack
        # n is the index of the needle
        h = 0
        n = 0
        # iterate through the haystack
        while h < len(haystack):
            # if the current element of haystack is equal to the first element of needle
```

```
            if haystack[h] == needle[0]:
                # iterate through the needle
                # while n is less than the length of needle
                # and h+n is less than the length of haystack
                # and the current element of haystack is equal to the current element o
                # increment n by one
                while n < len(needle) and h + n < len(haystack) and haystack[h + n] ==
                    n += 1

                # if n is equal to the length of needle
                if n == len(needle):
                    return h

            # increment h by one
            h += 1

        return -1
```

feel free to change the `haystack` and `needle` variables to get different values.

```
# Example usage
if __name__ == "__main__":
    # Create an instance of the Solution class
    solution = Solution()

    # Example input
    haystack = "hello" #change
    needle = "ll" #change

    # Call the strStr method
    index = solution.strStr(haystack, needle)
```

```
# Print the result
print("Index of first occurrence of needle in haystack:", index)
```