Chapter8_Exercises

May 29, 2017

1 Chapter 8 Exercises

1.1 Exercise 8.2

There is a string method called count that is similar to the function in Section 8.7. Read the documentation of this method and write an invocation that counts the number of a's in 'banana'.

```
In [1]: word = 'banana'

        count = word.count('a')

        print(count)
```

```
3
```

1.2 Exercise 8.3

A string slice can take a third index that specifies the "step size"; that is, the number of spaces between successive characters. A step size of 2 means every other character; 3 means every third, etc.

A step size of -1 goes through the word backwards, so the slice [::-1] generates a reversed string. Use this idiom to write a one-line version of is_palindrome from Exercise 6.3.

```
In [5]: def is_palindrome(word):
            return word[::-1] == word

        print('The truth is {}'.format(is_palindrome('racecar')))
        print('The truth is {}'.format(is_palindrome('gog')))
        print('The truth is {}'.format(is_palindrome('something something European
```

```
The truth is True
The truth is True
The truth is False
```

1.3 Exercise 8.4

The following functions are all intended to check whether a string contains any lowercase letters, but at least some of them are wrong. For each function, describe what the function actually does (assuming that the parameter is a string).

```
In [12]: def any_lowercase1(s):
             for c in s:
                 if c.islower():
                     return True
                 else:
                     return False
```

Returns true if there are any lowercase characters in the string s, false otherwise

```
In [16]: def any_lowercase2(s):
                 for c in s:
                     if 'c'.islower():
                         return 'True'
                     else:
                         return 'False'

         print(any_lowercase2('A'))
```

```
True
```

Always returns true (unless None)

```
In [23]: def any_lowercase3(s):
             for c in s:
                 flag = c.islower()
             return flag
```

Returns true if the last character in the string s is lowercase, false otherwise

```
In [24]: def any_lowercase4(s):
             flag = False
             for c in s:
                 flag = flag or c.islower()
             return flag
```

Returns true if there are any lowercase characters in the string s, false otherwise

```
In [25]: def any_lowercase5(s):
             for c in s:
                 if not c.islower():
                     return False
             return True
```

Returns true if there are no uppercase characters, false otherwise

1.4 Exercise 8.5

A Caesar cypher is a weak form of encryption that involves "rotating" each letter by a fixed number of places. To rotate a letter means to shift it through the alphabet, wrapping around to the beginning if necessary, so 'A' rotated by 3 is 'D' and 'Z' rotated by 1 is 'A'. To rotate a word, rotate each letter by the same amount. For example, "cheer" rotated by 7 is "jolly" and "melon" rotated by -10 is "cubed". In the movie 2001: A Space Odyssey, the ship computer is called HAL, which is IBM rotated by -1. Write a function called rotate_word that takes a string and an integer as parameters, and returns a new string that contains the letters from the original string rotated by the given amount. You might want to use the built-in function ord, which converts a character to a numeric code, and chr, which converts numeric codes to characters. Letters of the alphabet are encoded in alphabetical order.

```
In [60]: def rotate_word(s, rot):
             new = ''
             for c in s:
                 if c != ' ':
                     term = ord(c) + rot
                     if c.islower() and term < 97:
                         term = 122 - (96 - term)
                     if c.islower() and term > 122:
                         term = 96 + term
                     if c.isupper() and term < 64:
                         term = 90 - (64 - term)
                     if c.isupper() and term > 90:
                         term = 64 + term
                     new += chr(term)
                 else:
                     new += ' '
             return new

         print(rotate_word('HAL',1))
         print(rotate_word('Cheer',7))
         print(rotate_word('melon',-10))
         print(rotate_word('Hello there',3))
```

```
IBM
Jolly
cubed
Khoor wkhuh
```