Find Uppercase Letter in String

In this lesson, you will learn how to find the uppercase letter in a string using both an iterative and recursive approach in Python.

We'll cover the following

In this lesson, given a string, we develop an algorithm to return the first occurring uppercase letter. We will solve this problem using an iterative and recursive approach:

For instance, for the strings:

```
str_1 = "lucidProgramming"
str_2 = "LucidProgramming"
str_3 = "lucidprogramming"
```

The algorithm should return `P` `L`, and output a message indicating that no capital letter was found for `str_1`, `str_2`, and `str_3`, respectively.

Iterative Approach

Let's have a look at the code in Python, which uses the iterative approach:

```
1  def find_uppercase_iterative(input_str):
2      for i in range(len(input_str)):
3          if input_str[i].isupper():
4              return input_str[i]
5      return "No uppercase character found"
```

find_uppercase_iterative(input_str)

The `for` loop on **line 2** runs for all the characters present in `input_str`. By

using the built-in function `isupper()`, every character of `input_str`, i.e., `input_str[i]` is checked if it is uppercase or not. If the condition on **line 3** evaluates to `True` for some `input_str[i]`, then that character is returned from the function on **line 4**. However, if the condition does not evaluate to `True` in any iteration of the `for` loop, `"No uppercase character found"` is returned from the function on **line 5** to indicate that there was no uppercase in `input_str`.

Recursive Approach

The iterative approach was very straightforward. Let's look at the recursive approach in the snippet below:

```
1  def find_uppercase_recursive(input_str, idx=0):
2    if input_str[idx].isupper():
3      return input_str[idx]
4    if idx == len(input_str) - 1:
5      return "No uppercase character found"
6    return find_uppercase_recursive(input_str, idx+1)
```

find_uppercase_recursive(input_str, idx=0)

`find_uppercase_recursive()` takes in `input_str` and `idx` as input parameters. To provide some starting point, the second parameter is written as `idx = 0` which will set `idx` to `0` if no second parameter is provided when the function is called.

The base case is present on **line 2** which returns `input_str[i]` if it is an uppercase. On the other hand, if we reach somewhere in the recursive calls where `idx` is equal to `len(input_str) - 1`, i.e., we have reached the end of the string but didn't find any character which was uppercase. Therefore, we return `"No uppercase character found"` to indicate so. However, if both the conditions on **line 2** and **line 4** are not `True`, we make a recursive call on **line 6** and pass `input_str` and `idx + 1` so that the next character is evaluated.

Let's go ahead and run these two codes in the code widget below.

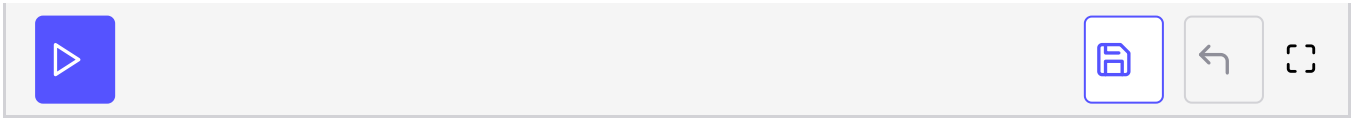
```
def find_uppercase_iterative(input_str):
    for i in range(len(input_str)):
        if input_str[i].isupper():
            return input_str[i]
    return "No uppercase character found"
```

```
def find_uppercase_recursive(input_str, idx=0):
    if input_str[idx].isupper():
        return input_str[idx]
    if idx == len(input_str) - 1:
        return "No uppercase character found"
    return find_uppercase_recursive(input_str, idx+1)

input_str_1 = "lucidProgramming"
input_str_2 = "LucidProgramming"
input_str_3 = "lucidprogramming"

print(find_uppercase_iterative(input_str_1))
print(find_uppercase_iterative(input_str_2))
print(find_uppercase_iterative(input_str_3))

print(find_uppercase_recursive(input_str_1))
print(find_uppercase_recursive(input_str_2))
print(find_uppercase_recursive(input_str_3))
```



That’s it for this problem. Yes, it was that simple! In the next lesson, we will discuss how to calculate the length of a string.