**Main Program**

The following four lines of code were provided by Project Gutenberg – they facilitate creating a local database of all the project Gutenberg titles.

Cache = get\_meta\_data\_cache()

Cache.open

Cache.populate

graph = cache.graph

Define an object of class ‘catalog’ named my\_catalog

my\_catalog = Invoke a function (get\_all\_titles) - the function retrieves from project Gutenberg website all book titles

Set a flag to check if user would like to search again

more\_titles = ‘y’

If more\_titles is true,

Define a variable book\_text.

Book\_text = Invoke a function search\_display\_options (send all available titles in variable my\_catalog as a parameter)

If book\_text is found,

Invoke a function display\_word\_cloud(send in book\_text as a parameter)

Else, ask the user if would like to search again.

**Classes and functions**

Class: book

Defines a class (book) with attributes: index, author, title, subject.

Methods:

get\_book\_author

get\_book\_title

get\_book\_index

get\_book\_subject

Class: book\_catalog

(book catalog) and create a list of books.

Methods:

1. get\_books

returns a list of books stored in this catalog

1. add\_book

add to my\_catalog a title passed in my\_book variable,

#populate search\_result\_catalog by author, title, or subject.

1. get\_book

return a book by index using my\_book variable

#check for errors using try except blocks.

1. get\_authors

return a sorted list of authors

1. get\_titles

returns a sorted list of titles

1. get\_subjects

returns a sorted list of subjects

1. display\_titles\_by\_authors

populate a dictionary with authors as key. And index and title as values

print author name, index and title values.

1. get\_all\_titles

get titles from Gutenberg library metadata by retrieving all titles

if title language is English,

then get book index,

book author,

book title,

and book subject, get rid of the new line and carriage return.

Add my\_book to my\_catalog

Return all titles in a new catalog.

1. display\_wordcloud

create a variable comment\_words

create tokens, convert into lowercase,

add the tokens to comment\_words

use the Wordcloud class with parameters for the canvas size,

background color, stopwords omission, min font size,

generate wordcloud from the comment\_words

plot image

1. search\_display\_options

set variable search\_result\_catalog to get value from the book\_catalog class

get input about search\_type either by author, title, or subject.

If invalid input tell the user.

Match the search with books through iteration.

Set match variable to false.

For my\_book in my\_catalog,

if the search is by author and the search term is in author’s name, search\_result\_catalog gets the book, match becomes true.

If the search is by title, and the search term is in the title,

search\_result\_catalog gets the book, match becomes true.

If the search is by subject, and the search term is in the subject,

search\_result\_catalog gets the book, match becomes true.

If match is true, then ask the user to enter title number.

Print displaying wordcloud.

If match is false, tell the user no matches found.

**Challenges**

After several runs, the Gutenberg API got corrupt and I had to populate the cache again by uncommenting and calling cache.populate() function. I also had to delete the local cahce database on the local filesystem (Gutenberg\_data subdirectory.)

Project Gutenberg does not provide a REST API. I was not able to exchange data using python JSON documents.

Was not able to download all titles as it took too long. Had to stop the program – comment cache.populate and only run on a small sample. Otherwise the program would take too long to run.

The wordcloud code I copied from https://www.datacamp.com/community/tutorials/wordcloud-python

had several lines that were not needed, including some libraries like pandas, numpy, and pillow that turned out to be extra and not needed.

**Instructions for users**

1. “Please select a search type: Author, Subject, Title [Aa/Ss/Tt]:” is displayed
2. The user enters input (for ex: ‘a’)
3. “Please enter a search term for an Author:” is displayed
4. The user enters a term
5. The program displays a list of titles by the selected author/authors/term with their indexes.
6. “Please type a title number from the above list:” is displayed
7. The user copies a book number
8. “Displaying Word Cloud in [Subject:” – “subject” – “title” – “author “ are displayed
9. The wordcloud is displayed
10. “Would you like to search again: [y/n]” is displayed
11. If user chooses y, the program continues for a new search. If the user chooses no the program stops.