Length of the Longest Consecutive Alphabetical String

```
class Solution:
    def longestContinuousSubstring(self, s: str) -> int:
        count = 1
        count_max = 1

        if s == "":
            return 0

        for i in range(1, len(s)):

            if ord(s[i])-ord(s[i-1]) == 1:
                count += 1
                count_max = max(count, count_max)
            else:
                count = 1

        return count_max
```

```
class Solution:
    def longestContinuousSubstring(self, s: str) -> int:
        alpha_az = "abcdefghijklmnopqrstuvwxyz"
        count = 1
        count_max = 1

        if s == "":
            return 0

        for i in range(1, len(s)):
```

```
            if alpha_az.index(s[i]) == alpha_az.index(s[i - 1])
                count += 1
                if count > count_max:
                    count_max = count
            else:
                count = 1

        return count_max
```

2nd code was significantly longer as it had to search both index of the alphabet, which was both O(n) operation

The first code is more better since converting to ord is O(1) and directly compares the index elements