

Basic String Methods

 coursera.org/learn/python-crash-course/supplement/bP5tN/basic-string-methods

In Python, strings are immutable. This means that they can't be modified. So if we wanted to fix a typo in a string, we can't simply modify the wrong character. We would have to create a new string with the typo corrected. We can also assign a new value to the variable holding our string.

If we aren't sure what the index of our typo is, we can use the string method `index` to locate it and return the index. Let's imagine we have the string **"lions tigers and bears"** in the variable **animals**. We can locate the index that contains the letter **g** using `animals.index("g")`, which will return the index; in this case 8. We can also use substrings to locate the index where the substring begins. `animals.index("bears")` would return 17, since that's the start of the substring. If there's more than one match for a substring, the index method will return the first match. If we try to locate a substring that doesn't exist in the string, we'll receive a **ValueError** explaining that the substring was not found.

We can avoid a `ValueError` by first checking if the substring exists in the string. This can be done using the **in** keyword. We saw this keyword earlier when we covered *for* loops. In this case, it's a conditional that will be either `True` or `False`. If the substring is found in the string, it will be `True`. If the substring is not found in the string, it will be `False`. Using our previous variable **animals**, we can do **"horses" in animals** to check if the substring "horses" is found in our variable. In this case, it would evaluate to `False`, since horses aren't included in our example string. If we did **"tigers" in animals**, we'd get `True`, since this substring is contained in our string.