ASumbaraju_Weeks9_10_Exercise

May 18, 2021

1 DSC 540-Week 9 & 10 Exercises

2 Aditya Sumbaraju

3 Date: 05/09/2021

4 Acticvity 9

[10]:
```
## Loading libraries
import urllib.request, urllib.parse, urllib.error

import requests

from bs4 import BeautifulSoup

import ssl

import re
```

[11]:
```
## Checking SSl certificate
# Ignore SSL certificate errors

ctx = ssl.create_default_context()

ctx.check_hostname = False

ctx.verify_mode = ssl.CERT_NONE
```

[12]:
```
# Read the HTML from the URL and pass on to BeautifulSoup

top100url = 'https://www.gutenberg.org/browse/scores/top'

response = requests.get(top100url)
```

[13]:
```
## Checking the status of the web request
def status_check(r):
    if r.status_code==200:
        print('Success!')
        return 1
    else:
        print('Failed!')
        return -1
```

[14]:
```
## checking status response
status_check(response)
```

```
Success!
```

[14]:
```
1
```

[15]:
```
## decoding response and and passing it on beautifulsoup

contents = response.content.decode(response.encoding)

soup = BeautifulSoup(contents, 'html.parser')
```

[16]:
```
## finding all link tags (href)
# Empty list to hold all the http links in the HTML page

lst_links=[]

# Find all the href tags and store them in the list of links

for link in soup.find_all('a'):
    lst_links.append(link.get('href'))
## print all the links
lst_links[:30]
```

[16]:
```
['/',
 '/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']
```

[17]:
```
## finding numeric digits in the links
booknum=[]
for i in range(19,119):
    link=lst_links[i]
    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 filenumber casted as integer

        booknum.append(int(n[0]))

## Printing the file numbers
print ('\nThe file numbers for the top 100 ebooks on Gutenberg are shown␣
↪below\n'+'-'*70)

## print the numbers
print(booknum)
```

```
The file numbers for the top 100 ebooks on Gutenberg are shown below
----------------------------------------------------------------------
[1, 1, 7, 7, 30, 30, 1342, 84, 11, 98, 64317, 2701, 65355, 174, 1661, 65351,
844, 57775, 1952, 345, 2542, 5200, 1260, 65350, 46, 43, 158, 205, 74, 219, 2591,
65347, 35899, 1400, 2852, 16, 2600, 1080, 65348, 4300, 76, 1232, 1250, 6130,
25344, 766, 55, 768, 5740, 6737, 65345, 1497, 65354, 514, 45, 65356, 120, 65343,
996, 408, 902, 244, 3825, 2554, 27827, 58585, 1184, 829, 5739, 1727, 20228,
2814, 1837, 17635, 65349, 43453, 135, 863, 30254, 2500, 65357, 100, 203, 161,
730, 3600, 6133, 36, 1399, 1998, 236, 16328]
```

[18]:
```
## What does the soup object's text look like
print(soup.text[:2000])
```

```
if (top != self):
    top.location.replace("http://www.gutenberg.org")
    alert('Project Gutenberg is a FREE service with NO membership required. If␣
 ↪you paid somebody else to get here,make them give you your money back!')
```

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 Procedures
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
2021-05-16144366
last 7 days1106228
last 30 days4859078

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

Pride and Prejudice by Jane Austen (1513)
Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley (1446)
Alice's Adventures in Wonderland by Lewis Carroll (746)
A Tale of Two Cities by Charles Dickens (710)
The Great Gatsby by F. Scott Fitzgerald (666)
Moby Dick; Or, The Whale by Herman Melville (609)
The Lost Giant and Other American Indian Tales Retold by Violet Moore Higgins
(597)
The Picture of Dorian Gray by Oscar Wilde (591)
The Adventures of Sherlock Holmes by Arthur Conan Doyle (576)
The Story of the Indian Mutiny by A. R. Hope Moncrieff (530)
The Importance of Being Earnest: A Trivial Comedy for Serious People by Oscar
Wilde (510)
Le jardin des supplices by Octave Mirbeau (506)

```
The Yellow Wallpaper by Charlotte Perkins Gilman (462)
Dracula by Bram Stoker (436)
Et dukkehjem. English by Henrik Ib
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-18-1fe4671d8b0a> in <module>
      2 print(soup.text[:2000])
      3
----> 4 if (top != self):
      5     top.location.replace("http://www.gutenberg.org")
      6     alert('Project Gutenberg is a FREE service with NO membership␣
 ↪required. If you paid somebody else to get here,make them give you your money␣
 ↪back!')

NameError: name 'top' is not defined
```

