Python course materials

# Advanced Strings

String objects have a variety of methods we can use to save time and add functionality. Let’s explore some of them in this lecture:

s = 'hello world'

## Changing case

We can use methods to capitalize the first word of a string, or change the case of the entire string.

# Capitalize first word in string  
s.capitalize()

'Hello world'

s.upper()

'HELLO WORLD'

s.lower()

'hello world'

Remember, strings are immutable. None of the above methods change the string in place, they only return modified copies of the original string.

s

'hello world'

To change a string requires reassignment:

s = s.upper()  
s

'HELLO WORLD'

s = s.lower()  
s

'hello world'

## Location and Counting

s.count('o') # returns the number of occurrences, without overlap

2

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

4

## Formatting

The center() method allows you to place your string ‘centered’ between a provided string with a certain length. Personally, I’ve never actually used this in code as it seems pretty esoteric…

s.center(20,'z')

'zzzzhello worldzzzzz'

The expandtabs() method will expand tab notations /code> into spaces:

'hello\thi'.expandtabs()

'hello hi'

## is check methods

These various methods below check if the string is some case. Let’s explore them:

s = 'hello'

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

s.isalnum()

True

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

s.isalpha()

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.

s.islower()

True

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

s.isspace()

False

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.

s.istitle()

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.

s.isupper()

False

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

s.endswith('o')

True

## Built-in Reg. Expressions

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.

s.split('e')

['h', 'llo']

s.partition('l')

('he', 'l', 'lo')

Great! You should now feel comfortable using the variety of methods that are built-in string objects!