**Description:** A mini search engine capable of searching the data, rank them and display. It has been written in python in visual studio. It has been implemented in object-oriented programming.

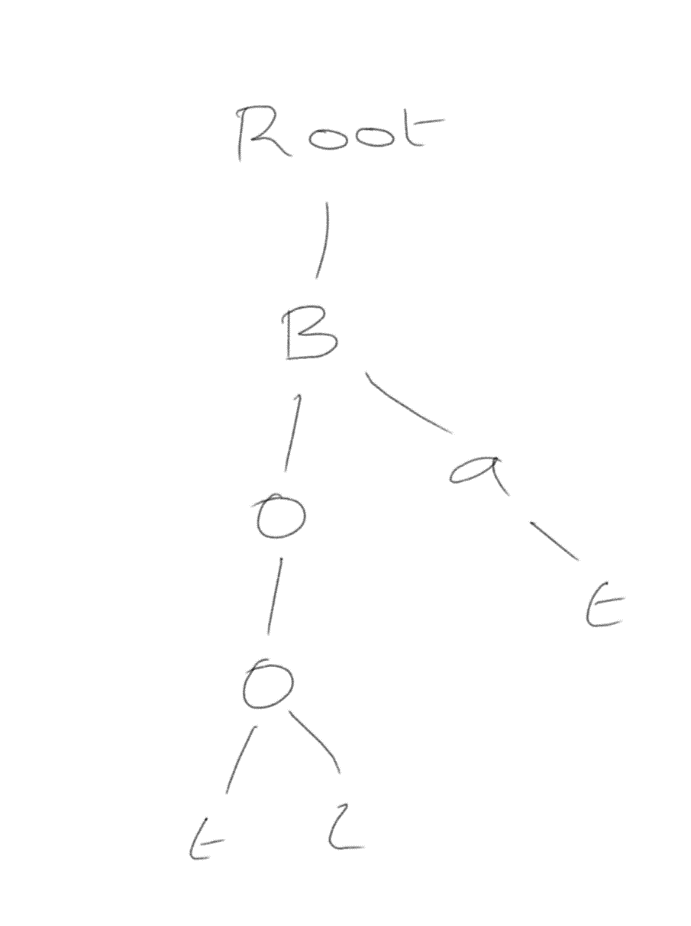
**System Requirements:** Visual studio

**Algorithms/libraries used**: Tire, Inverted File, TD-IDF, Beutifulsoup

Trie:

The Trie data structure has been implemented for stop words and document data with following structure.

Ex: boot, bat, bool



All the nodes are stored in Ascii values.

Each node also has values such if it marks end of word, is it stop word, the inverted file dictionary

Inverted File:

Every word has the documents which the appears and positions of it as key value pairs (k ,l) where k is all the documents where the word occurs and l is a list of all the positions at which the documents occur.

TF-IDF:

This is the algorithm used for the ranking of the pages. TF is Term frequency which calculates the total occurrence of the word divided by the total words. IDF is the inverse document frequency which gives the rareness of the word among all the documents

Beutifulsoup:

This library is used for extracting the data from the html file. First data is read from a local server and then passed onto the beautiful soup for extracting all the information from the file.

**Classes Description:**

Trienode: The class which has the nodes for trie. Every node in trie is the object of this database.

Trie: Contains method for insertion and searching the Trie tree

Stopwords: This classes inserts all the stop words to the tree and has a method to search using the trie class

Document: This class inserts all the document data to the trie tree. It also has a method to search the tree using trie class

Scraping: This class uses beutifulsoup for extracting the data from html files. First files are loaded from local host and then scraped for the purpose

InitializeEngine: This classes initializes the search engine by calling functions from stopword and documents so all the data required for performing search queries is gathered

Ranking: This classes computes the TF-IDF for the every word which is passed to it.

Search: This class computes the final rank of every page by computing ranks for every word and calculating the final ranks

PresentationLayer: This layer gets all the ranks of the documents and prints a brief description of the page and the URL of the page

**Steps for running the program:**

* Open the file CS600.sln which is solution file for Microsoft visual studio
* Run the program
* Type the query in console window
* For the first there may be 1-2 seconds delay as the search engine is being loaded with all the data
* From the second query the outputs are fast
* Type exit if you want to close the program

**Control Flow:**

1. First the data is initialized i.e., The stopword and documents are loaded into Trie tree
2. Each stopword is inserted into the trie
3. For every document, data is scraped using the scraping class and inserted into Trie
4. When the Query is passed each word is striped from it and rank is formed
5. Ranks for all the words in query and final rank is computed
6. Description is scraped from all the URL which has rank more than zero
7. The description and URL is printed on the console based on their ranks.

