More on Strings

Some Commonly Used String Methods

count(): Returns the number of times a specified value occurs in a string	
startswith(): Returns true if the string starts with the specified value	
endswith(): Returns true if the string ends with the specified value	Form validation of an email ID
isalpha(): Returns True if all characters in the string are in the alphabet	Usage: Form validation
isdigit(): Returns True if all characters in the string are digits	Usage: Form validation
isspace(): Returns True if all characters in the string are whitespaces	Usage: Form validation
islower(): Returns True if all characters in the string are lower case isupper(): Returns True if all characters in the string are upper case	
lower(): Converts a string into lower case upper(): Converts a string into upper case	Used in palindrome check.
split(): Splits the string at the specified separator, and returns a list	Usage: File processing
splitlines(): Splits the string at line breaks and returns a list	Usage: File processing
strip(): Returns a trimmed version of the string	
zfill(): Fills the string with a specified number of 0 values at the beginning	Left padding a string with 0s (ex. phn no)

Count of each alphabet

x = “A line about Python...”
for i in string.ascii_letters: print(i, x.count(i))

a 39
b 6
c 21
d 11
e 71
f 6
g 15
h 27
i 31
j 1
k 3
l 27
m 6
n 40
o 32
p 10

Count of characters, count of words and count of sentences in a given string

x = "A line about Python String from the book 'Pg 191, Learning Python (O'Reilly, 5e)': Strictly speaking, Python strings are categorized as immutable sequences, meaning that the characters they contain have a left-to-right positional order and that they cannot be changed in place. In fact, strings are the first representative of the larger class of objects called sequences that we will study here. Pay special attention to the sequence operations introduced in this post, because they will work the same on other sequence types we'll explore later, such as lists and tuples. Note: All string methods returns new values. They do not change the original string."

```
print(len(x))
print(len(x.split())) # it by default splits on space
print(len(x.split("."))) # this splits the string on full stop
```

```
658
105
6
```

split()

Date of birth:
23-07-2023 -> extract date or month or year
20/07/2023 -> extract date or month or year

20 Jun 2023 -> extract date or month or year

05.01.2015 -> extract date or month or year
5.1.2015 (do it using date formatting)

Number of ways in which one can retrieve a date from the above values:
Way 1: slice[]
Way 2: split()
Way 3: Date Formatting
d = dateutil.parser.parse("5.1.2015", dayfirst=True)

Find first occurence of ‘that’, and find all occurences of the word ‘that’

```
x.find('that') # This gives you the starting index of first occurence

165

print(x[x.find('that') : x.find('that') + 15])

that the charac

pattern = 'that'
for match in re.finditer(pattern, x):
    s = match.start()
    e = match.end()
    # print('String match "%s" at slice %d:%d' % (x[s:e], s, e))
    print('String match "{}" at slice {}:{}'.format(x[s:e], s, e))

String match "that" at slice 165:169
String match "that" at slice 240:244
String match "that" at slice 372:376
```

Find first occurence of ‘in’, and find all occurences of the word ‘in’

Startswith() and Endswith()

```
# How to check if a phone number is from a particular country?
# Condition for a number to come from a particular country is it’s starting country code
print('+917651179969'.startswith('+91')) # India
print('+917651179969'.startswith('+92')) # Pakistan
print('+17644479969'.startswith('+1')) # US and Canada

True
False
True

# How to check if a person’s DOB is from 2003? Assuming that DOB is following a pattern...

dates_of_birth = ['01/01/2003', '02/01/2004', '07/07/2003',
                  '03/02/2003', '04/03/2004', '05/03/2004']
for i in dates_of_birth:
    if i.endswith('2003'): print(i)
```

Split and SplitLines

```
# split()
string = 'Jack Smith Junior is a good boy'
string.split()

# splitlines()
string2 = """Jack Smith Junior is a good boy
He loves programming"""
string2.splitlines()
```

zfill()

Q: There is some identifier that is incremental in nature and that is supposed to be of certain length. We want to achieve this in a systematic and graceful manner.

```
empid_1 = '1015'
empid_2 = '99234'
empid_3 = '197234'
empids = [empid_1, empid_2, empid_3]

formatted_empids = [i.zfill(7) for i in empids] # List comprehension
formatted_empids
['0001015', '0099234', '0197234']
```

Note About String in Python

A line about Python String from the book "Pg 191, Learning Python (O'Reilly, 5e)":

Strictly speaking, Python strings are categorized as immutable sequences, meaning that the characters they contain have a left-to-right positional order and that they cannot be changed in place. In fact, strings are the first representative of the larger class of objects called sequences that we will study here. Pay special attention to the sequence operations introduced in this post, because they will work the same on other sequence types we'll explore later, such as lists and tuples.

Note: All string methods returns new values. They do not change the original string.