Week 6 & 7

July 30, 2023

Data Wrangling with Python: Activity 9, page 294

1. Import the necessary libraries, including regex and beautiful soup.

[2]:
```
import urllib.request, urllib.parse, urllib.error
import requests
from bs4 import BeautifulSoup
import ssl
import re
```

2. Check the SSL Certificate

[3]:
```
# Ignore SSL certificate errors
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE
```

3. Read the HTML from the URL

[4]:
```
# Creates url and gets respons
url = 'https://www.gutenberg.org/browse/scores/top#books-last1'
response = requests.get(url)
```

4. Write a small function to chek the status of the web request

[5]:
```
# Checks status of request
def status_check(r):
    if r.status_code==200:
        print("Success!")
        return 1
    else:
        print("Failed!")
        return -1
```

[6]:
```
status_check(response)
```

```
Success!
```

[6]:
```
1
```

5. Decode the respose and pass this on to BeautifulSoup for HTML parsing.

[7]:
```
# Decodes response
decoded = response.content.decode(response.encoding)
```

[8]:
```
# Creates soup object
soup = BeautifulSoup(decoded, 'html.parser')
```

6. Find all the **href** tags and store them in the list of links. Check what the list looks like- print the first 30 elements.

[9]:
```
# Blank list
links_list = []
```

[10]:
```
# Find all the href tags and append them in the list of links
for link in soup.find_all('a'):
    links_list.append(link.get('href'))
```

[11]:
```
links_list[:30]
```

[11]:
```
['/',
 '/about/',
 '/about/',
 '/policy/collection_development.html',
 '/about/contact_information.html',
 '/about/background/',
 '/policy/permission.html',
 '/policy/privacy_policy.html',
 '/policy/terms_of_use.html',
 '/ebooks/',
 '/ebooks/',
 '/ebooks/bookshelf/',
 '/browse/scores/top',
 '/ebooks/offline_catalogs.html',
 '/help/',
 '/help/',
 '/help/copyright.html',
 '/help/errata.html',
 '/help/file_formats.html',
 '/help/faq.html',
 '/policy/',
 '/help/public_domain_ebook_submission.html',
 '/help/submitting_your_own_work.html',
 '/help/mobile.html',
 '/attic/',
 '/donate/',
 '/donate/',
 '#books-last1',
```

```
 '#authors-last1',
 '#books-last7']
```

[38]:
```
# The first book
links_list[33]
```

[38]:
```
'/ebooks/8086'
```

[37]:
```
# The last book
links_list[132]
```

[37]:
```
'/ebooks/35'
```

7. Use regular expression to find the numeric digits in these links. These are the file numbers for the top 100 eBooks.

8. Initialize the empty list to hold the file numbers over an appropriate range and use **regex** to find the numeric digits in the link **href** string. Use **finall** method.

[23]:
```
# Creates blank list
book_numbers = []
```

[27]:
```
# Loops through the range of books
for i in range(33,132):
    # Takes link
    link=links_list[i]
    # Strips link
    link=link.strip()
    # Regular expression to find the numeric digits in the link (href) string
    n=re.findall('[0-9]+',link)
    if len(n)==1:
        # Append the number to list
        book_numbers.append(int(n[0]))
```

[36]:
```
print('File Numbers for books: \n', book_numbers)
```

```
File Numbers for books:
 [8086, 1513, 2701, 2641, 145, 37106, 100, 16389, 67979, 6761, 394, 2160, 1259,
4085, 6593, 5197, 1342, 26184, 71255, 11, 71254, 3188, 84, 1661, 345, 174,
71253, 1232, 71252, 1184, 71256, 98, 5200, 35899, 2600, 4300, 64317, 5998,
28054, 1399, 27827, 71257, 1952, 1998, 30254, 1727, 2554, 1080, 2542, 21415,
71261, 6130, 74, 821, 3206, 2680, 2591, 58585, 844, 1400, 4363, 1497, 1260, 996,
120, 76, 158, 45, 42108, 43, 24869, 244, 768, 71258, 205, 5740, 55, 71260,
33283, 8800, 514, 135, 40686, 2814, 161, 71, 25344, 36, 67098, 8492, 10, 600,
10007, 3296, 36020, 16, 766, 236, 71259]
```

9. What does the **soup** object's text look like? Use the **.text** method and print only the first 2000 characters (do not print the whole thing, as it is too long.

