JSON PROCESSING

- **Steps to run the utility :**
    - Enter the name of the json file: file_name.json

      eg: nested_example.json

    - Enter key to search(or 'q' to quit): key_to_search

      eg: otherstuff

- **Functions used:**
    - find_more_value() :

      This function takes the de-serialized data and key to be searched as parameter. Function checks whether the value of the key is of dictionary or list type and iterates recursively to fetch the value of the key. This function returns list of (key, value) pairs found in the data.

    - find_type_of_data() :

      Inside this function we call the function find_more_value(). First, this function checks if the data is a single document or a array of documents, and based on that it calls the function find_more_value().

    - pp_json() :

      This function prints the json data as well as value of the key in readable format.

- **Terms used in the utility :**

1. JSON Processing in python:

The full-form of JSON is JavaScript Object Notation. It means that a script (executable) file which is made of text in a programming language is used to store and transfer the data. Python supports JSON through a built-in package called json. It is similar to the dictionary in Python.

= > import json

2. De-serializing JSON:

The De-serialization is opposite of Serialization, i.e. conversion of JSON object into their respective Python objects. The load() method is used for it. If you have used Json data from another program or obtained as a string format of Json, then it can easily be de-serialized with load(), which is usually used to load from string, otherwise the root object is in list or dict.

= > data = json.load(json_file)

3. Serializing JSON:

The process of encoding JSON is usually called **serialization**. This term refers to the transformation of data into a series of bytes (hence serial) to be stored or transmitted across a network. To handle the data flow in a file, the JSON library in Python uses dump() function to convert the Python objects into their respective JSON object, so it makes easy to write data to files. See the following table given below.

= > json.dumps(json_thing, sort_keys=sort, indent=indents)

PYTHON OBJECT	JSON OBJECT
dict	object
list, tuple	array
str	string
int, long, float	numbers
True	true
False	false
None	null

4. pprint : Data pretty printer in Python

The **pprint** module provides a capability to "pretty-print" arbitrary Python data structures in a well-formatted and more readable way!