# Python exercises:

Level 1

Explore the OS & sys module and:

1. Find out if a file called as “sample.txt” exists in the current directory, if it does then delete the file.
2. Read the names of all the files in current folder with .txt extension & store it in a list.
3. Create a directory with few python files. Add this special directory to current list of sys.path and check if the python files available in that directory are accessible as modules.

Level 2:

1. Write a program to read a text file and paste alternate lines into a new file.
2. Write a program to read two files and combine each line from first file with the corresponding line in second file. Create a third file with this new data.
3. Create a module called as maths.py

Create 5 functions representing 5 mathematical operations (addition, square etc…)

any user should be able to import

Manage wrong inputs provided by the user with relevant error messages.

Level 3:

1. Read the file called ‘emails\_blue.txt’, create a dictionary with of number of email ids from the same domain.

sample output:

{'att.net': 124, 'yahoo.ca': 142, 'msn.com': 128,………… }

1. Read the file called ‘emails\_blue.txt’ and find out if any person has created same email IDs in different domains.

for example, johndoe@gmail.com , johndoe@yahoo.ca, ….

Make a dictionary of such ids, with id as key; and values being domain names as a list.

{‘johndoe’: [‘gmail.com’, yahoo.ca’], …..}

Please not the dictionary should contain only those names which have more than one id with same name

1. The following information about a list of government employees of Chicago city are mentioned in the file called employee.txt:

Name

Job Titles

Department

Full or Part-Time (F or P)

Hourly (meaning they are paid hourly)

Typical Hours (hours worked per week)

Hourly Rate (amount paid per hour)

Find out how many employees work full time and how many part time.

List the top 10 employees in terms of total earnings per week.

Put the data of all the crossing guard employees into a new file.

1. Create a module called mathematics, within which a class called as fractions, with provisions for numerators & denominators

For example:

obja = fraction(12, 7) # represents 12/7

objb = fraction(6, 5) # represents 6/5

Within the class create overloaded operator functions for arithmetic operations like addition (+) and subtraction (-)

For example:

objc = obja + objb # objc should now represent 97/35

objd = obja - objb # objd should now represent 13/35

Add appropriate display methods to display the values in fraction form