Project Documentation – SI 507 Final Project

Project code
Link to github: https://github.com/mdhamid1/finalproject_SI507

README info:

REQUIRED PACKAGES (shown via import statements in code):

```
import requests
import time
import hashlib
import json
from bs4 import BeautifulSoup
import sqlite3
import plotly.graph_objects as go
```

SPECIAL INSTRUCTIONS:
If the marvel_characters.db file does not exist and the program is being run for the first time then only the below two functions should be run. Currently these functions are commented out in the code to avoid running these functions by default.
-create_database_table() - this function will newly create two tables in the .db file. If this funciton is run repeatedly then it will overwrite the data input on each run into the database. As such ** only run this function if the marvel_characters.db file does not exist

-populate_database() - this function will populate the database with all the data in the caching json file. This funciton should also only be run once, after all the data has been scraped from the marvel website. Running this function repeatedly will keep adding redundant data into the database.

In order to run the program, no additional information is required. I am using an API to extract information about the events that various Marvel characters have been present in. This API requires four parameters to run:
-'apikey'
-'ts' - this is the timestamp
-'hash' - a hash value created from the md5 hash function. In order to obtain the appropriate hash value for the API to work the timestamp + private api key + public api key must be passed into the hash function.
-'name' - name of marvel character

All four of these parameters have been calcualted in the finalproject.py script within the get_event_info() function and no further action is required.

HOW TO INTERACT WITH PROGRAM:
Interacting with the program is very simple. The program will ask the user to enter the name of a Marvel character. If the Marvel character is found in the database then the program will return a

console summary of the character that includes its name, link to character webpage, powers, and events the character was present in. In addition, the program will also create a bar plot for the character that summarizes the powers of the character. This way the program allows an easy way to compare the powers of each Marvel character as the user can scroll through the various tabs for each Marvel character. If the marvel character is not found in the database, or if the entry is invalid then the program will ignore the entry and ask for another input. If the user types in "exit" the program will thank the user and exit.

Data Sources

Most of the character information has been extracted from the marvel website: https://www.marvel.com/characters

Origin of Data/How I extracted the Data/Initial Data Format:

There are several pages that have been scraped/crawled. The first set is 73 pages from the above link. The name of each character has been extracted from the above page and saved to a dictionary called character_dict. This leads to almost 3000 marvel characters. Then for each of these almost 3000 characters the script follows a link to their information page. It saves the link to this page in character_dict in a location corresponding to the character name. Then the powers of the marvel character are scraped from this page and saved in the character_dict. An example of this page is : https://www.marvel.com/characters/3-d-man-chandler.
The strategy I used to extract the data was as follows: at https://www.marvel.com/characters there were 73 pages, however there was no link allowing me to access the next pages in the code of the webpage. This meant that I would be unable to move through the various pages using BeautifulSoup. The solution to this was that while I was checking the network tab in the Inspect properties interface on Google Chrome I noticed that clicking on the page number at the bottom of the webpage was receiving a response in the form of a URL I could access. Clicking on this URL in the network tab of the inspect property took me to a page that had a JSON representation of information about the Marvel characters' names. This URL had a offset and limit value parameter. Varying the offset value allowed me to access each new page, with the rest of the URL remaining unchanged. (An example of this URL: https://www.marvel.com/v1/pagination/grid_cards?offset=38&limit=38&entityType=character&sortField=title&sortDirection=asc). I wrote a for loop to change the offset value on each iteration and passed that value into an f-string containing the URL string such that I could access the new page to extract the next set of character names when needed.
After this was done I was able to use BeautifulSoup to access the links to the Marvel characters' information page and scrape/crawl that to extract information about their powers. After this, I created a function called get_event_info() that called the Marvel API and allowed me to access the major events in the comics each Marvel character was a part of. All of the information that was scraped and also called from the API was saved in the character_dict dictionary. I also implemented caching such that once a character name and information link was scraped this information would be saved in the character_dict and this character_dict would be saved in a JSON file called character_cache.json. **It should be noted that scraping the internal links to all 3000 Marvel characters was taking too long so I ended up scraping the internal links for**

about 360 Marvel characters. After this, I created two tables by passing SQL create table queries via python to the database. Once the tables were created I wrote SQL insert statements to put the data in the tables.

<ins>Summary of Data:</ins>

Records available: almost 3000 + an individual internal link related to each of these almost 3000 characters.

Records retrieved: 360 character names + powers information from 360 individual internal links

<ins>Description of Records:</ins>

Character name (string): name of Marvel character. This will be a text field for my project and will be in the first table in the database.

Events (dictionary): contains events character has been involved in, and link to more info about the event. Events will be in a second table in the database. The event field will be a string. There will be a foreign key in this table that corresponds to the primary key Id for the marvel character in the first table in the database.

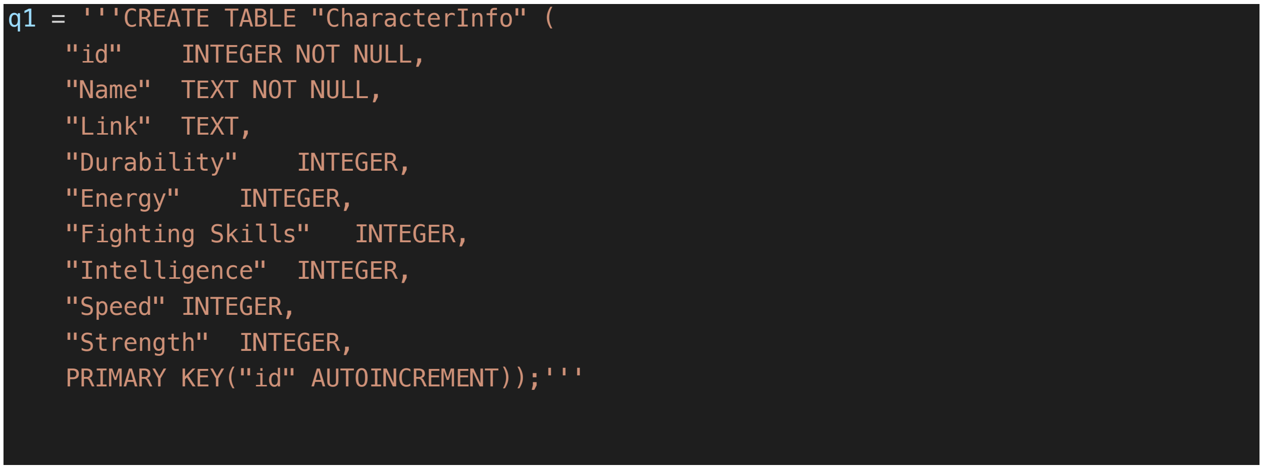
Link (string): link to more information about the character. This will be a text field in the first table in the database.

Powers (dictionary): a dictionary that consists of string and integer value pairs corresponding to the powers of the character. There will be a new field in the database for each power. This field will be an integer field and will be in the first table.

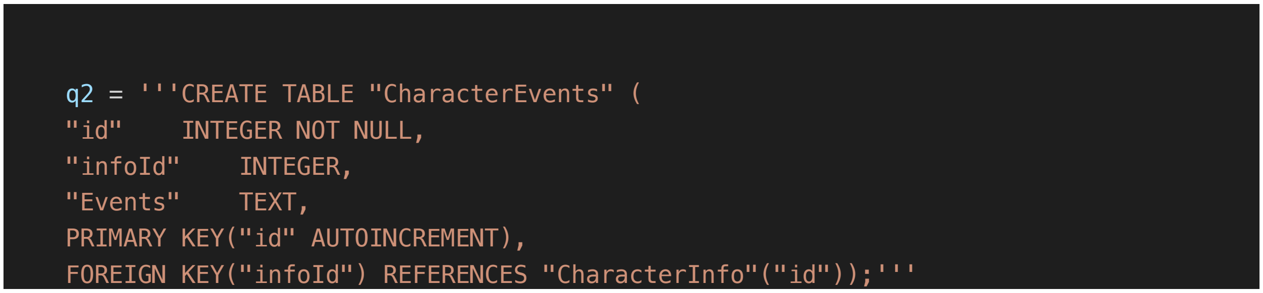
Database

<ins>Database Scheme:</ins>

Copied from script:



```
q1 = '''CREATE TABLE "CharacterInfo" (
    "id"    INTEGER NOT NULL,
    "Name"  TEXT NOT NULL,
    "Link"  TEXT,
    "Durability"    INTEGER,
    "Energy"    INTEGER,
    "Fighting Skills"   INTEGER,
    "Intelligence"  INTEGER,
    "Speed" INTEGER,
    "Strength"  INTEGER,
    PRIMARY KEY("id" AUTOINCREMENT));'''
```



```
q2 = '''CREATE TABLE "CharacterEvents" (
"id"    INTEGER NOT NULL,
"infoId"    INTEGER,
"Events"    TEXT,
PRIMARY KEY("id" AUTOINCREMENT),
FOREIGN KEY("infoId") REFERENCES "CharacterInfo"("id"));'''
```

The foreign key is present in the CharacterEvents (second table) table. The foreign key is linked to the primary key in the CharacterInfo (first table) table. This foreign key denotes which characters were mentioned in which events. It is a manty to many relationship.

Screenshots of data from Database:

CharacterInfo table (null values indicate that there were no powers found for this character on the website):

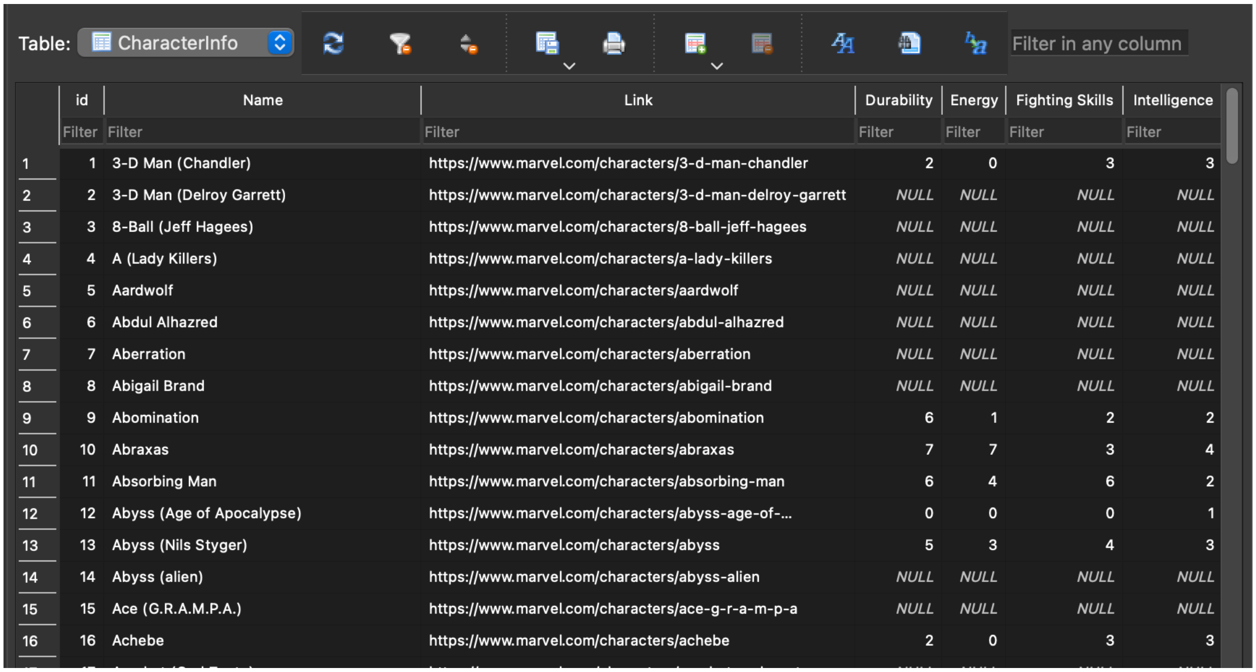
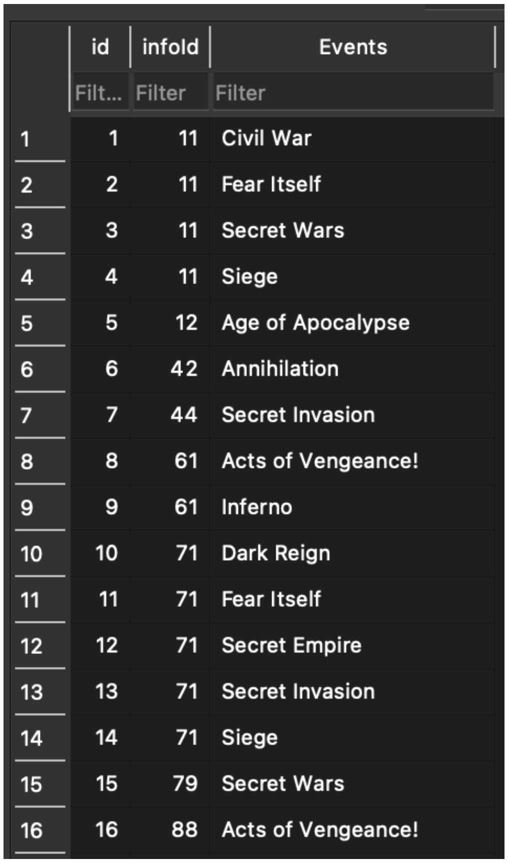


Table: CharacterInfo

	id	Name	Link	Durability	Energy	Fighting Skills	Intelligence
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	3-D Man (Chandler)	https://www.marvel.com/characters/3-d-man-chandler	2	0	3	3
2	2	3-D Man (Delroy Garrett)	https://www.marvel.com/characters/3-d-man-delroy-garrett	NULL	NULL	NULL	NULL
3	3	8-Ball (Jeff Hagees)	https://www.marvel.com/characters/8-ball-jeff-hagees	NULL	NULL	NULL	NULL
4	4	A (Lady Killers)	https://www.marvel.com/characters/a-lady-killers	NULL	NULL	NULL	NULL
5	5	Aardwolf	https://www.marvel.com/characters/aardwolf	NULL	NULL	NULL	NULL
6	6	Abdul Alhazred	https://www.marvel.com/characters/abdul-alhazred	NULL	NULL	NULL	NULL
7	7	Aberration	https://www.marvel.com/characters/aberration	NULL	NULL	NULL	NULL
8	8	Abigail Brand	https://www.marvel.com/characters/abigail-brand	NULL	NULL	NULL	NULL
9	9	Abomination	https://www.marvel.com/characters/abomination	6	1	2	2
10	10	Abraxas	https://www.marvel.com/characters/abraxas	7	7	3	4
11	11	Absorbing Man	https://www.marvel.com/characters/absorbing-man	6	4	6	2
12	12	Abyss (Age of Apocalypse)	https://www.marvel.com/characters/abyss-age-of-...	0	0	0	1
13	13	Abyss (Nils Styger)	https://www.marvel.com/characters/abyss	5	3	4	3
14	14	Abyss (alien)	https://www.marvel.com/characters/abyss-alien	NULL	NULL	NULL	NULL
15	15	Ace (G.R.A.M.P.A.)	https://www.marvel.com/characters/ace-g-r-a-m-p-a	NULL	NULL	NULL	NULL
16	16	Achebe	https://www.marvel.com/characters/achebe	2	0	3	3

CharacterEvents table:


Filt...
Filter
Filter

	id	infold	Events
1	1	11	Civil War
2	2	11	Fear Itself
3	3	11	Secret Wars
4	4	11	Siege
5	5	12	Age of Apocalypse
6	6	42	Annihilation
7	7	44	Secret Invasion
8	8	61	Acts of Vengeance!
9	9	61	Inferno
10	10	71	Dark Reign
11	11	71	Fear Itself
12	12	71	Secret Empire
13	13	71	Secret Invasion
14	14	71	Siege
15	15	79	Secret Wars
16	16	88	Acts of Vengeance!

Interaction and Presentation Options

How to inteact with the program

Interacting with the program is very simple. The program will ask the user to enter the name of a Marvel character. If the Marvel character is found in the database then the program will return a console summary of the character that includes its name, link to character webpage, powers, and events the character was present in. In addition, the program will also create a bar plot for the character that summarizes the powers of the character. This way the program allows an easy way to compare the powers of each Marvel character as the user can scroll through the various tabs for each Marvel character. If the marvel character is not found in the database, or if the entry is invalid then the program will ignore the entry and ask for another input. If the user types in "exit" the program will thank the user and exit.

Interactive presentation technologies

-Plotly - used for creating bar plots of the characters' powers

-command line prompts – used for asking user which character they would like to know about as well as exiting the program