[19]:
```
## Search in the extracted text
# Temp empty list of Ebook names

lst_titles_temp=[]
start_idx=soup.text.splitlines().index('Top 100 EBooks yesterday')
for i in range(100):
    lst_titles_temp.append(soup.text.splitlines()[start_idx+2+i])

lst_titles=[]

for i in range(100):
    id1,id2=re.match('^[a-zA-Z ]*',lst_titles_temp[i]).span()
    lst_titles.append(lst_titles_temp[i][id1:id2])

for l in lst_titles:
    print(l)
```

```
Top
Top
Top
Top


Top

Pride and Prejudice by Jane Austen
Frankenstein
Alice
A Tale of Two Cities by Charles Dickens
The Great Gatsby by F
```

```
Moby Dick
The Lost Giant and Other American Indian Tales Retold by Violet Moore Higgins
The Picture of Dorian Gray by Oscar Wilde
The Adventures of Sherlock Holmes by Arthur Conan Doyle
The Story of the Indian Mutiny by A
The Importance of Being Earnest
Le jardin des supplices by Octave Mirbeau
The Yellow Wallpaper by Charlotte Perkins Gilman
Dracula by Bram Stoker
Et dukkehjem
Metamorphosis by Franz Kafka
Jane Eyre
John Holder
A Christmas Carol in Prose
The Strange Case of Dr
Emma by Jane Austen
Walden
The Adventures of Tom Sawyer
Heart of Darkness by Joseph Conrad
Grimms
I
The Philippines a Century Hence by Jos
Great Expectations by Charles Dickens
The Hound of the Baskervilles by Arthur Conan Doyle
Peter Pan by J
War and Peace by graf Leo Tolstoy
A Modest Proposal by Jonathan Swift
Barnstormer by Tom W
Ulysses by James Joyce
Adventures of Huckleberry Finn by Mark Twain
Il Principe
Anthem by Ayn Rand
The Iliad by Homer
The Scarlet Letter by Nathaniel Hawthorne
David Copperfield by Charles Dickens
The Wonderful Wizard of Oz by L
Wuthering Heights by Emily Bront
Tractatus Logico
Noli me t
Under Three Flags by Bert Leston Taylor and Alvin T
The Republic by Plato
Lily Speed
Little Women by Louisa May Alcott
Anne of Green Gables by L
The Girl
Treasure Island by Robert Louis Stevenson
The Ambassador
Don Quixote by Miguel de Cervantes Saavedra
```

```
The Souls of Black Folk by W
The Happy Prince
A Study in Scarlet by Arthur Conan Doyle
Pygmalion by Bernard Shaw
Prestuplenie i nakazanie
The Kama Sutra of Vatsyayana by Vatsyayana
The Prophet by Kahlil Gibran
The Count of Monte Cristo
Gulliver
Korean
The Odyssey by Homer
Noli Me Tangere by Jos
Dubliners by James Joyce
The Prince and the Pauper by Mark Twain
Repertory of The Comedie Humaine
La conqu
A Pickle for the Knowing Ones by Timothy Dexter
Les Mis
The Mysterious Affair at Styles by Agatha Christie
The Romance of Lust
Siddhartha by Hermann Hesse
Ismael
The Complete Works of William Shakespeare by William Shakespeare
Uncle Tom
Sense and Sensibility by Jane Austen
Oliver Twist by Charles Dickens
Essays of Michel de Montaigne
Ars
The War of the Worlds by H
Anna Karenina by graf Leo Tolstoy
Also sprach Zarathustra
The Jungle Book by Rudyard Kipling
Beowulf
A Japanese Boy by Shigemi Shiukichi
The Awakening
Calculus Made Easy by Silvanus P
The Call of the Wild by Jack London
The Secret Garden by Frances Hodgson Burnett
An Index of The Divine Comedy by Dante by Dante Alighieri
```

5 Activity 10

[42]:
```python
## importing modules
import urllib.request, urllib.parse, urllib.error

import json
```

[43]:
```
with open('APIkey.json') as f:
    keys = json.load(f)
    omdbapi = keys['OMDBapi']
```

[44]:
```
## assigning the url
serviceurl = 'http://www.omdbapi.com/?'
```

[45]:
```
## creating a variable called apikey
apikey = '&apikey='+omdbapi
```

[52]:
```
#7: Writing a utility function called print_json to print the movie data from a␣
↪JSON file

def print_json(json_file):
    #List of the categories in the JSON files returned by the OMDb API
    json_categories = ['Title', 'Year', 'Rated', 'Released', 'Runtime',␣
↪'Genre', 'Director', 'Writer', 'Actors', 'Plot',
                    'Language', 'Country', 'Awards', 'Ratings', 'Metascore',␣
↪'imdbRating', 'imdbVotes', 'imdbID']

    #Iterating over the list to print each category followed by its value from␣
↪the JSON file
    for k in json_categories:
        print(f'{k} : {json_file[k]}')
```

