```
""" Use this file to develop and test your Assignment 4 functions S1, 2020."""


Note 1: for questions 1 and 6 of the assignment you will need to make sure you have
downloaded the input text files from the Assignment website. Make sure that the
input files are inside the same folder as your programs.


Note 2:  Two helper functions
Several of the testing codes for the functions in this assignment make use of the
print_dict_in_key_order(a_dict) function which prints dictionary key:value pairs in
sorted key order.

The testing code for function 4 in this assignment makes use of the
remove_less_than_2(a_dict) function which removes any key:value pairs which have a
corresponding value of 1 from the dictionary.

#---------------------
# Two helper functions
#---------------------
def print_dict_in_key_order(a_dict):
    all_keys = list(a_dict.keys())
    all_keys.sort()
    for key in all_keys:
        print(key, ":", a_dict[key])

def remove_less_than_2(a_dict):
    all_keys = list(a_dict.keys())
    for key in all_keys:
        if a_dict[key] == 1:
            del a_dict[key]

#----------------------------------------------------
# 1111111111111111111111111111111111111
# get_dictionary_from_file()
#----------------------------------------------------
"""
Define the get_dictionary_from_file(filename) function which is passed a filename
as a parameter. Each line of the file contains a word followed by " : " followed by
the meaning of the word. A " : " always separates the word from its meaning. An
example file content is:

    allegator : someone who alleges.
    ecdysiast : an exotic dancer, a stripper.
    eructation : a burp, belch.
    lickety-split : as fast as possible.
    lickspittle : a servile person, a toady.

The function returns a dictionary where each word is the key and the corresponding
value is the meaning.

Note: the keys and their corresponding values should not contain any leading or
trailing spaces (use the strip() method).

For example:

    the_dict = get_dictionary_from_file("WordsAndMeanings1.txt")
    for word in ["lickspittle", "allegator", "lickety-split"]:
        if word in the_dict:
            print(word, "=", the_dict[word])

prints:
```

```
lickspittle = a servile person, a toady.
allegator = some who alleges.
lickety-split = as fast as possible.
"""


#----------------------------------------------------
# 2222222222222222222222222222222222
# print_rows()
#----------------------------------------------------
"""
Define the print_rows(row_dict) function which is passed a Python dictionary as a
parameter. The keys of the dictionary are single letters (lowercase or uppercase)
and the corresponding values are tuples containing two integers, e.g. {'a': (4, 3),
'c': (5, 0), 'b': (-2, 5)}. The function prints ONE line of text. For each
key:value pair in the dictionary, the function prints the key in uppercase
surrounded by stars. The number of stars on either side of the key is given by the
smaller of the numbers in the tuple and the number of times the key is printed is
given by the larger of the numbers in the tuple, e.g. the key:value pair 's':(3, 2)
would print the string "**SSS**".




Notes
• The keys must be printed in alphabetical order (use the sort() method to sort the
list of keys).
• If either of the numbers in the corresponding tuple is less than 1 then nothing
is printed for that key:value pair.

For example:

    print_rows({'a': (4, 3), 'c': (5, 0), 'b': (-2, 5)})
    print_rows({'d': (12, -3), 'c': (1, 2), 'b': (3, -4), 'f': (11, 6)})

prints:

***AAAA***
*CC*******FFFFFFFFFFF******
"""


#----------------------------------------------------
#3333333333333333333333333333333333333333333333333333
# print_highest_frequency_keys()
#----------------------------------------------------
"""
Define the print_highest_frequency_keys(frequency_dict, key_length)
function which is passed two parameters:

    • frequency_dict: a dictionary containing words and their corresponding
frequency, e.g.
                    {"and":15, "tiger":7, "frog":1, "cat":15, "tests":2,
"dog":2, "bat":14, "rat":15, "talon":7}
    • key_length: an integer, the length of the dictionary keys to be considered.

The function first prints the key_length parameter followed by the string, " letter
keys: ". The function then considers only the keys in the dictionary which have a
length given by, key_length, and, of those keys, only the ones with the highest
frequency are printed in sorted order. Finally the highest frequency is printed.

For example, using the example dictionary above, if key_length is 5 then only keys
with 5 letters are considered, i.e. {"tiger":7, "tests":2, "talon":7} , and of
those, only the keys "talon" and "tiger" have the highest frequency of 7, therefore
```

the function prints:

