Every Boilermaker Engineer Codes: 101
Entry-Level Programming in Python

Lecture 08a

Dr. John H. Cole
<jhcole@purdue.edu>

PURDUE UNIVERSITY®

COLLEGE OF ENGINEERING

Spring 2021

Part I

Strings

Table of Contents

Accessing Each Character

Each character in a string can be accessed one at a time in a loop.

Terminal

```
>>> for c in 'Spam':
...     print(c)
...
S
p
a
m
>>>
```

Terminal

```
>>> for c in 'Spam':
...     ord(c)
...
83
112
97
109
>>>
```

The builtin function `ord(c)` returns the unicode integer of a single character; `chr(n)` does the inverse.

Strings are Immutable Sequences

Strings are immutable sequences so they can do everything immutable sequences can do.

`s[i], s[i:j:k]`	indexing and slicing
`s1 + s2`	concatenation
`s*n`	repetition
`s1 in s2`	inclusion testing
`len(s)`	measuring
`s1.count(s2)`	counting
`s1.index(s2)`	searching
`min(s), max(s)`	comparing
`sorted(s), reversed(s)`	ordering

For strings `s`, `s1`, `s2`, and indices `i`, `j`, `k`.

For more details, see lecture 7 or read the documentation at docs.python.org/3/library/stdtypes.html.

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

`s.lower()`	returns a copy of s converted to lower case
`s.upper()`	returns a copy of s converted to upper case

Terminal

```
>>> s = 'Spam and Eggs'
>>> s.lower()
'spam and eggs'
>>> s
'Spam and Eggs'
```

Terminal

```
>>> s = 'Spam and Eggs'
>>> s.upper()
'SPAM AND EGGS'
>>> s
'Spam and Eggs'
```

s.title() AND s.capitalize()

`s.title()`	returns a copy of s with the first letter of each word capitalized and the rest lowercased
`s.capitalize()`	returns a copy of s with the first letter of the string capitalized and the rest lowercased

Terminal

```
>>> s = 'Spam and Eggs'
>>> s.title()
'Spam And Eggs'
>>> s
'Spam and Eggs'
```

Terminal

```
>>> s = 'Spam and Eggs'
>>> s.capitalize()
'Spam and eggs'
>>> s
'Spam and Eggs'
```

Checking Status

`s.isspace()`	whitespace (e.g. space, tab, newline)
`s.isupper()`	uppercase letters (e.g. A through Z)
`s.islower()`	lowercase letters (e.g a through z)
`s.isalpha()`	alphabetic characters (e.g. upper and lowercase)
`s.isdigit()`	digits (e.g. 0 through 9)
`s.isalnum()`	alphanumeric (e.g. alphabetic and digits)

- return True if all the characters in s match the condition
- return False if any character doesn't match or there are no characters

Terminal

```
>>> 'SpAM'.isupper()
False
```

Terminal

```
>>> '123'.isdigit()
True
```

s.split()

`s.split()`	returns a list of substrings of s separated at whitespace.
`s.split(sep)`	returns a list of substrings of s separated at sep.

Terminal

```
>>> 'Spam and Eggs'.split()
['Spam', 'and', 'Eggs']
>>> s = 'www.wow.example.com'
>>> s.split('.')
['www', 'wow', 'example', 'com']
>>> 'w'.split('w')
['', '']
>>> s.split('w')
['', '', '', '.', 'o', '.example.com']
```

Splitting Exercise

Use the `split` method to complete this program so that it prints the total number of words in the string.

Editor - split_exercise.py

```python
string = 'one two three four five'


print(f'There are {n} words.')
```

Terminal

```
$ python split_exercise.py
There are 5 words.
```

Splitting Exercise

Use the split method to complete this program so that it prints the total number of words in the string.

Editor - split_exercise.py

```python
string = 'one two three four five'
word_list = string.split()
n = len(word_list)
print(f'There are {n} words.')
```

Terminal

```
$ python split_exercise.py
There are 5 words.
```

s.join(iterable)

`s.join(iterable)` returns the concatenation of the strings in iterable separated by s

- `join` is the "right" way to concatenate strings

Terminal

```
>>> ''.join(['S', 'p', 'a', 'm'])
'Spam'
>>> ', '.join(['Spam', 'Spam', 'egg', 'and Spam'])
'Spam, Spam, egg, and Spam'
>>> l = ['www', 'wow', 'example', 'com']
>>> '.'.join(l)
'www.wow.example.com'
```

Joining Exercise

Use the `join` method to complete this program so that it prints each word in `word_list` on a separate line. Hint: construct a string with a newline character between each word in the list.

Editor - join_exercise.py

```python
word_list = ['one', 'two', 'three', 'four']

print(string)
```

Terminal

```
$ python join_exercise.py
one
two
three
four
```

Joining Exercise

Use the `join` method to complete this program so that it prints each word in `word_list` on a separate line. Hint: construct a string with a newline character between each word in the list.

