Electronics & ICT Academy
National Institute of Technology, Warangal

Post Graduate Diploma in
Machine Learning and Artificial Intelligence

Python for AI-ML

Question Bank 2

edureka!

1. Write a Python program to fetch the title of the URL specified.

In [1]:

```
from bs4 import BeautifulSoup
import requests

url = input("Enter URL: ")
req = requests.session()
content = req.get(url)
soup = BeautifulSoup(content.text, 'html.parser')
#print(soup.prettify())
print(soup.title)
```

```
Enter URL: https://wikipedia.org (https://wikipedia.org)
<title>Wikipedia</title>
```

2. Write a Python program to print content of elements that contain a specified string of a given web page.

In [2]:

```
import requests
import re
from bs4 import BeautifulSoup as bs
url = 'https://www.python.org/'
reqs = requests.get(url)
soup = bs(reqs.text,'html.parser')
print("\nContent of elements that contain 'Python' string:")
str1 = soup.find_all(string=re.compile('Python'))
for txt in str1:
    print(" ".join(txt.split()))
```

```
Content of elements that contain 'Python' string:
Welcome to Python.org
Python
The Python Network
Python Brochure
Python Books
Python Essays
Python Conferences
Python Logo
Python Wiki
Python News
Python Events
Python Events Archive
# Python 3: Fibonacci series up to n
The core of extensible programming is defining functions. Python allows mand
atory and optional arguments, keyword arguments, and even arbitrary argument
lists.
More about defining functions in Python 3
# Python 3: List comprehensions
Lists (known as arrays in other languages) are one of the compound data type
s that Python understands. Lists can be indexed, sliced and manipulated with
other built-in functions.
More about lists in Python 3
# Python 3: Simple arithmetic
Calculations are simple with Python, and expression syntax is straightforwar
d: the operators
More about simple math functions in Python 3
# Python 3: Simple output (with Unicode)
>>> print("Hello, I'm Python!")
Hello, I'm Python!
What is your name? Python Hi, Python.
Experienced programmers in any other language can pick up Python very quickl
y, and beginners find the clean syntax and indentation structure easy to lea
rn.
with our Python 3 overview.
Python knows the usual control flow statements that other languages speak —
More control flow tools in Python 3
Python is a programming language that lets you work quickly
Whether you're new to programming or an experienced developer, it's easy to
learn and use Python.
Python source code and installers are available for download for all version
s!
Python 3.7.4
Documentation for Python's standard library, along with tutorials and guide
s, are available online.
Looking for work or have a Python related position that you're trying to hir
e for? Our
```

```
Python 3.5.8rc1 is now available
Python 3.8.0b4 is now available for testing
Python Software Foundation Fellow Members for Q1 & Q2 2019
Humble Bundle by No Starch supports the Python Software Foundation!
"Python is all about automating repetitive tasks, leaving more time for your
other SEO efforts."
Using Python scripts to analyse SEO and broken links on your site
Use Python for…
wxPython
IPython
Python Enhancement Proposals
: The future of Python
Python Software Foundation
The mission of the Python Software Foundation is to promote, protect, and ad
vance the Python programming language, and to support and facilitate the gro
wth of a diverse and international community of Python programmers.
Python Brochure
Python Books
Python Essays
Python Conferences
Python Logo
Python Wiki
Python News
Python Events
Python Events Archive
Python Software Foundation
```

3. Write a Python program to download an image from the URL provided.

In [3]:

```
import requests
url=input("Enter URL of the image required: ")
req = requests.get(url, stream=True)
req.raise_for_status()
with open('REQDownload.jpg', 'wb') as fd:
    for chunk in req.iter_content(chunk_size=50000):
        print('Received a Chunk')
        fd.write(chunk)
```

```
Enter URL of the image required: https://miro.medium.com/max/490/0*RYfebqkzW
Z3RToOh.png (https://miro.medium.com/max/490/0*RYfebqkzWZ3RToOh.png)
Received a Chunk
Received a Chunk
Received a Chunk
Received a Chunk
```

4. Write a Python program to extract all URL's from the specified link.

In [4]:

```
import requests
from bs4 import BeautifulSoup
url=input("Enter the URL to scrape: ")
r=requests.get(url)
c=r.content
soup=BeautifulSoup(c,"html.parser")
for link in soup.find_all('a'):
    print(link.get('href'))
```

```
Enter the URL to scrape: https://wikipedia.org (https://wikipedia.org)
//en.wikipedia.org/
//es.wikipedia.org/
//ja.wikipedia.org/
//de.wikipedia.org/
//ru.wikipedia.org/
//fr.wikipedia.org/
//it.wikipedia.org/
//zh.wikipedia.org/
//pt.wikipedia.org/
//pl.wikipedia.org/
//de.wikipedia.org/
//en.wikipedia.org/
//es.wikipedia.org/
//fr.wikipedia.org/
//it.wikipedia.org/
//nl.wikipedia.org/
//ja.wikipedia.org/
//pl.wikipedia.org/
```

5. Write a Python program to test if a given page is found or not on the server.

In [5]:

```
from urllib3.exceptions import HTTPError as BaseHTTPError
import requests

class RequestException(IOError):
    """This is the basis to raise errors"""

class HTTPError(RequestException):
    """An HTTP error occurred."""

class URLError(RequestException):
    """An HTTP error occurred."""

try:
    html = requests.get("http://www.example.com/")
except HTTPError as e:
    print("HTTP error")
except URLError as e:
    print("Server not found!")
else:
    print("HTML Details")
    print(html.content)
```

```
HTML Details
b'<!doctype html>\n<html>\n<head>\n    <title>Example Domain</title>\n\n
<meta charset="utf-8" />\n    <meta http-equiv="Content-type" content="text/
html; charset=utf-8" />\n    <meta name="viewport" content="width=device-wid
th, initial-scale=1" />\n    <style type="text/css">\n    body {\n        ba
ckground-color: #f0f0f2;\n        margin: 0;\n        padding: 0;\n        f
ont-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;\n
\n    }\n    div {\n        width: 600px;\n        margin: 5em auto;\n
padding: 50px;\n        background-color: #fff;\n        border-radius: 1e
m;\n    }\n    a:link, a:visited {\n        color: #38488f;\n        text-de
coration: none;\n    }\n    @media (max-width: 700px) {\n        body {\n
background-color: #fff;\n        }\n        div {\n            width: aut
o;\n            margin: 0 auto;\n            border-radius: 0;\n
padding: 1em;\n        }\n    }\n    </style>    \n</head>\n\n<body>\n<div>
\n    <h1>Example Domain</h1>\n    <p>This domain is established to be used
for illustrative examples in documents. You may use this\n    domain in exam
ples without prior coordination or asking for permission.</p>\n    <p><a hre
f="http://www.iana.org/domains/example">More information...</a></p>\n</div>
\n</body>\n</html>\n'
```

6. Write a python program to fetch data using a test API and display it.

HINT: Test Api is, https://jsonplaceholder.typicode.com/posts (https://jsonplaceholder.typicode.com/posts). Use get().

In [6]:

```
import requests
import json

# Make it a bit prettier..
print("-" * 30)
print( "This will show the Most Popular Posts on Social Media")
print("-" * 30)
# Get the feed
data = requests.get("https://jsonplaceholder.typicode.com/posts").json()
# Loop through the result.
for item in data[0]['title']:
    print("Post Title: ", (item))
```

```
------------------------------
This will show the Most Popular Posts on Social Media
------------------------------
Post Title:  s
Post Title:  u
Post Title:  n
Post Title:  t
Post Title:
Post Title:  a
Post Title:  u
Post Title:  t
Post Title:
Post Title:  f
Post Title:  a
Post Title:  c
Post Title:  e
Post Title:  r
Post Title:  e
Post Title:
Post Title:  r
Post Title:  e
Post Title:  p
Post Title:  e
Post Title:  l
Post Title:  l
Post Title:  a
Post Title:  t
Post Title:
Post Title:  p
Post Title:  r
Post Title:  o
Post Title:  v
Post Title:  i
Post Title:  d
Post Title:  e
Post Title:  n
Post Title:  t
Post Title:
Post Title:  o
Post Title:  c
Post Title:  c
Post Title:  a
Post Title:  e
Post Title:  c
Post Title:  a
```

```
Post Title:  t
Post Title:  i
Post Title:
Post Title:  e
Post Title:  x
Post Title:  c
Post Title:  e
Post Title:  p
Post Title:  t
Post Title:  u
Post Title:  r
Post Title:  i
Post Title:
Post Title:  o
Post Title:  p
Post Title:  t
Post Title:  i
Post Title:  o
Post Title:
Post Title:  r
Post Title:  e
Post Title:  p
Post Title:  r
Post Title:  e
Post Title:  h
Post Title:  e
Post Title:  n
Post Title:  d
Post Title:  e
Post Title:  r
Post Title:  i
Post Title:  t
```

7. Write a Python program to fetch visitor data on basis of browser of the past 90 days on data.gov

In [7]:

```
import requests
r = requests.get("https://analytics.usa.gov/data/live/browsers.json")
print("90 days of visits broken down by browser for all sites:")
print(r.json()['totals']['browser'])
```

```
90 days of visits broken down by browser for all sites:
{'Chrome': 1594938529, 'Safari': 1025371945, 'Internet Explorer': 25411984
7, 'Firefox': 138636924, 'Edge': 136335943, 'Samsung Internet': 89384948,
'Safari (in-app)': 35875729, 'Android Webview': 30933842, 'Opera': 1155336
4, 'Amazon Silk': 7884796, 'Opera Mini': 5168229, 'UC Browser': 3853923,
'Android Browser': 790376, 'YaBrowser': 764984, 'Coc Coc': 390919, 'Mozill
a Compatible Agent': 651536, 'Puffin': 229300, 'BlackBerry': 152321, 'Mozi
lla': 146941, 'SeaMonkey': 146557, 'Lore Document Collector 225ed41853c72a
372394acc33e4887b2': 3630, 'Maxthon': 94596, 'Playstation 4': 132686, 'Lor
e Document Collector def2979cc8866209a3751fa344350378': 1623, 'Lore Docume
nt Collector 7ae51970a8d7ea9bbbc98ecd9ab4f1ab': 1761, 'Amazon.com': 95826,
'Mercari_d': 329644, 'Lore Document Collector 1bb1dbb613d0db2041e35f52fea6
72c7': 32211, 'Lore Document Collector d5057a753cc168ab2754b07bf5968c8c':
1521, 'BestBuy': 33333, 'MagentaNews': 44432, 'Iron': 22399, 'Carousel': 3
0918, 'Lore Document Collector 71baa0b71725058671c6e86ca7f06181': 407, 'St
atusCake_Pagespeed_Indev': 23694, 'UCWEB': 17802, 'Lore Document Collector
f543adb427162d1c8888da167914499a': 275, 'DoximityWebView': 8294, 'Papers':
64166, 'Nintendo Browser': 17953, 'osee2unifiedRelease': 14836, 'Lore Docu
ment Collector 5a20b182ca368a24e6462a7ac6947051': 1385, 'Lore Document Col
```

8. Write a Python program to verify SSL certificates for HTTPS requests using requests module.

NOTE: Requests verifies SSL certificates for HTTPS requests, just like a web browser. By default, SSL verification is enabled, and Requests will throw a SSLError if it's unable to verify the certificate.

In [8]:

```
import requests
url=input("Enter a HTTPS URL: ")
print(requests.get(url))
```

```
Enter a HTTPS URL: https://wikipedia.org (https://wikipedia.org)
<Response [200]>
```

9. Write a Python program to get the number earthquakes detected worldwide with magnitude greater than 4.5 by the USGS.

API: http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.5_month.csv (http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.5_month.csv)

In [9]:

```
import csv
import requests
csvurl = 'http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/4.5_month.csv'
rows = list(csv.DictReader(requests.get(csvurl).text.splitlines()))
print("The number of magnitude 4.5+ earthquakes detected worldwide by the USGS:", len(rows)
```

```
The number of magnitude 4.5+ earthquakes detected worldwide by the USGS: 398
```

10. Write a Python program to get movie name, year and a brief summary of 10 random movies.

In [10]:

```
from bs4 import BeautifulSoup
import requests
import random
def get_imd_movies(url):
    page = requests.get(url)
    soup = BeautifulSoup(page.text, 'html.parser')
    movies = soup.find_all("td", class_="titleColumn")
    random.shuffle(movies)
    return movies
def get_imd_summary(url):
    movie_page = requests.get(url)
    soup = BeautifulSoup(movie_page.text, 'html.parser')
    return soup.find("div", class_="summary_text").contents[0].strip()

def get_imd_movie_info(movie):
    movie_title = movie.a.contents[0]
    movie_year = movie.span.contents[0]
    movie_url = 'http://www.imdb.com' + movie.a['href']
    return movie_title, movie_year, movie_url

def imd_movie_picker():
    ctr=0
    print("-----------------------------------------")
    for movie in get_imd_movies('http://www.imdb.com/chart/top'):
        movie_title, movie_year, movie_url = get_imd_movie_info(movie)
        movie_summary = get_imd_summary(movie_url)
        print(movie_title, movie_year)
        print(movie_summary)
        print("-----------------------------------------")
        ctr=ctr+1
        if (ctr==10):
          break;
if __name__ == '__main__':
    imd_movie_picker()
```

```
-----------------------------------------
Vertigo (1958)
A former police detective juggles wrestling with his personal demons and b
ecoming obsessed with a hauntingly beautiful woman.
-----------------------------------------
Network (1976)
A television network cynically exploits a deranged former anchor's ravings
and revelations about the news media for its own profit.
-----------------------------------------
Il buono, il brutto, il cattivo (1966)
A bounty hunting scam joins two men in an uneasy alliance against a third
in a race to find a fortune in gold buried in a remote cemetery.
-----------------------------------------
PK (2014)
An alien on Earth loses the only device he can use to communicate with his
spaceship. His innocent nature and child-like questions force the country
to evaluate the impact of religion on its people.
-----------------------------------------
Eternal Sunshine of the Spotless Mind (2004)
When their relationship turns sour, a couple undergoes a medical procedure
to have each other erased from their memories.
-----------------------------------------
Harry Potter and the Deathly Hallows: Part 2 (2011)
Harry, Ron, and Hermione search for Voldemort's remaining Horcruxes in the
```

```
ir effort to destroy the Dark Lord as the final battle rages on at Hogwart
s.
-----------------------------------------
Star Wars: Episode V - The Empire Strikes Back (1980)
After the Rebels are brutally overpowered by the Empire on the ice planet
Hoth, Luke Skywalker begins Jedi training with Yoda, while his friends are
pursued by Darth Vader.
-----------------------------------------
Ah-ga-ssi (2016)
A woman is hired as a handmaiden to a Japanese heiress, but secretly she i
s involved in a plot to defraud her.
-----------------------------------------
Drishyam (2015)
Desperate measures are taken by a man who tries to save his family from th
e dark side of the law, after they commit an unexpected crime.
-----------------------------------------
Logan (2017)
In a future where mutants are nearly extinct, an elderly and weary Logan l
eads a quiet life. But when Laura, a mutant child pursued by scientists, c
omes to him for help, he must get her to safety.
-----------------------------------------
```

11. Write a Python program to find the live weather report (temperature, wind speed) of a given city.

API:

http://api.openweathermap.org/data/2.5/weather?'+query+'&APPID=b35975e18dc93725acb092f7272cc6b8
(http://api.openweathermap.org/data/2.5/weather?'+query+'&APPID=b35975e18dc93725acb092f7272cc6b8

In [11]:

```
import requests

def weather_data(query):
    res=requests.get('http://api.openweathermap.org/data/2.5/weather?'+query+'&APPID=b35975
    return res.json();
def print_weather(result,city):
    print("{}'s temperature: {}°C ".format(city,result['main']['temp']))
    print("Wind speed: {} m/s".format(result['wind']['speed']))
    print("Description: {}".format(result['weather'][0]['description']))
    print("Weather: {}".format(result['weather'][0]['main']))
def main():
    city=input('Enter the city:')
    print()
    try:
        query='q='+city;
        w_data=weather_data(query);
        print_weather(w_data, city)
        print()
    except:
         print('City name not found...')
if __name__=='__main__':
    main()
```

```
Enter the city:Bangalore

Bangalore's temperature: 26.78°C
Wind speed: 4.1 m/s
Description: scattered clouds
Weather: Clouds
```

12. Write a Python program to list all language names and number of related articles in the order they appear in wikipedia.org.

In [12]:

```
from urllib.request import urlopen
import requests
from bs4 import BeautifulSoup
html = requests.get('https://www.wikipedia.org/')
bs = BeautifulSoup(html.text, "html.parser")
nameList = bs.findAll('a', {'class' : 'link-box'})
for name in nameList:
  print(name.get_text())
```

```
English
5 930 000+ articles


EspaÃ±ol
1 545 000+ artÃculos


æ□¥æ□¬èª□
1 168 000+ è¨□äº□


Deutsch
2 343 000+ Artikel


Ð Ñ□Ñ□Ñ□ÐºÐ¸Ð¹
1 568 000+ Ñ□Ñ□Ð°Ñ□ÐµÐ¹


FranÃ§ais
2 139 000+ articles


Italiano
1 552 000+ voci


ä¸æ□□
1 073 000+ æ¢□ç□®


PortuguÃªs
1 013 000+ artigos


Polski
1 359 000+ haseÅ□
```

13. Write a Python program to count number of tweets by a given Twitter account.

In [13]:

```
from bs4 import BeautifulSoup
import requests

handle = input('Input your account name on Twitter: ')
temp = requests.get('https://twitter.com/'+handle)
bs = BeautifulSoup(temp.text,'lxml')

try:
    tweet_box = bs.find('li',{'class':'ProfileNav-item ProfileNav-item--tweets is-active'})
    tweets= tweet_box.find('a').find('span',{'class':'ProfileNav-value'})
    print("{} tweets {} number of tweets.".format(handle,tweets.get('data-count')))

except:
    print('Account name not found...')
```

```
Input your account name on Twitter: narendramodi
narendramodi tweets 24543 number of tweets.
```

14. Write a Python program to that retrieves an arbitrary Wikipedia page of the users choice and creates a list of links on that page.