[39]:
```
print(soup.text[:2000])
```

```
Top 100 | Project Gutenberg




Menu


About




About Project Gutenberg
Collection Development
Contact Us
```

History & Philosophy
Permissions & License
Privacy Policy
Terms of Use

Search and Browse

Book Search
Bookshelves
Frequently Downloaded
Offline Catalogs

Help

All help topics →
Copyright How-To
Errata, Fixes and Bug Reports
File Formats
Frequently Asked Questions
Policies →
Public Domain eBook Submission
Submitting Your Own Work
Tablets, Phones and eReaders
The Attic →

Donate

Donation

Frequently Viewed or Downloaded
These listings are based on the number of times each eBook gets downloaded.
Multiple downloads from the same Internet address on the same day count as one download, and addresses that download more than 100 eBooks in a day are considered robots and are not counted.

Downloaded Books
2023-07-23225348
last 7 days1474969
last 30 days5828812

Top 100 EBooks yesterday
Top 100 Authors yesterday
Top 100 EBooks last 7 days
Top 100 Authors last 7 days
Top 100 EBooks last 30 days
Top 100 Authors last 30 days

Top 100 EBooks yesterday

Down and Out in the Magic Kingdom by Cory Doctorow (25079)
Romeo and Juliet by William Shakespeare (2735)
Moby Dick; Or, The Whale by Herman Melville (2496)
A Room with a View by E. M. Forster (2275)
Middlemarch by George Eliot (2182)
Little Women; Or, Meg, Jo, Beth, and Amy by Louisa May Alcott (2146)
The Complete Works of William Shakespeare by William Shakespeare (2106)
The Enchanted April by Elizabeth Von Arnim (2038)
The Blue Castle: a novel by L. M. Montgomery (2020)
The Adventures of Ferdinand Count Fathom - Complete by T. Smollett (1927)
Cranford by Elizabeth Cleghorn Gaskell (1898)
The Expedition of Humphry Clinker by T. Smollett (1881)
Twenty Years After by Alexandre Dumas (1875)
The Adventures of Roderick Random by T. Smollett (1865)
History of Tom Jones, a Foundling by Henry Fielding (1833)
My Life - Volume

10. Search in the extracted test (using a regular expression) from the soup object to find the names of the top 100 eBooks (yesterday's ranking).

11. Create a starting index. It should point at the text "Top 100 Ebooks yesterday". Hint: Use splitlines() method of the soup.text. It splits the lines of the text of the soup object.

[95]:
```
# Temp empty list of Ebook names
lst_titles_temp=[]
```

[96]:
```
# Splits the soup object
tart_idx=soup.text.splitlines().index('Top 100 EBooks yesterday')
```

12. Loop 1-100 to add the strings of next 100 lines to this temporary list. Hint: splitlines() method

[97]:
```
# Gets titles from list
for i in range(8,110):
    lst_titles_temp.append(soup.text.splitlines()[start_idx+2+i])
```

[98]:
```
# Prints items in list
for l in lst_titles_temp:
    print(l)
```

```
Down and Out in the Magic Kingdom by Cory Doctorow (25079)
Romeo and Juliet by William Shakespeare (2735)
Moby Dick; Or, The Whale by Herman Melville (2496)
A Room with a View by E. M.  Forster (2275)
Middlemarch by George Eliot (2182)
Little Women; Or, Meg, Jo, Beth, and Amy by Louisa May Alcott (2146)
The Complete Works of William Shakespeare by William Shakespeare (2106)
The Enchanted April by Elizabeth Von Arnim (2038)
The Blue Castle: a novel by L. M.  Montgomery (2020)
The Adventures of Ferdinand Count Fathom - Complete by T.  Smollett (1927)
Cranford by Elizabeth Cleghorn Gaskell (1898)
The Expedition of Humphry Clinker by T.  Smollett (1881)
Twenty Years After by Alexandre Dumas (1875)
The Adventures of Roderick Random by T.  Smollett (1865)
History of Tom Jones, a Foundling by Henry Fielding (1833)
My Life - Volume 1 by Richard Wagner (1792)
Pride and Prejudice by Jane Austen (1529)
Simple Sabotage Field Manual by United States. Office of Strategic Services
(1174)
The green girl by Jack Williamson (944)
Alice's Adventures in Wonderland by Lewis Carroll (807)
Ukraine : by Stephen Rudnitsky (749)
Mark Twain's Speeches by Mark Twain (675)
Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley (657)
The Adventures of Sherlock Holmes by Arthur Conan Doyle (609)
Dracula by Bram Stoker (594)
```

The Picture of Dorian Gray by Oscar Wilde (564)
Cleopatra's needle by Sir Erasmus Wilson (556)
The Prince by Niccolò Machiavelli (509)
Occult science in medicine by Franz Hartmann (494)
The Count of Monte Cristo, Illustrated by Alexandre Dumas (490)
Selected etchings by Piranesi, series 2 (482)
A Tale of Two Cities by Charles Dickens (468)
Metamorphosis by Franz Kafka (448)
The Philippines a Century Hence by José Rizal (424)
War and Peace by graf Leo Tolstoy (420)
Ulysses by James Joyce (406)
The Great Gatsby by F. Scott Fitzgerald (406)
Waverley; Or, 'Tis Sixty Years Since by Walter Scott (400)
The Brothers Karamazov by Fyodor Dostoyevsky (386)
Anna Karenina by graf Leo Tolstoy (384)
The Kama Sutra of Vatsyayana by Vatsyayana (375)
The Alo Man : by Louise Lamprey and Mara L. Pratt-Chadwick (368)
The Yellow Wallpaper by Charlotte Perkins Gilman (358)
Thus Spake Zarathustra: A Book for All and None by Friedrich Wilhelm Nietzsche (357)
The Romance of Lust: A classic Victorian erotic novel by Anonymous (350)
The Odyssey by Homer (347)
Crime and Punishment by Fyodor Dostoyevsky (346)
A Modest Proposal by Jonathan Swift (334)
A Doll's House : a play by Henrik Ibsen (330)
The Young Visiters or, Mr. Salteena's Plan by Daisy Ashford (329)
The white mail by Cy Warman (322)
The Iliad by Homer (322)
The Adventures of Tom Sawyer, Complete by Mark Twain (310)
Dombey and Son by Charles Dickens (301)
Moby Multiple Language Lists of Common Words by Grady Ward (299)
Meditations by Emperor of Rome Marcus Aurelius (299)
Grimms' Fairy Tales by Jacob Grimm and Wilhelm Grimm (298)
The Prophet by Kahlil Gibran (294)
The Importance of Being Earnest: A Trivial Comedy for Serious People by Oscar Wilde (293)
Great Expectations by Charles Dickens (292)
Beyond Good and Evil by Friedrich Wilhelm Nietzsche (290)
The Republic by Plato (278)
Jane Eyre: An Autobiography by Charlotte Brontë (277)
Don Quixote by Miguel de Cervantes Saavedra (272)
Treasure Island by Robert Louis Stevenson (262)
Adventures of Huckleberry Finn by Mark Twain (261)
Emma by Jane Austen (256)
Anne of Green Gables by L. M. Montgomery (256)
The slang dictionary : by John Camden Hotten (252)
The Strange Case of Dr. Jekyll and Mr. Hyde by Robert Louis Stevenson (248)
The Rámáyan of Válmíki, translated into English verse by Valmiki (247)

```
A Study in Scarlet by Arthur Conan Doyle (247)
Wuthering Heights by Emily Brontë (240)
Celtic Scotland, Volume I (of 3) : by W. F.  Skene (235)
Walden, and On The Duty Of Civil Disobedience by Henry David Thoreau (234)
Tractatus Logico-Philosophicus by Ludwig Wittgenstein (232)
The Wonderful Wizard of Oz by L. Frank  Baum (232)
Essays and soliloquies by Miguel de Unamuno (229)
Calculus Made Easy by Silvanus P.  Thompson (227)
The divine comedy by Dante Alighieri (226)
Little Women by Louisa May Alcott (224)
Les Misérables by Victor Hugo (220)
Demonology and Devil-lore by Moncure Daniel Conway (219)
Dubliners by James Joyce (217)
Sense and Sensibility by Jane Austen (216)
On the Duty of Civil Disobedience by Henry David Thoreau (209)
The Scarlet Letter by Nathaniel Hawthorne (207)
The War of the Worlds by H. G.  Wells (206)
Winnie-the-Pooh by A. A.  Milne (205)
The King in Yellow by Robert W.  Chambers (204)
The King James Version of the Bible (200)
Notes from the Underground by Fyodor Dostoyevsky (198)
Carmilla by Joseph Sheridan Le Fanu (196)
The Confessions of St. Augustine by Bishop of Hippo Saint Augustine (195)
Slave Narratives: A Folk History of Slavery in the United States from Interviews
with Former Slaves, (194)
Peter Pan by J. M.  Barrie (190)
David Copperfield by Charles Dickens (190)
The Jungle Book by Rudyard Kipling (185)
A tour through Holland : by Sir John Carr (185)
The Time Machine by H. G.  Wells (184)
```

13. Use regular expression to extract only text from the name strings and append to an empty list. Hint: Use match and span to find indices and use them

[99]:
```
# Creates blank list
lst_titles=[]
for i in range(100):
    # Takes only strings from title
    id1,id2=re.match('^[a-zA-Z ]*',lst_titles_temp[i]).span()
    # Adds to list
    lst_titles.append(lst_titles_temp[i][id1:id2])
```

[100]:
```
# Displays title
for l in lst_titles:
    print(l)
```

Down and Out in the Magic Kingdom by Cory Doctorow
Romeo and Juliet by William Shakespeare
Moby Dick
A Room with a View by E
Middlemarch by George Eliot
Little Women
The Complete Works of William Shakespeare by William Shakespeare
The Enchanted April by Elizabeth Von Arnim
The Blue Castle
The Adventures of Ferdinand Count Fathom
Cranford by Elizabeth Cleghorn Gaskell
The Expedition of Humphry Clinker by T
Twenty Years After by Alexandre Dumas
The Adventures of Roderick Random by T
History of Tom Jones
My Life
Pride and Prejudice by Jane Austen
Simple Sabotage Field Manual by United States
The green girl by Jack Williamson
Alice
Ukraine
Mark Twain
Frankenstein
The Adventures of Sherlock Holmes by Arthur Conan Doyle
Dracula by Bram Stoker
The Picture of Dorian Gray by Oscar Wilde
Cleopatra
The Prince by Niccol
Occult science in medicine by Franz Hartmann
The Count of Monte Cristo
Selected etchings by Piranesi
A Tale of Two Cities by Charles Dickens
Metamorphosis by Franz Kafka
The Philippines a Century Hence by Jos
War and Peace by graf Leo Tolstoy
Ulysses by James Joyce
The Great Gatsby by F
Waverley
The Brothers Karamazov by Fyodor Dostoyevsky
Anna Karenina by graf Leo Tolstoy
The Kama Sutra of Vatsyayana by Vatsyayana
The Alo Man
The Yellow Wallpaper by Charlotte Perkins Gilman
Thus Spake Zarathustra
The Romance of Lust
The Odyssey by Homer
Crime and Punishment by Fyodor Dostoyevsky
A Modest Proposal by Jonathan Swift

A Doll
The Young Visiters or
The white mail by Cy Warman
The Iliad by Homer
The Adventures of Tom Sawyer
Dombey and Son by Charles Dickens
Moby Multiple Language Lists of Common Words by Grady Ward
Meditations by Emperor of Rome Marcus Aurelius
Grimms
The Prophet by Kahlil Gibran
The Importance of Being Earnest
Great Expectations by Charles Dickens
Beyond Good and Evil by Friedrich Wilhelm Nietzsche
The Republic by Plato
Jane Eyre
Don Quixote by Miguel de Cervantes Saavedra
Treasure Island by Robert Louis Stevenson
Adventures of Huckleberry Finn by Mark Twain
Emma by Jane Austen
Anne of Green Gables by L
The slang dictionary
The Strange Case of Dr
The R
A Study in Scarlet by Arthur Conan Doyle
Wuthering Heights by Emily Bront
Celtic Scotland
Walden
Tractatus Logico
The Wonderful Wizard of Oz by L
Essays and soliloquies by Miguel de Unamuno
Calculus Made Easy by Silvanus P
The divine comedy by Dante Alighieri
Little Women by Louisa May Alcott
Les Mis
Demonology and Devil
Dubliners by James Joyce
Sense and Sensibility by Jane Austen
On the Duty of Civil Disobedience by Henry David Thoreau
The Scarlet Letter by Nathaniel Hawthorne
The War of the Worlds by H
Winnie
The King in Yellow by Robert W
The King James Version of the Bible
Notes from the Underground by Fyodor Dostoyevsky
Carmilla by Joseph Sheridan Le Fanu
The Confessions of St
Slave Narratives
Peter Pan by J

```
David Copperfield by Charles Dickens
The Jungle Book by Rudyard Kipling
A tour through Holland
The Time Machine by H
```

Data Wrangling with Python: Activity 10, page 295

1. Import **urllib.requests, urllib.parse, urrlib.error,** and **json**.

[7]:
```
import urllib.request, urllib.parse, urllib.error
import json
```

2. Load the secret API key (you have to get one from the OMDb webstie and use that; it has a daily limit of 1,000. from a JSON file stored in the same folder in a variable, by using **json.loads**.

3. Obatin a key and store it in JSON as **APIkeys.json**.

Key is stored in the following path OneDrive - Bellevue University/DSC 540 Data Preperation/Week6_7

4. Open the **APIkeys.json** file.

[16]:
```
# Opens APIKeys json
with open('APIkeys.json') as f:
    # Loads data
    keys = json.load(f)
    # Extracts key
    omdbapi = keys['OMDBapi']
```

5. Assign the OMDb portal as a string to a variable.

6. Create a variable called **apikey** with the last portion of the url **(* apikey=secretapikey**, where **secretapikey** is your own API key.

[35]:
```
# Sets url
omdb_url = 'http://www.omdbapi.com/?'
# Formats api key
apikey = 'apikey='+omdbapi+'&'
```

7. Write a utility function called **print_json** to print the movie data from a JSON file (which we will get from the portal).

[18]:
```
# JSON example of response from website
with open('example.json') as f2:
    keys_list = json.load(f2)
```

[20]:
```
# Displays keys in the json file
keys_list.keys()
```

[20]:
```
dict_keys(['Title', 'Year', 'Rated', 'Released', 'Runtime', 'Genre', 'Director',
'Writer', 'Actors', 'Plot', 'Language', 'Country', 'Awards', 'Poster',
'Ratings', 'Metascore', 'imdbRating', 'imdbVotes', 'imdbID', 'Type', 'DVD',
'BoxOffice', 'Production', 'Website', 'Response'])
```

[21]:
```
# Formats the key disply
def print_json(json_data):
    keys_list
    print("-"*50)

    for k in keys_list:
        if k in list(json_data.keys()):
            print(f"{k}: {json_data[k]}")
    print("-"*50)
```

8. Write a utility function to download a poster of the movie based on the information from the jason dataset and save in your local folder. Use **os** module. The poster data is stored in the JSON key **'Poster'**. Use the Python command open to open a file and write the poster data. Close the file after done. This function will save the poster data as an image file.

[32]:
```
def save_poster(json_data):
    import os
    title = json_data['Title']
    poster_url = json_data['Poster']
    # Splits the poster url
    poster_file_extension=poster_url.split('.')[-1]
    # Reads the image file from web
    poster_data = urllib.request.urlopen(poster_url).read()

    # Adds poster string to current directory
    savelocation=os.getcwd()+'\\'+'Posters'+'\\'
    # Creates new directory if the directory does not exist
    if not os.path.isdir(savelocation):
        os.mkdir(savelocation)
    # Creates file
    filename=savelocation+str(title)+'.'+poster_file_extension
    # Opens
    f=open(filename,'wb')
    # Creates poster
    f.write(poster_data)
    f.close()
```

9. Write a utility function **search_movie** to search a movie by its name, print the downloaded **JSON** data and save the movie poster in the local folder. Use try-except loop for this. Use the previously created **serviceurl** and **apikey** variables. You have to pass on a dictionary with a key **t** and the movie name as the corresponding value to **urllib.parse.urlencode()** function and then add the **serviceurl** and **apikey** to the output of the function to construct the full URL. This URL will be used for accessing the data. The JSON data has a key called

Response. If it is **True**, that means the read was successful. Check this before processing the data. If not successful, then print the **JSON** key **Error**, which will contain the appropriate error message returned by the movie database.

[36]:
```
def search_movie(title):
    try:
        # URL Call
        url = omdb_url +apikey+ urllib.parse.urlencode({'t': str(title)})
        print(f'Retrieving the data of "{title}" now... ')
        print(url)
        # Opens url
        uh = urllib.request.urlopen(url)
        data = uh.read()
        json_data=json.loads(data)

        if json_data['Response']=='True':
            print_json(json_data)
            # if poster exists, download poster
            if json_data['Poster']!='N/A':
                save_poster(json_data)
        # Error handle
        else:
            print("Error encountered: ",json_data['Error'])
    # Error handle
    except urllib.error.URLError as e:
        print(f"ERROR: {e.reason}")
```

10. Test **search_movie** function by entering **Titanic**

[38]:
```
search_movie("Titanic")
```

```
Retrieving the data of "Titanic" now…
http://www.omdbapi.com/?apikey=6a73f5de&t=Titanic
----------------------------------------------------
Title: Titanic
Year: 1997
Rated: PG-13
Released: 19 Dec 1997
Runtime: 194 min
Genre: Drama, Romance
Director: James Cameron
Writer: James Cameron
Actors: Leonardo DiCaprio, Kate Winslet, Billy Zane
Plot: A seventeen-year-old aristocrat falls in love with a kind but poor artist
aboard the luxurious, ill-fated R.M.S. Titanic.
Language: English, Swedish, Italian, French
Country: United States, Mexico
Awards: Won 11 Oscars. 126 wins & 83 nominations total
```

```
Poster: https://m.media-amazon.com/images/M/MV5BMDdmZGU3NDQtY2E5My00ZTliLWIzOTUt
MTY4ZGI1YjdiNjk3XkEyXkFqcGdeQXVyNTA4NzY1MzY@._V1_SX300.jpg
Ratings: [{'Source': 'Internet Movie Database', 'Value': '7.9/10'}, {'Source':
'Rotten Tomatoes', 'Value': '88%'}, {'Source': 'Metacritic', 'Value': '75/100'}]
Metascore: 75
imdbRating: 7.9
imdbVotes: 1,228,124
imdbID: tt0120338
Type: movie
DVD: 08 Jan 2002
BoxOffice: $674,292,608
Production: N/A
Website: N/A
Response: True
--------------------------------------------------
```

