

# Strings & Text Processing

---

 [codeinplace2021.github.io/pythonreader/en/strings](https://codeinplace2021.github.io/pythonreader/en/strings)

---

## Strings are immutable

---

What does this mean? Let's say we have the string:

```
s = "brahm"
```

Now, let's say that Brahm wants to change the string, so that the 'b' in his name is capitalized, like this: `"Brahm"`

To do that, Brahm might want to write a line of code like this, the same way you would change an element of a list:

```
s[0] = "B"  
#this won't work!
```

But when he does this, Brahm runs into an error message:

```
TypeError: 'str' object does not support item assignment
```

This is a key difference between strings and lists: strings can't be mutated. The only way to change a string variable is to reassign it and make a new string.

So, Brahm can create a new string to capitalize the 'b' in his name:

```
str = "Brahm"
```

Immutability guarantees that string parameters won't change, like in this example below. Click through the slides to see how the strings are stored in the computer's memory:

## String Functions

---

`.upper()` and `.lower()`

---

`.upper()` returns a string with all the letters in upper case, and `.lower()` returns a string with all the letters in lower case.

```
s = "So long and thanks for all the fish"  
>>> s.upper()  
" SO LONG AND THANKS FOR ALL THE FISH "  
  
>>> s.lower()  
" so long and thanks for all the fish "
```

`.replace(char1, char2)`

---

`.replace(char1, char2)` replaces all instances of the first parameter with the second parameter

```
s = "So long and thanks for all the fish"
>>> s.replace("a", "e")
" So long end thenks for ell the fish "

>>> s.replace("s", "")
" o long and thank for all the fih "
```

---

## `.find(char)`

`.find(char)` returns the first index of where the parameter (a character) occurs in the string. If the character is not in the string, the function returns -1.

```
s = "So long and thanks for all the fish"
>>> s.find("n")
5

>>> s.find("x")
-1
```

---

## `.strip()`

`.strip()` removes all the white space at the beginning and end of the string.

```
s = "      So long and thanks for all the fish      "
>>> s.strip()
"So long and thanks for all the fish"
```

---

## `.split()`

`.split` splits the string into a list. If you don't pass any parameters, the string is split by spaces:

```
s = "So long and thanks for all the fish"
>>> s.split()
["So", "long", "and", "thanks", "for", "all", "the", "fish"]
```

If you do pass a parameter, the string will be split by the parameter that you pass. So, if you pass in a comma as a parameter, the string will be separated by commas.

```
names = "Bruce,Diana,Victor,Barry,Clark,Arthur,Hal"
>>> names.split(",")
["Bruce", "Diana", "Victor", "Barry", "Clark", "Arthur", "Hal"]
```

---

## Some more string functions:

```

>>> chant = "wakanda forever"
>>> stop_balrog = "YOU SHALL NOT PASS"
>>> spaces = "   "
>>> number = "42"

>>> chant.startswith("wak")
True
>>> stop_balrog.startswith("you")
False
>>> chant.endswith("ver")
True

>>> chant.title()
"Wakanda Forever"

>>> chant.islower()
True
>>> spaces.isspace()
True
>>> number.isdigit()
True

```

Because strings are immutable, these functions don't change the string and return a new string instead.

## How to Process A String

---

Processing a string involves transforming or inspecting the contents of the string

**If you need both the index (i) and the character (char), use a for i in range loop**

**If you need just the character (char), use a for-each loop**

```

for i in range(len(s)):
    char = s[i]
    # process char

```

```

for char in s:
    # process char

```

## Reversing a string

---

We can use the code we wrote to reverse a string to check if a string is a palindrome! A palindrome is a word that reads the same way from the front and from the back. Here are some examples: kayak, A man, a plan, a canal - Panama! Here's a palindrome in Hindi:

कडक

```

def reverse_string(s):
    reverse = ""
    for ch in s:
        reverse = ch + reverse
    return reverse

def is_palindrome(str):
    normalized = normalize(str)
    rev = reverse_string(normalized)
    return normalized == rev

def normalize(str):
    '''
    This function removes all spaces and punctuation from a string
    '''
    normalized = ''
    for ch in str:
        if ch.isalpha():
            normalized += ch.lower()
    return normalized

```

## Solving a Puzzle

---

Let's return to the story of Bright Simons, who came up with the mPedigree problem. A significant problem people with malaria face is with counterfeit medicine. In 2013, the UN estimated that 700,000 people die each year because they are given counterfeit drugs. Bright Simons wanted to find a way to prevent malaria patients from receiving counterfeit medicine.

Here's how Bright Simon's solution works: Each box of medicine from the manufacturer has a label with a number on it. Someone can then take that number, and text it to mPedigree. If that number comes from a manufacturer, mPedigree will let them know. If that number comes from a counterfeiter who just put a random number on the box, mPedigree can tell the patient to not take the medicine.

You now have all the knowledge needed to implement Bright Simon's solution in Python. How can we write a function that generates these numbers for his solution?