INSTRUCTIONS :

1) First install the required dependencies.
   a) Nltk
   b) Speech_recognition
   c) If on windows, you need to set up unicurses.
2) Then run the ultimain.py module
3) Search is a simple matter of typing out the query. The result will update dynamically, and you will have one result selected by default. To open the file in your default text editor, just press Enter.
4) You can move up and down the list of selections with the up and down arrow keys
5) For our **BETA feature**, **voice search**, enter the command **:v**. Pressing **‘:’** will automatically take you to the command-entry mode. Press enter after entering the command (**:v**), and you will receive a prompt to speak the required input. Then search progresses as usual
6) To exit, either enter the command **‘:q’**, or press the **‘Esc’** key.

DOCUMENTATION

Query Module

Functions:

1)def query(string):

This function receives the input that was taken from the user, as a string. It calls the relevant internal functions to tokenize (nltk.tokenize.casual_tokenize(string)) and normalize the query (lemmatize), then perform a vector space comparison with the document database, to find out relevant documents and rank them accordingly. This function then returns a list of the top 10 most relevant documents.

If a query term is not found in the corpus, the function tries to form a substring match, matching the initial part of the term from the corpus. We assume that the initial part of the query would be correct, which is statistically shown to hold true in most cases.

Ultimain Module

This module checks for the operating system (sys.platform) of the user and accordingly runs the files (main.py or main_win.py). This is needed as python has some specific modules pertaining to a specific operating system (such as curses or unicurses)

Document Module

This module defines a class **DocDB**, of which only one instance would be stored in the **.pyirdocs.db** file. We are thinking of making this search engine extensible and configurable, so there would be one **.pyirdocs.db** file at the root of every corpus that the search engine would search in. The only configuration needed is to edit the **~/.pyirconf** file to point it to the required root directory.

The decision to make a **DocDB** class was taken just in order to continue with the philosophy of modularity that we have adopted, and so as to group similar functionality and data together

The members of the class are -
id_list - This is a list of tuples of (docID, docName). We use this for translations from docID to docName
idf - This is a Python dictionary, This dictionary would be indexed by all the normalized words. Each word would be mapped to a float value which would represent the idf of the word for the collection of documents
tf - This is a Python dictionary. This dictionary would be indexed by all the normalized words. Each word would be mapped to a list of tuples of (docID, tf)

Voice Recognition Module(voice_recog.py)

This module is to provide an additional feature to the user.It recognizes speech using Google Speech Recognition. Furthermore, additional engines such as wit.ai can be added to improve and extend functionality

Initialization Module

This is the initialization module of our project. It has functionality for the initial configuration of the database for the first run of the search engine.

First, we need to read a config file if it exists, else create it.
The config file will be stored as **~/.pyirconf**
This config file will basically contain the path to the root directory for the corpus

Functions:

def init():
This sets up the initial configuration file for the project in an interactive manner. It also parses the entire corpus and creates a **.pyirdocs.db** at a location specified by the user, if it does not already exist. This file would be the result of parsing the database and would contain structures (like the inverted index)

1)def _initdb():
This is called by the init() function. It initializes the creation of the **.pyirdocs.db** file.

2)def _initconf()
This is called by the init() function. It interactively creates the **.pyirconf** file.

Processor Module

This module defines the preprocessing that the engine will perform on the documents in order to create the **.pyirdocs.db** file (the path to which will have been loaded to the global variable **_DB_PATH**. This module uses the following external modules/libs-
Nltk - For tokenization and normalization and lemmatization
Math - For logarithms (for idf)
Pickle - For writing to the files in binary format
Collections -> Counter - For grouping common tokens together

Functions:

1) **preprocess(of_obj)** -
   This file takes as input a file object for the **.pyirdocs.db** file and calls all relevant, required functions within the module to perform the required preprocessing on all documents in the directory (except the .db file itself). It then stores the result of the preprocessing in the **.pyirdocs.db** file, in the form of an object of the documents.DocDB class.
2) **_preprocess_idf(docdb_obj, N):**
   This function is called by the primary preprocess function. It calculates the IDF using math.log10(N / num_doc) where N is the total number of documents in the corpus and num_doc is the number of documents which have that particular token.
3) **_preprocess_magnitude(docdb_obj):**
   This function is called by the primary preprocess function. And it calculates the magnitude of the tf-idf vector of the document.The magnitude is dependent only on the terms that the document contains.
4) **_preprocess_file(docnum, docdb_obj, document):**
   This function is called by the primary preprocess function and it reads the entire document, tokenizes the words, lemmatizes the tokens.

Global Variables Module

This modules just contains some global variables which would be used
by different modules of the project. This allows for easier configuration
at a central point, instead of manually changing occurrences of variables
at different places

Currently, it contains these variables -

```
_PROJ_NAME  - The name of the project that will be display to the user
_VERSION    - The version of the project
_AUTHORS    - The people who have contributed to this project
_LICENSE    - The license for this project
_CONF_PATH  - The absolute path to the .pyirconf file
_DOCS_PATH  - The absolute path to the directory which contains the documents
_DOC_DB_OBJ - An object of the DocDB class that is the result of
              preprocessing the documents
```