11. Test search_movie function by entering **"Random_error"** (obviously this will not be found and you should be able to check whether your error catching code is working properly)

[39]:
```
search_movie('Random_error')
```

```
Retrieving the data of "Random_error" now…
http://www.omdbapi.com/?apikey=6a73f5de&t=Random_error
Error encountered:  Movie not found!
```

Connect to an API of your choice and do a simple data pull - you can use any API - except the API you have selected for your project.

```
a. In previous versions of this course we have always used Twitter, but with recent organizatio

b. Connect to the API and do a "Get" call/operation on the API to return a subset of data from
```

[52]:
```
# Creates ket
with open('weather.json') as f3:
    keys_weather = json.load(f3)
    appid = keys['appid']
```

[61]:
```
# Creates url and applies api
weather_url = 'https://api.openweathermap.org/data/2.5/weather?'
weather_appid = '&appid='+appid
```

[62]:
```
# JSON example of response from website
with open('weather_example.json') as f4:
    keys_list2 = json.load(f4)
```

[63]:
```
keys_list2.keys()
```

[63]:
```
dict_keys(['coord', 'weather', 'base', 'main', 'visibility', 'wind', 'clouds',
'dt', 'sys', 'id', 'name', 'cod'])
```

[64]:
```
# Formats keys and displays
def print_json2(json_data):
    keys_list2
    print("-"*50)
    for k in keys_list2:
        if k in list(json_data.keys()):
            print(f"{k}: {json_data[k]}")
    print("-"*50)
```

[69]:
```
def search_weather(city):
    try:
        # URL Call
        url = weather_url + urllib.parse.urlencode({'q':␣
 ↪str(city)})+weather_appid
        print(f'Retrieving the data of "{city}" now... ')
        print(url)
        # Opens URL
        uh = urllib.request.urlopen(url)
        data = uh.read()
        json_data=json.loads(data)
        # Prints data
        print_json2(json_data)
    # Error Handle
    except urllib.error.URLError as e:
        print(f"ERROR: {e.reason}")
```

[70]:
```
search_weather('London')
```

```
Retrieving the data of "London" now…
https://api.openweathermap.org/data/2.5/weather?q=London&appid=c95457742f9936906
058d38ad9153a1b
--------------------------------------------------
coord: {'lon': -0.1257, 'lat': 51.5085}
weather: [{'id': 802, 'main': 'Clouds', 'description': 'scattered clouds',
'icon': '03n'}]
base: stations
main: {'temp': 285.82, 'feels_like': 285.15, 'temp_min': 283.35, 'temp_max':
287.6, 'pressure': 1011, 'humidity': 77}
visibility: 10000
wind: {'speed': 0.45, 'deg': 187, 'gust': 0.89}
clouds: {'all': 30}
dt: 1690331827
sys: {'type': 2, 'id': 2075535, 'country': 'GB', 'sunrise': 1690344883,
'sunset': 1690401542}
id: 2643743
name: London
cod: 200
```

---

[75]:
```
import requests
```

[76]:
```
def get_request(city, key):
    # URL Call
    response = requests.get(weather_url + urllib.parse.urlencode({'q':␣
 ↪str(city)})+weather_appid)
    # Handle if connection goes through
    if response.status_code == 200:
        data = response.json()
        subset = data[key]
        # Ruturns data from key
        return subset
    # Error Handle
    else:
        print("Error:", response.status_code)
        return None
```

[77]:
```
get_request('London', 'coord')
```

[77]:
```
{'lon': -0.1257, 'lat': 51.5085}
```

Using one of the datasets provided, or a dataset of your own, choose 3 of the following visualizations to complete. You must submit via PDF along with your code. You are free to use Matplotlib, Seaborn or another package if you prefer.

a. Line
b. Scatter
c. Bar
d. Histogram
e. Density Plot
f. Pie Chart

[108]:
```
import pandas as pd
import numpy as np
```

[88]:
```
# Read in Data
df = pd.DataFrame(np.random.randint(0,100,size=(15, 15)),␣
 ↪columns=list('ABCDEFGHIJKLMNO'))
```

