# CSE101 – File IO, tuple, dictionary

## Goals

This tutorial will help you:

1. The basics of File I/O in Python

2. How to do very basic analysis of text

3. Understand tuple in Python

4. Understand dictionary in Python

## Background

Many tools are being built these days to analyze text in human language. Companies are developing language translation applications. Intelligence organizations are analyzing intercepted communication searching for ‘keywords’ that may indicate a threat. Applications are being developed to read and understand text and deliver very relevant results on keyword searches (deep learning).

The field in CS that works with such tasks falls under what is called ‘Natural Language Processing’ or more recently ‘Computational Linguistics’.

Several decades ago, a program was developed called Eliza. This program acted like a psychoanalyst and would ask questions about a person’s mood and then answer their responses with additional questions. The program seemed sentient (intelligent) but was using a trick of finding keywords and asking a ‘patterned’ question to probe for more details.

Such programs have become far more sophisticated today (4-5 decades later).

What we will be doing is searching text for several keywords and producing some statistics about them.

## File IO Tasks

1. **Create PyCharm project**

In Pycharm, create a ‘Project’. You can name the project whatever you want. Then, on the project name (in the project explorer on the left side of the PyCharm display), right click on the project name and select New->Python Package. Give that a name (I suggest ‘Lab6’).

1. **Create a python program file.**

Right-click on your package name (‘Lab6’?) and select New->Python File. Give the program the name textAnalysis. Note that you do NOT include the ‘.py’ extension when providing the python file name.

1. **(Write a program to read the contents of a text file)**

Review the methods in python to open a file (open()), read a line of text (.readline()), and close a file (.close()). For information on the python file operations, review this web page (section 7.2 only): <https://docs.python.org/3/tutorial/inputoutput.html>

**Important note!** When you open a file, you need either an absolute path or relative path to the file. If you put the file in the same folder as your python source, you can simply use “debate.txt” as the filename.

Write a program that opens a file called debate.txt. The file is provided in the lab assignment on blackboard.

Read each line and print it to the screen (hint: use a while loop until there is no more data when you use .readline()). Test your program by running it and verifying that it prints the contents of the file.

1. **(Add code to ‘tokenize’ the words)**

First, remove the code that prints the line to the console. Now you can add code to split each line into words. The .split() operation does that and produces a list with each word as elements of the list. Add that and use a ‘foreach’ loop to examine each word. Count each word in the file (hint, you can use len() on the list of words after the line is split on spaces). Use an if-elif statement to compare each word to 4 different words we are tallying counts for. You can create 4 separate count variables, 1 for each word (i.e. education\_count, school\_count, etc). In the next lectures and chapter of the book you will learn more about ‘dictionaries’ which are types in Python that will help you with storing information indexed by strings as well as integers. That would make this problem much easier to solve.

The 4 words we are searching for and counting are:

• economy

• jobs

• education

• school

In order to catch all copies of the words, make sure you ‘lower case’ each word in the list before comparing it with each of these words (see the .lower() method available on strings). When the entire file has been read, tokenized, and words counted, close the file and print a summary of the results.

Presumably, if we add the counts for the words ‘economy’ and ‘jobs’ and compare that to the sum of counts for ‘education’ and ‘school’, we might get an indication of how important these topics were (relative to each other) in the presidential debate.

Print out:

1. The total words in the file

2. The counts of each of the 4 words

3. The sum of occurrences of economy and jobs

4. The sum of occurrences of education and school

1. **(Test code)**

Your code should produce the following output:

words: \_\_\_\_\_\_\_\_\_\_

education count: \_\_\_\_

school count: \_\_\_\_

economy count: \_\_\_\_

jobs count: \_\_\_\_

education + school: \_\_\_\_

economy + jobs: \_\_\_\_

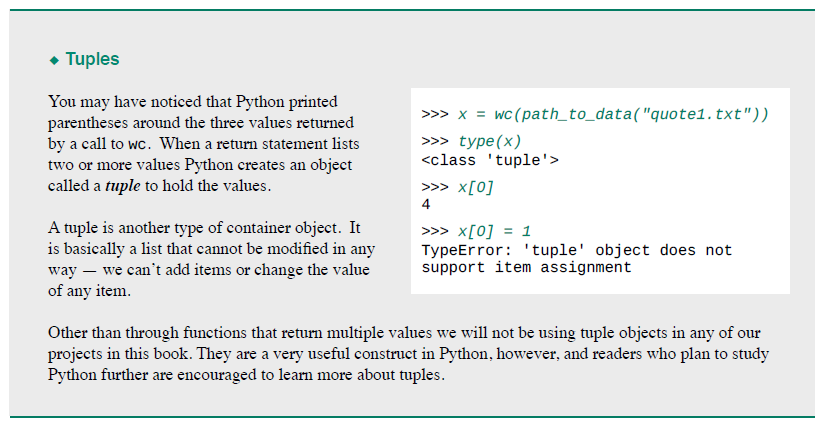
1. **Tutorial project**

Follow tutorial on Page 157 of the textbook “Explorations in Computing” by John S. Conery

## Tuple Tasks

1. **Tutorial project**

Follow tutorial on Page 157 of the textbook “Explorations in Computing” by John S. Conery



## Dictionary Tasks

1. **Tutorial project**

Follow tutorial on Page 164 of the textbook “Explorations in Computing” by John S. Conery

