**HOW-TO:**

Executable: poodle.py <- run this

Additional files: ignore.txt (edit this file and add words to ignore when scraping if you wish), graph.txt (stores url graph data structure), ranks.txt (stores rank data structure), and index.txt (index data structure)

After poodle greets you with an interesting, randomly selected message, enter “-help” to see what options await you! Otherwise, enter “-build” to get started building a database.

# Code

### Github: <https://github.com/Benjy96/Python_Search_Engine>

The **main components** are structured into “regions” within the poodle.py file. For example, if you want to find the methods related to the poodle engine architecture and UI, use control-f: - poodle –

## Most interesting code snippets:

* **Multi-keyword search:** I designed the multi-keyword search to use a recursive technique. The search method takes in the user’s entered term. If the term has a ‘,’ it is split into a list storing multiple terms. The method then calls itself, passing in individual terms as the parameter. The key here is that **only individual terms** can then progress into the actual **search operations**, printing their results. The original caller (base - function with a list) then exits, because it is not an **individual term**, and cannot enter the if statement that only accepts **individual** terms (to run the search operation). (Control-f: #RECURSIVE)
* **Poodle Index:** The main poodle functions are called using a dictionary, similar to a switch statement in other programming languages
* **Sorting search results by rank:** I use a lambda expression to set the **key** parameter of the sort method. I use a list x = [url, urlRank], and sort it by doing x.sort(key = lambda x: x[1]) – this means the parameter “x” (the list) is passed to the function, and the **first element** is returned by the lambda function. Setting this as the key means the list is **sorted by page rank**. (Control-f: lambda)

## All suggested extra functionalities complete:

* Punctuation – string translate method to remove it
* Ignore list – using a set instead of a list (faster lookup)
* Case-insensitive search – we create a new index with lower case keys
  + Search – using ***lambda*** expression to sort list by second element
* Max crawler depth set by user when building database
* Multi-keyword search – split/separated by ‘,’
  + DONE LIKE RECURSION
  + Each small, individual search term goes through the process
  + The BASE CASE FOR MULTIPLE SEARCHES stores a list of terms
    - Splits the list
    - If list exists, does nothing except call sub-searches
    - If no list, go through individual process (like sub-searches)
* Usability – POODLE command line tool
  + Invalid commands and URLs caught
  + Default Max 10 results displayed – can be changed

## Most useful development functionality not included in spec:

* Debug mode – enables “under the hood” printing to view operations as they happen (e.g. printing a crawled page as it gets crawled) – assists developer with debugging, or lets user see updates in Realtime, rather than having results printing only after they have been finished. Simply add poodleDebugOutput anywhere you want a debug message (potentially) printed, and once the user sets the debug mode, these messages will be evaluated and printed

## Note on Crawling Terminology

Whenever you set the “max depth”, I have decided that a depth of **1** will only crawl the first page. For example, 0 clicks away from the index. A depth of 2 will crawl the index, and the pages **on the** index page (1 click away each). I prefer this, as a depth of 0, in computer science terms, means do nothing.

## POODLE Options

