## 

## 

**Experiment 2:**

**Strings, Lists, Tuples, and Dictionaries**

CPE106L (Software Design Laboratory)

**Terenz Ace C. Flores**

**Kyeong Nan Gim**

**Rafael Lorenzo R. Muyot**

**Louella Amor Ramos**

**Roger Angelo D. Salinas**

Group No.: **1**

Section: **B3**

## **PreLab**



|  |
| --- |
| **Readings, Insights, and Reflection**  Note: All text in red should be removed in the final lab report. Change the font color from RED to black.  < **What to Include?**  Readings include: METIS books, pertinent websites. Provide below your Insights and Reflection. Paragraph format> |
| ***<Include ISBNs and pages of used METIS books and pertinent urls>***  ***<Insights and Reflections here…>*** |

**Answers to Questions**

1. **b. 20**

2. **b. [20,30]**

3. **a. 1**

4. **b. [10,20,30,40,50]**

5. **b. [10, 5, 30]**

6. **c. [10, 15, 20, 30]**

7. **b. ["name", "age"]**

8. **b. None**

9. **b. pop**

10. **b. string and tuples**

## **InLab**



**Objectives**

Examples:

1. **Debug** the sample programs on tuples, dictionary, etc.
2. **Use** Anaconda and Linux terminal in running python statements.
3. **Use** Visual Studio Code in debugging.
4. **Compare** C++ and Python data structures.

(**Avoid** these verbs: To know, to understand)

1. Objective 1
2. Objective 2

* **Tools Used** 
  + Anaconda
  + Git Terminal
* **Procedure.**
* Steps Performed with edited screenshots of tools used (Example: Using Anaconda, sample run, debugging with **DISCUSSIONS** (DON’T copy and paste from the reference METIS book and/or official websites)**.** Use the source in the Lab Guide.
* Use the given Data Files
* **DO not include source in the your screengrabs nor include in your lab report the given source codes as shown below… but you may include code snippet and debugging display.**

|  |
| --- |
| **DO NOT include this complete sample codes in your LAB REPORT discussion.**  """  File: mode.py  Prints the mode of a set of numbers in a file.  """  fileName = input("Enter the file name: ")  f = open(fileName, 'r')    # Input the text, convert its to words to uppercase, and  # add the words to a list  words = []  for line in f:  wordsInLine = line.split()  for word in wordsInLine:  words.append(word.upper())  # Obtain the set of unique words and their  # frequencies, saving these associations in  # a dictionary  theDictionary = {}  for word in words:  number = theDictionary.get(word, None)  if number == None:  # word entered for the first time  theDictionary[word] = 1  else:  # word already seen, increment its number  theDictionary[word] = number + 1  # Find the mode by obtaining the maximum value  # in the dictionary and determining its key  theMaximum = max(theDictionary.values())  for key in theDictionary:  if theDictionary[key] == theMaximum:  print("The mode is", key)  break |

* **IMPORTANT**: Figure numbers and brief description.

|  |
| --- |
| Discussion here…. **DON’T forget to mention the image in the screengrabs**.  As shown in Figure 1…. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat .  Figure 1. Using conda command to display created environments. There are seven conda environments.  Figure 2. Debugging error message showing ‘FileNotFoundError’ exception  Discussion here…. DON’T forget to mention the image in the screengrabs.  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim… **as shown in Figure 2**.  Figure 3. Class diagram of Student class showing attributes and operations  Discussion here for screenshot #3…. DON’T forget to mention the image in the screengrabs.  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim… **as shown in Figure 3**.  More discussions here…  Example: **Compare** C++ and Python data structures.  **Use the particular chapter, pages of the METIS book as well as the reference urls. Does not have to contain a screengrab.**  Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. |

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## **PostLab**



This section should start on a new page. [**What to Include?]**

**Text in RED are to be removed in the final lab report**

**Programming Problems** (Leaders will assign each member to work on)

Includes edited screenshots of tools used (Using Anaconda, Git Terminal, and Visual Studio Code sample run, debugging with **DISCUSSIONS** (DON’T copy and paste from the ebook)**.** Use the source in the Lab Guide. You may include source codes of Roth. **IMPORTANT**: Figure numbers and labels.

1. Filename: **stats.py**

A group of statisticians at a local college has asked you to create a set of functions that compute the median and mode of a set of numbers, as defined in the below sample programs:

* [mode.py](https://mymailmapuaedu-my.sharepoint.com/:u:/g/personal/dapadilla_mapua_edu_ph/EWQI7kSWI-xFiYfm2KyofXoBBV_Zjex0RtarScWwn57pag?e=CD4HuL)
* [median.py](https://mymailmapuaedu-my.sharepoint.com/:u:/g/personal/dapadilla_mapua_edu_ph/EQ7kUrEOI0RIgplQhfEYnTABvArfAp403AaoreENtzVfgw?e=yo1hzy)

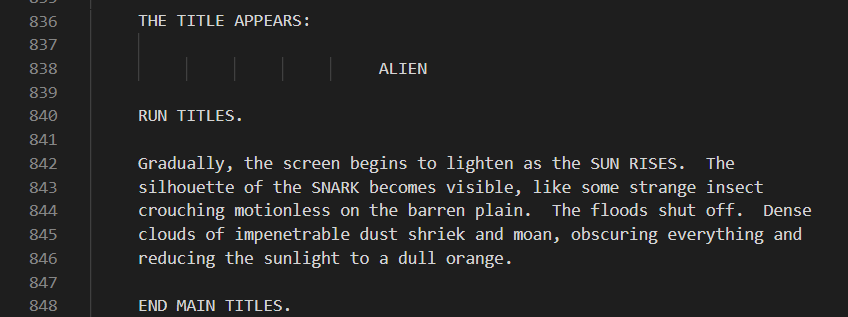
Define these functions in a module named **stats.py**. Also include a function named mean, which computes the average of a set of numbers. Each function should expect a list of numbers as an argument and return a single number. Each function should return 0 if the list is empty. Include a main function that tests the three statistical functions with a given list.

1. Filename: **LR2\_2.py**

Write a program that allows the user to navigate the lines of text in a file. The program should prompt the user for a filename and input the lines of text into a list. The program then enters a loop in which it prints the number of lines in the file and prompts the user for a line number. Actual line numbers range from 1 to the number of lines in the file. If the input is 0, the program quits. Otherwise, the program prints the line associated with that number.

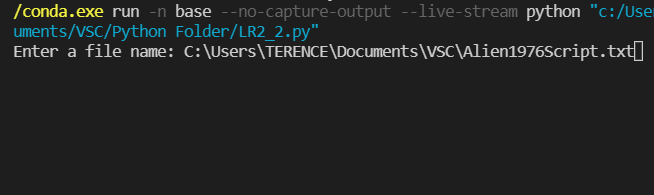
First, identify which text file will be used to test the program. **As seen in Figure X**, we used a copy of the script for the 1976 movie, Alien.

Figure X. Snippet of the Alien (1976) movie script in the Alien1976Script.txt file.



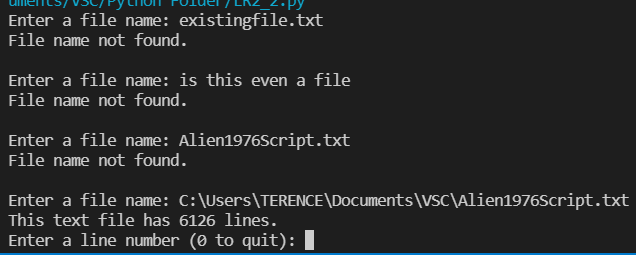
When launching the program, the user will be prompted to enter a file name consisting of several text lines. The user may input the file name directly if it is in the same file directory as the python file or input the complete directory path if it is stored in a different folder. An example of a complete directory path is **shown in Figure X**.

Figure X. Example of a complete directory path including the text file.



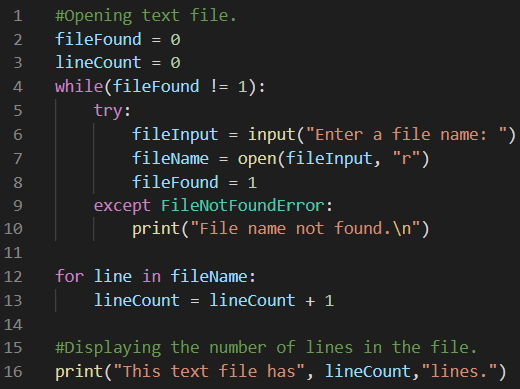
If the user enters a file name that cannot be found or does not exist, the program will alert the user and ask for another file name **as shown in Figure X**.

Figure X. Snippet of wrong and correct file name inputs.



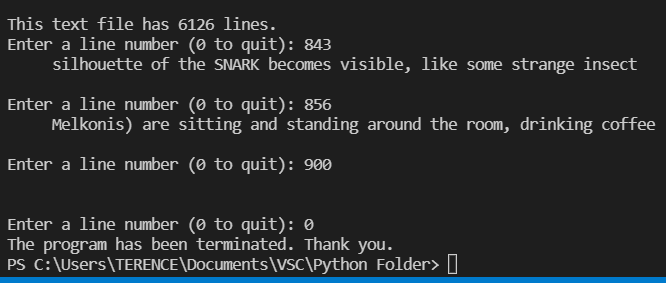
The program was able to identify if the file name is right or wrong by using the try-except block on the first while loop, **as seen in Figure X**. In the except block, the “FileNotFoundError” exception triggers whenever the correct file cannot be found. The block will only break from the conditional statement if the user inputs a valid file name.

Figure X. Snippet of wrong and correct file name inputs.



Once an appropriate text file has been found, the program will output the number of lines it contains and prompt the user for which line number to be displayed. The program will be stuck on a loop that is prompting the user for a line number until the user types 0 to terminate the program **shown in Figure X**.

Figure X. Displaying the corresponding line content per line number and terminating the program.



Another while loop statement is used for the program to locate the requested line number in the file and display the content of the corresponding line. The loop will break once the user enters 0 to end the program.

1. Filename: **generator\_modified.py**

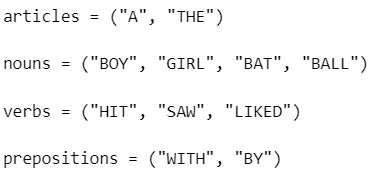
Modify the sentence-generator program of Case Study 5.3:

* METIS book: 9781337671019, page 150.
* Python source code: [generator.py](https://mymailmapuaedu-my.sharepoint.com/:u:/g/personal/dapadilla_mapua_edu_ph/EWb2bLxiCAROunKE_BC2ux8BFjuKrS3b-6YYBLrs5zDObw?e=YCZWOK)

so that it inputs its vocabulary from a set of text files at startup. The filenames are **nouns.txt**, **verbs. txt, articles.txt**, and **prepositions.txt**. (Hint: Define a single new function, getWords. This function should expect a filename as an argument. The function should open an input file with this name, define a temporary list, read words from the file, and add them to the list. The function should then convert the list to a tuple and return this tuple. Call the function with an actual filename to initialize each of the four variables for the vocabulary.)

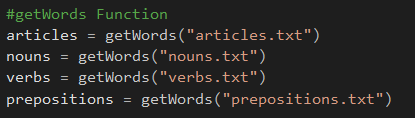
We create the input files that will be opened by the program to generate the sentence. From the initial code given it can be seen the words to be put in each file is already given as can be seen in the figure X below.

Figure X. Guide words from the initial code.



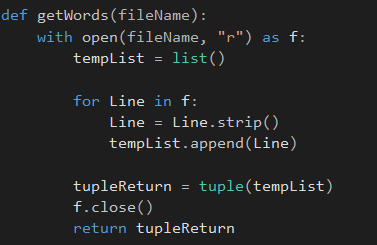
The files nouns.txt, verbs. txt, articles.txt, and prepositions.txt are then created, which is then proceeded to be placed in the main saved folder for the python program. When the 3 input files are created, the getWords function is created to call on the 4 main input files as can be seen in figure X.

Figure X. getWords function is created to call on the 4 input files as the main argument in the program.



After creating the function that call on the 4 main input files, a tupple is created along with the with statement to call on the file stream in a cleaner way. A for function is called to create a temporary list for the line which is then proceeded to be contained in the tupple which manages the call function.

Figure X. Call on function for the input files that is stored in a temporary list contained in the tupple.



Testing the program, it can be seen that it works like the initial program however the array that was once used is now contained in an input file used as an argument by the program.

Figure X. output of the generator.py, with 4 input files as the main argument creating a sentence generator.

