Assignment 9

More About Strings

LILLEBAELT ACADEMY OF
PROFESSIONAL HIGHER EDUCATION

Author
Martin Grønholdt
mart80c7@edu.eal.dk

Sunday 22 January 2017

Table of Contents

Introduction

The programs in this hand-in is about list and their use.

All files for this hand in are available at:
https://github.com/deadbok/eal_programming/tree/master/Assignment 9

Error handling

All programs that requests user input, handle bad input by asking the user, to use only the correct data type, where after it exits.

```
Enter the amount of a purchase: 2hjjhg

Please use only numbers.
```

Example output of a program when the user enters an incorrect value

All programs using file I/O will show an error message if something goes wrong during file access.

2. Sum of Digits in a String

This program calculate the sum of a string of digits.

prog2.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
'''
Name: Program 2 "Sum of Digits in a String"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-01-22)

A program that ask for a string non separated of numbers, and sums of each
digit of the input.
'''
def get_numbers():
    """
    Get a string of numbers from the user.

    :return: A list all numbers.
    """
    # List for the numbers.

    numbers = list()

    number_str = input('Input a string of non-separated digits: ')

    try:
        #Convert each character to an int, fail and complain if something
        #cannot be converted.
        for ch in number_str:
            numbers.append(int(ch))
    except ValueError:
        # Complain when something unexpected was entered.
        print('\n"{}" is not a number. Please use only numbers.'.format(ch))
        exit(1)

    return numbers

def main():
    '''
    Program main entry point.
    '''

    numbers = get_numbers()
    print("\nThe numbers are: {}".format(str(numbers).strip('[]')))
    print("The sum is: {:0.2f}".format(sum(numbers)))

# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result

```
Input a string of non-separated digits: 123456789

The numbers are: 1, 2, 3, 4, 5, 6, 7, 8, 9
The sum is: 45.00
```

Output of the program.

```
Input a string of non-separated digits: 5/*56

/ is not a number. Please use only numbers.
```

Output of the program when wrong input is given.

3. Date Printer

Program to convert from one date format to another. It uses the datatime package for the conversion, which also validates the input.

prog3.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
'''
Name: Program 1 "Date Printer"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-01-22)

A program that reads a date in the form mm/dd/yyyy from the user and reformat
it to the form March 12, 2012.
'''
from datetime import datetime

def get_date():
    """
    Get a date of the form mm/dd/yyyy, complain if input is wrong.

    :return: A datetime object.
    """
    date_str = input('Input a date in the form mm/dd/yyyy: ')

    # Do sanity check of the input.
    try:
        # Use the datetime object function strptime to from correct form,
        # gives us free validation of the input.
        date = datetime.strptime(date_str, '%m/%d/%Y')
    except ValueError:
        # Complain when something unexpected was entered.
        print('\nPlease enter a date of the form mm/dd/yyyy.')
        exit(1)

    return date

def main():
    '''
    Main entry point.
    '''
    date = get_date()
    # Use the datetime object function strftime to convert to the correct form.
    print('\nThe date is: {}'.format(date.strftime('%B %d, %Y')))


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result

```
Input a date in the form mm/dd/yyyy: 12/20/1978

The date is: December 20, 1978
```

Console output of the program.

```
Input a date in the form mm/dd/yyyy: 20/12/1978

Please enter a date of the form mm/dd/yyyy.
```

Output of the program when wrong input is given.

6. Average Number of Words

This program reads a number of sentences from a file, calculates the average word count for all sentences.

prog6.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 4 "Average Number of Words"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-01-22)

Read a file with one sentence per line, and calculates the average number of
words per sentence.
"""


def load_sentences(filename='textErrorList.txt'):
    """
    Load sentences from a file separated by lines.

    :return: List of sentences.
    """
    print('Loading text data...')
    # List of sentences
    sentences = list()
    # Load the file
    try:
        with open(filename, 'r') as data_file:
            for line in data_file:
                # Strip line endings
                sentences.append(line.strip())
    except IOError as ex:
        # Complain when something goes wrong with the file access.
        print('Exception: {}'.format(str(ex)))
        print('Error loading text.')
        exit(1)
    print('OK\n')
    return (sentences)


def count_words(sentences):
    """
    Count words in a list of sentences separated by space.

    :param sentences: List of sentences.
    :return: Number of words.
    """
    # Count all words and print what is going on.
    i = 1
    total_n_words = 0
    for sentence in sentences:
        # Split word by spaces
        words = sentence.split()
        n_words = len(words)
```

```
        total_n_words += n_words
        print('Line {:2}, {:2} words:\t{}'.format(i, n_words,
                                                  str(words).strip("[]")))
        i += 1
    return (total_n_words)


def main():
    """
    Program main entry point.
    """
    sentences = load_sentences()
    n_words = count_words(sentences)
    # Print result.
    print('\nAverage number of words per sentence: {:.2f}'.format(
        n_words / len(sentences)))


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result

```
Loading text data...
OK

Line  1,  3 words:  '#0', 'No', 'Error'
Line  2,  3 words:  '#100', 'System', 'Error'
Line  3,  4 words:  '#101', 'Not', 'Enough', 'Memory'
Line  4,  3 words:  '#102', 'Memory', 'Error'
Line  5,  4 words:  '#200', 'Not', 'An', 'Image'
…
Line 78,  5 words:  '#3001', 'ROI:', 'Not', 'a', 'polygon'
Line 79,  8 words:  '#3002', 'LCD:', 'string', 'read', 'is', 'not', 'a', 'number'
Line 80,  6 words:  '#3003', 'Barcode:', 'check', 'character', 'not', 'valid'
Line 81,  4 words:  '#3004', 'Meter:', 'parallel', 'lines'
Line 82,  6 words:  '#3005', 'Browser:', 'Not', 'a', 'Browser', 'image.'

Average number of words per sentence: 5.18
```

Output of the program (truncated).

```
Loading text data...
Exception: [Errno 2] No such file or directory: 'textErrorList.txt'
Error loading text.
```

Output when the programs fails reading the data file.

10. Most frequent character.

Display most frequent character in a string. I have chosen to output the additional characters, when the 1st place is shared.

Prog10.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 9 "Most Frequent Character"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-01-22)

Display the most frequent character in a string entered by the user.
"""


def ch_freqency(data):
    """
    Count the occurrence of each character in the data, and return the count in
    a dictionary.

    :param data: Input string
    :return: List of tuples (character, count).
    """
    # Dictionary to keep count
    counts = dict()
    # Strip whitespaces and lowercase.
    data = ''.join(data.split()).lower()
    # Run through all characters.
    for ch in data:
        # Create a new entry if it is not there, else add.
        if ch not in counts.keys():
            counts[ch] = 1
        else:
            counts[ch] += 1

    # The final list.
    ret = list()
    # Sort the dictionary entries by value, and them to a list as tuples.
    for w in sorted(counts, key=counts.get, reverse=True):
        ret.append((w, counts[w]))

    return ret


def main():
    '''
    Program main entry point.
    '''
    # Get string from user.
    user_str = input('Input a string: ')

    # Count the characters
    counts = ch_freqency(user_str)
```

```
    # Get the highest count from the sorted list.
    max = counts[0][1]
    # List of other characters with max count.
    other_ch = list()
    # Just a counter.
    i = 1
    # Current entry
    count = counts[i]
    # Find all other characters that occur as many times as the most occurring.
    while count[1] == max:
        other_ch.append(count[0])
        i += 1
        count = counts[i]

    # Print the result.
    print('Most frequent character "{}" occurs {} times'.format(counts[0][0],
                                                                counts[0][1]),
          end='')

    # Print contenders or end the line.
    if len(other_ch):
        print(' (characters that occur the same number of times: {}).'.format(
            str(other_ch).strip("[]")))
    else:
        print('.')


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result

```
Input a string: Martin Bo Kristensen Grønholdt
Most frequent character "n" occurs 4 times.
```

Output of the program with a clear winner.

```
Input a string: This is the last exercise.
Most frequent character "s" occurs 4 times (characters that occur the same
number of times: 'e').
```

Output of the program with multiple contestants.

Conclusion

In this hand in the beauty of Python is mostly in the seemingly little things, like using the datetime object, printing lists by stripping off unwanted characters from the string representation.