[92]:
```
df.head()
```

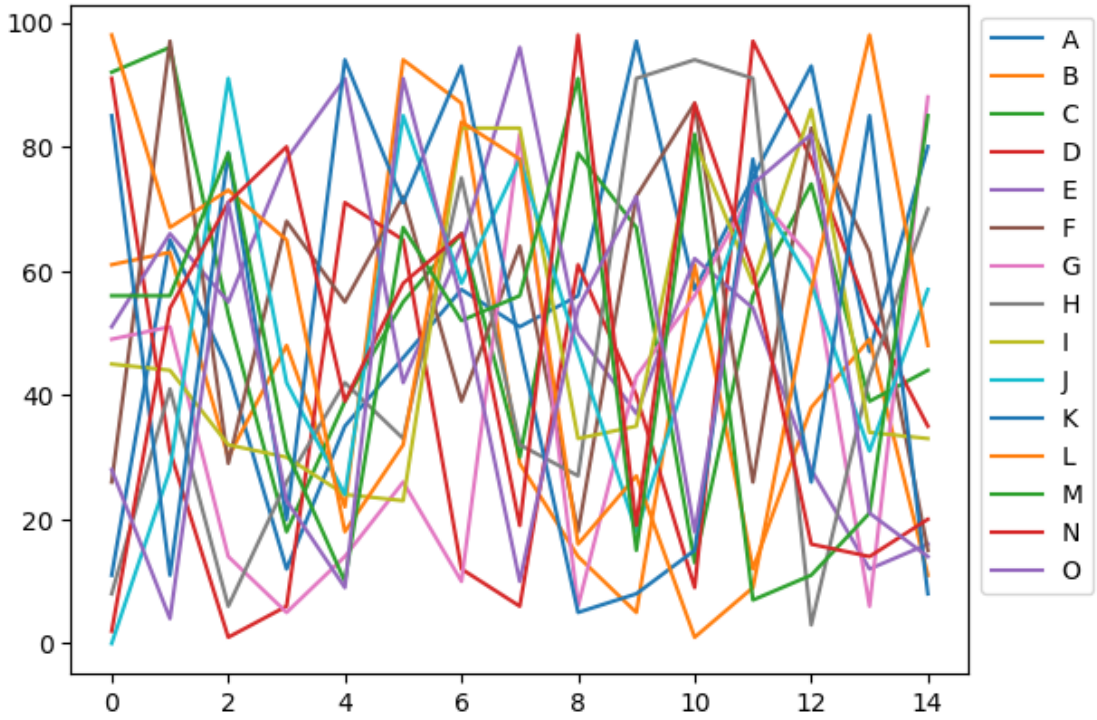
[92]:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0	11	61	92	91	51	26	49	8	45	0	85	98	56	2	28
1	65	63	96	31	66	97	51	41	44	28	11	67	56	54	4
2	44	30	53	1	55	29	14	6	32	91	79	73	79	71	71
3	12	48	18	6	78	68	5	26	30	42	20	65	31	80	23

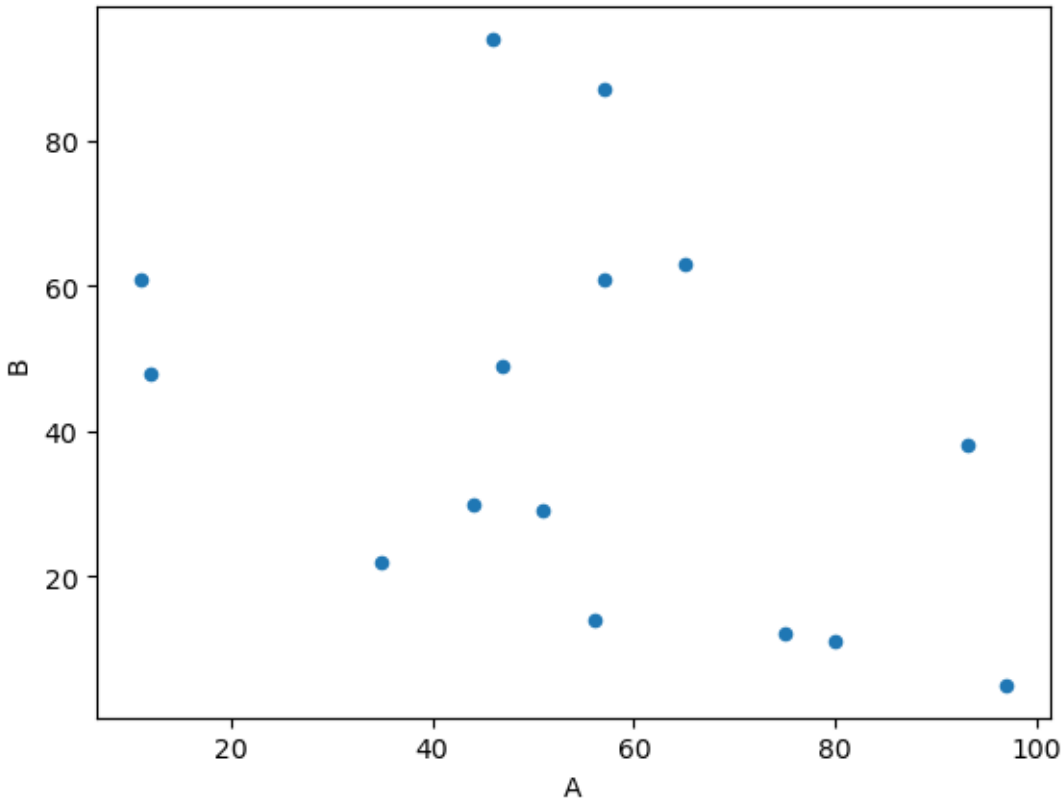
```
4  35  22  39  71  91  55  14  42  24  24  94  18  10  39   9
```

