CompSci 101 Assignment 4

Due: 4:30pm, June 4

Worth: 3% of your final mark

Topic: Python dict objects

This assignment is marked out of 30

CodeRunner Assignments

Assignment 4 is completely marked on CodeRunner. **There is no other submission required.**

https://www.cs.auckland.ac.nz/courses/compsci101s1c/assignments/

Assignment 4 – Complete 7 functions

For Assignment 4, I have posted program containing the skeleton and testing code for the 7 assignment questions. Download these programs from the CompSci 101 assignments website:

https://www.cs.auckland.ac.nz/courses/compsci101s1c/assignments/

Develop the solution to each function in each program.

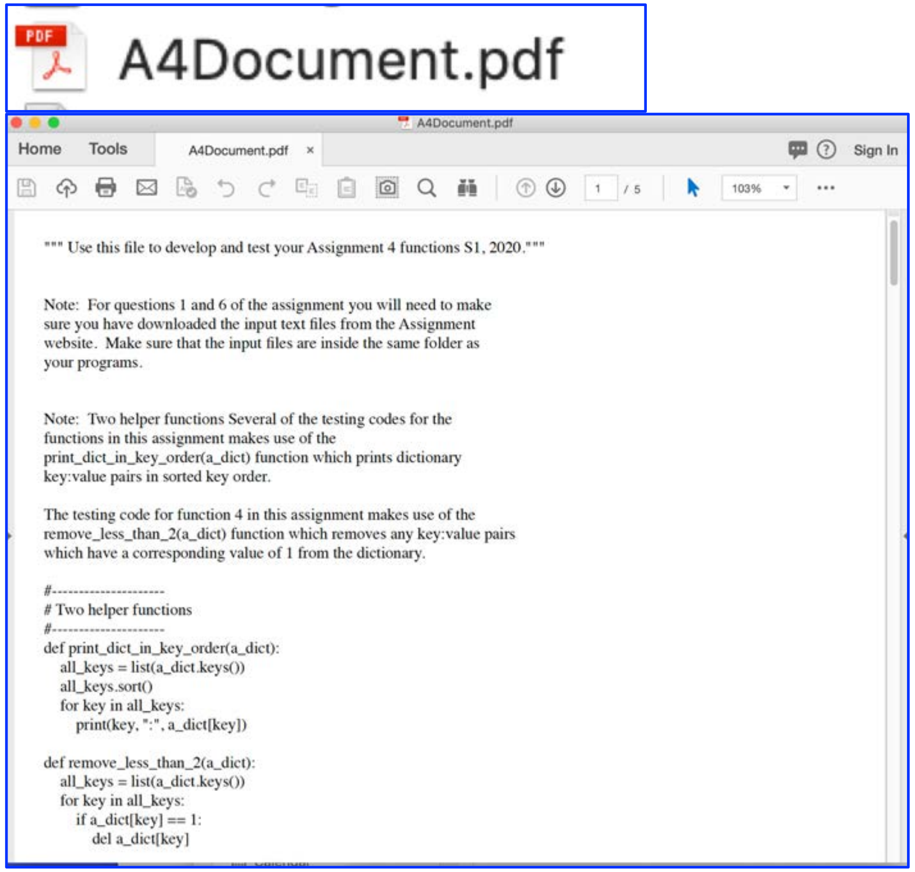
Once you are happy that your function executes correctly, submit **the whole function** to CodeRunner. You will receive immediate feedback from CodeRunner telling you if you have passed the tests for that question. You can submit as many times as you like. You need to submit one function at a time.

A4 Skeleton Files

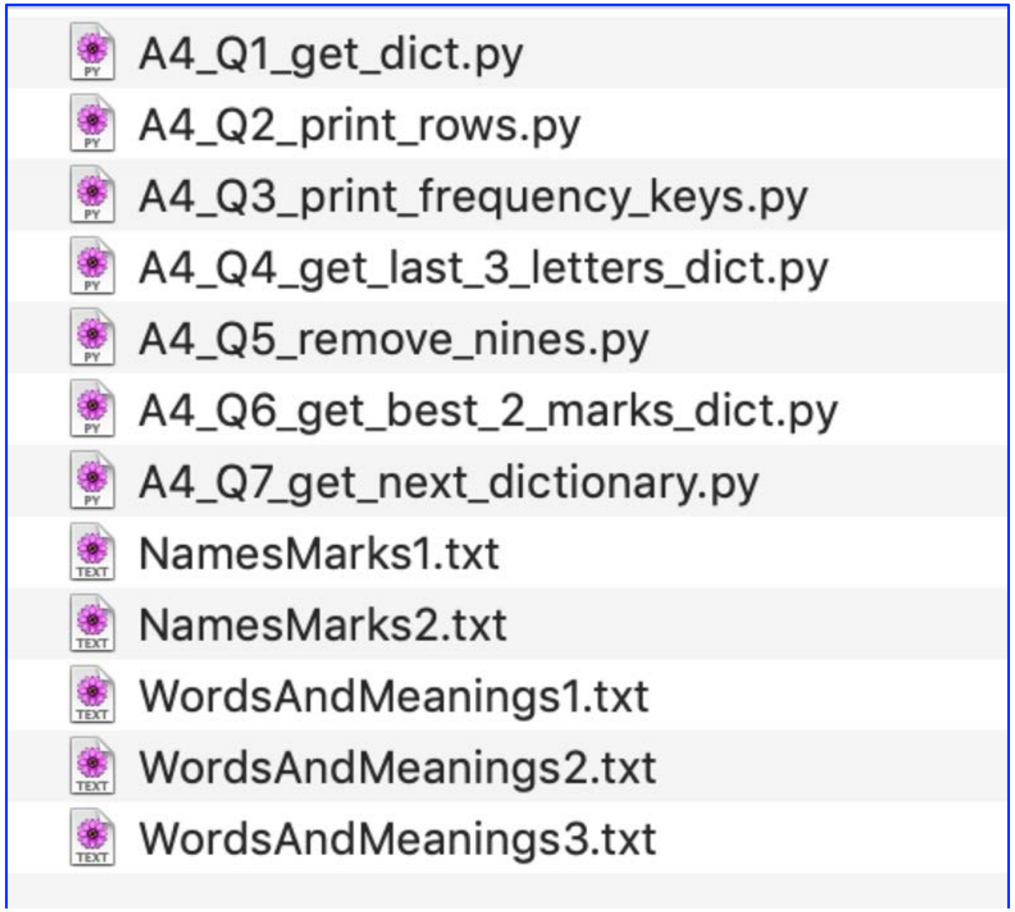
https://www.cs.auckland.ac.nz/courses/compsci101s1c/assignments/

Download:

- **the Assignment 4 document,**
- **the 7 skeleton Python files**
- **input files for testing Questions 1 and 6.**



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


Note:  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:  Two helper functions Several of the testing codes for the
functions in this assignment makes 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]
```



A4_Q1_get_dict.py
A4_Q2_print_rows.py
A4_Q3_print_frequency_keys.py
A4_Q4_get_last_3_letters_dict.py
A4_Q5_remove_nines.py
A4_Q6_get_best_2_marks_dict.py
A4_Q7_get_next_dictionary.py
NamesMarks1.txt
NamesMarks2.txt
WordsAndMeanings1.txt
WordsAndMeanings2.txt
WordsAndMeanings3.txt

A4 Two Helper Functions

Some of the functions in this assignment makes use of the function:

print_dict_in_key_order(a_dict)

which prints dictionary key:value pairs in sorted key order.

```
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])
```

The testing code for function 4 in this assignment makes use of the function:

remove_less_than_2(a_dict)

which removes any key:value pairs which have a corresponding value of 1 from the dictionary.