[53]:
```
import os

#Function to download the poster
def download_poster(json_file):
    #Pulling the movie title and poster link from the JSON file
    title = str(json_file['Title'])
    poster_link = json_file['Poster']

    #Reading the poster image data by passing the poster link through the␣
↪urllib.request library
    poster = UR.urlopen(poster_link).read()

    #Defining the image file's save location in the current working directory
    save_location = os.getcwd() + '\\'

    #Getting the file extension from the poster URL by grabbing the last value␣
↪from the URL split by its dots
    #Usually images will end in '.JPEG' or '.PNG' so the last set of characters␣
↪after the last dot should be the extension.
    poster_file_extension = poster_link.split('.')[-1]
    #If the file extension is one of the most common image extensions, the file␣
↪will be downloaded
    if poster_file_extension.upper() == 'JPEG' or poster_file_extension.upper()␣
↪== 'JPG' or poster_file_extension.upper() == 'PNG':
        #Defining what the saved file will be named
        filename = '{}{}{}{}'.format(save_location, title, '.',␣
↪poster_file_extension)

        #Opening a brand new file, writing the image data to it, then closing it
        with open(filename,'wb') as f:
            f.write(poster)
            f.close()
```

[54]:
```
## utility function for searching
def search_movie(title):
    try:
        url = serviceurl + urllib.parse.urlencode({'t': str(title)})+apikey
        print(f'Retrieving the data of {title} now... ')
        print(url)
        uh = urllib.request.urlopen(url)
        data = uh.read()
        json_data=json.loads(data)
        if json_data['Response']=='True':
            print_json(json_data)
            # Asks user whether to download the poster of the movie

        if json_data['Poster']!='N/A':
            save_poster(json_data)
        else:
            print('Error encountered: ',json_data['Error'])

    except urllib.error.URLError as e:
        print(f'ERROR: {e.reason}')
```

[57]:
```
#Writing a function to display the downloaded poster directly in Jupyter␣
↪Notebook.

def show_image(movie_name):
    try:
        from IPython.display import Image, display
        display(Image(filename = '{}.jpg'.format(movie_name)))
    except:
        print('ERROR!')
```

[58]:
```
#Testing the functions on the movie Titanic
titanic = 'Titanic'
search_movie(titanic)
```

```python
show_image(titanic)
print(' ')

#Testing to see if an error occurs
#search_movie('No Movie Found')
show_image('No Movie Found')
```

```
Retrieving the data of Titanic now…
http://www.omdbapi.com/?t=Titanic&apikey=cd4d1cd1
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, Kathy Bates
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 : USA, Mexico, Australia, Canada
Awards : Won 11 Oscars. Another 114 wins & 83 nominations.
Ratings : [{'Source': 'Internet Movie Database', 'Value': '7.8/10'}, {'Source':
'Rotten Tomatoes', 'Value': '89%'}, {'Source': 'Metacritic', 'Value': '75/100'}]
Metascore : 75
imdbRating : 7.8
imdbVotes : 1,060,049
imdbID : tt0120338
```

[61]:
```python
# Import the Twython class
from twython import Twython
import json

# Load credentials from json file
with open("twitter_credentials.json", "r") as file:
    creds = json.load(file)

# Instantiate an object
```

```python
python_tweets = Twython(creds['CONSUMER_KEY'], creds['CONSUMER_SECRET'])

# Create our query
query1 = {'q': 'Bellevue University',
        'result_type': 'recent',
         'count': '100',
        'lang': 'en',
        }
query2 = {'q': 'Data Science',
        'result_type': 'recent',
         'count': '100',
        'lang': 'en',
        }
```

[64]:
```python
import pandas as pd

# Search tweets
dict_ = {'user': [], 'date': [], 'text': [], 'favorite_count': []}
for status in python_tweets.search(**query1)['statuses']:
    dict_['user'].append(status['user']['screen_name'])
    dict_['date'].append(status['created_at'])
    dict_['text'].append(status['text'])
    dict_['favorite_count'].append(status['favorite_count'])

# Structure data in a pandas DataFrame for easier manipulation
df = pd.DataFrame(dict_)
df.sort_values(by='favorite_count', inplace=True, ascending=False)
df
```

[64]:
```
               user                            date  \
20   jamison_gruber  Sat May 15 17:35:58 +0000 2021
32  ChieftainNation  Wed May 12 17:47:16 +0000 2021
1      JournalHardin  Sun May 16 20:01:10 +0000 2021
5          BellevueU  Sun May 16 17:00:56 +0000 2021
2      JournalHardin  Sun May 16 19:58:14 +0000 2021
28     Okotoks_Dawgs  Thu May 13 18:00:14 +0000 2021
36          ksidzyik  Tue May 11 19:55:00 +0000 2021
42   BellevueLeader  Tue May 11 06:20:12 +0000 2021
26   corporatelearn  Thu May 13 20:00:56 +0000 2021
3      JournalHardin  Sun May 16 19:56:57 +0000 2021
10   ManuelGFalcon2  Sat May 15 19:23:10 +0000 2021
11   ManuelGFalcon2  Sat May 15 19:21:27 +0000 2021
40            jkyndt  Tue May 11 16:38:53 +0000 2021
30   Austin_Plourde  Thu May 13 01:15:52 +0000 2021
27         wcbleague  Thu May 13 18:02:50 +0000 2021
41   BellevueLeader  Tue May 11 06:35:09 +0000 2021
29         JimNekuda  Thu May 13 16:28:41 +0000 2021
```

```
38  MilfordEaglesVB  Tue May 11 17:50:24 +0000 2021
31  ProudChieftains  Wed May 12 21:56:18 +0000 2021
37         Emmmm_41  Tue May 11 17:53:33 +0000 2021
33        BellevueU  Wed May 12 16:47:40 +0000 2021
34  Vanblaricon5Sam  Wed May 12 02:10:28 +0000 2021
35           crishm  Tue May 11 20:37:12 +0000 2021
25       PatiMoore_  Thu May 13 20:01:21 +0000 2021
39           jkyndt  Tue May 11 17:50:24 +0000 2021
0   SantoshAnampal1  Sun May 16 22:14:35 +0000 2021
24   CDNBaseballNet  Fri May 14 13:41:31 +0000 2021
13         tonysroe  Sat May 15 19:14:43 +0000 2021
4     jacobknauss_  Sun May 16 17:24:07 +0000 2021
6          IHLBates  Sun May 16 08:21:07 +0000 2021
7     lanefeierfeil  Sun May 16 01:07:23 +0000 2021
8        kbaxter014  Sat May 15 22:59:45 +0000 2021
9   elliottbaseball  Sat May 15 19:35:04 +0000 2021
12       dgruber34  Sat May 15 19:19:18 +0000 2021
14    BereniceMonge  Sat May 15 18:05:53 +0000 2021
23     BallMechanic  Fri May 14 15:39:09 +0000 2021
15   WilliamKyleIII  Sat May 15 17:50:51 +0000 2021
16    JaxsonJones17  Sat May 15 17:47:34 +0000 2021
17     BrysonPatlan  Sat May 15 17:42:11 +0000 2021
18          J0shDix  Sat May 15 17:41:27 +0000 2021
19   CrystenaKeesee  Sat May 15 17:41:26 +0000 2021
22           crishm  Fri May 14 21:33:57 +0000 2021
21        Mwambeta1  Sat May 15 14:26:28 +0000 2021

                                                 text  favorite_count
20  Blessed to receive an offer from Bellevue Univ…             116
32  Congratulations to Sarah Felten for signing wi…              28
1   Plattsmouth graduates Chris Casart and Cole Wa…               9
5   The Bellevue University campus is in full bloo…               8
2   Conestoga graduate Jenna Curtis played in four…               7
28  Dawgs WCBL RHP Corey Jackson has been an absol…               7
36  I love seeing @BellevueU students do cool thin…               6
42  It is not too often that an undergraduate rese…               3
26  Bellevue University, along with 22 other insti…               3
3   Plattsmouth graduate Sydni Haugaard played in …               3
10  Congratulations, Bellevue University Track Tea…               1
11  Congratulations, Bellevue University Track Tea…               1
40  Bellevue University student published for Heli…               1
30  RT @ksidzyik: I love seeing @BellevueU student…               0
27  RT @Okotoks_Dawgs: Dawgs WCBL RHP Corey Jackso…               0
41  The University of Nebraska at Omaha awarded de…               0
29  Bellevue University to Join the Federal Academ…               0
38  Attention all incoming 6th-8th graders for the…               0
31  RT @ChieftainNation: Congratulations to Sarah …               0
```

```
37  RT @MilfordEaglesVB: Attention all incoming 6t…               0
33  RT @ksidzyik: I love seeing @BellevueU student…               0
34  RT @ksidzyik: I love seeing @BellevueU student…               0
35  RT @ksidzyik: I love seeing @BellevueU student…               0
25  RT @corporatelearn: Bellevue University, along…               0
39  RT @BellevueLeader: It is not too often that a…               0
0   RT @BellevueU: The Bellevue University campus …               0
24  RT @Okotoks_Dawgs: Dawgs WCBL RHP Corey Jackso…               0
13  RT @jamison_gruber: Blessed to receive an offe…               0
4   RT @jamison_gruber: Blessed to receive an offe…               0
6   RT @jamison_gruber: Blessed to receive an offe…               0
7   RT @jamison_gruber: Blessed to receive an offe…               0
8   RT @jamison_gruber: Blessed to receive an offe…               0
9   RT @Okotoks_Dawgs: Dawgs WCBL RHP Corey Jackso…               0
12  RT @jamison_gruber: Blessed to receive an offe…               0
14  RT @jamison_gruber: Blessed to receive an offe…               0
23  RT @Okotoks_Dawgs: Dawgs WCBL RHP Corey Jackso…               0
15  RT @jamison_gruber: Blessed to receive an offe…               0
16  RT @jamison_gruber: Blessed to receive an offe…               0
17  RT @jamison_gruber: Blessed to receive an offe…               0
18  RT @jamison_gruber: Blessed to receive an offe…               0
19  RT @jamison_gruber: Blessed to receive an offe…               0
22  RT @corporatelearn: Bellevue University, along…               0
21  Imagine a line from Mlolongo to Westlands with…               0
```

[67]:
```python
dict_ = {'user': [], 'date': [], 'text': [], 'favorite_count': []}
for status in python_tweets.search(**query2)['statuses']:
    dict_['user'].append(status['user']['screen_name'])
    dict_['date'].append(status['created_at'])
    dict_['text'].append(status['text'])
    dict_['favorite_count'].append(status['favorite_count'])

# Structure data in a pandas DataFrame for easier manipulation
df1 = pd.DataFrame(dict_)
df1.sort_values(by='favorite_count', inplace=True, ascending=False)
df1
```

[67]:
```
               user                            date  \
99      MaxRLambert  Tue May 18 01:57:50 +0000 2021
50    alibaba_cloud  Tue May 18 02:10:00 +0000 2021
53         AgentW45  Tue May 18 02:09:21 +0000 2021
23     ifihadastick  Tue May 18 02:15:45 +0000 2021
65     pam_pinklady  Tue May 18 02:06:53 +0000 2021
..              …                               …
33    help_academia  Tue May 18 02:12:45 +0000 2021
32  SUPERIORWRITE17  Tue May 18 02:13:01 +0000 2021
31  ProminentTutors  Tue May 18 02:13:11 +0000 2021
```

```
30  DigitalPatriot0  Tue May 18 02:13:13 +0000 2021
0      EllipsisVoid  Tue May 18 02:22:15 +0000 2021

                                                 text  favorite_count
99  We made the front page of the @scsentinel!\n\n…              14
50  #BREAKING #AlibabaCloud's self-developed cloud…               4
53  @neontaster Am I the only one that reads that …               3
23  At the end of April, I had 65% of Population I…               3
65  RT @libertytarian: #Fauci &amp; #CDC say vacci…               0
..                                                …             …
33  Hire us to help you with your coursework\nFina…               0
32  Let The Prophesionals Handle Your;\nPowerPoint…               0
31  Hire us to help you with your coursework\nFina…               0
30  RT @gamesblazer06: This is when you know Masks…               0
0   @tonyjellison @Studio_Michaud @thisisinsider @…               0

[100 rows x 4 columns]
```

5.1 Exercise 4

[104]:
```
## load packages
import pandas as pd
import seaborn as sns
```

[107]:
```
## import data and apply cleansing rules
carPriceData = pd.read_csv('C:/BN_Cloud_Projects/DSC540/project_data/
↪CarPrice_Assignment.csv')
carPriceData.head()
```

[107]:
```
   car_ID  symboling                   CarName fueltype aspiration doornumber  \
0       1          3        alfa-romero giulia      gas        std        two
1       2          3       alfa-romero stelvio      gas        std        two
2       3          1  alfa-romero Quadrifoglio      gas        std        two
3       4          2               audi 100 ls      gas        std       four
4       5          2                audi 100ls      gas        std       four

       carbody drivewheel enginelocation  wheelbase  …  enginesize  \
0  convertible        rwd          front       88.6  …         130
1  convertible        rwd          front       88.6  …         130
2    hatchback        rwd          front       94.5  …         152
3        sedan        fwd          front       99.8  …         109
4        sedan        4wd          front       99.4  …         136

   fuelsystem  boreratio  stroke compressionratio horsepower  peakrpm citympg  \
0        mpfi       3.47    2.68              9.0        111     5000      21
1        mpfi       3.47    2.68              9.0        111     5000      21
2        mpfi       2.68    3.47              9.0        154     5000      19
```

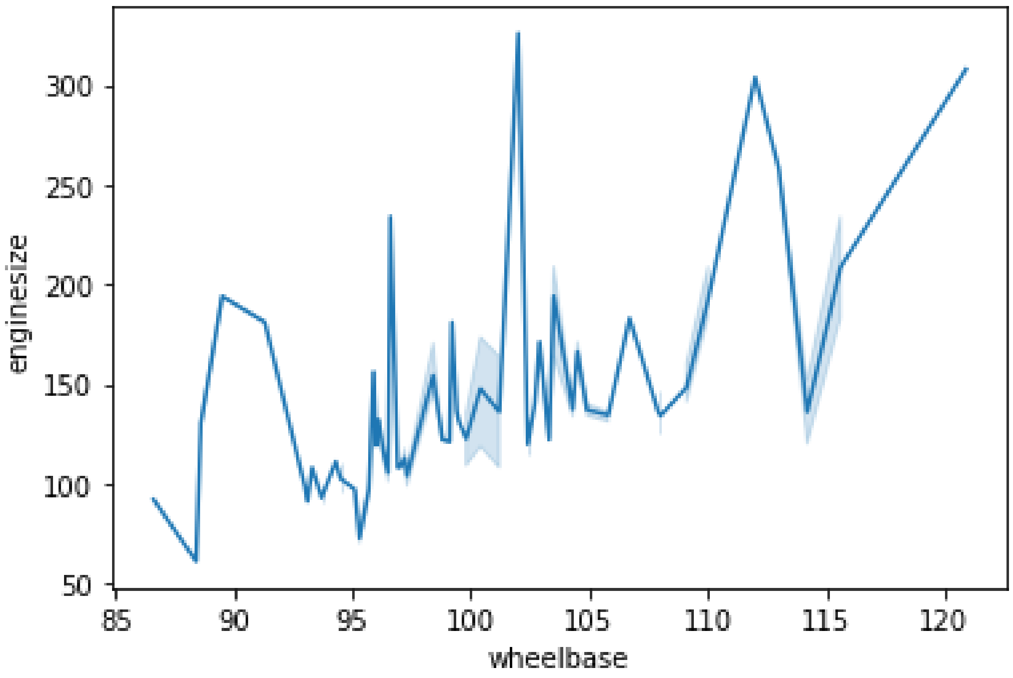
```
3        mpfi        3.19    3.40              10.0        102      5500       24
4        mpfi        3.19    3.40               8.0        115      5500       18

   highwaympg    price
0          27  13495.0
1          27  16500.0
2          26  16500.0
3          30  13950.0
4          22  17450.0

[5 rows x 26 columns]
```

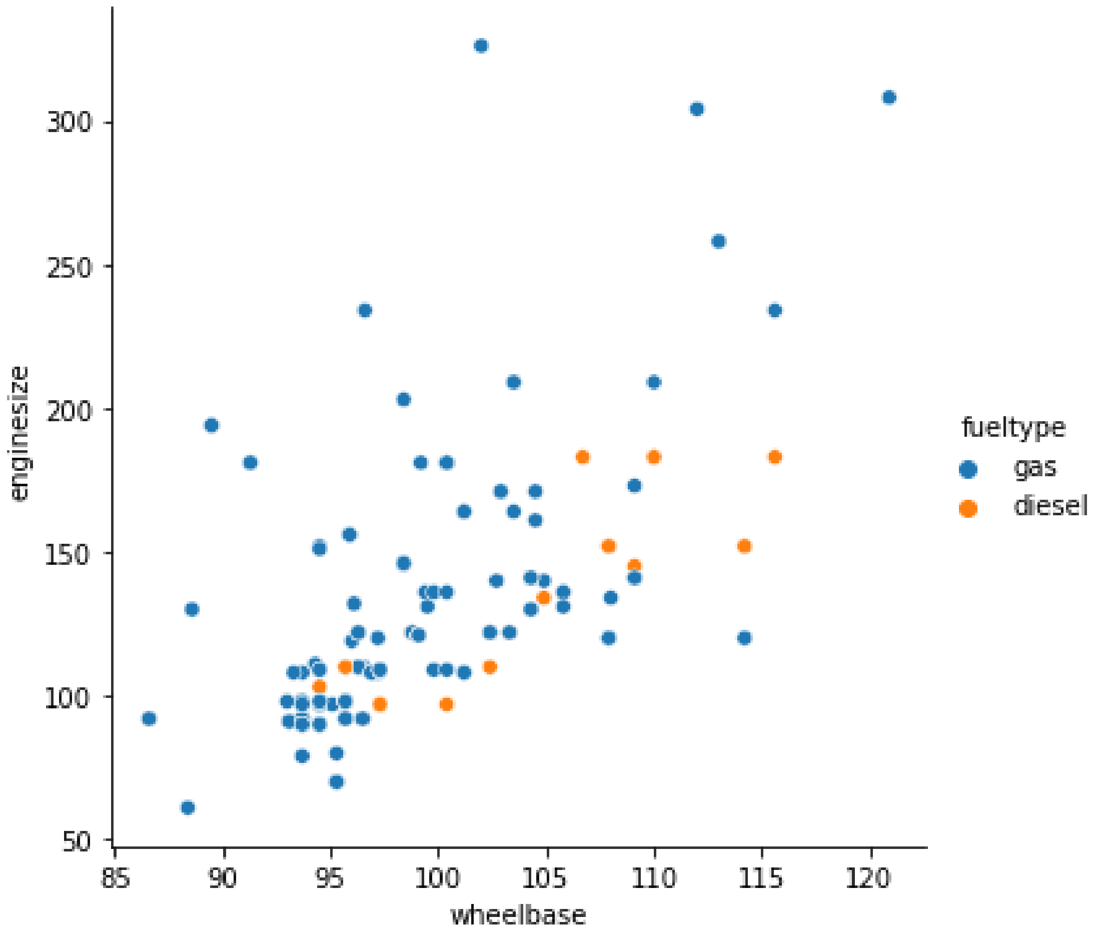
[108]:
```
## line chart
sns.lineplot(data=carPriceData, x="wheelbase", y="enginesize")
```

