😊 Advanced Strings

1. String objects have a variety of methods we can use to save time and add functionality.
2. Let's start by calling a string s = 'hello world'.

```
[1]: s = 'hello world'
```

3. We can use methods to capitalize the first word of a string or change the case of the entire string.
4. Remember, strings are immutable. None of the below methods change the string in place, they only return modified copies of the original string.

```
[2]: # Capitalize first word in string
     s.capitalize()
```

```
[2]: 'Hello world'
```

```
[3]: s.upper()
```

```
[3]: 'HELLO WORLD'
```

```
[4]: s.lower()
```

```
[4]: 'hello world'
```

5. Now we are going to change the strings for we need to reassign our string.

```
[5]: s
```

```
[5]: 'hello world'
```

To change a string requires reassignment:

```
[6]: s = s.upper()
     s
```

```
[6]: 'HELLO WORLD'
```

```
[7]: s = s.lower()
     s
```

```
[7]: 'hello world'
```

6. Below, we can see that using the count method, we can return the number of occurrences and the starting index position.

```
[9]: s.count('o') # returns the number of occurrences, without overlap

[9]: 2

[10]: s.find('o') # returns the starting index position of the first occurence

[10]: 4
```

7. The center() method allows you to place your string 'centered' between a provided string with a certain length.

```
[11]: s.center(20,'z')

[11]: 'zzzzhello worldzzzzz'
```

The `expandtabs()` method will expand tab notations `\t` into spaces:

```
[12]: 'hello\thi'.expandtabs()

[12]: 'hello   hi'
```

8. We are going to look at some **is check methods**. These various methods below check if the **string is some case**.

```
[13]: s = 'hello'
```

`isalnum()` will return True if all characters in **s** are alphanumeric

```
[14]: s.isalnum()

[14]: True
```

`isalpha()` will return True if all characters in **s** are alphabetic

```
[15]: s.isalpha()

[15]: True
```

`islower()` will return True if all cased characters in **s** are lowercase and there is at least one cased character in **s**, False otherwise.

```
[16]: s.islower()

[16]: True
```

`isspace()` will return True if all characters in **s** are whitespace.

```
[17]: s.isspace()

[17]: False
```

9. **istitle()** will return True if **s** is a title-cased string and there is at least one character in **s**, i.e. uppercase characters may only follow uncased characters and lowercase characters only cased ones. It returns False otherwise.

```
[18]: s.istitle()
```

```
[18]: False
```

`isupper()` will return True if all cased characters in **s** are uppercase and there is at least one cased character in **s**, False otherwise.

```
[19]: s.isupper()
```

```
[19]: False
```

Another method is `endswith()` which is essentially the same as a boolean check on `s[-1]`

```
[20]: s.endswith('o')
```

```
[20]: True
```

10. Strings have some built-in methods that can resemble regular expression operations. We can use split() to split the string at a certain element and return a list of the results. We can use partition() to return a tuple that includes the first occurrence of the separator sandwiched between the first half and the end half.

```
[21]: s.split('e')
```

```
[21]: ['h', 'llo']
```

```
[22]: s.partition('l')
```

```
[22]: ('he', 'l', 'lo')
```