[106]:
```
# Line Plot
line_plot = df.plot()
line_plot.legend(loc='upper left', bbox_to_anchor=(1.0, 1.0))
```

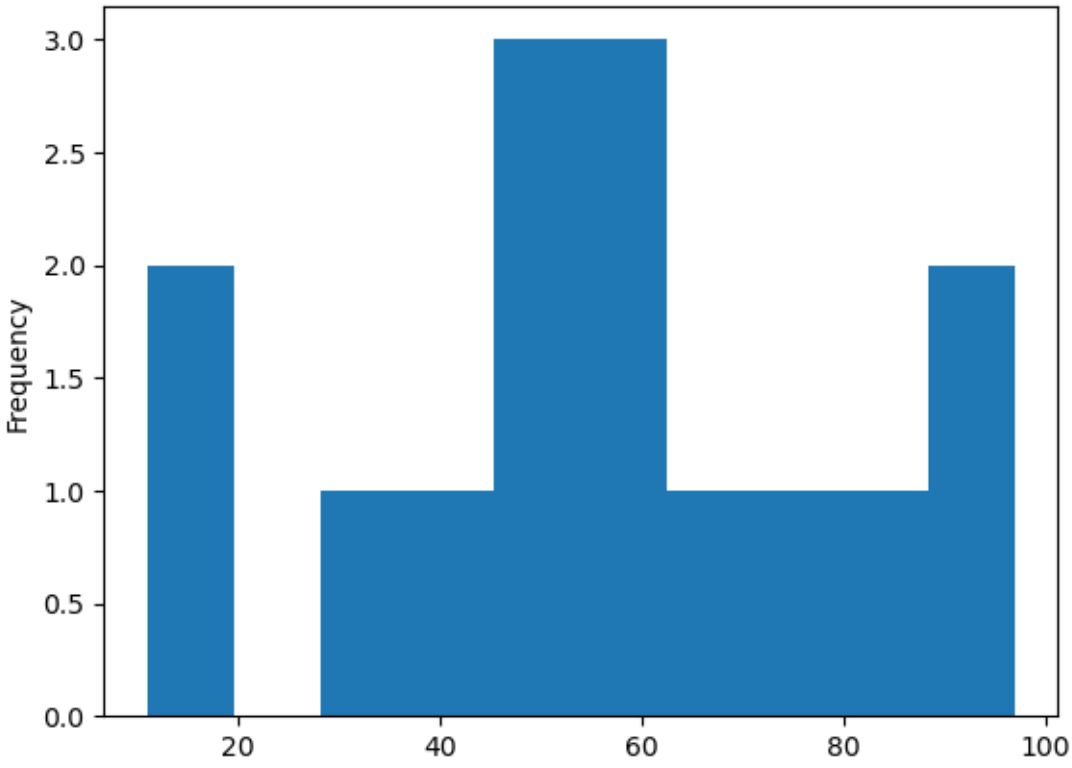
[106]:
```
<matplotlib.legend.Legend at 0x7f91901bf460>
```



[107]:
```
# Scatter Plot
scatter_plot = df.plot(kind= 'scatter', x = 'A', y ='B')
```



[109]:
```
# Histogram
hist = df['A'].plot(kind = 'hist')
```


Frequency
3.0
2.5
2.0
1.5
1.0
0.5
0.0
20
40
60
80
100