

File I/O and text processing

Introduction

In this assignment we will learn about navigating the file system, file I/O, and efficient use of lists and dictionaries.

This assignment is broken up into three tasks.

1. Write a function `getfilelist()` that takes a pathname as input and returns a list of **.txt** files thereunder.
2. Write a function `getwordfreqs()` that takes a filename as input and returns a dictionary of words and their frequencies.
3. Write a function `getcommonwords()` that takes as input a list of dictionaries and returns a list of of the most frequent words common to all dictionaries.

Reading material

Chapters 1-6, 11 in Beginning Python

Test harness and data

The submission system will test your implementation on a set of test data. To facilitate your coding, you have access to **part** of this test data as well as a test harness. Download these below:

- books_student.zip (file-io/books_student.zip)
- a4_test.py (file-io/a4_test.py)

Unzip the test data to a folder on your own PC (**making sure to extract pathnames!**), and test your code using it. For example, on a Linux system, unzip and run your code on the command line:

```
$ mkdir assignment_4
$ cd assignment_4
$ unzip /path/to/books_student.zip
$ ls
books  a4_test.py
$ ls books
folder_00  folder_01  folder_02  folder_03  folder_04
$ python3 a4_test.py books
```

Submission

You can submit one answer at a time for testing, but for your final submission you must submit all three parts together. The submission system uses Python 3, so it's best to test using this, but anything above Python 2.7 should work fine.

Question 1: Listing files and folders (30%)

The Python OS (<https://docs.python.org/3.5/library/os.html>) module handles access to the underlying operating system, among other things. This makes it possible to write platform-independent code without having to think too much about support for all possible operating systems.

Use the OS module in this subtask. There are many ways to solve the problem. Think about efficiency, and read the documentation carefully so as not to reimplement existing functionality!

Traverse the folder hierarchy below the specified pathname and return a list where each element is path to a **.txt** file. Other file types must not be returned in this list!

For example, given the following structure:

```
books
\
|---- folder_01
|       \---- 11-0.txt
|              74-0.txt
|---- folder_00
|       \---- pg27827.txt
|              pg5200.txt
|---- folder_02
|       \---- 76-0.txt
|              84-0.txt
\---- folder_03
       \---- 219-0.txt
              98-0.txt
```

your function should return the following sorted list:

```
['./books/folder_00/pg27827.txt', './books/folder_00/pg5200.txt', './books/folde
```

Your output will be dependent on whether you run Microsoft Windows, MacOS, or Linux, the only constraint is that each item in the returned list must be a relative path to a filename that can be passed directly to Python's `open()` function call and succeed. **NB: Return the list, don't just print the list to standard out!**

Name your function `getfilelist()` with a string as an argument, and returning the aforementioned list. This method will be called from our test harness, so it must be defined correctly.

1

Question 2: Dictionary of word frequencies (35%)

A Python dictionary (<https://docs.python.org/3.5/library/stdtypes.html#mapping-types-dict>) is a data structure that maps hashable values (a key) to objects (a value). We want to use a dictionary to create a list of word frequencies from a given text file. The dictionary should have the format `key: value` where key is the word and value is the frequency.

Python's `collections.Counter`

(<https://docs.python.org/3.5/library/collections.html#collections.Counter>) module is a useful place to start here. `collections.Counter` is available in both Python 2.7 and Python 3.

Consider the text from *The Metamorphosis* by Franz Kafka:

The first thing he wanted to do was to get up in peace without being disturbed, to get dressed, and most of all to have his breakfast. Only then would he consider what to do next, as he was well aware that he would not bring his thoughts to any sensible conclusions by lying in bed. He remembered that he had often felt a slight pain in bed, perhaps caused by lying awkwardly, but that had always turned out to be pure imagination and he wondered how his imaginings would slowly resolve themselves today. He did not have the slightest doubt that the change in his voice was nothing more than the first sign of a serious cold, which was an occupational hazard for travelling salesmen.

If we remove punctuation, convert all words to lower-case, we have the following word frequencies:

```
8 he
7 to
4 was
4 the
4 that
4 in
4 his
3 would
2 of
2 not
2 lying
2 have
2 had
2 get
2 first
2 do
2 by
2 bed
2 and
2 a
1 wondered
1 without
1 which
1 what
1 well
```

Write the function `getwordfreqs()` which takes as input a pathname and returns a dictionary with words as keys and frequencies as values.

Be careful with the text processing. Remember to:

1. Only consider words consisting of alphanumeric characters
2. Convert all text to lower-case before counting
3. If you have issues with character encodings, do a little searching for a solution. E.g.: here (<https://stackoverflow.com/questions/9233027/unicodedecodeerror-charmap-codec-cant-decode-byte-x-in-position-y-character>).

Also, don't forget to **close your file!**

1

Question 3: Most frequent common words shared between dictionaries (35%)

For this part you'll have to get your hands dirty with lists and dictionaries.

Consider the three dictionaries with word frequencies sorted in descending order:

words1	words2	words3
-----	-----	-----
big: 5	small: 7	small: 10
small: 5	little: 5	big: 8
tiny: 5	tiny: 4	tiny: 9
little: 3	big: 1	little: 7
huge: 3	normal: 1	huge: 7
normal: 1	huge: 1	normal: 1

If we consider common words in the top 3 of each list, we have the result:

small, tiny

In this final part, you are to write the function `getcommonwords()` which takes a list of dictionaries and returns a list of words which are common to each dictionary, and within the top 10 words of that dictionary. Put simply, if the dictionary is sorted in descending order, consider only the first 10 entries.

1


Submit

Information

Author(s)	Donn Morrison
Deadline	25/09/2018 23:59:00
Status	Not yet attempted
Grade	0.0%
Grading weight	1.0
Attempts	0
Submission limit	No limitation

Submitting as

➤ **Nadine Obwaller**

 Classroom : Default classroom
(/aggregation/IFUD1056-IINI4014)

For evaluation

 Best submission

 No submission

Submission history

No submission