In [14]:

```
import requests
from bs4 import BeautifulSoup

page=input("Enter Wikisearch Key: ")
html = requests.get("https://en.wikipedia.org/wiki/"+page)
bsObj = BeautifulSoup(html.text)
for link in bsObj.findAll("a"):
  if 'href' in link.attrs:
    print(link.attrs['href'])
```

```
/wiki/Category:Disambiguation_pages
/wiki/Category:Disambiguation_pages_with_short_description
/wiki/Category:All_article_disambiguation_pages
/wiki/Category:All_disambiguation_pages
/wiki/Category:Animal_common_name_disambiguation_pages
/wiki/Special:MyTalk
/wiki/Special:MyContributions
/w/index.php?title=Special:CreateAccount&returnto=Python
/w/index.php?title=Special:UserLogin&returnto=Python
/wiki/Python
/wiki/Talk:Python
/wiki/Python
/w/index.php?title=Python&action=edit
/w/index.php?title=Python&action=history
/wiki/Main_Page
/wiki/Main_Page
/wiki/Portal:Contents
/wiki/Portal:Featured_content
/wiki/Portal:Current_events
/wiki/Special:Random
```

15. Write a Python program to get the number of security alerts issued by US-CERT in the current year.

Source: https://www.us-cert.gov/ncas/alerts (https://www.us-cert.gov/ncas/alerts)

In [15]:

```
import requests
from lxml import html
url = 'https://www.us-cert.gov/ncas/alerts'
doc = html.fromstring(requests.get(url).text)
print("The number of security alerts issued by US-CERT in the current year:")
print(len(doc.cssselect('.item-list li')))
```

```
The number of security alerts issued by US-CERT in the current year:
30
```

16. Write a python program to remove all entries with a null value present.

In [16]:

```
import pandas as pd
df = pd.read_csv("lap_times.csv")
pd.isnull(df).any()
print(df.shape)
df = df.dropna(how='any',axis=0)
pd.isnull(df).any()
print(df.shape)
print(df)
```

```
(100, 5)
(97, 5)
    Position  Year         Make                          Model     Time
0          1  2009      Radical                       SR8 LM  06:48.0
1          2  2005      Radical                          SR8  06:56.1
2          3  2013      Porsche                    918 Spyder  06:57.0
3          4  2015  Lamborghini  Aventador LP 750-4 Superveloce  06:59.7
4          5  2015       Nissan                    GT-R Nismo  07:08.7
..       ...   ...          ...                           ...      ...
95        96  2003      Ferrari                       360 CS  07:56.0
96        97  2009      Ferrari                   California GT  07:56.0
97        98  2009      Porsche    Panamera Sport Chrono Turbo  07:56.0
98        99  2009      Porsche                 Panamera Turbo  07:56.0
99       100  2002    Chevrolet                   Corvette Z06  07:56.0

[97 rows x 5 columns]
```

17. Write a python program to replace all missing

values in the dataset with a desired value accepted from the user.

In [17]:

```
import pandas
import numpy as np
data = {'one': pandas.Series([1, 2, 5],
         index=['a', 'b', 'e']),
        'two': pandas.Series([1, 2, 3, 4],
         index=['a', 'b', 'c', 'd'])}
table = pandas.DataFrame(data)
print("Before:\n",table)
print()
buff=int(input("Enter a buffer value: "))
table['one'] = table['one'].replace(np.nan, buff)
table['two'].fillna(buff,inplace = True)
print("After:\n",table)
```

```
Before:
    one  two
a  1.0  1.0
b  2.0  2.0
c  NaN  3.0
d  NaN  4.0
e  5.0  NaN

Enter a buffer value: 999
After:
      one    two
a    1.0    1.0
b    2.0    2.0
c  999.0    3.0
d  999.0    4.0
e    5.0  999.0
```

18. Develop a Python code to impute a mean value to all the null fields present in the data provided.

In [18]:

```
import pandas
import numpy as np
data = {'one': pandas.Series([1, 2, 5],
                             index=['a', 'b', 'e']),
        'two': pandas.Series([1, 2, 3, 4],
                             index=['a', 'b', 'c', 'd'])}
table = pandas.DataFrame(data)
print("Before:\n",table)
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(table)
imputed_data = imputer.transform(table.values)
print("\nAfter:\n",pandas.DataFrame(imputed_data, columns=['One','Two']))
```

```
Before:
    one  two
a  1.0  1.0
b  2.0  2.0
c  NaN  3.0
d  NaN  4.0
e  5.0  NaN

After:
        One  Two
0  1.000000  1.0
1  2.000000  2.0
2  2.666667  3.0
3  2.666667  4.0
4  5.000000  2.5
```

19. Write a pythonic method to convert categorical data into numerical data.

In [19]:

```
import pandas as pd
data = {'Customer_id': pd.Series([1,2,3,4,5]),
        'Loan_type': pd.Series(['Home Loan','Personal Loan','Education Loan','Home Loan','C
        'Income': pd.Series(['30K','25K','15K','40K','35K'])}
loan_info = pd.DataFrame(data)
print(loan_info)
print()
loan_info = pd.get_dummies(loan_info, prefix_sep='_', drop_first=True)
print(loan_info)
```

```
   Customer_id       Loan_type Income
0            1       Home Loan    30K
1            2   Personal Loan    25K
2            3  Education Loan    15K
3            4       Home Loan    40K
4            5     Credit Loan    35K

   Customer_id  Loan_type_Education Loan  Loan_type_Home Loan  \
0            1                         0                    1
1            2                         0                    0
2            3                         1                    0
3            4                         0                    1
4            5                         0                    0

   Loan_type_Personal Loan  Income_25K  Income_30K  Income_35K  Income_40K
0                        0           0           1           0           0
1                        1           1           0           0           0
2                        0           0           0           0           0
3                        0           0           0           0           1
4                        0           0           0           1           0
```

20. Write a pyhton program to convert all the categorical data with ordinal values in the dataset into numerical form.

HINT: Use LabelEncoder().

In [20]:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

data = {'Customer_id': pd.Series([1,2,3,4,5]),
        'Loan_type': pd.Series(['Home Loan',
                                'Personal Loan',
                                'Education Loan',
                                'Home Loan',
                                'Credit Loan']),
        'Income': pd.Series(['30K','25K','15K',
                             '40K','35K'])}
loan_info = pd.DataFrame(data)
print(loan_info)
print()
labelencoder = LabelEncoder()
loan_info_upd=loan_info.apply(labelencoder.fit_transform)
print("Transformed Numerical value: ")
print(loan_info_upd)
```

```
   Customer_id       Loan_type Income
0            1       Home Loan    30K
1            2   Personal Loan    25K
2            3  Education Loan    15K
3            4       Home Loan    40K
4            5     Credit Loan    35K

Transformed Numerical value:
   Customer_id  Loan_type  Income
0            0          2       2
1            1          3       1
2            2          1       0
3            3          2       4
4            4          0       3
```

21. Implement OneHotEncoder from sklearn to convert categorical data into numerically parsable data.

In [21]:

```
from numpy import argmax
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
# define example
data = ['cold', 'cold', 'warm', 'cold', 'hot', 'hot', 'warm', 'cold', 'warm', 'hot']
values = list(data)
print(values)
print()
# integer encode
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values)
print("Integer Encoding: ")
print(integer_encoded)
print()
# binary encode
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
print("OneHotEncoding: ")
print(onehot_encoded)
print()
# invert first example
inverted = label_encoder.inverse_transform([argmax(onehot_encoded[0, :])])
print("Inverting to Categorical: ")
print(inverted)
```

```
['cold', 'cold', 'warm', 'cold', 'hot', 'hot', 'warm', 'cold', 'warm', 'ho
t']

Integer Encoding:
[0 0 2 0 1 1 2 0 2 1]

OneHotEncoding:
[[1. 0. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 1. 0.]]

Inverting to Categorical:
['cold']

c:\users\nikhilmeduri\appdata\local\programs\python\python37-32\lib\site-pac
kages\sklearn\preprocessing\_encoders.py:415: FutureWarning: The handling of
integer data will change in version 0.22. Currently, the categories are dete
rmined based on the range [0, max(values)], while in the future they will be
determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify
"categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the cat
egories to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

22. Implement a pythonic code to standardize the data given as input and the display the standardized data.

In [22]:

```
from sklearn.preprocessing import StandardScaler

X=[[10],[15],[22],[33],[25],[34],[56]]
Y=[[101],[105],[222],[333],[225],[334],[556]]
print("Before standardisation X values are ", X)
print()
print("Before standardisation Y values are ", Y)
print()
sc_X = StandardScaler()
X = sc_X.fit_transform(X)
Y = sc_X.fit_transform(Y)

print("After standardisation X values are: \n", X)
print()
print("After standardisation Y values are:\n", Y)
```

```
Before standardisation X values are  [[10], [15], [22], [33], [25], [34], [5
6]]

Before standardisation Y values are  [[101], [105], [222], [333], [225], [33
4], [556]]

After standardisation X values are:
 [[-1.27049317]
 [-0.91475508]
 [-0.41672176]
 [ 0.36590203]
 [-0.20327891]
 [ 0.43704965]
 [ 2.00229723]]

After standardisation Y values are:
 [[-1.14102498]
 [-1.11369504]
 [-0.31429431]
 [ 0.44411152]
 [-0.29379685]
 [ 0.450944  ]
 [ 1.96775565]]
```

23. Write a Python code to utilise the MinMax Scaler method to reiterate all values in a provided range.

In [23]:

```
import numpy as np
from sklearn import preprocessing
data1 = np.array([[-117.3],
                  [27.5],
                  [0],
                  [-20.9],
                  [1000]])
minmax_scale = preprocessing.MinMaxScaler(feature_range=(2, 3))
scaled = minmax_scale.fit_transform(data1)
scaled
```

Out[23]:

```
array([[2.        ],
       [2.12959814],
       [2.10498523],
       [2.08627942],
       [3.        ]])
```

24. Implement a python code such that you convert numerical array into an array with numbers in the interqaurtile range.

HINT: Use the RobustScaler() function.

In [24]:

```
import numpy as np
from sklearn import preprocessing
data1 = np.array([[-101.3],
                  [27.5],
                  [1],
                  [-200.9],
                  [1000]])
robust_scaler = preprocessing.RobustScaler()
scaled = robust_scaler.fit_transform(data1)
scaled
```

Out[24]:

```
array([[-0.79425466],
       [ 0.20574534],
       [ 0.        ],
       [-1.56754658],
       [ 7.75621118]])
```

25. Write a pythonic code to normalize the arrays/data provided by the user.

In [25]:

```
import numpy as np
from sklearn import preprocessing
data1 = np.array(
    [[5.1,3.5,1.4,0.2],
     [4.9,3.0,1.4,0.2],
     [4.7,3.2,1.3,0.2],
     [4.6,3.1,1.5,0.2],
     [5.0,3.6,1.4,0.2]])
print("Original Data: ")
print(data1)
print()
normalized_data = preprocessing.normalize(data1)
print("Normalized Data: ")
print(normalized_data)
```

```
Original Data:
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]

Normalized Data:
[[0.80377277 0.55160877 0.22064351 0.0315205 ]
 [0.82813287 0.50702013 0.23660939 0.03380134]
 [0.80533308 0.54831188 0.2227517  0.03426949]
 [0.80003025 0.53915082 0.26087943 0.03478392]
 [0.790965   0.5694948  0.2214702  0.0316386 ]]
```

26. Develop a python code to generate sample data out of existing data and append it to the dataset.

In [26]:

```
import pandas as pd
# making data frame from csv file
data = pd.read_csv("employees.csv")
# generating one row
rows = data.sample(frac =.25)
# checking if sample is 0.25 times data or not
if (0.25*(len(data))== len(rows)):
    print( "Cool")
    print(len(data), len(rows))
# display
rows
```

```
Cool
1000 250
```

Out[26]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
418	Julia	Female	3/2/1982	12:52 PM	36403	2.664	True	Finance
5	Dennis	Male	4/18/1987	1:35 AM	115163	10.125	False	Legal
368	Marilyn	Female	12/17/1982	12:10 PM	147183	8.748	False	Business Development
307	Marilyn	Female	9/26/1981	2:23 AM	86386	2.937	False	Distribution
910	Melissa	Female	10/22/2002	1:20 AM	45223	8.879	True	Legal
...	...	...	...	...	...	...	...	...
927	Philip	Male	3/11/2005	10:56 AM	103557	16.014	True	Business Development
807	Mary	Female	11/6/2011	8:32 AM	115057	2.089	False	Finance
876	Terry	NaN	9/11/1992	4:41 PM	41238	8.219	False	Marketing
810	Ralph	Male	2/10/2000	4:48 PM	89854	7.227	False	Business Development
358	Scott	Male	12/17/2011	3:45 AM	90429	4.450	False	Product

250 rows × 8 columns

27. Use filter() function to subset all columns in a dataframe which has the letter provided by user in its name.

In [27]:

```
import pandas as pd

# Creating the dataframe
df = pd.read_csv("employees.csv")
# Using regular expression to extract all
# columns which has letter 'a' or 'A' in its name.
regpat=input("Enter letter to filter out: ")
df.filter(regex = regpat)
```

```
Enter letter to filter out: L
```

Out[27]:

	Last Login Time
0	12:42 PM
1	6:53 AM
2	11:17 AM
3	1:00 PM
4	4:47 PM
...	...
995	6:09 AM
996	6:30 AM
997	12:39 PM
998	4:45 PM
999	6:24 PM

1000 rows × 1 columns

28. Develop a Pythonic code to generate sample data and replace all the null values with dummy data.

In [28]:

```
import pandas as pd
# making data frame from csv file
data = pd.read_csv("employees.csv")
df = pd.DataFrame(data)
# generating one row
rows = data.sample(frac =.25)
# checking if sample is 0.25 times data or not
if (0.25*(len(data))== len(rows)):
    print( "Cool")
    print(len(data), len(rows))
# display
buff=input("Enter a buffer value: ")
res = df.apply(lambda x: x.fillna(00) if x.dtype.kind in 'biufcOSUV' else x.fillna(buff))
#data=data['First Name'].fillna(buff,inplace = True)
print("After:\n")
rows
```

```
Cool
1000 250
Enter a buffer value: 999
After:
```

Out[28]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus %	Senior Management	Team
196	Steven	Male	11/10/1985	10:53 AM	62719	19.127	False	Client Services
324	Ruby	Female	12/10/1988	8:27 PM	76707	6.031	False	Business Development
887	David	Male	12/5/2009	8:48 AM	92242	15.407	False	Legal
373	Kenneth	Male	4/13/1999	10:28 PM	81839	12.072	False	Sales
844	Maria	NaN	6/19/1985	1:48 AM	148857	8.738	False	Legal
...	...	...	...	...	...	...	...	...
704	Thomas	Male	9/7/1991	9:51 AM	65251	11.211	False	Distribution
168	Peter	NaN	9/3/1987	5:59 PM	38989	7.017	True	Marketing
316	Marie	Female	10/3/2013	8:05 PM	123711	10.966	False	Product
728	Dorothy	Female	10/8/2008	12:22 AM	82744	19.111	True	Client Services
918	Ryan	Male	4/27/1999	3:41 AM	85858	19.475	False	Client Services

250 rows × 8 columns

In [ ]:

```
```

1. Write a program to create a pandas series from range X to Y.

Input: Range X and Y is given in single line seperated with a space

```
10 20
```

Ouput:

```
0    10
1    11
2    12
3    13
4    14
5    15
6    16
7    17
8    18
9    19
10   20
dtype: int32
```

In [1]:

```
import pandas as pd
import numpy as np
X,Y=map(int,input().split(' '))
series=pd.Series(np.arange(X,Y+1))
print(series)
```

```
10 20
0     10
1     11
2     12
3     13
4     14
5     15
6     16
7     17
8     18
9     19
10    20
dtype: int32
```

2. Write a Python program to add, subtract, multiple and divide two Pandas Series.

Input: Two series in each line

```
1, 2, 3, 4, 5
6, 7, 8, 9, 10
```

Output:

```
Addition:  0     7
    1     9
    2    11
    3    13
    4    15
    dtype: int64
    Substraction:  0   -5
    1   -5
    2   -5
    3   -5
    4   -5
    dtype: int64
    Multiplication:  0     6
    1    14
    2    24
    3    36
    4    50
    dtype: int64
    Division:  0    0.166667
    1    0.285714
    2    0.375000
    3    0.444444
    4    0.500000
    dtype: float64
```

In [2]:

```
import pandas as pd
l1=map(int,input().split(','))
l2=map(int,input().split(','))
S1=pd.Series(l1)
S2=pd.Series(l2)
print('Addition: ',S1+S2)
print('Substraction: ',S1-S2)
print('Multiplication: ',S1*S2)
print('Division: ',S1/S2)
```

```
1,2,3,4,5
6,7,8,9,10
Addition:  0     7
1     9
2    11
3    13
4    15
dtype: int64
Substraction:  0   -5
1   -5
2   -5
3   -5
4   -5
dtype: int64
Multiplication:  0     6
1    14
2    24
3    36
4    50
dtype: int64
Division:  0    0.166667
1    0.285714
2    0.375000
3    0.444444
4    0.500000
dtype: float64
```