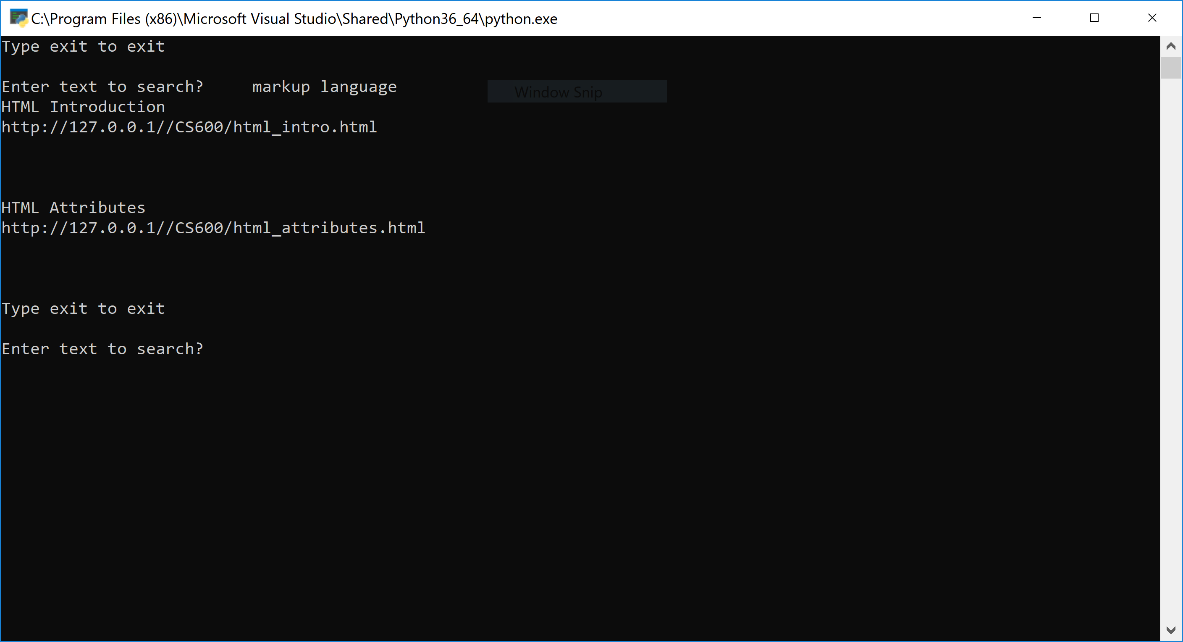
**Input data:**

Input data is taken from w3schools html tutorials. It is of 10 web pages in which all the web pages all inter linked

**Output:**

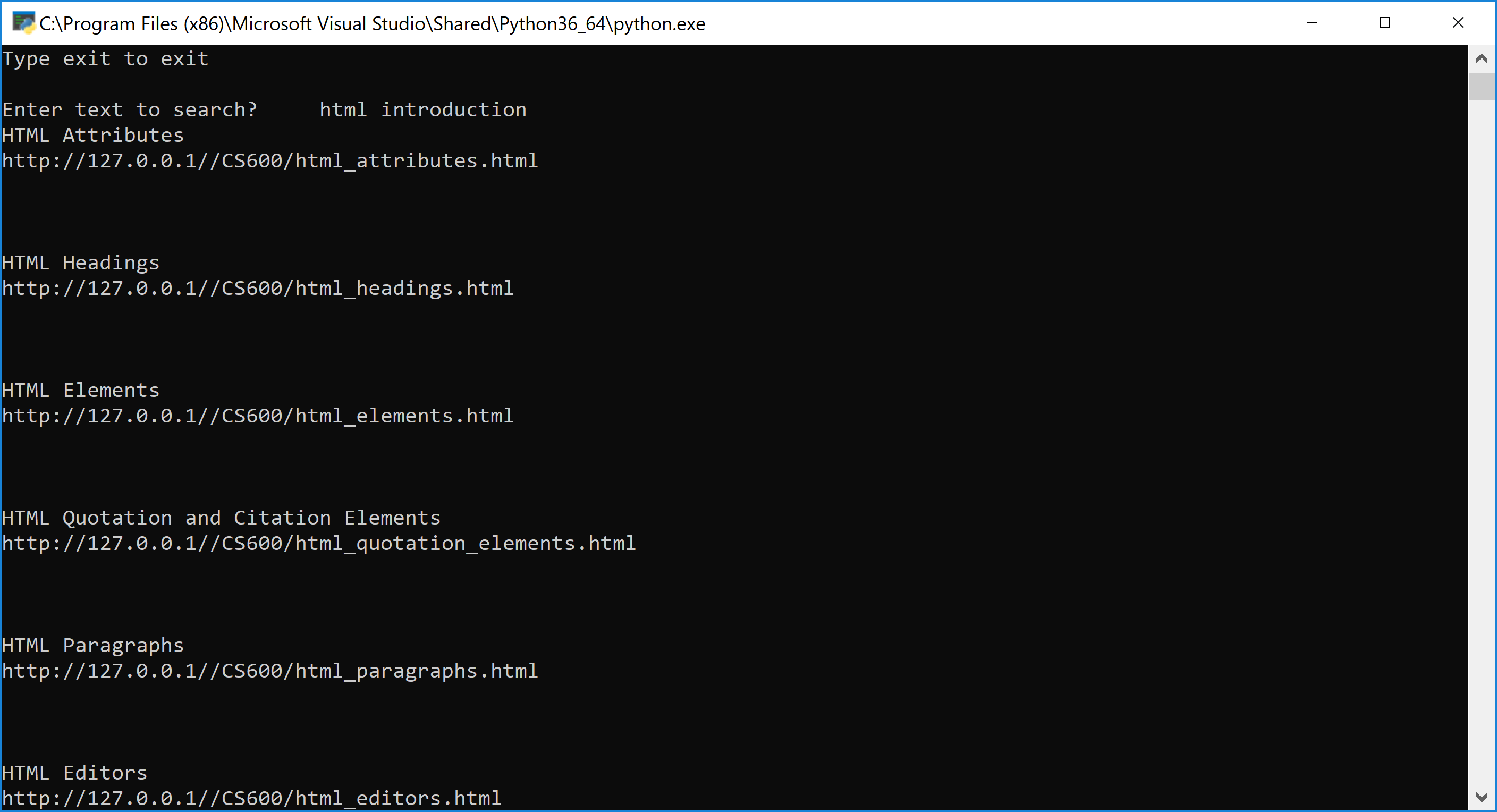
Query: markup language

Output:



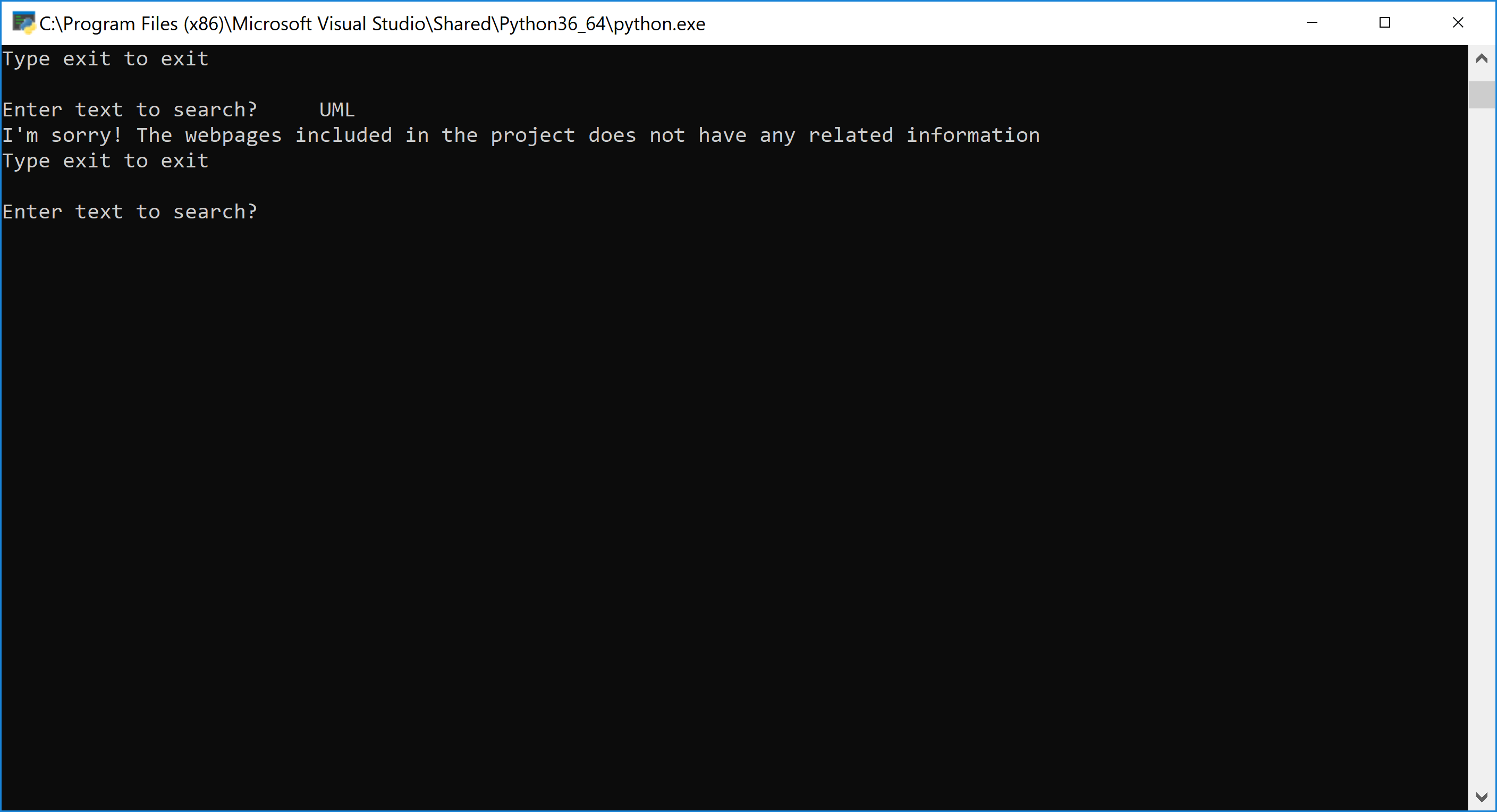
Query: html introduction

Output:



Query: UML

Output:



Query: exit

Output:

