Project Name:

Messy Data-Cleaning Disney-Movie-Dataset With Imdbapi

Table of Contents

Demo

Load the webpage

In [2]:
```
r = requests.get("https://en.wikipedia.org/wiki/Toy_Story_3")

# Convert to a beautiful soup object
soup = bs(r.content)

# Print out the HTML
contents = soup.prettify()
print(contents)
```

```
<!DOCTYPE html>
<html class="client-nojs" dir="ltr" lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   Toy Story 3 - Wikipedia
  </title>
  <script>
   document.documentElement.className="client-js";RLCONF={"wgBreakFrames":!1,"wgSeparatorTransformTable":["",""],"wgDigitTran
sformTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","February","March","April","May","June","Jul
y","August","September","October","November","December"],"wgRequestId":"d58ddccf-39fa-4477-b7bf-13af1060304a","wgCSPNonce":!
1,"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":!1,"wgNamespaceNumber":0,"wgPageName":"Toy_Story_3","wgTitle":"Toy S
tory 3","wgCurRevisionId":982658067,"wgRevisionId":982658067,"wgArticleId":1213838,"wgIsArticle":!0,"wgIsRedirect":!1,"wgActi
on":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["Wikipedia indefinitely semi-protected pages","Articles wit
h short description","Short description matches Wikidata","Good articles","Use American English from June 2020","All Wikipedi
a articles written in American English","Use mdy dates from June 2020","Template film date with 2 release dates",
"Articles with hAudio microformats","Album infoboxes lacking a cover","Album articles lacking alt text for covers","Album cha
rt usages for Mexico","CS1 maint: ref=harv","Commons category link from Wikidata","Official website different in Wikidata and
Wikipedia","Wikipedia articles with LCCN identifiers","Wikipedia articles with VIAF identifiers","Wikipedia articles with Wor
ldCat-VIAF identifiers","2010 films","English-language films","2010 3D films","2010 computer-animated films","2010s American
```

In [8]:
```
get_info_box("https://en.wikipedia.org/wiki/One_Little_Indian_(film)")
```

Out[8]:
```
{'title': 'One Little Indian',
 'Directed by': 'Bernard McEveety',
 'Produced by': 'Winston Hibler',
 'Written by': 'Harry Spalding',
 'Starring': ['James Garner',
  'Vera Miles',
  'Pat Hingle',
  'Morgan Woodward',
  'Jodie Foster'],
 'Music by': 'Jerry Goldsmith',
 'Cinematography': 'Charles F. Wheeler',
 'Edited by': 'Robert Stafford',
 'Production company': 'Walt Disney Productions',
 'Distributed by': 'Buena Vista Distribution',
 'Release date': ['June 20, 1973'],
 'Running time': '90 Minutes',
 'Country': 'United States',
 'Language': 'English',
 'Box office': '$2 million'}
```