3. From the raw data below create a Pandas Series

```
[ '  Aron', 'Jackson   ', '   Ahree   ', 'Sam']
```

- Print all elements after stripping spaces from the left and right
- Print all the elements after removing spaces from the left only
- Print all the elements after removing spaces from the right only

In [3]:

```
import pandas as pd
sa=pd.Series([ '  Aron', 'Jackson    ', '    Ahree    ', 'Sam'])
for i in sa:
    print(str(i).strip(),end='|')
print('\n')
for i in sa:
    print(str(i).lstrip(),end='|')
print('\n')
for i in sa:
    print(str(i).rstrip(),end='|')
```

```
Aron|Jackson|Ahree|Sam|

Aron|Jackson    |Ahree    |Sam|

  Aron|Jackson|    Ahree|Sam|
```

4. Write a program to convert a dictionary into Pandas Series.

Input:{'Sam':89,'Aron':82,'Gray':78,'Isla':93,'Ahree':87}

Output:

```
Sam      89
Aron     82
Gray     78
Isla     93
Ahree    87
dtype: int64
```

In [4]:

```
import pandas as pd
my_dict={'Sam':89,'Aron':82,'Gray':78,'Isla':93,'Ahree':87}
series=pd.Series(my_dict)
print(series)
```

```
Sam      89
Aron     82
Gray     78
Isla     93
Ahree    87
dtype: int64
```

5. Write a program to convert the below dictionary into DataFrame and print head(2) and tail(2) of the DataFrame.

Input:

```
{'Name':['Sameer','Leona','Samuel','Jackson','Gray','Sylphia'],
 'Class':[11,11,12,12,11,12],
'Age':[17,17,18,19,17,21,23]
}
```

Output:

```
    Name      Class     Age
0   Sameer      11       17
1   Leona       11       17
    Name      Class     Age
4   Gray        11       21
5   Sylphia     12       23
```

In [5]:

```
import pandas as pd
my_dict={'Name':['Sameer','Leona','Samuel','Jackson','Gray','Sylphia'],
         'Class':[11,11,12,12,11,12], 'Age':[17,17,18,17,21,23] }
df=pd.DataFrame(my_dict)
print(df.head(2))
print(df.tail(2))
```

```
     Name  Class  Age
0  Sameer     11   17
1   Leona     11   17
      Name  Class  Age
4     Gray     11   21
5  Sylphia     12   23
```

6. Convert the given nested list into single series and print the output in sorted form.

Input:[[1,23,12,31,14,12],[32,43,32,42],[65,75,65,57,41,33,68,52]]

Output:

```
0      1.0
2     12.0
5     12.0
4     14.0
1     23.0
3     31.0
8     32.0
6     32.0
15    33.0
14    41.0
9     42.0
7     43.0
17    52.0
13    57.0
10    65.0
12    65.0
16    68.0
11    75.0
dtype: float64
```

In [6]:

```
import pandas as pd
series=pd.Series([[1,23,12,31,14,12],[32,43,32,42],[65,75,65,57,41,33,68,52]])
series=series.apply(pd.Series).stack()
series=series.reset_index(drop=True)
print(series.sort_values(axis=0))
```

```
0      1.0
2     12.0
5     12.0
4     14.0
1     23.0
3     31.0
8     32.0
6     32.0
15    33.0
14    41.0
9     42.0
7     43.0
17    52.0
13    57.0
10    65.0
12    65.0
16    68.0
11    75.0
dtype: float64
```

7. Create a series from 1 to 1000 and select only numbers divisble by 7 and 17.

Output:

```
118    119
237    238
356    357
475    476
594    595
713    714
832    833
951    952
dtype: int32
```

In [7]:

```
import pandas as pd
series=pd.Series(np.arange(1,1001))
print(series[(series % 7==0) & (series%17==0)])
```

```
118    119
237    238
356    357
475    476
594    595
713    714
832    833
951    952
dtype: int32
```

8. Create a program that reads the below dictionary as a DataFrame and iterate over columns and rows.

In [8]:

```
import pandas as pd
my_dict={'Name':['Sameer','Leona','Samuel','Jackson','Gray','Sylphia'],
         'Class':[11,11,12,12,11,12], 'Age':[17,17,18,17,21,23] }
df=pd.DataFrame(my_dict)
for col in df.iteritems():
    print(col)
for row in df.iterrows():
    print(row)
```

```
('Name', 0     Sameer
1      Leona
2     Samuel
3    Jackson
4       Gray
5    Sylphia
Name: Name, dtype: object)
('Class', 0    11
1    11
2    12
3    12
4    11
5    12
Name: Class, dtype: int64)
('Age', 0    17
1    17
2    18
3    17
4    21
5    23
Name: Age, dtype: int64)
(0, Name     Sameer
Class        11
Age          17
Name: 0, dtype: object)
(1, Name     Leona
Class       11
Age         17
Name: 1, dtype: object)
(2, Name     Samuel
Class        12
Age          18
Name: 2, dtype: object)
(3, Name     Jackson
Class         12
Age           17
Name: 3, dtype: object)
(4, Name     Gray
Class      11
Age        21
Name: 4, dtype: object)
(5, Name     Sylphia
Class         12
Age           23
Name: 5, dtype: object)
```

9. Samantha has a dataset of top 50 songs from spotify named 'top50spotify.csv'. Import the dataset as a DataFrame and drop the first column and save it as 'top50.csv'

In [9]:

```
import pandas as pd
top50=pd.read_csv('top50spotify.csv')
top50.head()
```

Out[9]:

	Unnamed: 0	Track.Name	Artist.Name	Genre	Beats.Per.Minute	Energy	Danceability	Lo
0	1	Señorita	Shawn Mendes	canadian pop	117	55	76	
1	2	China	Anuel AA	reggaeton flow	105	81	79	
2	3	boyfriend (with Social House)	Ariana Grande	dance pop	190	80	40	
3	4	Beautiful People (feat. Khalid)	Ed Sheeran	pop	93	65	64	
4	5	Goodbyes (Feat. Young Thug)	Post Malone	dfw rap	150	65	58	

In [10]:

```
col=list(top50.columns)
col[0]='temp'
top50.columns=col
top50=top50.drop('temp',axis=1)
top50.to_csv('top50.csv')
top50.head()
```

Out[10]:

	Track.Name	Artist.Name	Genre	Beats.Per.Minute	Energy	Danceability	Loudness..dB..
0	Señorita	Shawn Mendes	canadian pop	117	55	76	-6
1	China	Anuel AA	reggaeton flow	105	81	79	-4
2	boyfriend (with Social House)	Ariana Grande	dance pop	190	80	40	-4
3	Beautiful People (feat. Khalid)	Ed Sheeran	pop	93	65	64	-8
4	Goodbyes (Feat. Young Thug)	Post Malone	dfw rap	150	65	58	-4

10. Find the average Energy and Length of first 10 songs

In [11]:

```
first_10=top50[:10]
first_10[['Energy','Length.']].sum()/10
```

Out[11]:

```
Energy      65.1
Length.    195.6
dtype: float64
```

11. Find the total length of songs, group by genre from top to bottom.

In [12]:

```
print(top50.groupby('Genre')['Length.'].sum().sort_values(axis=0,ascending=False))
```

```
Genre
dance pop           1621
pop                 1368
latin               1126
edm                  656
reggaeton flow       611
canadian hip hop     579
panamanian pop       514
reggaeton            427
brostep              396
electropop           389
canadian pop         382
dfw rap              333
country rap          272
australian pop       210
atl hip hop          200
boy band             181
escape room          173
big room             164
r&b en espanol       162
pop house            153
trap music           131
Name: Length., dtype: int64
```

12. Print the artist name with the most number of tracks in one genre.

[**Hint**: Group by artist name and genre]

In [13]:

```
import pandas as pd
keys=['Genre','Artist','N_Tracks']
new_df=pd.DataFrame(columns=keys)
i=0
for x,y in top50.groupby(['Genre','Artist.Name']):
    new_df.loc[i]=[x[0],x[1],y['Track.Name'].count()]
    i=i+1
```

In [14]:

```
new_df[new_df.N_Tracks==new_df.N_Tracks.max()]
```

Out[14]:

	Genre	Artist	N_Tracks
27	pop	Ed Sheeran	4

13. Print the data of the tracks created by the artist from the previous question.

In [15]:

```
top50[top50['Artist.Name']=='Ed Sheeran']
```

Out[15]:

	Track.Name	Artist.Name	Genre	Beats.Per.Minute	Energy	Danceability	Loudness..dB..
3	Beautiful People (feat. Khalid)	Ed Sheeran	pop	93	65	64	-8
5	I Don't Care (with Justin Bieber)	Ed Sheeran	pop	102	68	80	-5
37	Antisocial (with Travis Scott)	Ed Sheeran	pop	152	82	72	-5
49	Cross Me (feat. Chance the Rapper & PnB Rock)	Ed Sheeran	pop	95	79	75	-6

14. Create a new column called Rating and inputs data as:

- If popularity is greater than average set as 'Good'
- If popularity is less than average set as 'Bad'

In [16]:

```
avg=top50.Popularity.mean()
rating=[]
for i in top50.Popularity:
    if i <= avg:
        rating.append('Bad')
    else:
        rating.append('Good')

top50['Rating']=rating

top50.head()
```

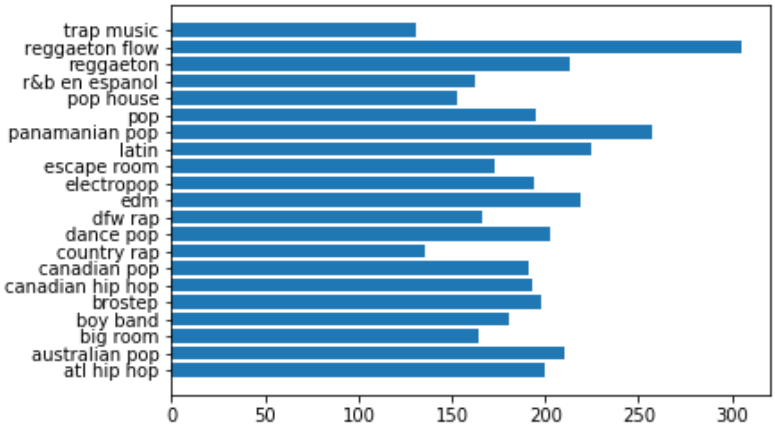
Out[16]:

	Track.Name	Artist.Name	Genre	Beats.Per.Minute	Energy	Danceability	Loudness..dB..
0	Señorita	Shawn Mendes	canadian pop	117	55	76	-6
1	China	Anuel AA	reggaeton flow	105	81	79	-4
2	boyfriend (with Social House)	Ariana Grande	dance pop	190	80	40	-4
3	Beautiful People (feat. Khalid)	Ed Sheeran	pop	93	65	64	-8
4	Goodbyes (Feat. Young Thug)	Post Malone	dfw rap	150	65	58	-4

15. Plot the average length of every genre on a horizontal bar chart.

In [18]:

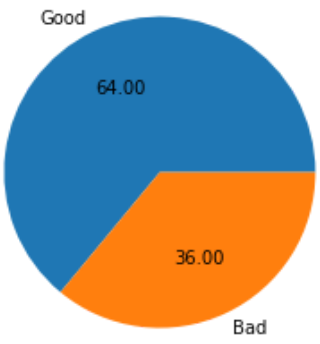
```
import matplotlib.pyplot as plt
my_dict=dict()
for group,data in top50.groupby('Genre'):
    my_dict[group]=data['Length.'].mean()
plt.barh(list(my_dict.keys()),list(my_dict.values()))
plt.show()
```



16. Plot a pie chart of ratings and display the labels.

In [19]:

```
b=top50.Rating[top50.Rating=='Bad'].count()
g=top50.Rating[top50.Rating=='Good'].count()
data=[g,b]
plt.pie(data,labels=['Good','Bad'],autopct='%0.2f')
plt.show()
```



17. Create a pandas series from below dictionary where indices are subjects:

```
{'English':{'Sam':60,'Jackson':74,'Ahree':85},
 'History':{'Gloria':83,'Sam':65,'Isla':78,'Aron':72,'Gray':61},
 'Geography':{'Jackson':92,'Gloria':95,'Isla':82,'Aron':75,'Ahree':76},
 'Mathematics':{'Sam':99,'Gloria':74,'Jackson':89,'Ahree':85,'Gray':95},
 'Science':{'Sam':89,'Aron':82,'Gray':78,'Isla':93,'Ahree':87}
}
```

Output:

	Sam	Jackson	Ahree	Gloria	Isla	Aron	Gray
English	60.0	74.0	85.0	NaN	NaN	NaN	NaN
History	65.0	NaN	NaN	83.0	78.0	72.0	61.0
Geography	NaN	92.0	76.0	95.0	82.0	75.0	NaN
Mathematics	99.0	89.0	85.0	74.0	NaN	NaN	95.0
Science	89.0	NaN	87.0	NaN	93.0	82.0	78.0

In [20]:

```
import pandas as pd
tests={'English':{'Sam':60,'Jackson':74,'Ahree':85},
       'History':{'Gloria':83,'Sam':65,'Isla':78,'Aron':72,'Gray':61},
       'Geography':{'Jackson':92,'Gloria':95,'Isla':82,'Aron':75,'Ahree':76},
       'Mathematics':{'Sam':99,'Gloria':74,'Jackson':89,'Ahree':85,'Gray':95},
       'Science':{'Sam':89,'Aron':82,'Gray':78,'Isla':93,'Ahree':87} }
series=pd.Series(tests)
series=series.apply(pd.Series)
series
```

Out[20]:

	Sam	Jackson	Ahree	Gloria	Isla	Aron	Gray
English	60.0	74.0	85.0	NaN	NaN	NaN	NaN
History	65.0	NaN	NaN	83.0	78.0	72.0	61.0
Geography	NaN	92.0	76.0	95.0	82.0	75.0	NaN
Mathematics	99.0	89.0	85.0	74.0	NaN	NaN	95.0
Science	89.0	NaN	87.0	NaN	93.0	82.0	78.0

18. Convert the above series into DataFrame and replace the null values with zeroes.

In [21]:

```
df=pd.DataFrame(series)
df=df.fillna(0)
df
```

Out[21]:

	Sam	Jackson	Ahree	Gloria	Isla	Aron	Gray
English	60.0	74.0	85.0	0.0	0.0	0.0	0.0
History	65.0	0.0	0.0	83.0	78.0	72.0	61.0
Geography	0.0	92.0	76.0	95.0	82.0	75.0	0.0
Mathematics	99.0	89.0	85.0	74.0	0.0	0.0	95.0
Science	89.0	0.0	87.0	0.0	93.0	82.0	78.0

19. Transpose the DataFrame and create a new column Average fill it by calculating average of all subjects.

In [22]:

```
df=df.transpose()
df
```

Out[22]:

	English	History	Geography	Mathematics	Science
Sam	60.0	65.0	0.0	99.0	89.0
Jackson	74.0	0.0	92.0	89.0	0.0
Ahree	85.0	0.0	76.0	85.0	87.0
Gloria	0.0	83.0	95.0	74.0	0.0
Isla	0.0	78.0	82.0	0.0	93.0
Aron	0.0	72.0	75.0	0.0	82.0
Gray	0.0	61.0	0.0	95.0	78.0

In [24]:

```
average=[]
for i in df.iterrows():
    average.append(i[1].mean())
df['Average']=average
df
```

Out[24]:

	English	History	Geography	Mathematics	Science	Average
Sam	60.0	65.0	0.0	99.0	89.0	62.6
Jackson	74.0	0.0	92.0	89.0	0.0	51.0
Ahree	85.0	0.0	76.0	85.0	87.0	66.6
Gloria	0.0	83.0	95.0	74.0	0.0	50.4
Isla	0.0	78.0	82.0	0.0	93.0	50.6
Aron	0.0	72.0	75.0	0.0	82.0	45.8
Gray	0.0	61.0	0.0	95.0	78.0	46.8

20. Gloria is planning to purchase a new car for herself. She inquired various sources and was left with 3 different dataset managed by different sources.

- Insurance_Car_data.csv
- Sales_Car_data.csv

Gloria only require Insurance and Sales data before making the purhcase so, merge both the datasets.

In [25]:

```
import pandas as pd
i_car=pd.read_csv('Insurance_Car_data.csv')
i_car.head()
```

Out[25]:

	Manufacturer	Model	Fuel capacity	Fuel efficiency	Price in thousands	Wheelbase
0	Acura	Integra	13.2	28.0	21.50	101.2
1	Acura	TL	17.2	25.0	28.40	108.1
2	Acura	CL	17.2	26.0	NaN	106.9
3	Acura	RL	18.0	22.0	42.00	114.6
4	Audi	A4	16.4	27.0	23.99	102.6

In [26]:

```
s_car=pd.read_csv('Sales_Car_data.csv')
s_car.head()
```

Out[26]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands
0	Acura	Integra	16.919	16.360	02-Feb-14	21.50
1	Acura	TL	39.384	19.875	06-Mar-15	28.40
2	Acura	CL	14.114	18.225	01-Apr-14	NaN
3	Acura	RL	8.588	29.725	03-Oct-15	42.00
4	Audi	A4	20.397	22.255	10-Aug-15	23.99

In [27]:

```
car=pd.merge(s_car,i_car,how='outer')
```

In [28]:

```
car.head()
```

Out[28]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands	Fuel capacity	Fuel efficiency	Wheelb
0	Acura	Integra	16.919	16.360	02-Feb-14	21.50	13.2	28.0	1
1	Acura	TL	39.384	19.875	06-Mar-15	28.40	17.2	25.0	1
2	Acura	CL	14.114	18.225	01-Apr-14	NaN	17.2	26.0	1
3	Acura	RL	8.588	29.725	03-Oct-15	42.00	18.0	22.0	1
4	Audi	A4	20.397	22.255	10-Aug-15	23.99	16.4	27.0	1

21. Remove any null values with mode if data is categorical else with the average.

In [29]:

```
car.isna().any()
```

Out[29]:

```
Manufacturer           False
Model                  False
Sales in thousands     False
4-year resale value     True
Latest Launch          False
Price in thousands      True
Fuel capacity           True
Fuel efficiency         True
Wheelbase               True
dtype: bool
```

In [30]:

```
import numpy as np
for i in car:
    if car[i].dtype==np.float64:

        car[i]=car[i].fillna(car[i].mean())
    else:
        car[i]=car[i].fillna(car[i].mode())
```

In [31]:

```
car.isna().any()
```

Out[31]:

```
Manufacturer           False
Model                  False
Sales in thousands     False
4-year resale value    False
Latest Launch          False
Price in thousands     False
Fuel capacity          False
Fuel efficiency        False
Wheelbase              False
dtype: bool
```

In [32]:

```
car.head()
```

Out[32]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands	Fuel capacity	Fuel efficiency	Wheelb
0	Acura	Integra	16.919	16.360	02-Feb-14	21.500000	13.2	28.0	1
1	Acura	TL	39.384	19.875	06-Mar-15	28.400000	17.2	25.0	1
2	Acura	CL	14.114	18.225	01-Apr-14	27.390755	17.2	26.0	1
3	Acura	RL	8.588	29.725	03-Oct-15	42.000000	18.0	22.0	1
4	Audi	A4	20.397	22.255	10-Aug-15	23.990000	16.4	27.0	1

22. Group by manufacturer and print the average 4-year resale value in descending order.

In [33]:

```
car.groupby('Manufacturer')['4-year resale value'].mean().sort_values(ascending=False)
```

Out[33]:

```
Manufacturer
Porsche          56.475000
Mercedes-Benz    29.648875
Audi             28.270000
BMW              27.624325
Lexus            25.607321
Cadillac         22.329595
Acura            21.046250
Lincoln          20.107658
Infiniti         19.690000
Saab             18.072975
Subaru           18.072975
Jaguar           18.072975
Volvo            18.072975
Oldsmobile       17.073492
Dodge            16.961634
Toyota           16.814775
Honda            15.557000
Chrysler         15.406139
Jeep             15.353333
Volkswagen       14.966329
Buick            14.941250
Nissan           14.886564
Pontiac          14.532163
Mitsubishi       14.262143
Mercury          13.970000
Chevrolet        13.768664
Ford             13.424816
Saturn           13.345190
Plymouth         11.911994
Hyundai           7.531667
Name: 4-year resale value, dtype: float64
```

23. Find the best fuel efficient Model and Manufacturer.

In [34]:

```
car[car['Fuel efficiency']==car['Fuel efficiency'].max()][['Manufacturer','Model']]
```

Out[34]:

	Manufacturer	Model
26	Chevrolet	Metro

24. Print details of models made by Audi using group by method.

Hint: String should be cleaned before operation.

In [35]:

```
car.Manufacturer=car.Manufacturer.str.strip()
group=car.groupby('Manufacturer')
group.get_group('Audi')
```

Out[35]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands	Fuel capacity	Fuel efficiency	Wheelb
4	Audi	A4	20.397	22.255	10-Aug-15	23.99	16.4	27.0	1(
5	Audi	A6	18.780	23.555	08-Sep-15	33.95	18.5	22.0	1(
6	Audi	A8	1.380	39.000	27-Feb-14	62.00	23.7	21.0	1'

25. Print the data of car where sales in thousands is between 200 to 300.

In [36]:

```
car[car['Sales in thousands'].between(200,300,inclusive=True)]
```

Out[36]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands	Fuel capacity	Fuel efficiency	Wh
40	Dodge	Ram Pickup	227.061	15.060	03-Jun-14	19.460	26.0	17.0	
49	Ford	Taurus	245.815	10.055	20-Dec-15	17.885	16.0	24.0	
52	Ford	Explorer	276.747	16.640	25-Apr-14	31.930	21.0	19.0	
55	Ford	Ranger	220.650	7.850	14-Jan-14	12.050	20.0	23.0	
58	Honda	Accord	230.902	13.210	20-May-14	15.350	17.1	27.0	
137	Toyota	Camry	247.994	13.245	02-Oct-15	17.518	18.5	27.0	

26. Random sample 20 rows from the dataset

In [37]:

```
car=car.sample(n=20)
car
```

Out[37]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Latest Launch	Price in thousands	Fuel capacity	Fuel efficiency
140	Toyota	Tacoma	84.087	9.575000	08-Jan-15	11.528000	15.1	23.0
75	Lexus	RX300	51.238	18.072975	01-Apr-14	34.605000	17.2	21.0
6	Audi	A8	1.380	39.000000	27-Feb-14	62.000000	23.7	21.0
155	Volvo	C70	3.493	18.072975	26-Apr-15	45.500000	18.5	23.0
152	Volvo	V40	3.545	18.072975	21-Sep-15	24.400000	15.8	25.0
2	Acura	CL	14.114	18.225000	01-Apr-14	27.390755	17.2	26.0
61	Honda	Odyssey	76.029	19.490000	02-Aug-14	26.000000	20.0	23.0
89	Mercury	Grand Marquis	81.174	14.875000	24-Jul-14	22.605000	19.0	21.0
133	Saturn	LS	49.989	18.072975	12-Apr-14	15.010000	13.1	28.0
39	Dodge	Viper	0.916	58.470000	08-Jul-15	69.725000	19.0	16.0
16	Cadillac	Eldorado	6.536	25.725000	27-Nov-15	39.665000	19.0	22.0
87	Mercury	Cougar	26.529	13.890000	23-Feb-14	16.540000	16.0	30.0
111	Oldsmobile	Aurora	14.690	19.890000	18-Feb-15	36.229000	18.5	22.0
110	Oldsmobile	Alero	80.255	18.072975	20-Oct-09	18.270000	15.0	27.0
51	Ford	Crown Victoria	63.403	14.210000	26-Sep-15	22.195000	19.0	21.0
69	Jeep	Grand Cherokee	157.040	18.810000	12-Oct-15	26.895000	20.5	19.0
35	Dodge	Neon	76.034	7.750000	12-Dec-15	12.640000	12.5	29.0
37	Dodge	Stratus	71.186	10.185000	31-Oct-15	20.230000	16.0	24.0
70	Lexus	ES300	24.072	26.975000	07-Sep-14	31.505000	18.5	23.0
88	Mercury	Sable	67.956	11.030000	22-Sep-14	19.035000	16.0	24.0

27. Given below are two DataFrame perform left and right joins respoectively.

```
df1 = pd.DataFrame({'Name': ['Sam', 'Aron', 'Smith','Lee','Gloria'] ,
                    'Age': [18, 27, 18, 27, 17] ,
                    'Hobby': ['music','binge-watching','dancing','music','musi
c']})


df2 = pd.DataFrame({'Name': ['Sam', 'Aron', 'Smith','Janna','Cohle'] ,
                    'Age': [23, 21, 21, 25, 18],
                    'Hobby': ['painting','dancing','dancing','gaming','binge-wat
ching']})
```

In [38]:

```
import pandas as pd
df1 = pd.DataFrame({'Name': ['Sam', 'Aron', 'Smith','Lee','Gloria'] ,
                'Age': [18, 27, 18, 27, 17] ,
                'Hobby': ['music','binge-watching','dancing','music','music']})
df2 = pd.DataFrame({'Name': ['Sam', 'Aron', 'Smith','Janna','Cohle'] ,
                    'Age': [23, 21, 21, 25, 18],
                    'Hobby': ['painting','dancing','dancing','gaming','binge-watching'
]})
print(pd.merge(df1,df2,how='left',left_on=['Name'], right_on=['Name']))
print()
print(pd.merge(df1,df2,how='right',left_on=['Name'], right_on=['Name']))
```

```
     Name  Age_x         Hobby_x  Age_y   Hobby_y
0     Sam     18           music   23.0  painting
1    Aron     27  binge-watching   21.0   dancing
2   Smith     18         dancing   21.0   dancing
3     Lee     27           music    NaN       NaN
4  Gloria     17           music    NaN       NaN

    Name  Age_x         Hobby_x  Age_y         Hobby_y
0    Sam   18.0           music     23        painting
1   Aron   27.0  binge-watching     21         dancing
2  Smith   18.0         dancing     21         dancing
3  Janna    NaN             NaN     25          gaming
4  Cohle    NaN             NaN     18  binge-watching
```

28. Create pandas dataframe having keys and ltable and rtable as below -

```
'key': ['One', 'Two'], 'ltable': [1, 2]
'key': ['One', 'Two'], 'rtable': [4, 5]
```

Merge both the tables based of key

In [39]:

```
import pandas pd
sj=pd.DataFrame({'key': ['One', 'Two'], 'ltable': [1, 2]})
sk=pd.DataFrame({'key': ['One', 'Two'], 'rtable': [4, 5]})
c=pd.concat([sj,sk],axis=1,keys=['One','Two'])
c
```

Out[39]:

	One		Two	
	key	ltable	key	rtable
0	One	1	One	4
1	Two	2	Two	5

29. Create a Python DataFrame from the below data:

```
Food        PortionSize Calories    K joules
Blueberries   100g      30           128
Cranberries   100g      15           65
Cherries         120g       39           168
Coconut          110g       351         1446
```

Rearrange columns as : Food, Calories, K joules, PortionSize

In [40]:

```
import pandas as pd
fruits=pd.DataFrame({'Food':['Blueberries','Cranberries','Cherrries','Coconut'],
                     'PortionSize':['100g','100g','120g','110g'],
                     'Calories': [30,15,39,351],
                     'K joules':[128,65,168,1446]})
fruits
```

Out[40]:

	Food	PortionSize	Calories	K joules
0	Blueberries	100g	30	128
1	Cranberries	100g	15	65
2	Cherrries	120g	39	168
3	Coconut	110g	351	1446

In [41]:

```
fruits.reindex(columns=['Food','Calories','K joules','PortionSize'])
```

Out[41]:

	Food	Calories	K joules	PortionSize
0	Blueberries	30	128	100g
1	Cranberries	15	65	100g
2	Cherrries	39	168	120g
3	Coconut	351	1446	110g

30. Calculate the percentile rank of the Test_score column and add a new column specifying the percentiles.

```
my_dict={'Name':['Gloria','Aron','Janna','Mia','Jackson','Sylphia','Jim'],
         'Test_score':[52,73,65,84,62,66,93]}
```

In [42]:

```
import pandas as pd
my_dict={'Name':['Gloria','Aron','Janna','Mia','Jackson','Sylphia','Jim'],
       'Test_score':[52,73,65,84,62,66,93]}
df=pd.DataFrame(my_dict)
df['Percentile']=df.Test_score.rank(pct=True)
df
```

Out[42]:

	Name	Test_score	Percentile
0	Gloria	52	0.142857
1	Aron	73	0.714286
2	Janna	65	0.428571
3	Mia	84	0.857143
4	Jackson	62	0.285714
5	Sylphia	66	0.571429
6	Jim	93	1.000000

31. Create a DataFrame from the given data.

```
my_dict = {
'Name':['Gloria','Aron','Janna','Mia','Jackson','Sylphia','Jim'],
   'Test_score':[52,73,65,84,62,66,93],
   'Test2_score':[73,58,66,85,42,87,81],
 'Gender':["F","M","F","F","M","F","M"]
   }
```

Rearrange the dataset by Names in alphabetical order and reset index.

In [43]:

```
import pandas as pd
my_dict = { 'Name':['Gloria','Aron','Janna','Mia','Jackson','Sylphia','Jim'],
            'Test_score':[52,73,65,84,62,66,93],
            'Test2_score':[73,58,66,85,42,87,81],
            'Gender':["F","M","F","F","M","F","M"] }
df=pd.DataFrame(my_dict)
df
```

Out[43]:

	Name	Test_score	Test2_score	Gender
0	Gloria	52	73	F
1	Aron	73	58	M
2	Janna	65	66	F
3	Mia	84	85	F
4	Jackson	62	42	M
5	Sylphia	66	87	F
6	Jim	93	81	M

In [44]:

```
df=df.sort_values('Name',axis=0)
df
```

Out[44]:

	Name	Test_score	Test2_score	Gender
1	Aron	73	58	M
0	Gloria	52	73	F
4	Jackson	62	42	M
2	Janna	65	66	F
6	Jim	93	81	M
3	Mia	84	85	F
5	Sylphia	66	87	F

In [45]:

```
df=df.reset_index(drop=True)
df
```

Out[45]:

	Name	Test_score	Test2_score	Gender
0	Aron	73	58	M
1	Gloria	52	73	F
2	Jackson	62	42	M
3	Janna	65	66	F
4	Jim	93	81	M
5	Mia	84	85	F
6	Sylphia	66	87	F

32. Import the laliga player stats dataset(laliga_player_stats.csv). Group the dataset by Position and print the average timing of each position.

In [46]:

```
import pandas as pd
stats=pd.read_csv('laliga_player_stats.csv')
stats.head()
```

Out[46]:

	Team	Position	Name	Minutes played	Games played	Yellow Cards	Red Cards	Goals scored	Penalties scored	Own goals
0	Athletic Club	Goalkeeper	Hodei Oleaga	0.000	0	0	0	0	0	0
1	Athletic Club	Goalkeeper	A. Remiro	0.000	0	0	0	0	0	0
2	Athletic Club	Goalkeeper	Herrerín	2.790	31	1	0	0	0	0
3	Athletic Club	Goalkeeper	Unai Simón	630.000	7	2	0	0	0	0
4	Athletic Club	Defender	Núñez	1.063	12	4	0	0	0	0

In [47]:

```
stats.groupby('Position')['Minutes played'].mean()
```

Out[47]:

```
Position
Defender      121.626973
Forward       181.430862
Goalkeeper    113.551719
Midfielder    108.873855
Name: Minutes played, dtype: float64
```

33. Print the name and position of the player who the scored most number of goals.

In [48]:

```
stats[stats['Goals scored']==max(stats['Goals scored'])][['Name','Position']]
```

Out[48]:

	Name	Position
138	Messi	Forward

34. Print the Team name which scored maximum number of goals

In [49]:

```
stats.groupby('Team')['Goals scored'].sum().sort_values(ascending=False).index[0]
```

Out[49]:

```
'FC Barcelona'
```

35. Find the Team with the least number of Yellow cards

In [50]:

```
stats.groupby('Team')['Yellow Cards'].sum().sort_values().index[0]
```

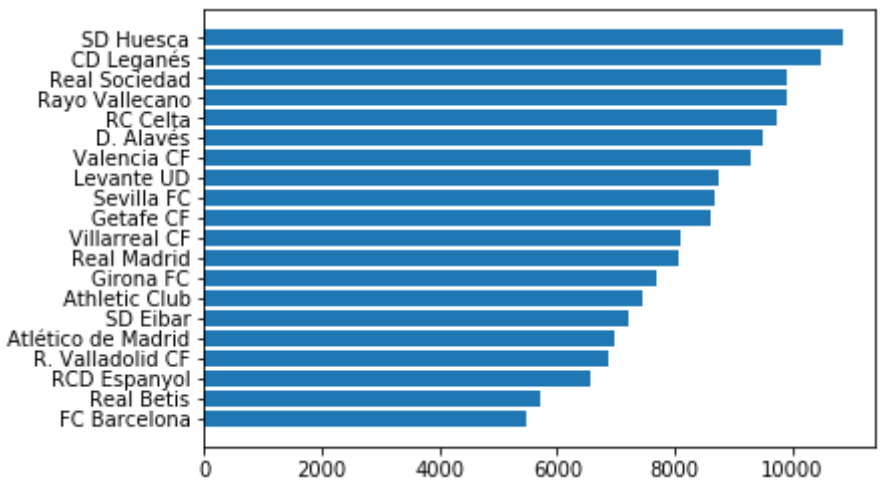
Out[50]:

```
'RC Celta'
```

36. Plot the Passes made by each team on a barplot after sorting them.

In [51]:

```
passes=stats.groupby('Team').Passes.sum().sort_values()
plt.barh(passes.index,passes)
plt.show()
```



1. Import the Manufacturer_Car_data.csv and remove any null values. Plot Curb weight on simple plot.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
m_car=pd.read_csv('Manufacturer_Car_data.csv')
m_car.head()
```

Out[1]:

	Manufacturer	Model	Width	Length	Curb weight	Fuel capacity	Fuel efficiency	Horsepower
0	Acura	Integra	67.3	172.4	2.639	13.2	28.0	140.0
1	Acura	TL	70.3	192.9	3.517	17.2	25.0	225.0
2	Acura	CL	70.6	192.0	3.470	17.2	26.0	225.0
3	Acura	RL	71.4	196.6	3.850	18.0	22.0	210.0
4	Audi	A4	68.2	178.0	2.998	16.4	27.0	150.0

In [2]:

```
m_car.isna().any()
```

Out[2]:

```
Manufacturer       False
Model              False
Width               True
Length              True
Curb weight         True
Fuel capacity       True
Fuel efficiency     True
Horsepower          True
dtype: bool
```

In [3]:

```
m_car.shape
```

Out[3]:

```
(157, 8)
```

In [4]:

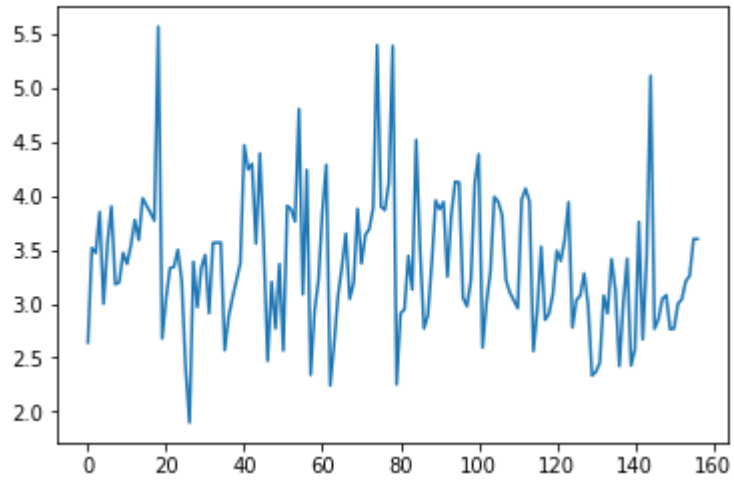
```
m_car=m_car.dropna()
```

In [5]:

```
import matplotlib.pyplot as plt
```

In [6]:

```
plt.plot(m_car['Curb weight'])
plt.show()
```



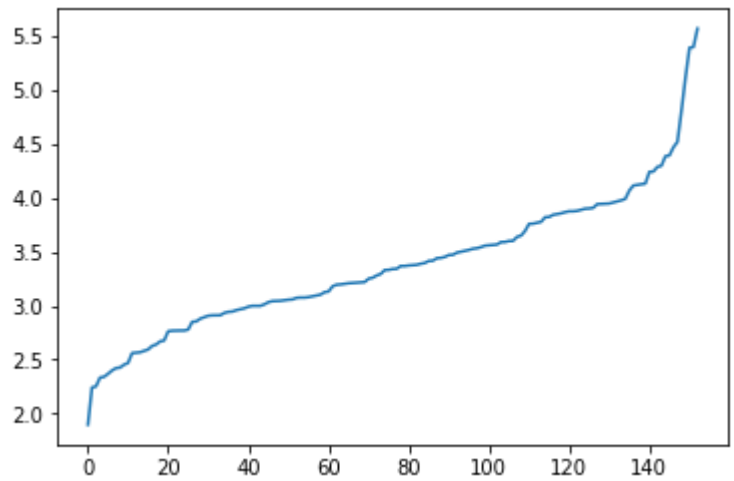
2. Sort the dataset by Curb weight and plot the simple plot again.

In [7]:

```
m_car=m_car.sort_values('Curb weight')
m_car=m_car.reset_index(drop=True)
plt.plot(m_car['Curb weight'])
```

Out[7]:

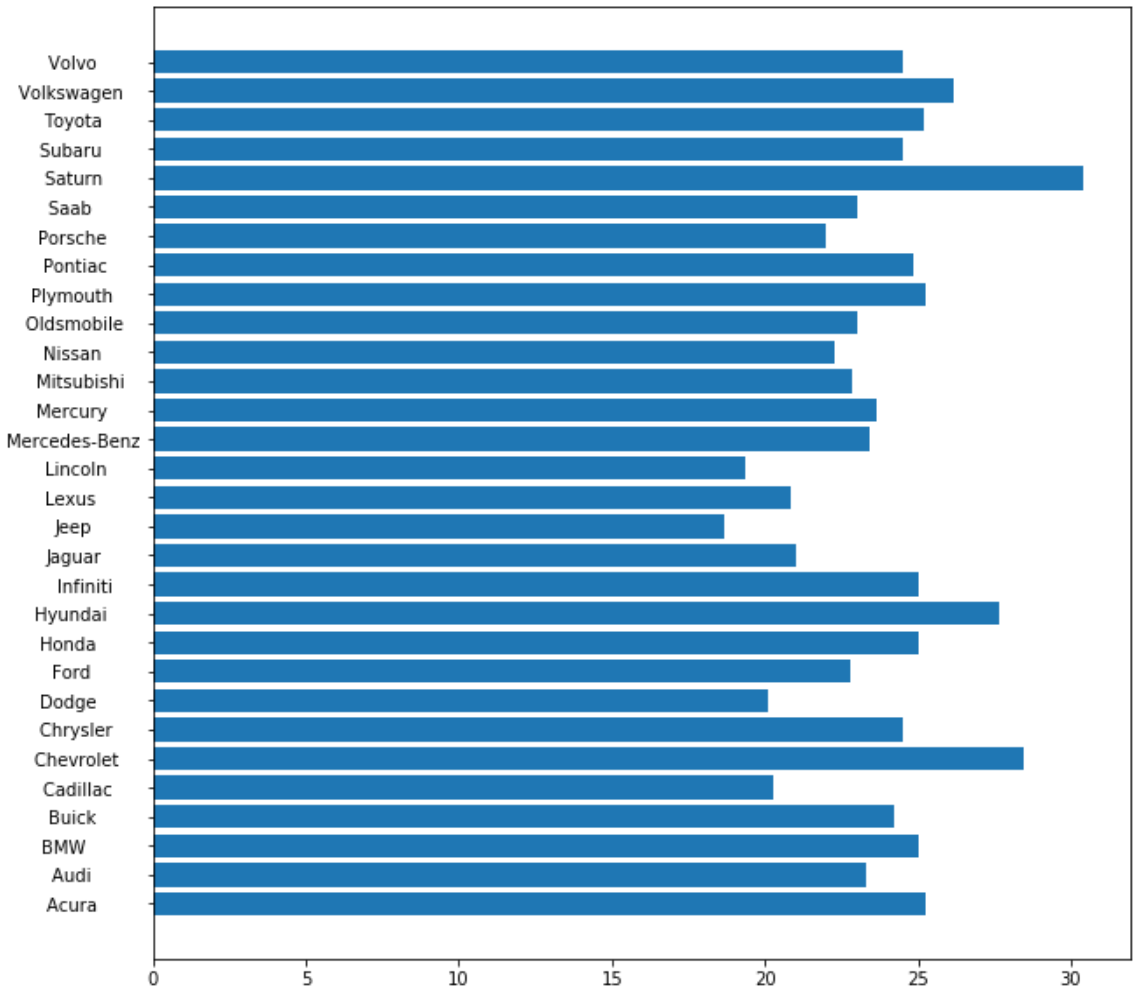
```
[<matplotlib.lines.Line2D at 0x23db77310f0>]
```



3. Group by the Manufacturer and use barplot to plot fuel efficiency and fuel capacity of each Manufacturer.

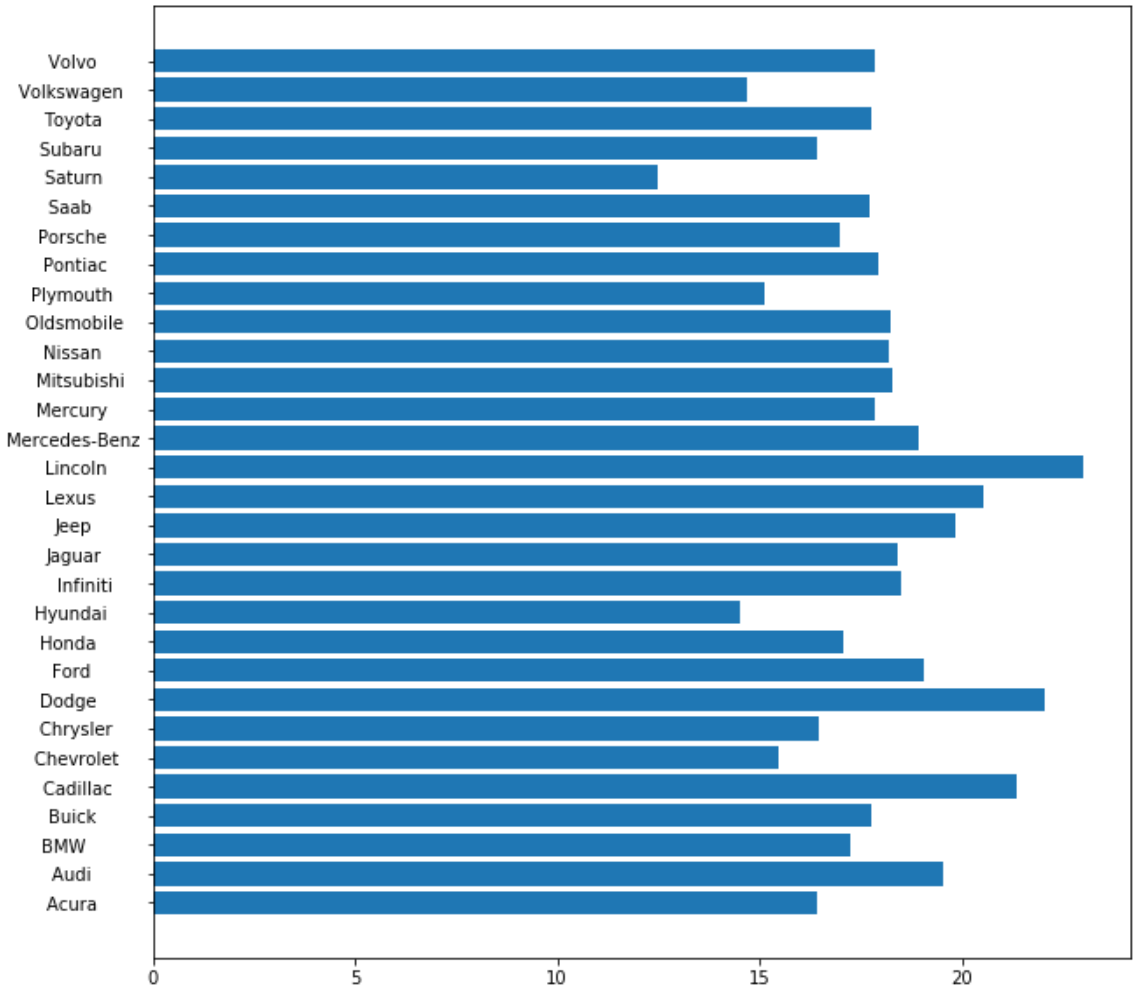
In [8]:

```
group=m_car.groupby('Manufacturer')
fuel_eff=[]
fuel_cap=[]
grp=[]
for gp,data in group:
    grp.append(gp)
    fuel_eff.append(data['Fuel efficiency'].mean())
    fuel_cap.append(data['Fuel capacity'].mean())
plt.figure(figsize=(10,10))
plt.barh(grp,fuel_eff)
plt.show()
```



In [9]:

```
plt.figure(figsize=(10,10))
plt.barh(grp,fuel_cap)
plt.show()
```



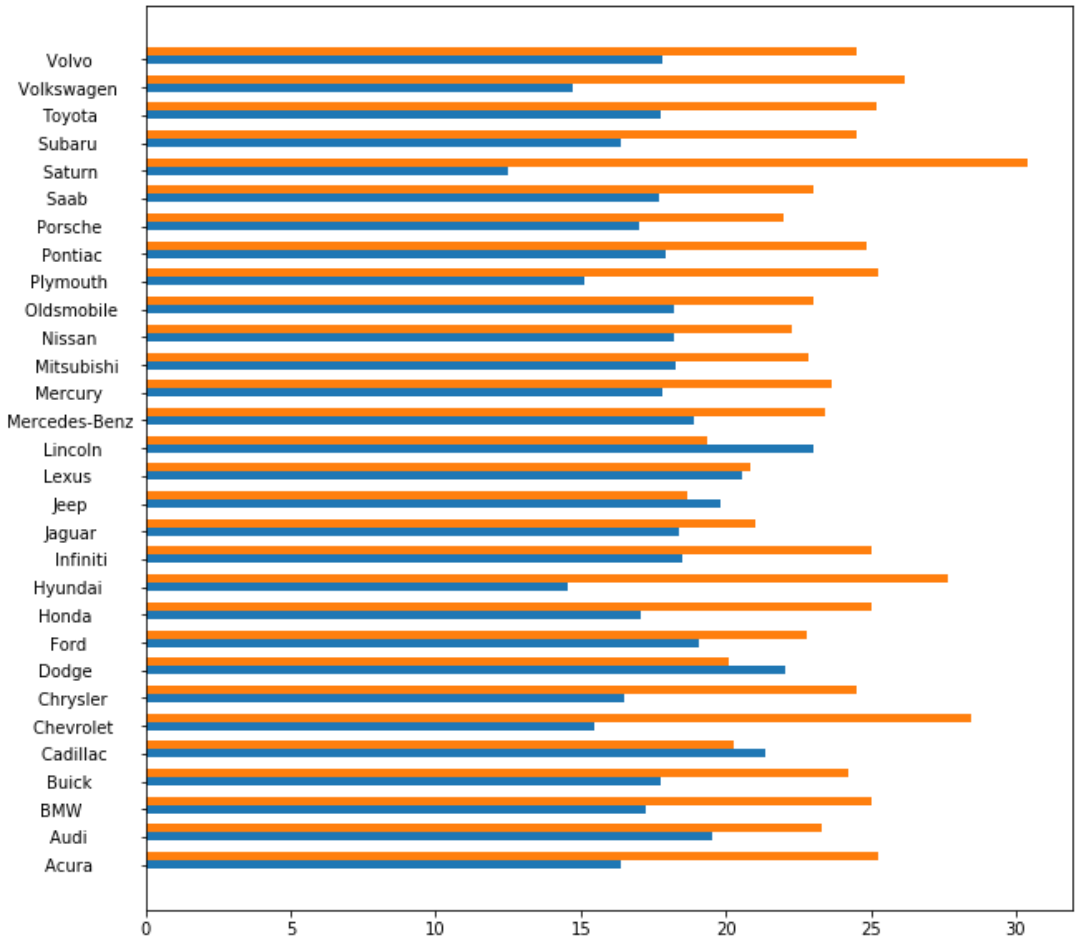
4. Using grouped bar chart plot the same in single horizontal barplot.

In [10]:

```
grp_height=0.3
plt.figure(figsize=(10,10))
plt.barh(np.arange(len(grp))-grp_height,fuel_cap,height=grp_height,tick_label=grp)
plt.barh(np.arange(len(grp)),fuel_eff,height=grp_height)
```

Out[10]:

```
<BarContainer object of 30 artists>
```



5. Import the supermarket_sales.csv and simple bar chart average gross income of all cities.

In [11]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [12]:

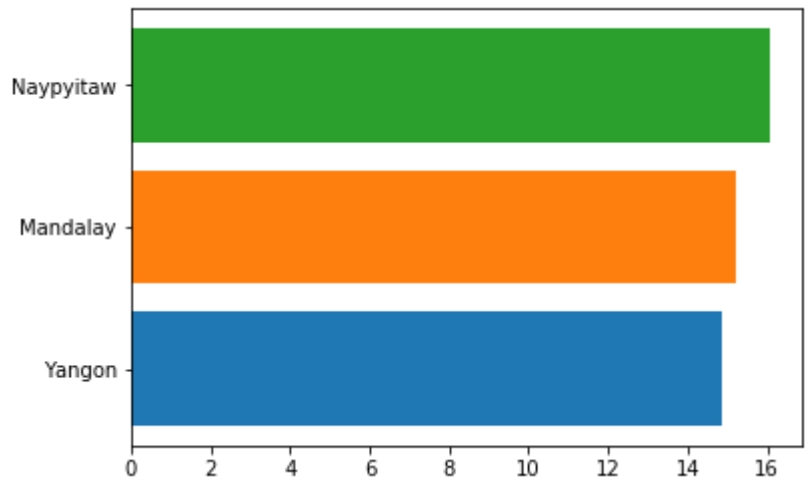
```
sales=pd.read_csv('supermarket_sales.csv')
sales.head()
```

Out[12]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634

In [13]:

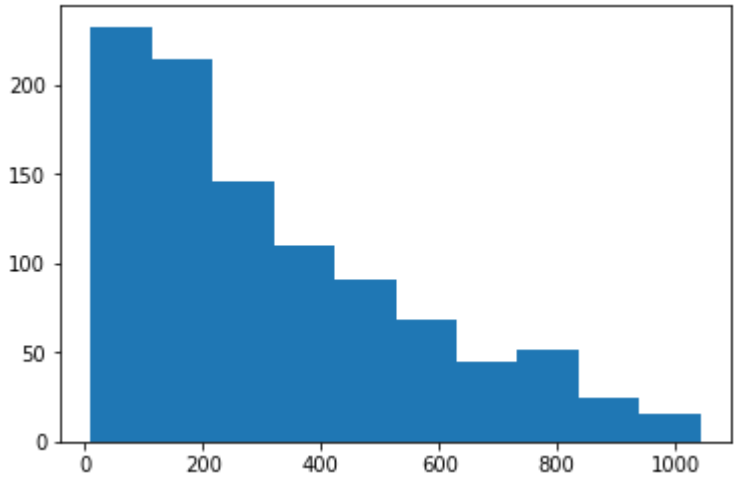
```
cities=set(sales.City)
for city in cities:
    plt.barh(city,sales[sales.City==city]['gross income'].mean())
```



6. Create a histogram for Total purchase

In [14]:

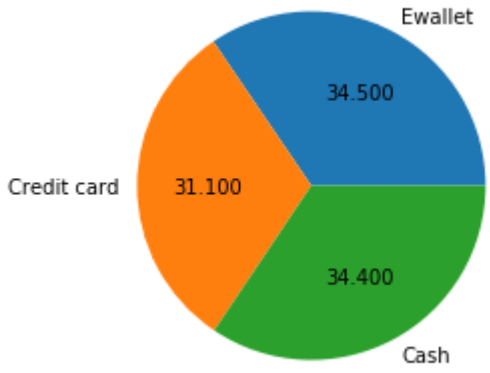
```
plt.hist(sales.Total)
plt.show()
```



7. Create a pie chart for payment to represent which mode of payment is used most.

In [15]:

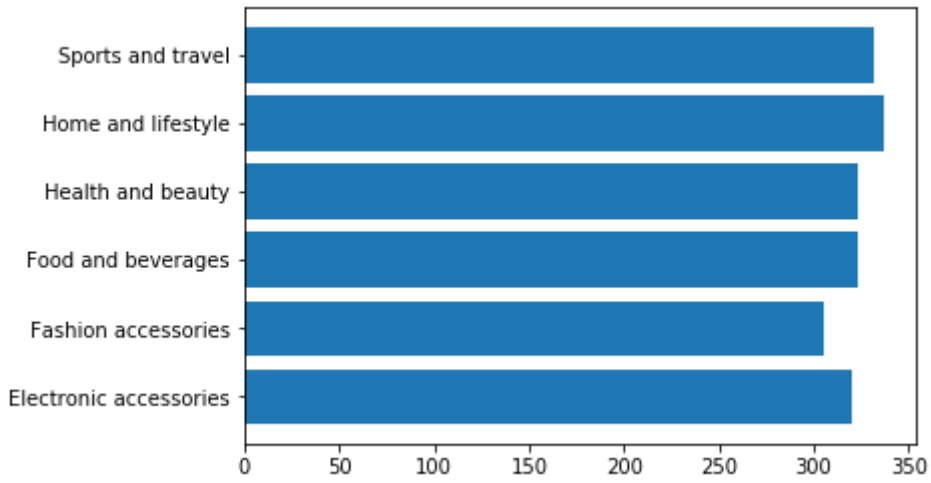
```
mop=list(set(sales.Payment))
val=[]
for mode in mop:
    val.append(sales[sales.Payment==mode].Payment.count())
plt.pie(val,labels=mop,autopct='%0.3f')
plt.show()
```



8. Group by Product Line and create a horizontal bar chart to represent mean total purchase in particular Product line.

In [16]:

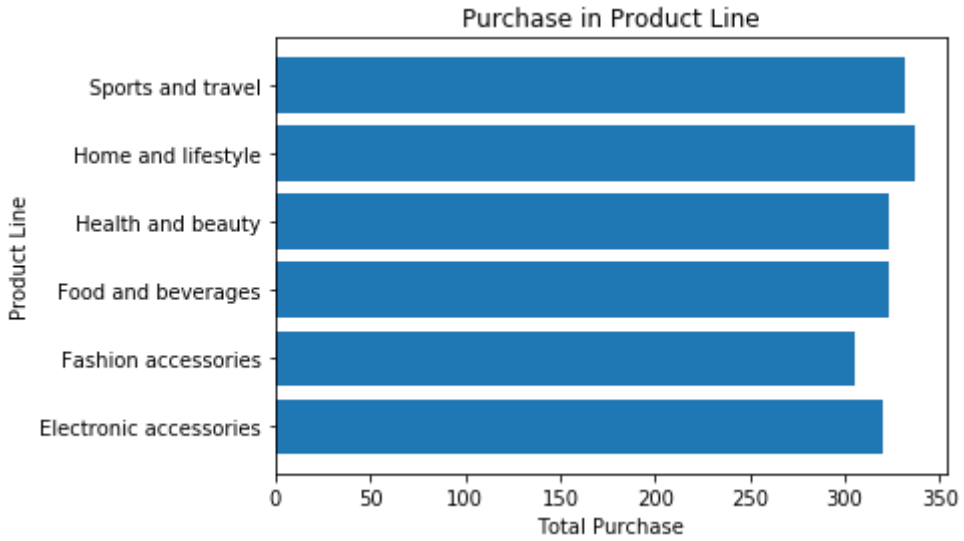
```
my_dict=dict()
for grp,data in sales.groupby('Product line'):
    my_dict[grp]=data.Total.mean()
plt.barh(list(my_dict.keys()),list(my_dict.values()))
plt.show()
```



9. Label X and Y axes alongwith Title of the graph and save it as 'graph.png'.

In [17]:

```
plt.barh(list(my_dict.keys()),list(my_dict.values()))
plt.title('Purchase in Product Line')
plt.xlabel('Total Purchase')
plt.ylabel('Product Line',rotation=90)
plt.show()
plt.savefig('graph.png')
```




```
<Figure size 432x288 with 0 Axes>
```

10. Create a line plot to represent the following equations:

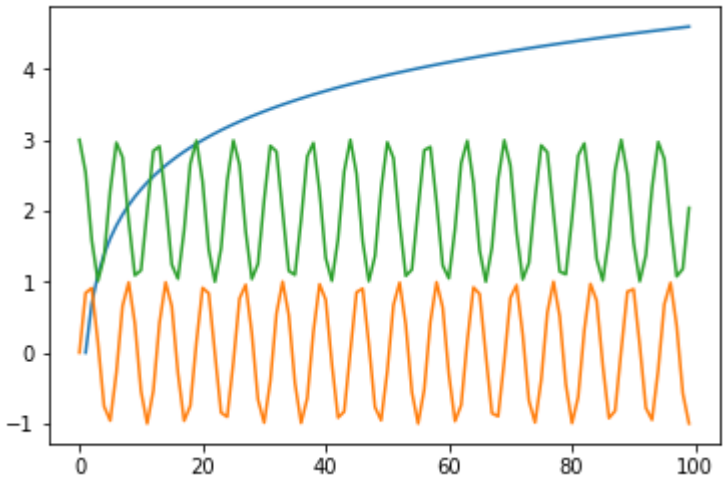
- y=cos(x)+2
- y=log(x)
- y=sin(x)

Use numpy to generate the data.

In [18]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
x=np.arange(100)
y=np.arange(100)
e1=np.log(x)
e2=np.sin(x)
e3=np.cos(x)+2
plt.plot(y,e1)
plt.plot(y,e2)
plt.plot(y,e3)
plt.show()
```

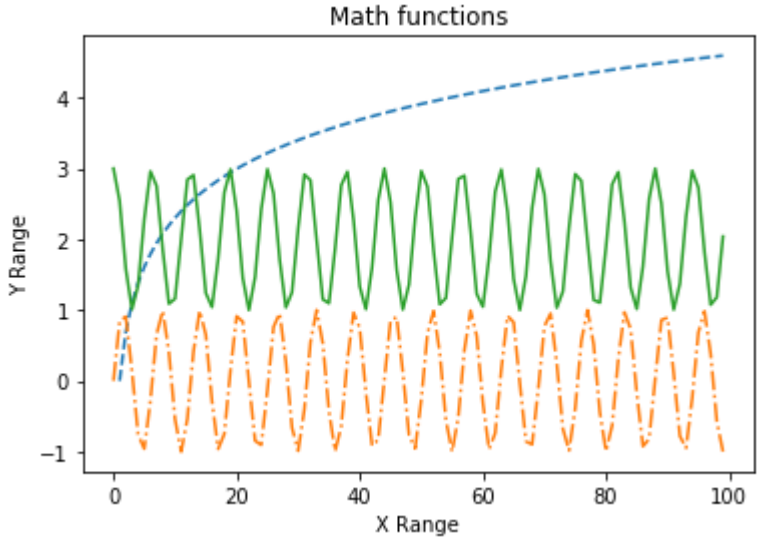
```
C:\Users\Sumeet_Kumar_Jain\Anaconda3\lib\site-packages\ipykernel_launcher.
py:6: RuntimeWarning: divide by zero encountered in log
```



11. Add labels and title to the graph alongwith use different line styles.

In [19]:

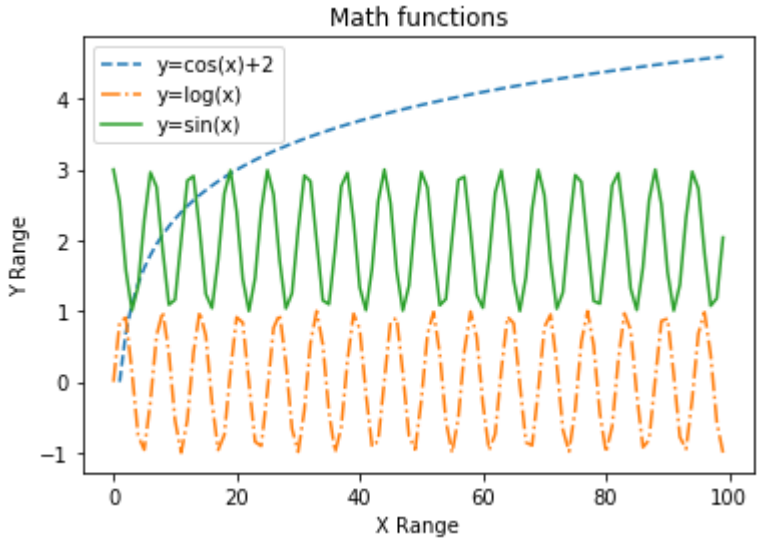
```
plt.plot(y,e1,'--')
plt.plot(y,e2,'-.')
plt.plot(y,e3,'-')
plt.xlabel('X Range')
plt.ylabel('Y Range')
plt.title('Math functions')
plt.show()
```



12. Create legends for the graph.

In [20]:

```
plt.plot(y,e1,'--')
plt.plot(y,e2,'-.')
plt.plot(y,e3,'-')
plt.xlabel('X Range')
plt.ylabel('Y Range')
plt.title('Math functions')
plt.legend(['y=cos(x)+2','y=log(x)','y=sin(x)'])
plt.show()
```



13. Import the dataset Mall_customers.csv and plot a histogram on Spending Score with 10 bins.

In [21]:

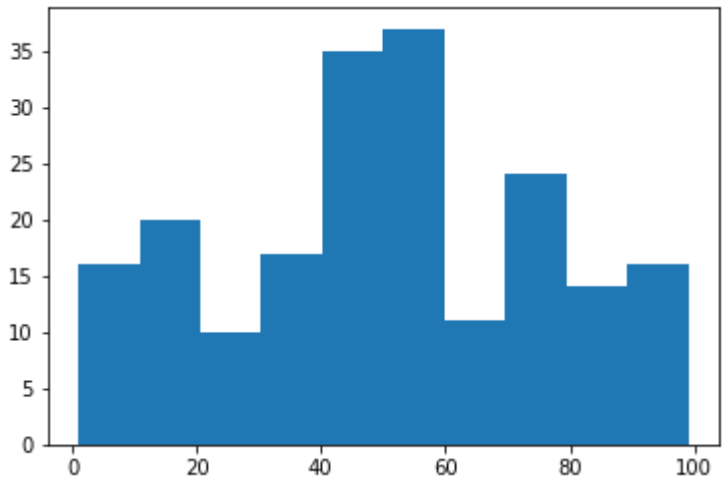
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
mall=pd.read_csv('Mall_Customers.csv')
mall.head()
```

Out[21]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [22]:

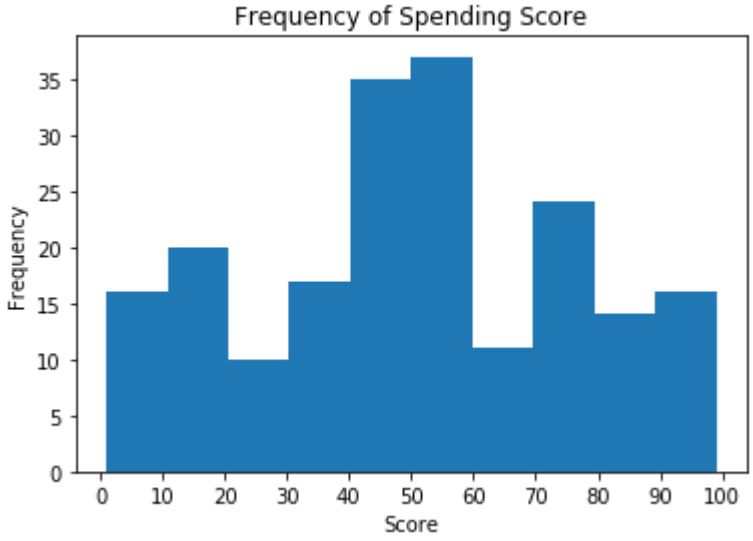
```
plt.hist(mall['Spending Score (1-100)'],bins=10)
plt.show()
```



14. Add labels and title alongwith the proper xticks. What does the histogram tell you?

In [23]:

```
# There are maximum number of customers who have a shopping score between 50-60
plt.hist(mall['Spending Score (1-100)'],bins=10)
plt.xlabel('Score')
plt.ylabel('Frequency')
plt.title('Frequency of Spending Score')
plt.xticks(np.arange(0,101,10))
plt.show()
```



15. Sylphia has a dataset of various cereals sold in the supermarket. Syliphia wants to create a representation that can easily represents the qualities of cereals. Import the cereal dataset and plot ratings of different types of MFR.

In [24]:

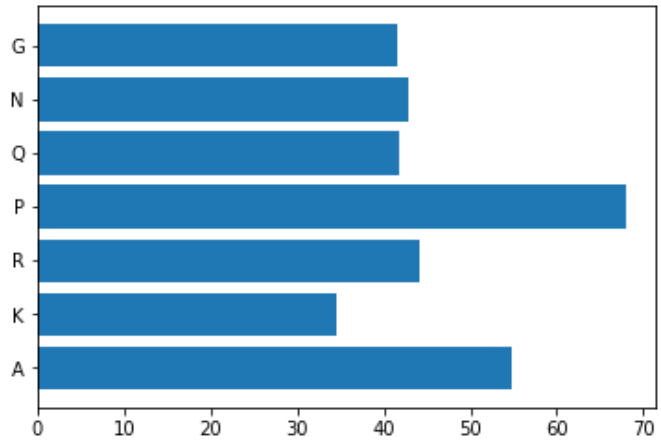
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
cereal=pd.read_csv('cereal.csv')
cereal.head()
```

Out[24]:

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins
0	100% Bran	N	C	70	4	1	130	10.0	5.0	6	280.0	25
1	100% Natural Bran	Q	C	120	3	5	15	2.0	8.0	8	135.0	0
2	All-Bran	K	C	70	4	1	260	9.0	7.0	5	320.0	25
3	All-Bran with Extra Fiber	K	C	50	4	0	140	14.0	8.0	0	330.0	25
4	Almond Delight	R	C	110	2	2	200	1.0	14.0	8	NaN	25

In [25]:

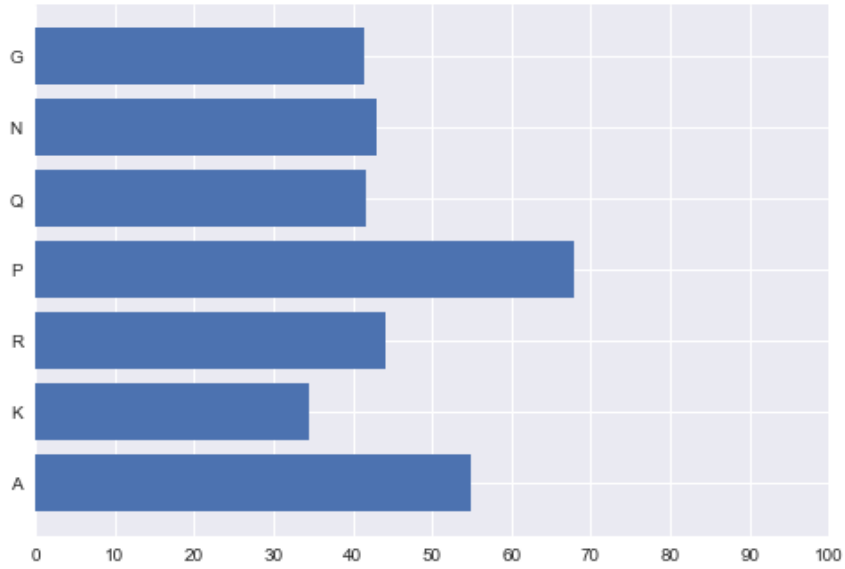
```
plt.barh(list(set(cereal.mfr)),cereal.groupby('mfr').rating.mean())
plt.show()
```



16. Use xticks range form 0-100 and change style of the graph to seaborn.

In [26]:

```
plt.style.use('seaborn')
plt.barh(list(set(cereal.mfr)),cereal.groupby('mfr').rating.mean())
plt.xticks(np.arange(0,101,10))
plt.show()
```



17. Fill the missin values with mode if data is categorical else with mean and using subplots line plot.

[calories, sodium, potass,rating]

In [27]:

```
cereal.isna().any()
```

Out[27]:

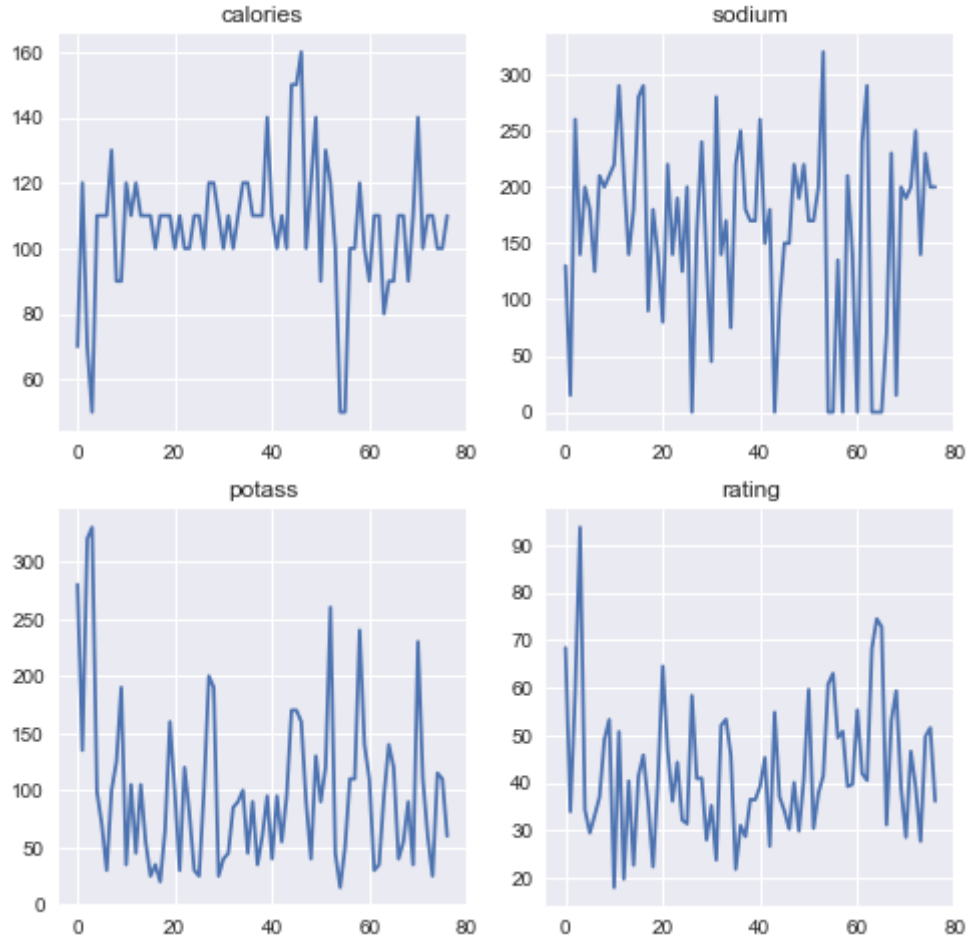
```
name        False
mfr         False
type        False
calories    False
protein     False
fat         False
sodium      False
fiber       False
carbo        True
sugars      False
potass       True
vitamins    False
shelf       False
weight      False
cups        False
rating      False
dtype: bool
```

In [28]:

```
cereal.carbo=cereal.carbo.fillna(cereal.carbo.mean())
cereal.potass=cereal.potass.fillna(cereal.potass.mean())
```

In [29]:

```
index=1 #index starts from 1 for subplot
plt.figure(figsize=(8,8))
for col in ['calories', 'sodium', 'potass','rating']:
    plt.subplot(2,2,index)
    plt.plot(cereal[col],)
    plt.title(col)
    index=index+1
```

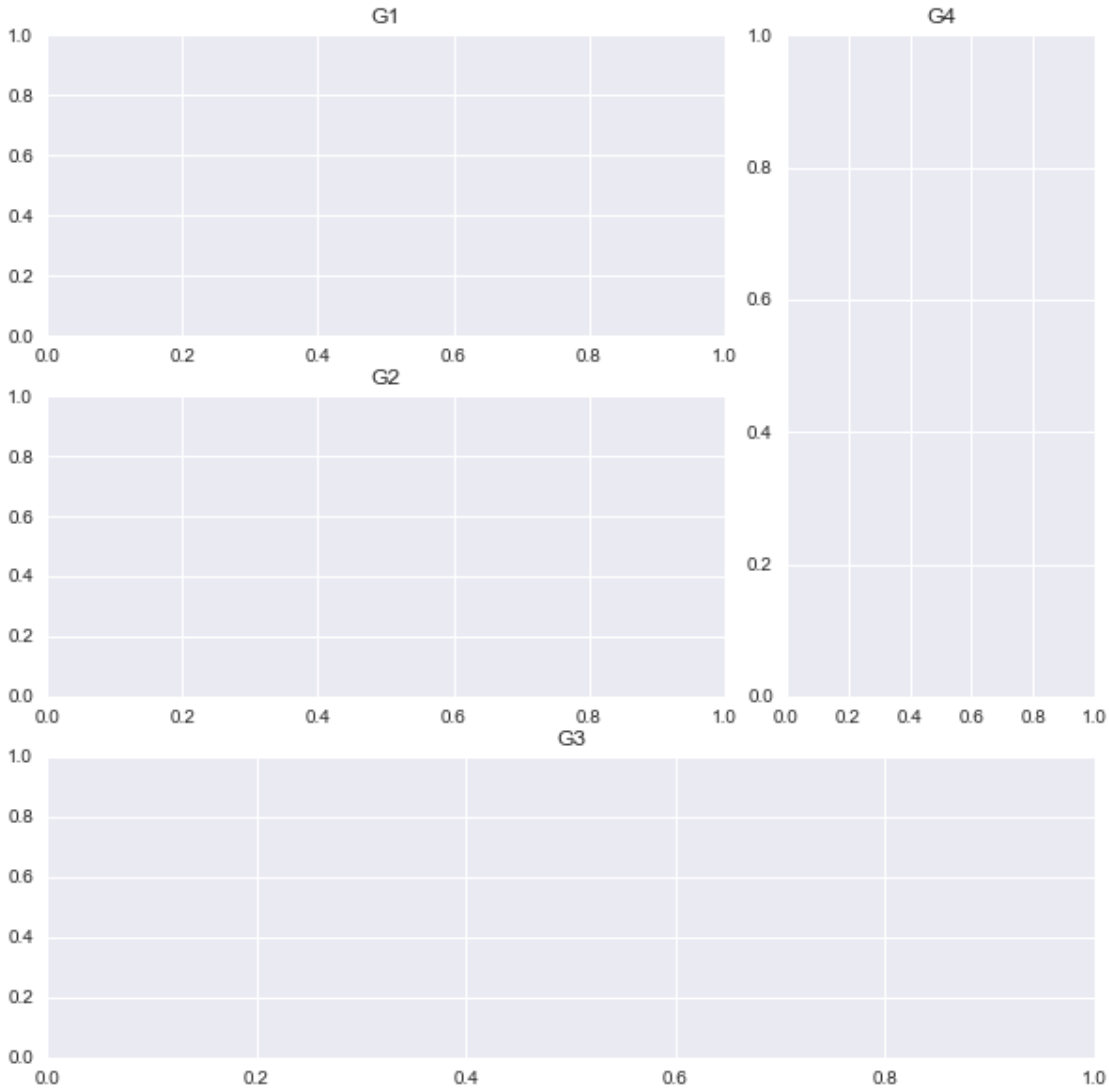


18. Create a Gridplot of (3,3) where the dimensions for subplot will be:

- [0,:2]
- [1,:2]
- [2,:]
- [:2,2]

In [30]:

```
my_grid=plt.GridSpec(3,3)
my_fig=plt.figure(figsize=(10,10))
my_fig.add_subplot(my_grid[0,:2])
plt.title('G1')
my_fig.add_subplot(my_grid[1,:2])
plt.title('G2')
my_fig.add_subplot(my_grid[2,:])
plt.title('G3')
my_fig.add_subplot(my_grid[:2,2])
plt.title('G4')
plt.show()
```

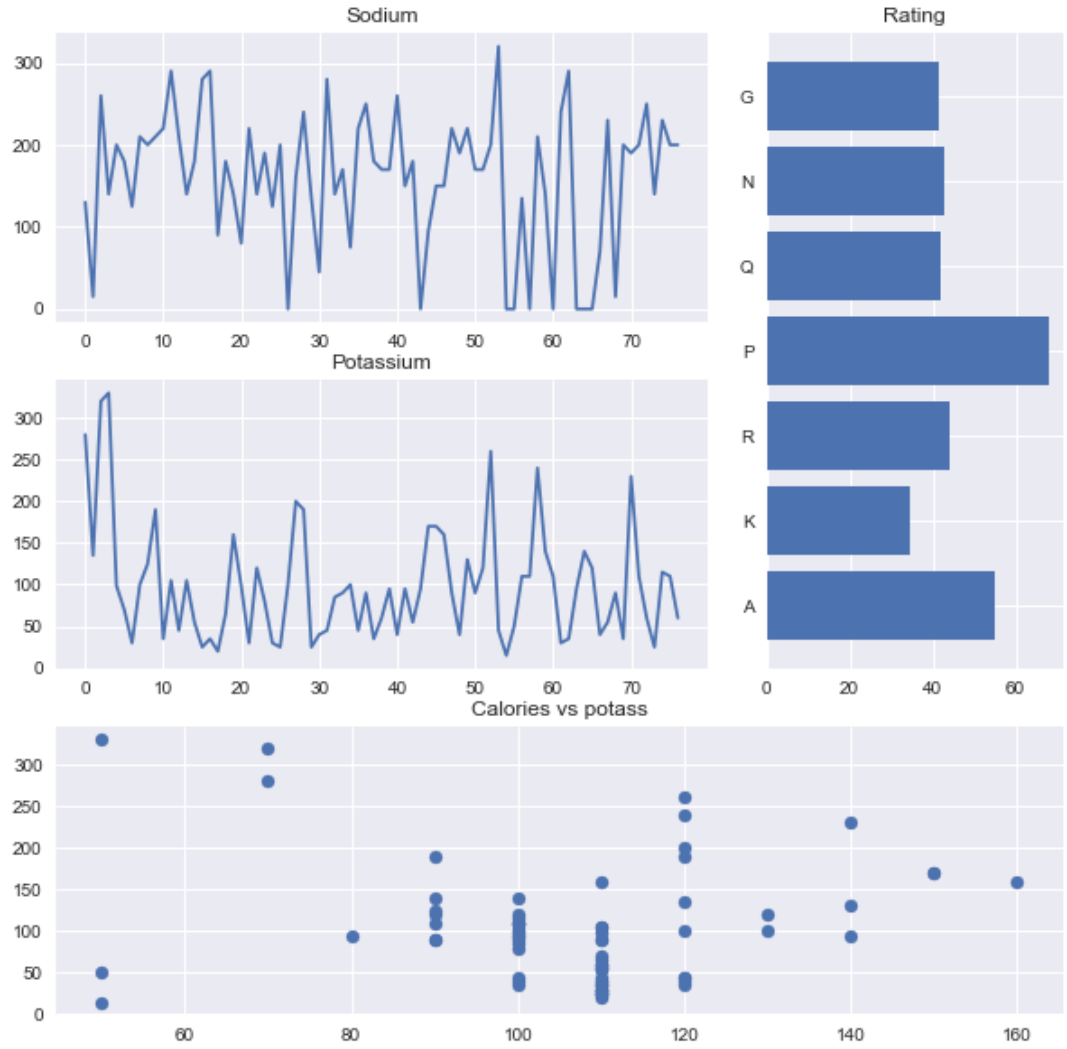


19. Plot the following graphs:

- Scatter plot calories vs potassium
- bar plot for ratings grouped by mfr
- line plots for sodium and calories

In [31]:

```
my_grid=plt.GridSpec(3,3)
my_fig=plt.figure(figsize=(10,10))
my_fig.add_subplot(my_grid[0,:2])
plt.title('Sodium')
plt.plot(cereal.sodium)
my_fig.add_subplot(my_grid[1,:2])
plt.title('Potassium')
plt.plot(cereal.potass)
my_fig.add_subplot(my_grid[2,:])
plt.title('Calories vs potass')
plt.scatter(cereal.calories,cereal.potass)
my_fig.add_subplot(my_grid[:2,2])
plt.title('Rating')
plt.barh(list(set(cereal.mfr)),cereal.groupby('mfr').rating.mean())
plt.show()
```



20. Import the rainfall.csv dataset and create a jointplot for JUN and JAN.

In [32]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
rain=pd.read_csv('rainfall.csv')
rain.head()
```

Out[32]:

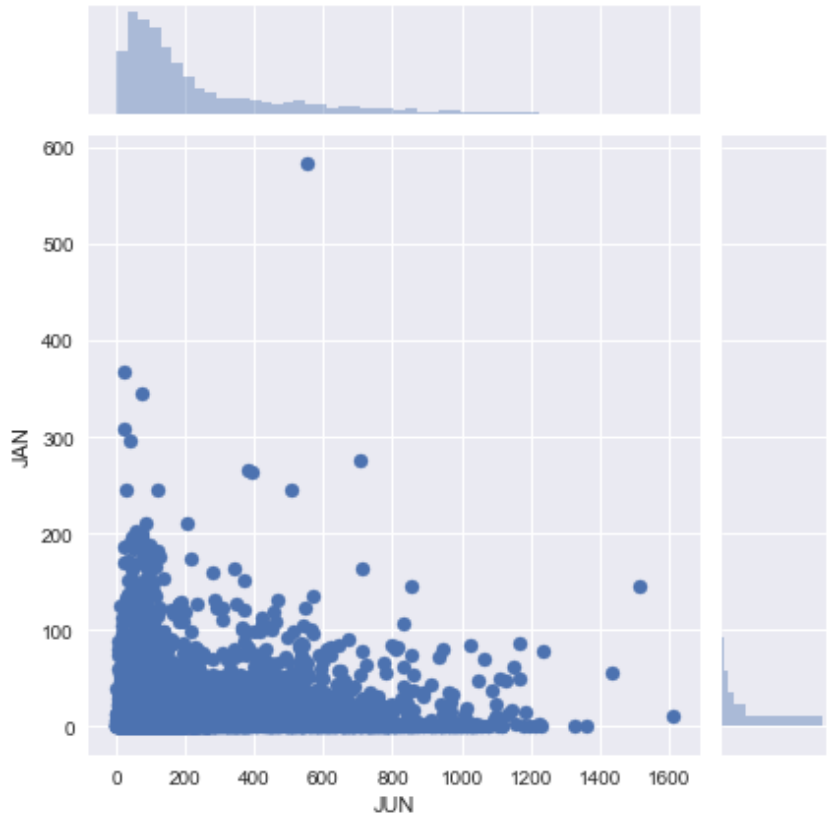
	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7

In [33]:

```
import seaborn as sns
```

In [34]:

```
sns.jointplot(rain.JUN,rain.JAN)
plt.show()
```



21. Generate a boxplot every SUBDIVISION

In [35]:

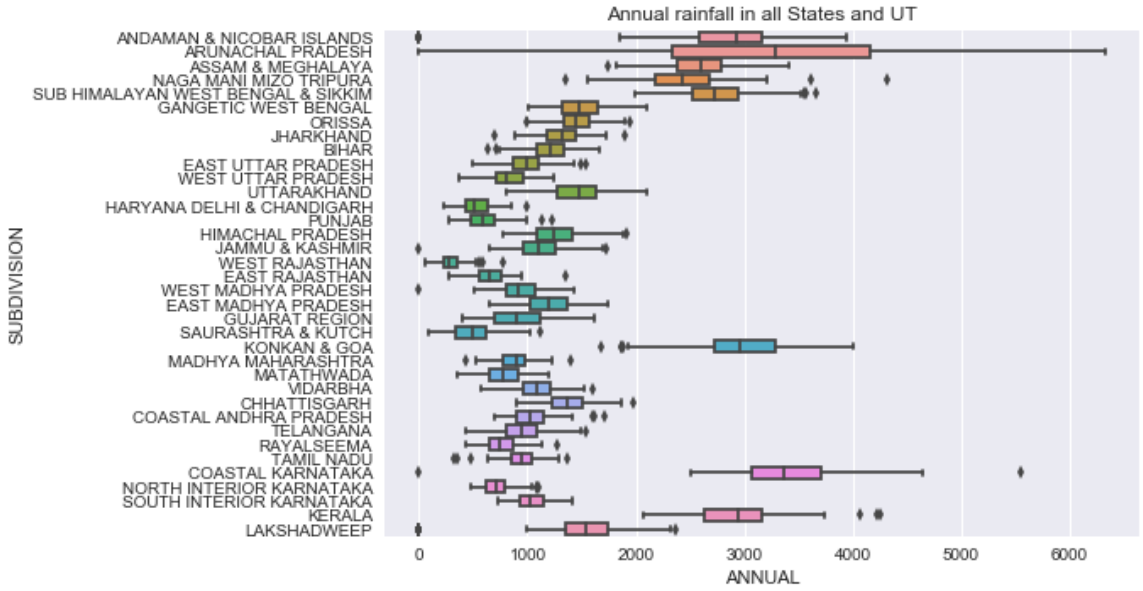
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
rain=rain.fillna(0)
```

In [36]:

```
sns.boxplot(y='SUBDIVISION', x='ANNUAL', data=rain)
plt.title('Annual rainfall in all States and UT')
```

Out[36]:

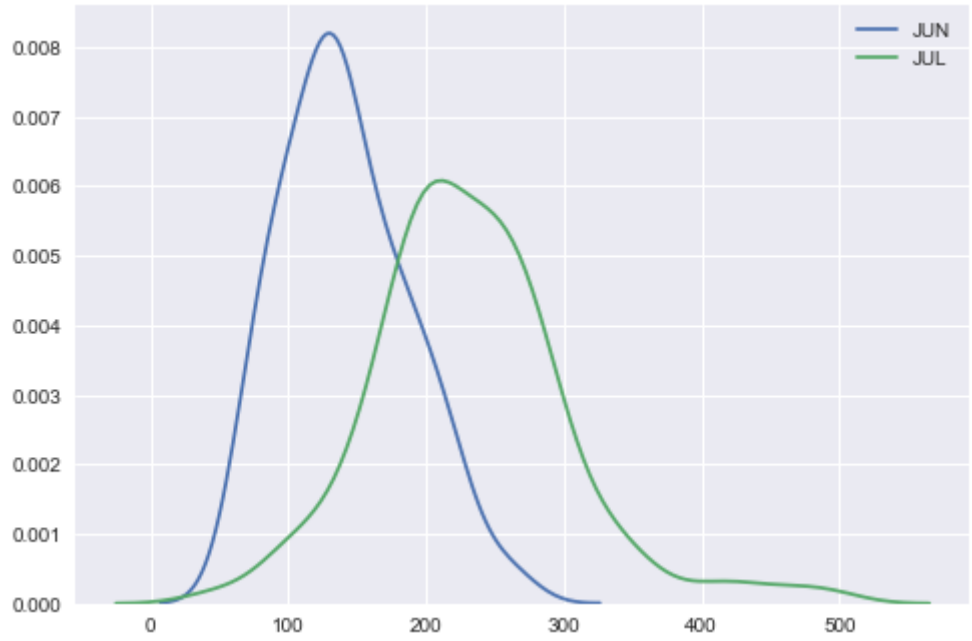
```
Text(0.5, 1.0, 'Annual rainfall in all States and UT')
```



22. Create a KDEplot for the month of JUN & JAN and take SUBDIVISION as SOUTH INTERIOR KARNATAKA. What conclusion can you draw from the plot?