Editor - join_exercise.py

```python
word_list = ['one', 'two', 'three', 'four']
string = '\n'.join(word_list)
print(string)
```

Terminal

```
$ python join_exercise.py
one
two
three
four
```

s.center(w[, c])

`s.center(w)`	return `s` centered in a string of width `w` padded with spaces
`s.center(w, c)`	return `s` centered in a string of width `w` padded with fill character `c`

- returns `s` if width is less than `len(s)`

Terminal

```
>>> 'Spam'.center(10)
'   Spam   '
>>> 'Spam'.center(10,'=')
'===Spam==='
```

Terminal

```
>>> 'Spam'.center(9,'=')
'===Spam=='
>>> 'Spam'.center(2)
'Spam'
```

`s.ljust(w[, c])` AND `s.rjust(w[, c])`

`s.ljust(w)`	return `s` *left* justified in a string of width `w` padded with spaces
`s.ljust(w, c)`	return `s` *left* justified in a string of width `w` padded with fill character `c`
`s.rjust(w)`	return `s` *right* justified in a string of width `w` padded with spaces
`s.rjust(w, c)`	return `s` *right* justified in a string of width `w` padded with fill character `c`

- returns `s` if width is less than `len(s)`

Terminal

```
>>> 'Spam'.ljust(10)
'Spam      '
>>> 'Spam'.ljust(10,'=')
'Spam======'
```

Terminal

```
>>> 'Spam'.rjust(10)
'      Spam'
>>> 'Spam'.rjust(10,'=')
'======Spam'
```

Padding Example

Editor - justify.py

```python
words = ['spam', 'sausage',
         'bacon', 'egg']

for word in words:
    print(word.center(16))


for word in words:
    print(word.rjust(16))


for word in words:
    print(word.ljust(16))
```

Terminal

```
$ python justify.py
      spam
    sausage
     bacon
      egg
            spam
         sausage
           bacon
             egg
spam
sausage
bacon
egg
```

s.strip([chars])

`s.strip()`	returns a copy of `s` with the leading and trailing whitespace removed
`s.strip(chars)`	returns a copy of `s` with the leading and trailing characters in `chars` removed • removes characters until reaching a character not in `chars`

Terminal

```
>>> s = '    Spam    '
>>> s.strip()
'Spam'
>>> s
'    Spam    '
```

Terminal

```
>>> s = 'www.wow.example.com'
>>> s.strip('cmow.')
'example'
>>> s.strip('cmow')
'.wow.example.'
>>> s.strip('cmow').strip('.')
'wow.example'
```

`s.lstrip([chars])` AND `s.rstrip([chars])`

`s.lstrip([chars])`	returns a copy of s with the *leading* characters in `chars` removed
`s.rstrip([chars])`	returns a copy of s with the *trailing* characters in `chars` removed

- optional argument `chars` defaults to whitespace
- removes characters until reaching a character not in `chars`

Terminal

```
>>> s = '    Spam    '
>>> s.lstrip()
'Spam    '
>>> s.rstrip()
'    Spam'
>>> s
'    Spam    '
```

Terminal

```
>>> s = 'www.wow.example.com'
>>> s.lstrip('cmow.')
'example.com'
>>> s.rstrip('cmow.')
'www.wow.example'
>>> s
'www.wow.example.com'
```

s.startswith(prefix[, i[, j]])

`s.startswith(prefix)`	True if `s` starts with `prefix`
`s.startswith(prefix, i)`	True if `s[i:]` starts with `prefix`
`s.startswith(prefix, i, j)`	True if `s[i:j]` starts with `prefix`

- `prefix` can also be a tuple of prefixes to look for

Terminal

```
>>> s = 'www.wow.example.com'
>>> s.startswith('wow')
False
>>> s.startswith('wow', 4)
True
>>> s.startswith('wow', 4, 7)
True
```

Terminal

```
>>> s
'www.wow.example.com'
>>> s[4:]
'wow.example.com'
>>> s[4:7]
'wow'
```

s.endswith(suffix[, i[, j]])

`s.endswith(suffix)`	True if `s` ends with `suffix`
`s.endswith(suffix, i)`	True if `s[i:]` ends with `suffix`
`s.endswith(suffix, i, j)`	True if `s[i:j]` ends with `suffix`

- `suffix` can also be a tuple of suffixes to look for

Terminal

```
>>> s = 'www.wow.example.com'
>>> s.endswith('com')
True
>>> s.endswith('com', 4)
True
>>> s.endswith('com', 4, 7)
False
```

Terminal

```
>>> s
'www.wow.example.com'
>>> s[4:]
'wow.example.com'
>>> s[4:7]
'wow'
```

s.find(sub[, i[, j]])

`s.find(sub)`	return lowest index where substring sub is found in `s`
`s.find(sub, i)`	return lowest index where substring sub is found in `s[i:]`
`s.find(sub, i, j)`	return lowest index where substring sub is found in `s[i:j]`

- returns -1 if sub is not found
- use `sub in s` if you do not need the index of the substring

Terminal

```
>>> s = 'www.wow.example.com'
>>> s.find('wow')
4
>>> s.find('Wow')
-1
```

Terminal

```
>>> s.find('e')
8
>>> s.find('e', 9)
14
```

s.replace(old, new)

`s.replace(old, new)` return a copy of s with each occurrence of substring `old` replaced by `new`

Terminal

```
>>> s = 'www.wow.example.com'
>>> s.replace('Wow', 'spam')
'www.wow.example.com'
>>> s.replace('wow', 'spam')
'www.spam.example.com'
>>> s.replace('.com', '.gov')
'www.wow.example.gov'
>>> 'abba'.replace('a', 'aba')
'ababbaba'
>>> 'ababbaba'.replace('a', 'aba')
'abababbabababa'
```

Thanks for watching!