In [18]:
```
print([movie.get('Running time (int)', 'N/A') for movie in movie_info_list])
```

```
[41, 83, 88, 126, 74, 64, 70, 42, 65, 71, 75, 94, 73, 75, 82, 68, 74, 96, 75, 84, 77, 92, 69, 81, 60, 127, 92, 76, 75, 73, 85,
81, 70, 90, 80, 75, 83, 83, 72, 97, 75, 104, 93, 105, 95, 97, 134, 69, 92, 126, 79, 97, 128, 74, 91, 105, 98, 130, 89, 93, 67,
98, 100, 118, 103, 110, 80, 79, 91, 91, 97, 118, 139, 92, 131, 87, 116, 93, 110, 110, 131, 101, 108, 84, 78, 75, 164, 106, 110,
99, 113, 108, 112, 93, 91, 93, 100, 100, 79, 96, 113, 89, 118, 92, 88, 92, 87, 93, 93, 93, 90, 83, 96, 88, 89, 91, 93, 92, 97,
100, 100, 89, 91, 112, 115, 95, 91, 95, 104, 74, 48, 77, 104, 128, 101, 94, 104, 90, 100, 88, 93, 98, 100, 112, 84, 98, 97, 11
4, 96, 100, 109, 83, 90, 107, 96, 103, 91, 95, 105, 113, 80, 101, 89, 74, 90, 89, 110, 74, 93, 84, 83, 69, 77, 107, 93, 88, 10
8, 84, 121, 89, 104, 90, 86, 84, 108, 107, 96, 98, 105, 108, 94, 106, 102, 88, 102, 102, 97, 111, 100, 96, 98, 78, 81, 108, 89,
99, 89, 81, 92, 100, 89, 79, 91, 101, 104, 103, 86, 105, 93, 92, 98, 95, 93, 87, 93, 87, 128, 86, 95, 114, 93, 83, 83, 88, 78,
112, 92, 74, 77, 82, 104, 113, 100, 78, 83, 96, 115, 86, 92, 99, 73, 128, 85, 88, 125, 96, 104, 95, 72, 75, 61, 117, 94, 100, 1
43, 97, 85, 86, 50, 74, 136, 89, 76, 40, 120, 84, 113, 115, 131, 100, 68, 95, 97, 101, 119, 100, 76, 120, 81, 143, 106, 40, 12
0, 99, 82, 117, 151, 104, 76, 92, 95, 94, 168, 111, 82, 87, 110, 107, 124, 74, 83, 150, 97, 91, 100, 111, 96, 99, 76, 98, 97, 9
9, 96, None, 90, 101, None, 101, 96, 88, 97, 108, 104, 116, 103, 109, 115, 74, 123, 108, 100, 101, 125, None, 90, 88, 109, 89,
104, 137, 106, 69, 103, 95, 132, 77, 96, 93, 104, 87, 101, 130, 77, 104, 149, 92, 102, 125, 107, 77, 124, 97, 84, 127, 81, 102,
124, 129, 106, 82, 130, 94, 154, 94, 117, 108, 106, 98, 114, 97, 117, 103, 124, 107, 161, 80, 129, 76, 129, 102, None, 162, 10
5, 109, 118, 104, 99, 112, 131, 112, 76, 128, 100, 118, 119, 104, 100, 103, 114, 99, 102, 107, 78, 89, 95, 160, 85, 100, 95, 8
7, 115, 99, None, 100, None, None, None, None]
```

In [24]:
```
movie_info_list[-50]
```

Out[24]:
```
{'title': 'Alice Through the Looking Glass',
 'Directed by': 'James Bobin',
 'Produced by': ['Joe Roth', 'Suzanne Todd', 'Jennifer Todd', 'Tim Burton'],
 'Written by': 'Linda Woolverton',
 'Based on': ['Characters', 'by', 'Lewis Carroll'],
 'Starring': ['Johnny Depp',
  'Anne Hathaway',
  'Mia Wasikowska',
  'Rhys Ifans',
  'Helena Bonham Carter',
  'Sacha Baron Cohen',
  'Alan Rickman',
  'Stephen Fry',
  'Michael Sheen',
  'Timothy Spall'],
 'Music by': 'Danny Elfman',
 'Cinematography': 'Stuart Dryburgh',
 'Edited by': 'Andrew Weisblum',
 'Production company': ['Walt Disney Pictures',
  'Roth Films',
  'Team Todd',
  'Tim Burton Productions'],
 'Distributed by': ['Walt Disney Studios', 'Motion Pictures'],
 'Release date': ['May 10, 2016 ( London )', 'May 27, 2016 (United States)'],
 'Running time': '114 minutes',
 'Country': 'United States',
 'Language': 'English',
 'Budget': '$170 million',
 'Box office': '$300 million',
 'Running time (int)': 114,
 'Budget (float)': 170000000.0,
 'Box office (float)': 300000000.0}
```

```
In [47]: running_times = df.sort_values(['Running time (int)'],  ascending=False)
         running_times.head(20)
```