[108]:
```
<AxesSubplot:xlabel='wheelbase', ylabel='enginesize'>
```



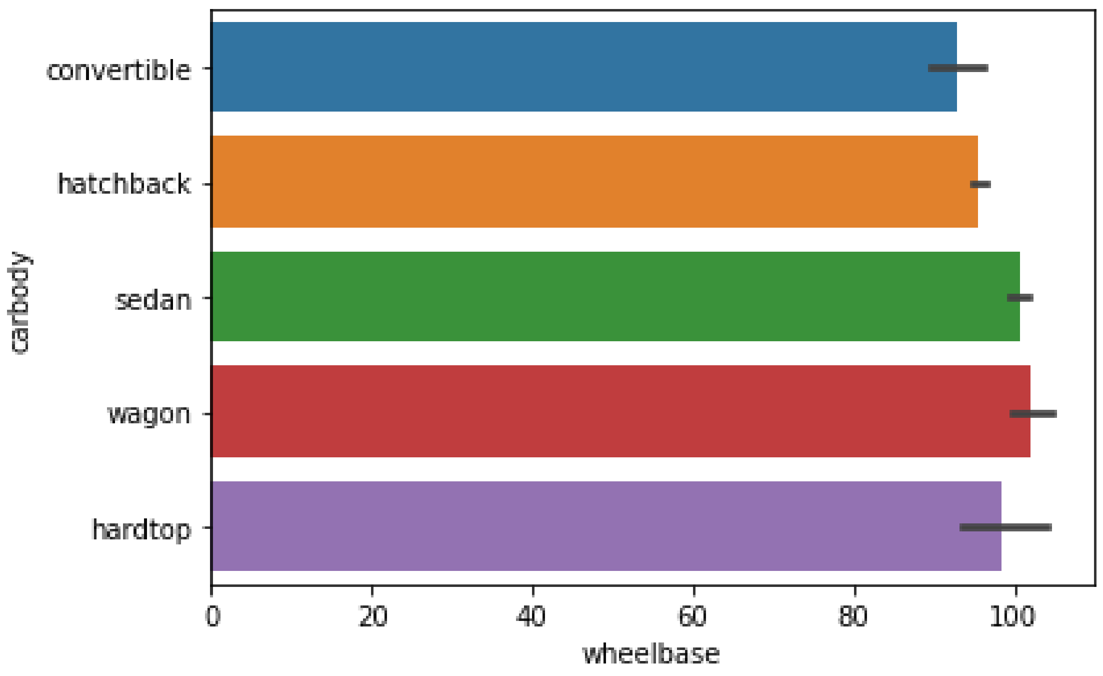
[109]:
```
## scatter plot
sns.relplot(data=carPriceData, x='wheelbase', y='enginesize',
hue='fueltype')
```

