notebook_6_string_processing

March 21, 2018

1 String Processing

1.1 Objectives

1. Use various string methods to transform a body of text.
2. Understand special (escaped) characters.
3. Learn to `split` and `join` strings
4. Process strings with list comprehensions

1.2 Useful string methods

- Use `lower` and `upper` to change case.
- Use `strip`, `rstrip`, and `lstrip` to strip whitespace
- Use `replace` to make changes to the text.

1.3 Changing case

- Strings are case-sensitive.
- Use `lower` to remove case considerations.

```
In [1]: "Hello".lower()
```

```
Out[1]: 'hello'
```

```
In [2]: name = "Todd Iverson"
        name.lower()
```

```
Out[2]: 'todd iverson'
```

1.4 Escaped Characters

- Python uses **escaped characters** for whitespace
- All escaped characters start with \
- Common characters include
  - `"\n"` is *newline*
  - `"\t"` is *tab*, etc.
  - `'\''` and `"\""`
  - `"\\"`

```
In [3]: from string import whitespace
        whitespace

Out[3]: ' \t\n\r\x0b\x0c'
```

1.5 Whitespace - Evaluating versus printing

```
In [4]: "\t"

Out[4]: '\t'

In [5]: print('\t')
```

```
In [6]: a_string = "This string\nhas\nmultiple\nlines"
        a_string

Out[6]: 'This string\nhas\nmultiple\nlines'

In [7]: print(a_string)

This string
has
multiple
lines
```

1.6 Removing whitespace

Since whitespace counts toward string length, we frequently strip it from the ends of a string

```
In [2]: raw_name = "    Todd\n\t\n"
        len(raw_name)

Out[2]: 11

In [3]: raw_name.strip()

Out[3]: 'Todd'
```

1.7 Chaining methods in one expression

- You can chain methods together using dot notation
- Think about the types of each part of the equation

```
In [6]: raw_name.strip().lower()

Out[6]: 'todd'
```

1.8 Exercise 1

1. Use `help` to explore the `replace` method
2. Make an example that chains `replace` with another string method

```
In [ ]:
```

```
In [ ]:
```

1.9 Splitting strings

- Split *cuts* a string into parts
- Returns a list of strings
- split_by character/sequence is removed
  - No argument == split on whitespace

```
In [24]: state = "Mississippi"
         state.split("i")
```

```
Out[24]: ['M', 'ss', 'ss', 'pp', '']
```

```
In [35]: split_str = state.split("ss")
         split_str
```

```
Out[35]: ['Mi', 'i', 'ippi']
```

```
In [37]: quote = '''I know something ain't right.
                    Sweetie, we're crooks. If everything were right, we'd be in jail.'''
         quote.lower().split()
```

```
Out[37]: ['i',
          'know',
          'something',
          "ain't",
          'right.',
          'sweetie,',
          "we're",
          'crooks.',
          'if',
          'everything',
          'were',
          'right,',
          "we'd",
          'be',
          'in',
          'jail.']
```

1.10 Joining strings

- Reverse of split
  - join a list of strings into one string
  - *glue* the characters together with base string

```
In [27]: split_str

Out[27]: ['Mi', 'i', 'ippi']

In [28]: "".join(split_str)

Out[28]: 'Miiippi'

In [29]: "-".join(split_str)

Out[29]: 'Mi-i-ippi'

In [40]: "***".join(split_str)

Out[40]: 'Mi***i***ippi'
```

1.11 Using list comprehensions and join

- List comprehension on a string processes each character
- Join the strings back together after processing

```
In [42]: [ch for ch in "Mississippi"]

Out[42]: ['M', 'i', 's', 's', 'i', 's', 's', 'i', 'p', 'p', 'i']

In [47]: no_vowels = [ch
                      for ch in "Mississippi".lower()
                      if ch not in "aeiou"]
         no_vowels

Out[47]: ['m', 's', 's', 's', 's', 'p', 'p']

In [49]: no_vowels = "".join([ch
                              for ch in "Mississippi".lower()
                              if ch not in "aeiou"])
         no_vowels

Out[49]: 'mssspp'

In [51]: "".join([2*ch for ch in "Mississippi"])

Out[51]: 'MMiissssiissssiippppii'
```

1.11.1 Exercise 2

Write a function that recognizes palindromes.

Hint: Use `all`, `reversed` and `zip`. You may with to read the `help` on `reversed`.

In [ ]:

1.12 Computing statistics

1. Clean up the text
    1. Remove punctuation and other items you want to ignore
2. Use `split` and a list comprehension to change words into their value/worth
3. Use reduction functions like `sum`, `len`, `all`, `any` to compute statistic

```
In [4]: # Note - I removed some punctuation
        quote = '''I know something aint right
                    Sweetie were crooks if everything were right we'd be in jail'''

In [6]: split_lower = quote.split()
        split_lower[:3]

Out[6]: ['I', 'know', 'something']

In [7]: # Split, make lowercase, and map to word length
        lengths = [len(w)
                   for w in split_lower]
        lengths

Out[7]: [1, 4, 9, 4, 5, 7, 4, 6, 2, 10, 4, 5, 4, 2, 2, 4]

In [8]: mean_word_length = sum(lengths)/len(lengths)
        mean_word_length

Out[8]: 4.5625
```

1.12.1 Exercise 3

Write a function that will count the number of vowels in a string.

Hint You will want to write a helper function that takes a character and returns `1` if it is a vowel (a,e,i,o,u) and `0` otherwise

In [ ]: