Lab 6.

Part_01: Dictionaries

Class work:

Exercise 1: Download a copy of the file www.py4e.com/code3/words.txt

Write a program that reads the words in *words.txt* and stores them as keys in a dictionary. It doesn't matter what the values are. Then you can use the in operator as a fast way to check whether a string is in the dictionary.

Exercise 2: Write a program that categorizes each mail message by which day of the week the commit was done. To do this look for lines that start with "From", then look for the third word and keep a running count of each of the days of the week. At the end of the program print out the contents of your dictionary (order does not matter).

Sample Line:

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

Sample Execution:

```
python dow.py
Enter a file name: mbox-short.txt
{'Fri': 20, 'Thu': 6, 'Sat': 1}
```

Exercise 3: Write a program to read through a mail log, build a histogram using a dictionary to count how many messages have come from each email address, and print the dictionary.

```
Enter file name: mbox-short.txt
{'gopal.ramasammycook@gmail.com': 1, 'louis@media.berkeley.edu': 3,
'cwen@iupui.edu': 5, 'antranig@caret.cam.ac.uk': 1,
'rjlowe@iupui.edu': 2, 'gsilver@umich.edu': 3,
'david.horwitz@uct.ac.za': 4, 'wagnermr@iupui.edu': 1,
'zqian@umich.edu': 4, 'stephen.marquard@uct.ac.za': 2,
'ray@media.berkeley.edu': 1}
```

Exercise 4: Add code to the above program to figure out who has the most messages in the file. After all the data has been read and the dictionary has been created, look through the dictionary using a maximum loop (see Chapter 5: Maximum and minimum loops) to find who has the most messages and print how many messages the person has.

```
Enter a file name: mbox-short.txt
cwen@iupui.edu 5
```

```
Enter a file name: mbox.txt
zqian@umich.edu 195
```

Exercise 5: This program records the domain name (instead of the address) where the message was sent from instead of who the mail came from (i.e., the whole email address). At the end of the program, print out the contents of your dictionary.

```
python schoolcount.py
Enter a file name: mbox-short.txt
{'media.berkeley.edu': 4, 'uct.ac.za': 6, 'umich.edu': 7,
'gmail.com': 1, 'caret.cam.ac.uk': 1, 'iupui.edu': 8}
```

Tasks:

1. Create a list of your favorite musicians.

2. Create a dictionary containing various data about you: height, favorite color, favorite actor, etc.

3. Write a program that asks the user for their weight, favorite color, or actor and returns the result of the dictionary created in the previous task.

4. Create a dictionary linking your favorite musicians to a list of your favorite songs written by them.

5. There are a number of dictionaries with intersecting keys (values are positive numbers). Write 2 functions that do the following operations with an array of dictionaries:

The 1st function max_dct(*dicts) forms a new dictionary according to the rule:

If there are duplicate keys in the source dictionaries, we select the maximum among their values and assign this key (for example, dictionary_1 has the key “a” with the value 5, and dictionary_2 has the key “a”, but with the value 9. Choose the maximum value, t i.e. 9, and assign the key “a” in the already new dictionary).

If the key is not repeated, then it is simply transferred with its value to the new dictionary (for example, the key “c” was found only in one dictionary, while others do not. Therefore, we transfer this key along with its value to the new dictionary). We return the generated dictionary.

The 2nd function sum_dct(*dicts) sums the values of duplicate keys. The values of the other keys remain the same. (Operations are carried out similar to the first function, but not the maxima are taken, but the sums of the values of the keys of the same name). The function returns the generated dictionary.

Part_02: Tuples.

Tasks:

1. **Grade Statistics:**
   - Create a tuple with grades of students in a math class (ranging from 1 to 10). Write a program that outputs the average grade, the maximum, and the minimum grades.
2. **Weather Analysis:**
   - Create a tuple with temperature data for a week. Write a program that calculates the average temperature, the number of days with positive and negative temperatures.
3. **Sales Distribution:**
   - Create a tuple with data on sales of products in different months. Write a program that outputs the month with the maximum and minimum sales.
4. **Coordinates of Points:**
   - Create a tuple of coordinates of points on a plane. Write a program that determines how many points are in each quadrant.
5. **Book List:**
   - Create a tuple representing a list of books in a library (title, author, publication year). Write a program that outputs a list of books published within a certain period.
6. **Student Statistics:**
   - Create a tuple with data about students (name, age, average grade). Write a program that outputs statistics on the ages of students.
7. **Operations with Complex Numbers:**
   - Create a tuple of two complex numbers. Write a program that performs addition, subtraction, and multiplication operations on these numbers.
8. **Order List:**
   - Create a tuple with information about orders in an online store (product name, quantity, price). Write a program that outputs the total cost of all orders.
9. **City Data:**
   - Create a tuple with data about cities (name, population, coordinates). Write a program that determines the city with the largest population.
10. **Subject Grades:**
    - Create a tuple representing student grades for different subjects. Write a program that outputs the subject with the highest and lowest average score.
11. **Age Group Distribution:**
    - Create a tuple with ages of people. Write a program that determines the number of people in different age groups (e.g., 0-18, 19-35, 36-50, 51+).
12. **Stock Price Analysis:**
    - Create a tuple with stock prices on the market for a month. Write a program that determines the month with the maximum price change.
13. **Solar System Planets:**
    - Create a tuple with data about the planets of the Solar System (name, diameter, distance from the Sun). Write a program that outputs the planet with the smallest and largest diameter.
14. **Sports Team Statistics:**
    - Create a tuple with data about sports team results (team name, number of wins, number of losses). Write a program that outputs the team with the best win/loss ratio.
15. **Character Count:**

- Create a tuple with multiple strings. Write a program that counts the occurrences of each character in all the strings.

16. Task List:

- Create a tuple with tasks to be completed (task name, status). Write a program that outputs the number of completed and incomplete tasks.

17. Time Analysis:

- Create a tuple with data about the start and end times of various events. Write a program that determines the total duration of the events.

18. Genetic Code:

- Create a tuple with a sequence of nucleotides in a genome. Write a program that finds repeating sequences in the genetic sequence.

19. Expense Analysis:

- Create a tuple with data about expenditures in different categories (groceries, transportation, entertainment) for a month. Write a program that determines the category with the highest and lowest expenditures.

20. Fibonacci Sequence:

- Write a program using a tuple that generates the Fibonacci sequence up to a certain limit.