Out[47]:

	title	Production company	Release date	Running time	Country	Language	Box office	Running time (int)	Budget (float)	Box office (float)	...	Narrated by	Cinematography	
302	Pirates of the Caribbean: At World's End	NaN	[May 19, 2007 (Disneyland Resort), May 25, 2...	168 minutes	United States	English	$961 million	168.0	300000000.0	9.610000e+08	...	NaN	Dariusz Wolski	
86	The Happiest Millionaire	Walt Disney Productions	[June 23, 1967, November 30, 1967]	[164 minutes, (, Los Angeles, premiere), 144 m...	United States	English	$5 million (US/ Canada rentals)	164.0	5000000.0	5.000000e+06	...	NaN	Edward Colman	
401	Jagga Jasoos	[Walt Disney Pictures India, Picture Shuru Ent...	[14 July 2017]	162 minutes	India	Hindi	833.5 million	162.0	NaN	NaN	...	NaN	Ravi Varman	
394	Dangal	[Aamir Khan Productions, Walt Disney Pictures ...	[21 December 2016 (United States), 23 December...	161 minutes	India	Hindi	[est., (,)]	161.0	NaN	NaN	...	Aparshakti Khurana	Setu (Satyajit Pande)	E
425	Hamilton	[Walt Disney Pictures, 5000 Broadway Productio...	[July 3, 2020]	160 minutes	United States	English	NaN	160.0	12500000.0	NaN	...	NaN	Declan Quinn	
382	ABCD 2	Walt Disney Pictures	[19 June 2015]	154 minutes	India	Hindi	est.	154.0	NaN	NaN	...	NaN	Vijay Kumar Arora	

Overview

This is diving into Similar to Real World Data Science Task of Data Cleaning Concept for movie dataset creation.

Cleaning your data, all outdated or incorrect information is gone – leaving you with the highest quality information. Data cleaning is a process used to determine inaccurate, incomplete, or unreasonable data and then improve quality by correcting detected errors and omissions.

This repository contains the code for Data Cleaning using python's various libraries.

It used bs4, json, pickle, re, omdbapi, os, urllib, pandas and requests libraries.

These libraries help to perform individually one particular functionality.

Pandas objects rely heavily on Numpy objects.

JSON is commonly used for transmitting data in web applications.

"Pickling" is the process whereby a Python object hierarchy is converted into a byte stream.

A *regular expression* is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern.

The purpose of creating this repository is to gain insights into process of cleaning data.

These python libraries raised knowledge in discovering these libraries with practical use of it.

It leads to growth in my ML repository.

This above few screenshots will help you to understand flow of output.

Motivation

The reason behind building this is, data scientists says that the initial steps of obtaining and cleaning data constitute 80% of the job. Therefore, if I am just stepping into this field or planning to step into this field, it is important to be able to deal with messy data, whether that means

missing values, inconsistent formatting, malformed records, or nonsensical outliers. Data cleansing is also important because it improves your data quality and in doing so, increases overall productivity. Since data is a major asset in many companies, inaccurate data can be dangerous. Incorrect data can reduce marketing effectiveness, thereby bringing down sales and efficiency. If the organization had clean data, then falling into such situations can be avoided. And data cleaning is the way to go. It removes major errors and inconsistencies that are inevitable when multiple sources of data are getting pulled into one dataset. Using tools to clean up data will make everyone more efficient. Fewer errors mean happier customers and fewer frustrated employees. Increased productivity and better decisions are other benefits of using data cleaning. Some general guidelines that all companies can follow to data clean include forming a data quality plan. By standardizing the data process, one will ensure a good point of entry and reduce the risk of duplication. Monitoring errors and fixing the data at the source can save both time and resources. Investing in tools that measure data accuracy is another wise way that can be adopted. A reliable third-party source can capture information directly from first-party sites. It would then clean and compile the data to provide more complete information for business intelligence and analytics. Hence, I continue to gain knowledge while practicing the same and spread literary wings in tech-heaven.

Technical Aspect

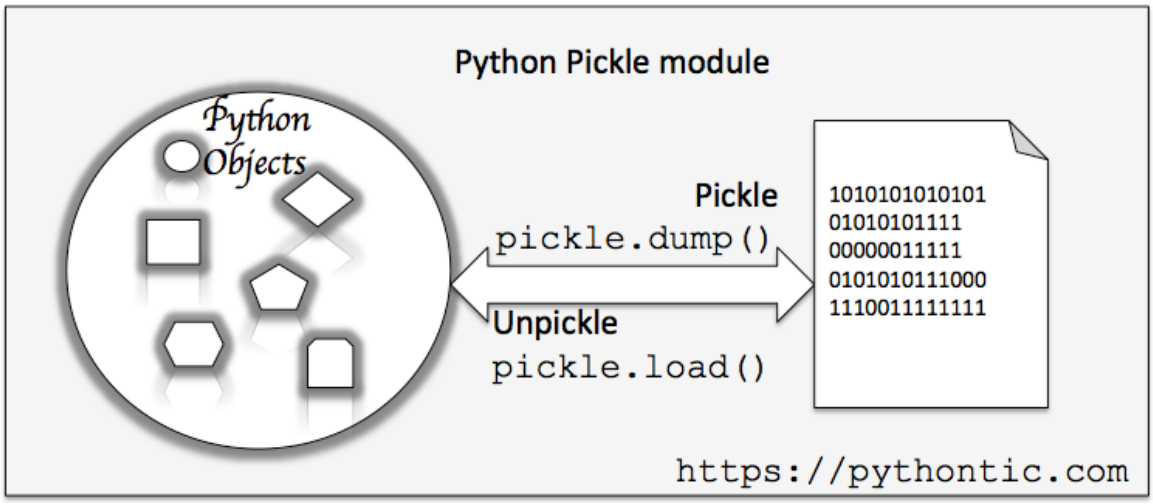
Pandas module mainly works with the tabular data. It contains DataFrame and Series. Pandas is 18 to 20 times slower than Numpy. Pandas is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data.

With requests module, you can add content like headers, form data, multipart files, and parameters via simple Python libraries. It also allows you to access the response data of Python in the same way. The requests library is the de facto standard for making HTTP requests in Python. It abstracts the complexities of making requests behind a beautiful, simple API so that you can focus on interacting with services and consuming data in your application.

BS4 commonly saves programmers hours or days of work. you have to pass something to BeautifulSoup to create a soup object. That could be a document or an URL. Beautiful Soup also relies on a parser, the default is lxml.

JavaScript Object Notation (JSON) is a standardized format commonly used to transfer data as text that can be sent over a network. It's used by lots of APIs and Databases, and it's easy for both humans and machines to read. JSON represents objects as name/value pairs, just like a Python dictionary. JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.



A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression.
Omdbapi module gets movie, series, episode data from the Open Movie Database (OMDb) api.

The gateway from Python to web is done through urllib module. It is a Python module for fetching URLs (Uniform Resource Locators). It offers a very simple interface, in the form of the *urlopen* function. This is capable of fetching URLs using a variety of different protocols.

os module provides a portable way of using operating system dependent functionality.