[109]:
```
<seaborn.axisgrid.FacetGrid at 0x14da8314e80>
```



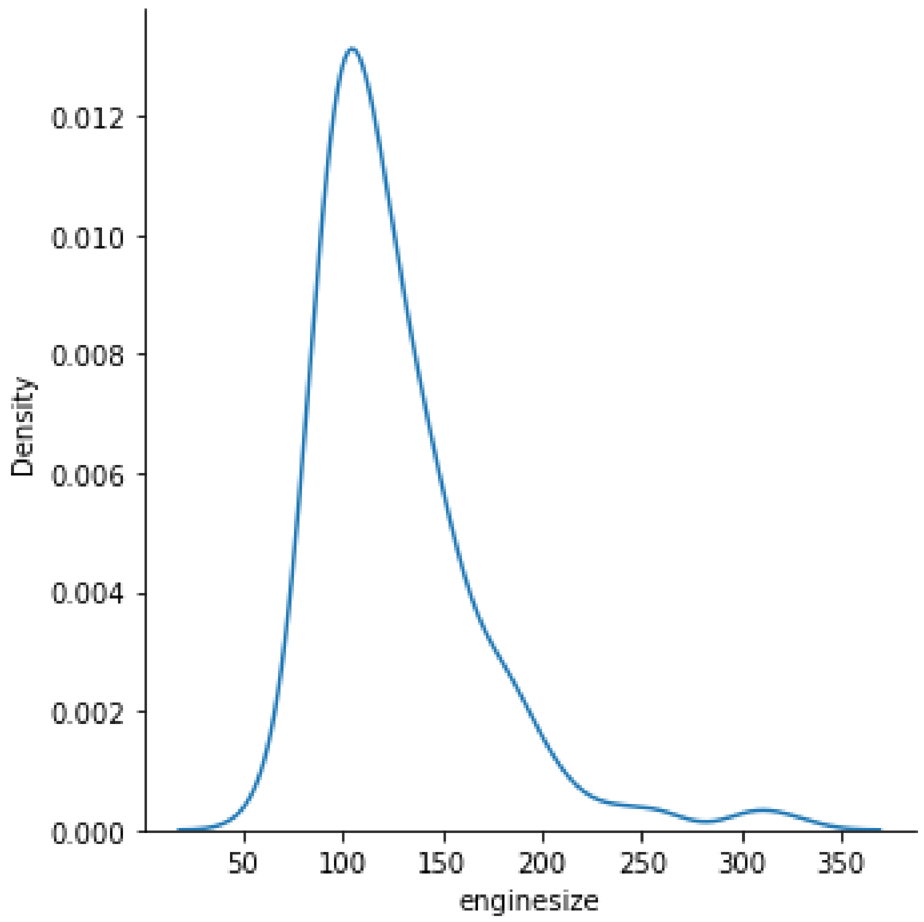
```
[110]: ## bar chart
       sns.barplot(x="wheelbase", y="carbody", data=carPriceData)
```

```
[110]: <AxesSubplot:xlabel='wheelbase', ylabel='carbody'>
```

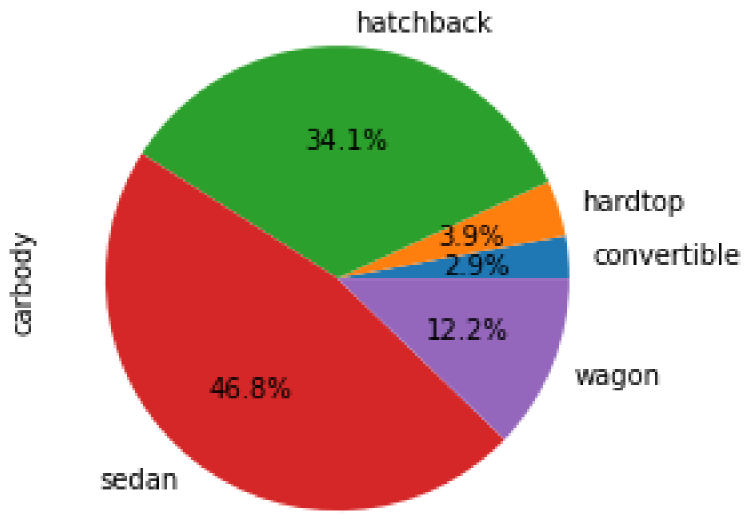


```
[111]: ## Density plot
       sns.displot(data=carPriceData, x='enginesize', kind='kde')
```

```
[111]: <seaborn.axisgrid.FacetGrid at 0x14daa598b80>
```



[112]:
```
## pie chart
data = carPriceData.groupby("carbody")["carbody"].count()
data.plot.pie(autopct="%.1f%%");
```


hatchback
34.1%
hardtop
3.9%
convertible
2.9%
12.2%
wagon
46.8%
sedan
carbody

```
[ ]:
```