```
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]
```

CompSci 101 Assignment 4

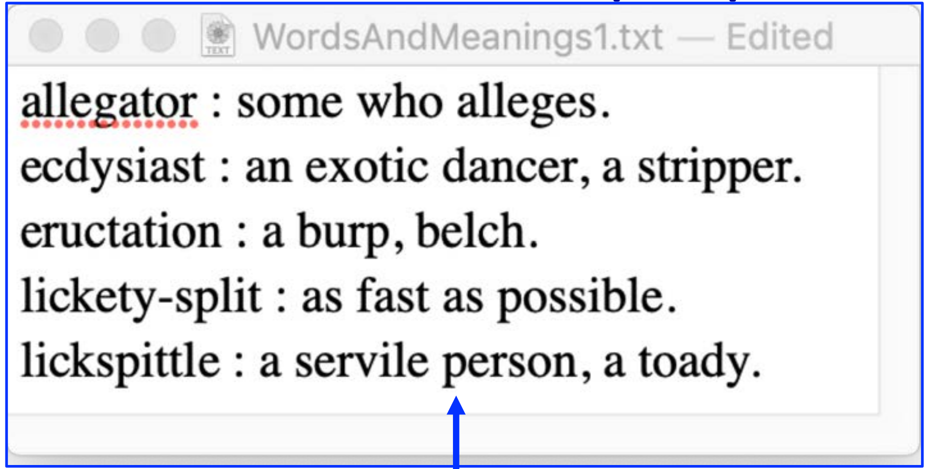
Some helpful information about Questions 1, 2 and 3

A4 Q1 – get_dictionary_from_file()

parameters – a string, the name of a file

returns – a dictionary object where each word is the key and the corresponding value is the meaning.

Example input file:


WordsAndMeanings1.txt — Edited

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

Notes
In the file each word and its meaning is on a separate line and ' : ' separates the word from its meaning.

```
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])
```

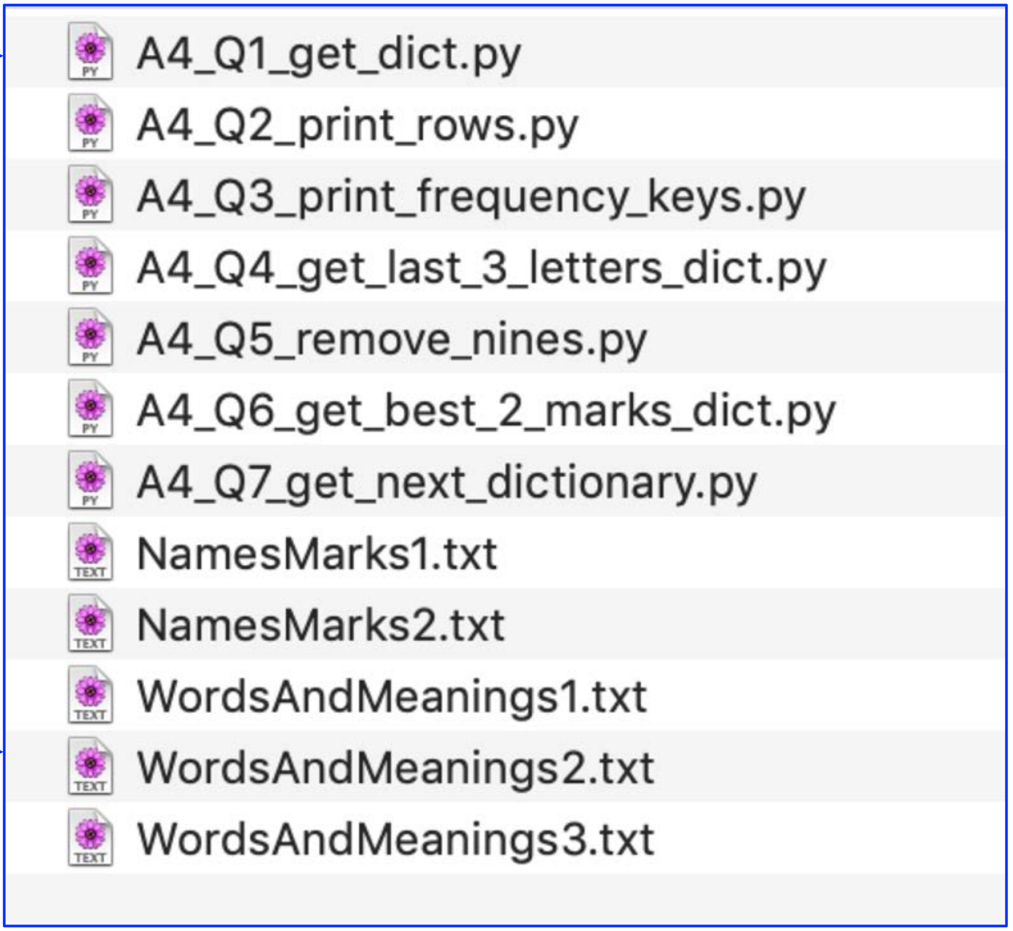
```
lickspittle = a servile person, a today.
allegator = some who alleges.
lickety-split = as fast as possible.
```

A4 Q1 – get_dictionary_from_file()

Make sure the example input files are in the same folder as your Python program

Question 1 Skeleton program

Three input files for testing Question 1



A4_Q1_get_dict.py
A4_Q2_print_rows.py
A4_Q3_print_frequency_keys.py
A4_Q4_get_last_3_letters_dict.py
A4_Q5_remove_nines.py
A4_Q6_get_best_2_marks_dict.py
A4_Q7_get_next_dictionary.py
NamesMarks1.txt
NamesMarks2.txt
WordsAndMeanings1.txt
WordsAndMeanings2.txt
WordsAndMeanings3.txt

A4 Q1 – get_dictionary_from_file()

How to read from a text file (Lecture 20)

1. Open the file for reading (the variable , `input_file`, is now the connection between your program and the file):

```
input_file = open(name_of_file, "r")
```

2. Read all the text from the file (the variable , `contents`, is now a string containing ALL the text from the file):

```
contents = input_file.read()
```

3. Make sure you close the file after you have finished:

```
input_file.close()
```

A4 Q1 – get_dictionary_from_file()

How to split a string into a list of its single parts or tokens (Lecture 21)

1. The `split()` method is used to break up a string into a list of strings (if there is no argument it separates the words using white space as the separator):

```
words = "The budget was unlimited,  but I exceeded it  "
word_list = words.split()
print(word_list)
```

```
['The', 'budget', 'was', 'unlimited,', 'but', 'I', 'exceeded', 'it']
```

2. The argument passed to the `split()` is the string which you wish to use as the separator in the text (e.g. `"\n"`, `":"`, `","`):

```
words = "The  budget was unlimited   \nbut I  exceeded\nit "
lines = words.split("\n")
print(lines)
```

