ESCI 386 – Scientific Programming, Analysis and Visualization with Python

Lesson 7 - Strings
Indexing Strings

• Strings are indexed just like lists and tuples.

```python
>>> s = 'Halleluiah'
>>> s[0]
'H'
>>> s[3:6]
'lel'
>>> s[-1]
'h'
>>> s[4:]
'eluiah'
```
Finding the Length of a String

• The len() function returns the lengths of a string.
Concatenation

• Strings are concatenated with the + operator.

```python
>>> 'hot' + 'dog'
'hotdog'
```
The join() method

- Strings also have a join() method.

```python
>>> ' '.join(['cat', 'dog'])
'cat dog'
>>> ', '.join(['cat', 'dog'])
'cat, dog'
>>> ' + '.join(['cat', 'dog'])
'cat + dog'
>>> ' and '.join(['cat', 'dog'])
'cat and dog'
```
Multiplications

• Strings can even be multiplied by integers.

```python
>>> 'rabbit '*5
'rabbit rabbit rabbit rabbit rabbit '
```
Methods for Retrieving Information About a String

- `s.count(ss)` – counts the occurrence of a substring `ss`
- `s.endswith(sfx)` – checks to see if string ends with the substring `sfx`
- `s.isalnum()` – checks to see if string contains only numbers and letters
- `s.isalpha()` – checks to see if string is all letters
- `s.isdigit()` – checks to see if string is all numbers
Methods for Retrieving Information About a String

• `s.islower()` – checks to see if string is all lower case

• `s.isspace()` – checks to see if string is all whitespace characters

• `s.istitle()` – checks to see if first letter of every word is capitalized

• `s.isupper()` – checks to see if string is all upper case

• `s.startswith(pfx)` – checks to see if string starts with the substring `pfx`
Searching for Text Within a String

• Strings can be searched for strings and substrings using the find() or index() methods.

• find() returns -1 if the substring is not found.

• index() returns an error if the substring is not found.
Examples

```python
>>> s = 'Hello and goodbye.'
>>> s.find('l')
2
>>> s.index('l')
2
>>> s.find('x')
-1
>>> s.index('x')
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    s.index('x')
ValueError: substring not found
```
Searching a Specified Portion of a String

• The find() and index() methods both search from the beginning of the string, and find the first occurrence of the substring.

• You can also specify a start and ending index to only search a portion of the string.
Searching Backwards Through a String

- The rfind() and rindex() methods work just like find() and index(), except the string is searched backwards from the end of the string.

```python
>>> s = 'Hello and goodbye.'
>>> s.rfind('o')
12
```
Adding, Removing, or Replacing Text

• You can replace/modify substrings within a string using the `replace()` method.

• The `replace()` method does not change the original string. It just creates a modified copy.
Examples

>>> s = 'Halleluiah'
>>> s.replace('l', 'x')
'Haxxexuiah'

>>> s.replace('l', 'xo')
>>> s.replace('l', 'xo')
'Haxoxoexouiah'

>>> s.replace('l', '')
'Haeuiah'

>>> s.replace('l', ' ')
'Ha  e uiah'
Stripping Strings

• `s.strip()` – This removes beginning and trailing whitespace characters, including newline characters. This is very handy when reading data in from files or from the `raw_input` method.

• The `s.lstrip()` and `s.rstrip()` methods do the same except `lstrip()` only removes from the beginning of the string, while `rstrip()` only removes from the end of the string.
Centering Text

• `s.center(w, [pad])` centers the text in a field of width w.

• The optional pad character will be added to the ends of the string.

```python
>>> s = 'Python'
>>> s.center(15)
'     Python    '
>>> s.center(15, '-')
'-----Python-----'
```
Justifying Text

- `s.rjust(w, [pad])` and `s.ljust(w, [pad])` work the same way as `center()`, but do either right or left justification.

```python
>>> s = 'Python'
>>> s.rjust(15)
'     Python'
>>> s.ljust(15,'x')
'Pythonxxxxxxxxxx'
```
Changing the Case of a String

- These methods return copies of the string with the cases of the letters altered:
  - `s.capitalize()` – Makes beginning letter uppercase
  - `s.lower()` – converts the entire string to lowercase.
  - `s.upper()` – converts the entire string to uppercase.
  - `s.swapcase()` – converts all lowercase letters to uppercase and vice versa
  - `s.title()` – converts the entire string to title case (capitalizes first letter of each word).
Splitting Strings

- `s.split(delimiter)` splits a string into pieces based on the delimiter specified.

```python
>>> s = 'cat, dog, mouse, bird'
>>> s.split(',')
['cat', ' dog', ' mouse', ' bird']
>>> s.split('o')
['cat, d', 'g, m', 'use, bird']
>>> s.split(' )
['cat,', 'dog,', 'mouse,', 'bird']
```
Splitting Strings with Multiple Whitespaces

• If a string contains multiple consecutive whitespaces, the split() method can give strange results.

• The returned list will have empty elements.

```python
>>> s = '  This string    has extra  whitespaces.   
>>> s.split(' ') [['', '', 'This', 'string', '', '', '', 'has', 'extra', '', 'whitespaces.', '', '', '']]
```
Solution...

- The re module has methods and functions that use regular expressions.

- We will cover this in depth later, but for now, here is how to handle splitting strings with multiple whitespaces.

>>> import re
>>> s = '  This string    has extra  whitespaces.   '
>>> s = s.strip()
>>> s
'This string    has extra  whitespaces.'
>>> re.split(r'\s+', s)
['This', 'string', 'has', 'extra', 'whitespaces.']

Getting rid of leading and trailing whitespace.