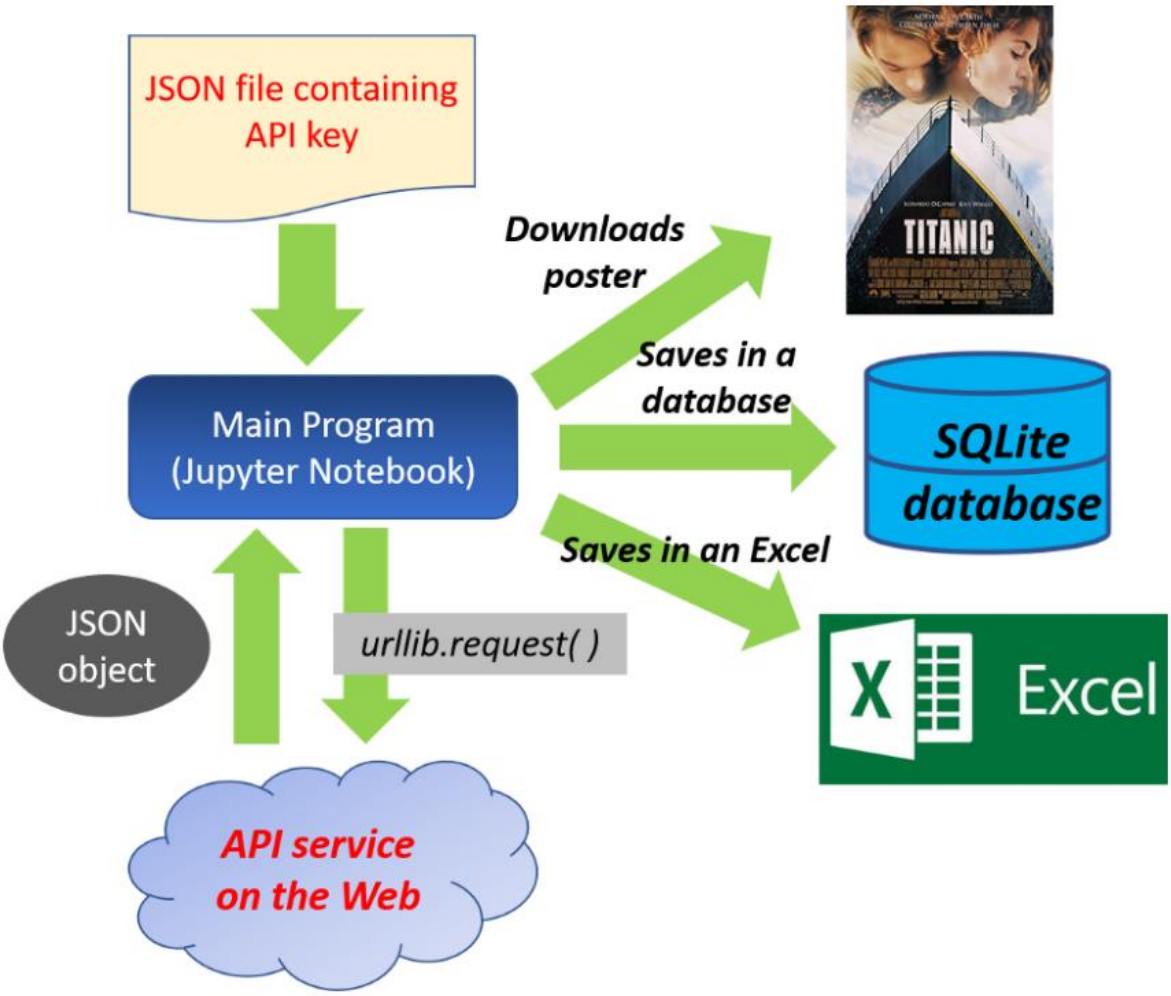
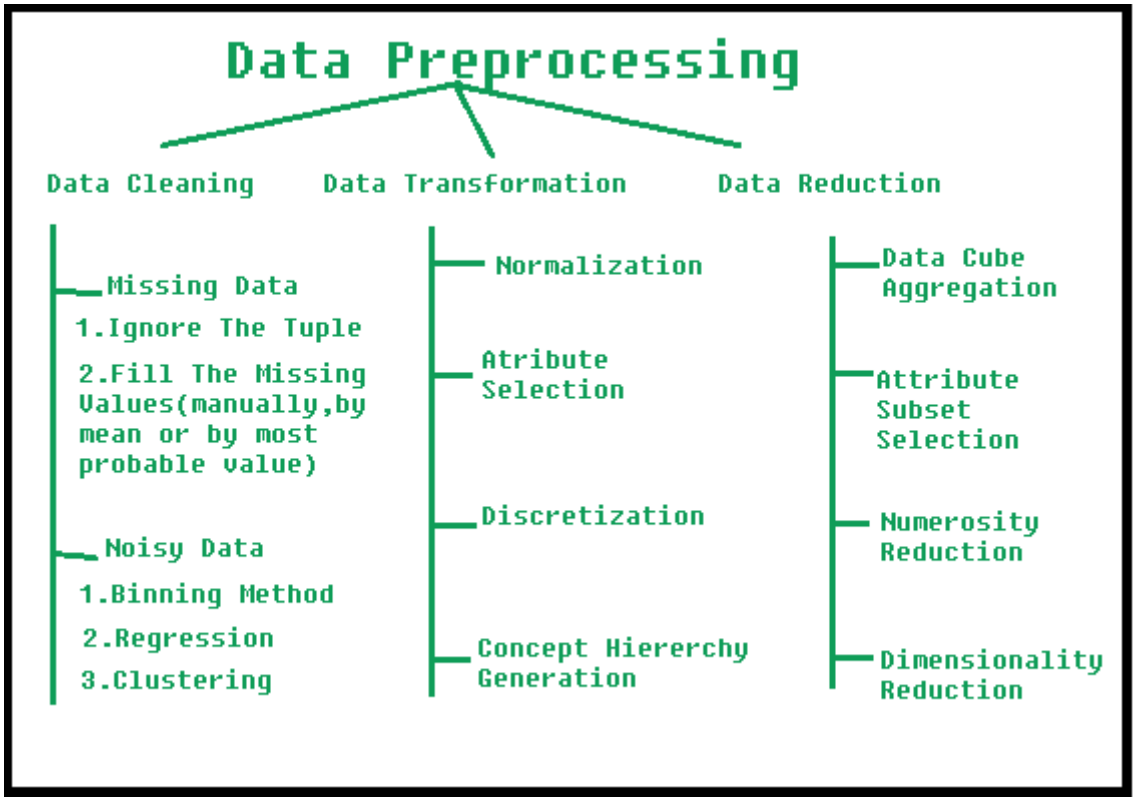


Fig: Main flow of the program

The basic idea is to send request to external API with a movie title that is entered by the user. The program then tries to download the data and if successful, prints it out.

You will notice that the program uses a secret API key for accessing the data.



Installation

Using intel core i5 9th generation with NVIDIA GFORECE GTX1650.
Windows 10 Environment Used.
Already Installed Anaconda Navigator for Python 3.x
The Code is written in Python 3.8.
If you don't have Python installed then please install Anaconda Navigator from its official site.
If you are using a lower version of Python you can upgrade using the pip package, ensuring you have the latest version of pip, *python* -m pip install --*upgrade pip and press Enter*.

Run/How to Use/Steps

Keep your internet connection on while running or accessing files and throughout too.
Follow this when you want to perform from scratch.

Create account here.

There's free API key available for OMDb API now.

http://www.omdbapi.com/apikey.aspx

Then check your mail, you will receive OMDb API Link.

Open Anaconda Prompt, Perform the following steps:

cd <PATH>
pip install bs4

pip install pandas
pip install requests
pip install omdbapi
pip install urllib3

You can also create requirement.txt file as, pip freeze > requirements.txt

run files.

Follow this when you want to just perform on local machine.
Download ZIP File.
Right-Click on ZIP file in download section and select Extract file option, which will unzip file.
Move unzip folder to desired folder/location be it D drive or desktop etc.
Open Anaconda Prompt, write cd <PATH> and press Enter.

eg: cd C:\Users\Monica\Desktop\Projects\Python Projects 1\16)Disney-data-science-messy-data-clean

In Anconda Prompt, pip install -r requirements.txt to install all packages.
Open in Jupyter Notebook, <filename>.ipynb
That is,

Open in Jupyter Notebook, 1)DataCleaning_DisneyMovieDataset_with_imdbapi.ipynb

This creates multiple files, Disney_data_final_dataset in 3 formats .csv, .json and .pickle. Also creates two checkpoints. All gets created in same working folder and then I created empty folder named Output_Files and transferred there.
Please be careful with spellings or numbers while typing filename and easier is just copy filename and then run it to avoid any silly errors.
Note: cd <PATH>
[Go to Folder where file is. Select the path from top and right-click and select copy option and paste it next to cd one space <path> and press enter, then you can access all files of that folder] [cd means change directory]

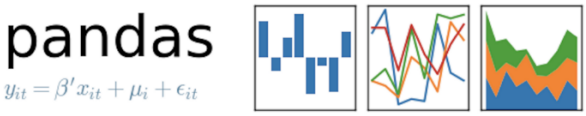
Directory Tree/Structure of Project

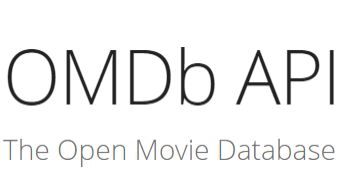
Folder: 16)Disney-data-science-messy-data-clean
1)DataCleaning_DisneyMovieDataset_with_imdbapi.ipynb
Folder: 16)Disney-data-science-messy-data-clean > helper
conversion.py
conversion_complete.py
save_and_load_dataset_checkpoints.py
test_money_conversion.py

To Do/Future Scope

can do analysis on dataset created.

Technologies Used/System Requirements/Tech Stack





Credits

KeithGalli Channel

https://realpython.com/python-data-cleaning-numpy-pandas/

https://medium.com/daily-python/python-script-to-consume-the-omdb-api-daily-python-15-aa9457f6d090

https://towardsdatascience.com/step-by-step-guide-to-build-your-own-mini-imdb-database-fc39af27d21b