```
['The  budget was unlimited   ', 'but I  exceeded', 'it ']
```

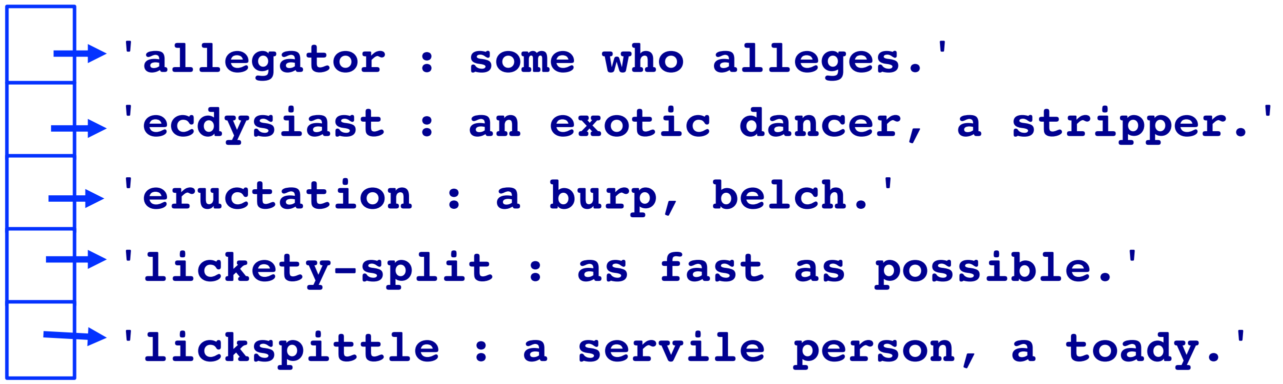
A4 Q1 – get_dictionary_from_file()

How to split a string into a list of its single parts or tokens (Lecture 21)

1. Split into a list of each line of text:

The string read from the file:

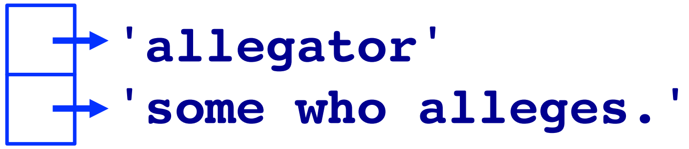
'allegator : some who alleges.\necdysiast : an exotic dancer, a stripper.\neructation : a burp, belch.\nlickety-split : as fast as possible.\nlickspittle : a servile person, a toady.'



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

2. Using " : " as the separator, split each string element of the list into a list of each part of the line

```
'allegator : some who alleges.'
```



```
'allegator'
'some who alleges.'
```

A4 Q1 – get_dictionary_from_file()

How to create a new dictionary, how to add key:value pairs to the new dict.

A dictionary is a collection of key:value pairs in any order:

```
{'a': 7, 'c': 5, 'b': -2, …}
```

1. Create an empty Python dictionary:

```
a_dict = {}    #or a_dict = dict()
```

2. Fill the dictionary with the key:value pairs. Usually this happens over and over, i.e. inside a loop. Corresponding value can be of any type (integer, string, list, tuple, etc.):

```
. . .
    a_dict[the_key] = the_corresponding_value
```

3. Usually the filled Python dict is returned as a result of the function:

```
return a_dict
```

A4 Q2 – print_rows()

parameter - a dictionary, e.g.

returns – None

```
{'a': (4,3), 'c': (5,0), 'b': (-2,5)}
```

For each key:value pair in the dictionary, the function prints the key in uppercase characters surrounded by stars.

Notes

- **Only keys with corresponding numbers which are greater than 0 are printed.**
- **Number of stars surrounding each key is given by the smaller of the two numbers in the corresponding tuple.**
- **Number of repetitions of the key is given by the bigger of the two numbers in the corresponding tuple.**
- **The keys are printed in sorted order.**
- **The printing is done on one line of output.**

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

```
*CC*******FFFFFFFFFFF******
```

A4 Q2 – print_rows()

How to visit each key of the dictionary in sorted order

1. Get a collection of just the keys from the dictionary:

```
collection_of_keys = some_dict.keys()
```

2. Get a list of the collection of keys:

```
list_of_keys = list(collection_of_keys)
```

OR

```
list_of_keys = list(some_dict.keys())
```

3. The list can now be sorted:

```
list_of_keys.sort()
```

4. Process each element of the sorted list using a loop:

```
for a_key in list_of_keys:
     #do what needs to be done with the key
```

A4 Q2 – print_rows()

Visit the values corresponding to each key of the dictionary (Lecture 22)

A dictionary is a collection of key:value pairs in any order:

```
{'a': (4,3), 'c': (5,0), 'b': (-2,5)}
```

Once you have the key, you can obtain and process the corresponding value for the key (this is usually done inside a loop):

```
a_dict = {'a': (4,3), 'c': (5,0), 'b': (-2,5), …}
for the_key in a_dict:
    print(a_dict[the_key])
```

A4 Q2 – print_rows()

How to get the smallest, biggest number in a tuple?

(Lecture 19)

```
{'a': (4,3), 'c': (5,0), 'b': (-2,5)}
```

For this question values corresponding to each key are tuples.

The `min()`, the `max()` and `sum()` functions can be used with Python tuples:

```
numbers_tuple = (25, 1)
smallest = min(numbers_tuple)
biggest = max(numbers_tuple)
total = sum(numbers_tuple)
```

Note: Use the repeat operator ('*') to repeat a string.

```
letter = 'A'
letters = letter * 10
print(letters)  #prints AAAAAAAAAA
```

A4 Q3 – print_highest_frequency_keys()

parameters – a dictionary and an integer

returns – None:

Example dictionary:

```
{'and':15, 'tiger':7,
'frog':1, 'cat':15,
'tests':2, 'dog':2,
'bat':14, 'rat':15,
'talon':7}
```

Prints information from the dictionary. Considers only the keys of the parameter integer length. Of those keys, the function finds the highest corresponding value. The function prints all the keys (of the correct length) which have the highest corresponding value. The keys are printed in sorted order.

```
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)
```

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

A4 Q3 – print_highest_frequency_keys()

Access each key:value pairs of a dictionary (Lecture 22)

Use a loop to process each key:value pair of a dictionary:

```
a_dict = {'and':15, 'tiger':7, 'frog':1, 'cat':15, …}
for the_key in a_dict:
     the_value = a_dict[the_key]
     ….
```

OR

```
a_dict = {'and':15, 'tiger':7, 'frog':1, 'cat':15, …}

for the_key, the_value in a_dict.items():
   ….
```

A4 Q3 – print_highest_frequency_keys()
Find the biggest corresponding value

1. Say that the biggest is the smallest possible value (or the first value):

```
biggest_so_far = 0 #the smallest possible corresponding value
```

2. Go through each key:value in the dictionary and if you find one which is bigger than the `biggest_so_far` change it:

```
a_dict = {'a': (4,3), 'c': (5,0), 'b': (-2,5), …}

for the_key, the_value in a_dict.items():
  if the current value is valid and bigger than biggest_so_far
       assign the current value to biggest_so_far
```

When the loop ends, `biggest_so_far` stores the biggest value found.

A4 Q3 – print_highest_frequency_keys()

Example dictionary:

```
{'and':15, 'tiger':7,
'frog':1, 'cat':15,
'tests':2, 'dog':2,
'bat':14, 'rat':15,
'talon':7}
```

What the question is asking:

print a list of the keys of a given length which have the biggest corresponding value.

1. Define an empty list.

2. Go through the dictionary and find the biggest corresponding value for valid keys (keys which have the correct length).

3. Go through the dictionary again and add the keys which are valid and have the correct length to a list. Sort the list.

A4 Q3 – print_highest_frequency_keys()

OR

Example dictionary:

```
{'and':15, 'tiger':7,
'frog':1, 'cat':15,
'tests':2, 'dog':2,
'bat':14, 'rat':15,
'talon':7}
```

What the question is asking:

Print a list of the keys of a certain length which have the biggest corresponding value.

1. Define an empty list. Define `biggest_so_far` as 0.
2. Go through each key:value pair in the dictionary:
    - If the key is valid and the corresponding value is equal to the `biggest_so_far`, append the key to the list,
    - If the key is valid and the corresponding value is bigger than the `biggest_so_far`, change biggest_so_far, clear the list and add the key to the cleared list.
3. Sort the list.