                    5 letter keys: ['tiger', 'talon'] 7

For example:

        word_frequencies = {"and":15, "tiger":7, "frog":1, "cat":15,
                "tests":2, "dog":2, "bat":14, "rat":15, "talon":7}
        print_highest_frequency_keys(word_frequencies, 5)

prints:

5 letter keys: ['talon', 'tiger'] 7
"""

#--
44444444444444444444444444444444444444444444444
get_last_three_letters_dict(text)
#--
"""
Define the get_last_three_letters_dict(text) function which is passed a string as a
parameter. The function first converts the parameter string to lower case and then
returns a dictionary object which has:

    • keys which are the last three letters of any of the words in the text
parameter which have a length greater than 2,
    • corresponding values which are the number of words in the text parameter
which end with these last three letters.

Notes
•   The testing code removes any key:value pairs in the returned dictionary where
the last three letters occur just the once.
•   The testing code uses the print_dict_in_key_order(a_dict) function to print the
key:value pairs of the returned dictionary in alphabetical key order.

For example:

        text = 'nubile singer linger finger juvenile tiger sing turnstile mobile tile'
        a_dict = get_last_three_letters_dict(text)
        remove_less_than_2(a_dict)
        print_dict_in_key_order(a_dict)

prints:

ger : 4
ile : 5
"""

#--
5555555555555555555555555555555555555555555555555
remove_nines()
#--
"""
Define the remove_nines(numbers_dict) function which is passed a dictionary as a
parameter. The dictionary object has:

    • keys which are strings,
    • corresponding values which are lists of integers.

The function removes any integers from the corresponding values which contain the
digit 9.

Note: the testing code uses the print_dict_in_key_order(a_dict) function to print

the key:value pairs of the returned dictionary in alphabetical key order.

For example:

```
    word_numbers_dict = {"fish": [9, 89, 76],   "rat":[2, 891, 4], 'dog' : []}
    remove_nines(word_numbers_dict)
    print_dict_in_key_order(word_numbers_dict)
```

prints:

```
dog : []
fish : [76]
rat : [2, 4]
"""
```

```
#-------------------------------------------------------
# 6666666666666666666666666666666666666666666666666666
# get_best_2_marks_dict()
#------------------------------
"""
```

Define the get_best_2_marks_dict(filename) function which is passed a filename as a
parameter. Each line of the file contains a name followed by at least two numbers
all separated by any number of spaces.
An example file ("NamesMarks1.txt") content is:

```
    Ralph     65     73
    Fancy  74  80 51 69   62
    Carolin  80   70   54   90
```

The function returns a dictionary of key:value pairs where each name is the key and
the corresponding value is a tuple containing the two biggest numbers from the
numbers following the name.

Note: the testing code uses the print_dict_in_key_order(a_dict) function to print
the key:value pairs of the returned dictionary in alphabetical key order.

For example:

```
    the_dict = get_best_2_marks_dict("NamesMarks1.txt")
    print_dict_in_key_order(the_dict)
```

prints

```
Carolin : (80, 90)
Fancy : (74, 80)
Ralph : (65, 73)
"""
```

```
#-------------------------------------------------------
# 7777777777777777777777777777777777777777777777777777
# get_next_words_dict()
#-------------------------------------------------------
"""
```

Define the get_next_words_dict() function which is passed a string of text as a
parameter. Firstly the function converts the parameter text into lowercase. The
function returns a dictionary with keys which are all the unique words from the
text, and corresponding values which are lists of all the unique words in the text
which come after the key word.

Notes
• The last word in the sentence will initially have the empty string as its next
word,

• You can assume that the text contains no punctuation,
 • Each list of next words must be sorted into ascending order,
 • The testing code uses the print_dict_in_key_order(a_dict) function to print the
key:value pairs of the returned dictionary in alphabetical key order.

For example:

```
    text = 'Easy come easy go go easy go easy'
    next_words_dict = get_next_words_dict(text)
    print_dict_in_key_order(next_words_dict)
```

prints:

```
come : ['easy']
easy : ['', 'come', 'go']
go : ['easy', 'go']
"""
#---------------------------------------------------
# The end :-)
#---------------------------------------------------
```