In [37]:

```
#More rainfall in the month in JUN
sns.kdeplot(rain[rain.SUBDIVISION=='SOUTH INTERIOR KARNATAKA'].JUN)
sns.kdeplot(rain[rain.SUBDIVISION=='SOUTH INTERIOR KARNATAKA'].JUL)
plt.show()
```

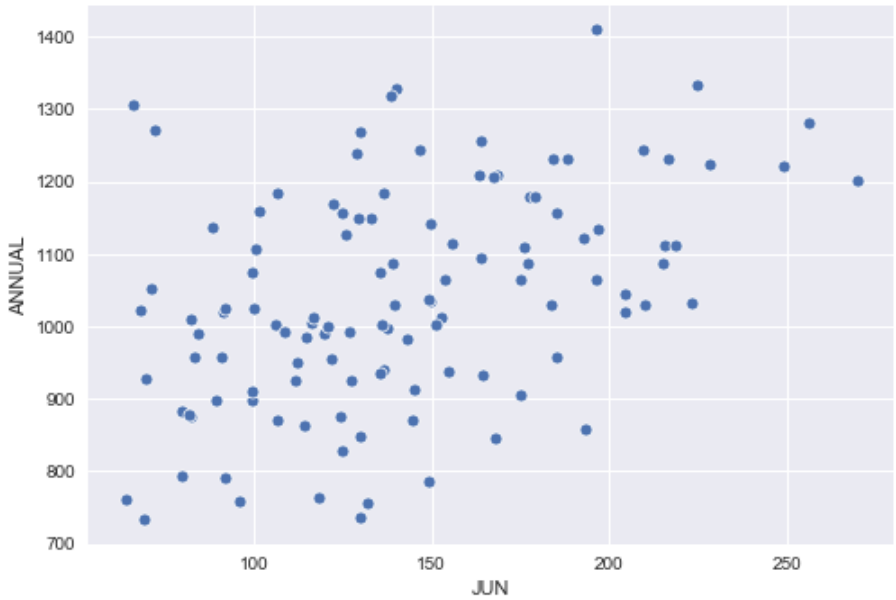


23. Using seaborn plot a scatter plot for SUBDIVISION=SOUTH INTERIOR KARNATAKA

- for the month JUN and ANNUAL rainfall
- for the month JAN and ANNUAL rainfall **Which is closely related to ANNUAL rainfall**

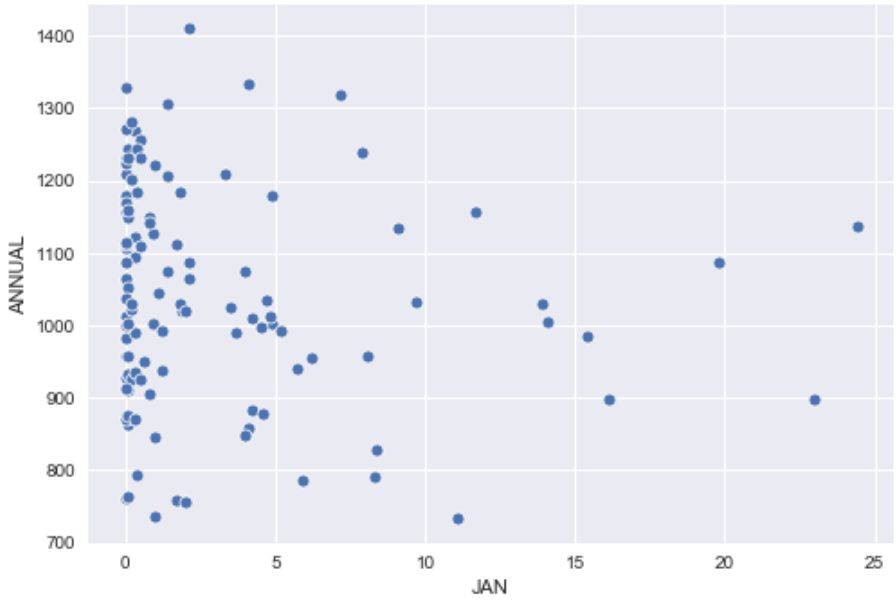
In [38]:

```
sns.scatterplot(rain[rain.SUBDIVISION=='SOUTH INTERIOR KARNATAKA'].JUN,rain[rain.SUBDIV
ISION=='SOUTH INTERIOR KARNATAKA'].ANNUAL)
plt.show()
```



In [39]:

```
sns.scatterplot(rain[rain.SUBDIVISION=='SOUTH INTERIOR KARNATAKA'].JAN,rain[rain.SUBDIV
ISION=='SOUTH INTERIOR KARNATAKA'].ANNUAL)
plt.show()
```



24. Using Cars dataset(Car_sales.csv) and generate a barchart for vehicle type to see the count of vehicle types.

In [40]:

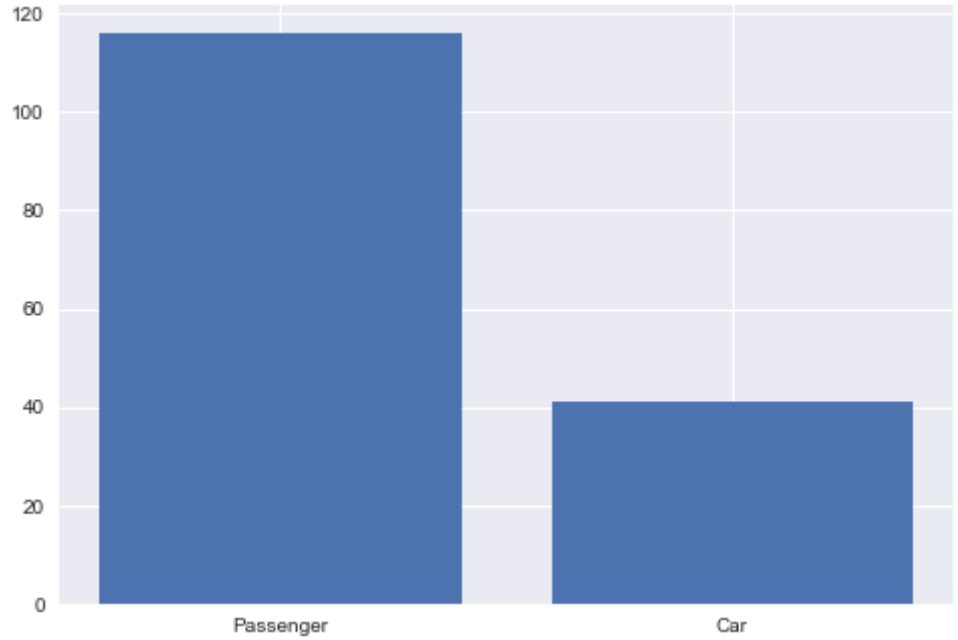
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
car=pd.read_csv('Car_sales.csv')
car.head()
```

Out[40]:

	Manufacturer	Model	Sales in thousands	4-year resale value	Vehicle type	Price in thousands	Engine size	Horsepower	Wh
0	Acura	Integra	16.919	16.360	Passenger	21.50	1.8	140.0	
1	Acura	TL	39.384	19.875	Passenger	28.40	3.2	225.0	
2	Acura	CL	14.114	18.225	Passenger	NaN	3.2	225.0	
3	Acura	RL	8.588	29.725	Passenger	42.00	3.5	210.0	
4	Audi	A4	20.397	22.255	Passenger	23.99	1.8	150.0	

In [41]:

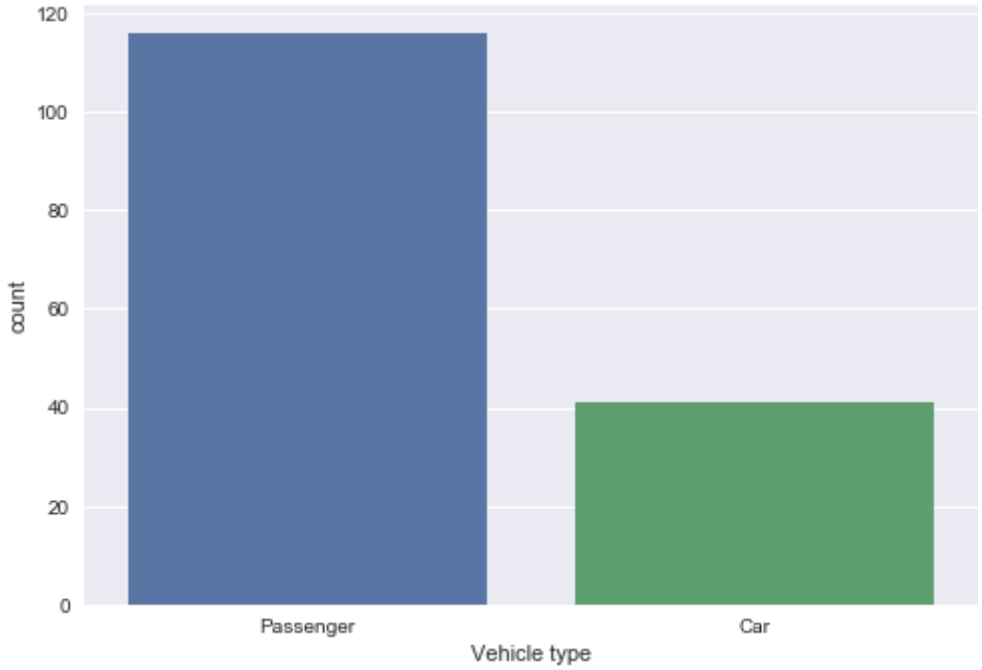
```
label=list(set(car['Vehicle type']))
count=[]
for i in label:
    count.append(car[car['Vehicle type']==i].Manufacturer.count())
plt.bar(label,count)
plt.show()
```



25. Generate a countplot for vehicle type to see the count of vehicle types.

In [42]:

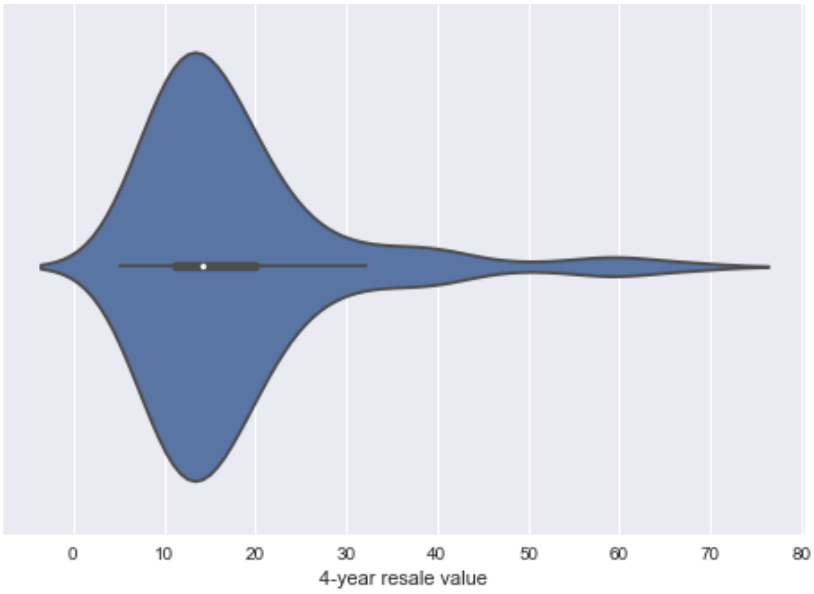
```
sns.countplot(car['Vehicle type'])
plt.show()
```

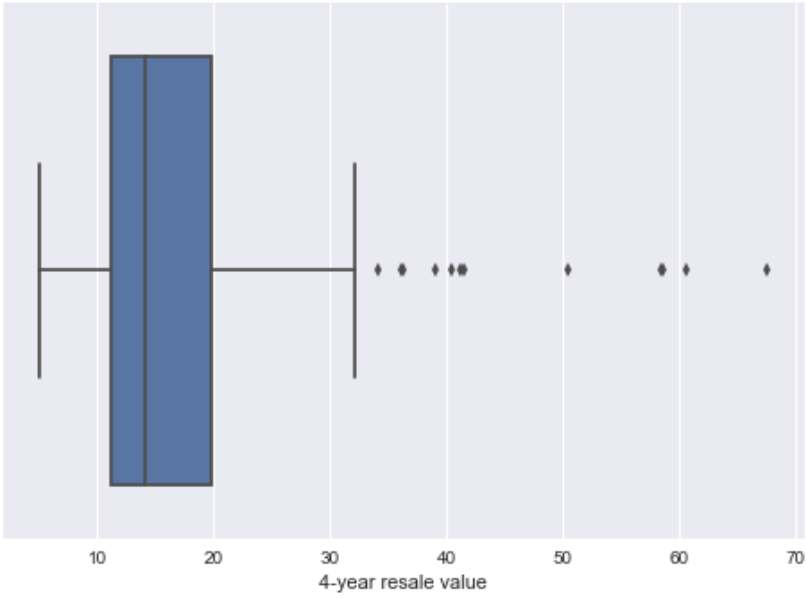


26. Create a violin plot and boxplot for 4-year resale value.

In [43]:

```
sns.violinplot(car['4-year resale value'])
plt.show()
sns.boxplot(car['4-year resale value'])
plt.show()
```


0
10
20
30
40
50
60
70
80
4-year resale value

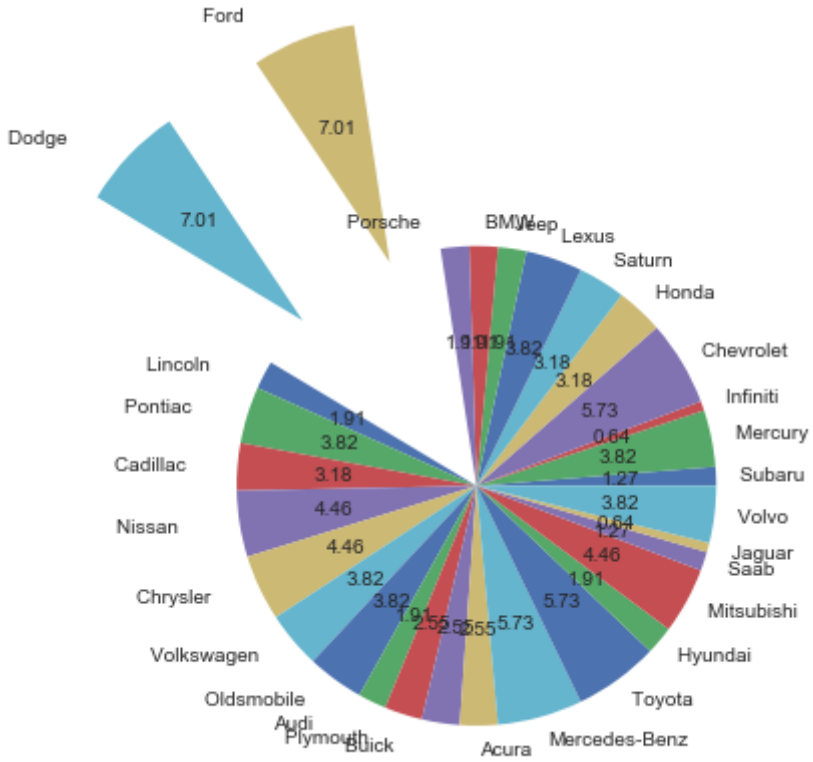

10
20
30
40
50
60
70
4-year resale value

27. Create pie chart and highlight(explode) manufacturer who made more than 10 models

In [44]:

```
val=[]
for i in set(car.Manufacturer):
    val.append(car[car.Manufacturer==i].Manufacturer.count())
temp_df=pd.DataFrame({'Manufacturer':list(set(car.Manufacturer)),'Count':val})


explode_max=np.zeros(len(val))
indx=list(temp_df[temp_df.Count>10].index)
explode_max[indx]=1
plt.pie(val,labels=list(set(car.Manufacturer)),autopct='%0.2f',explode=explode_max)
plt.show()
```

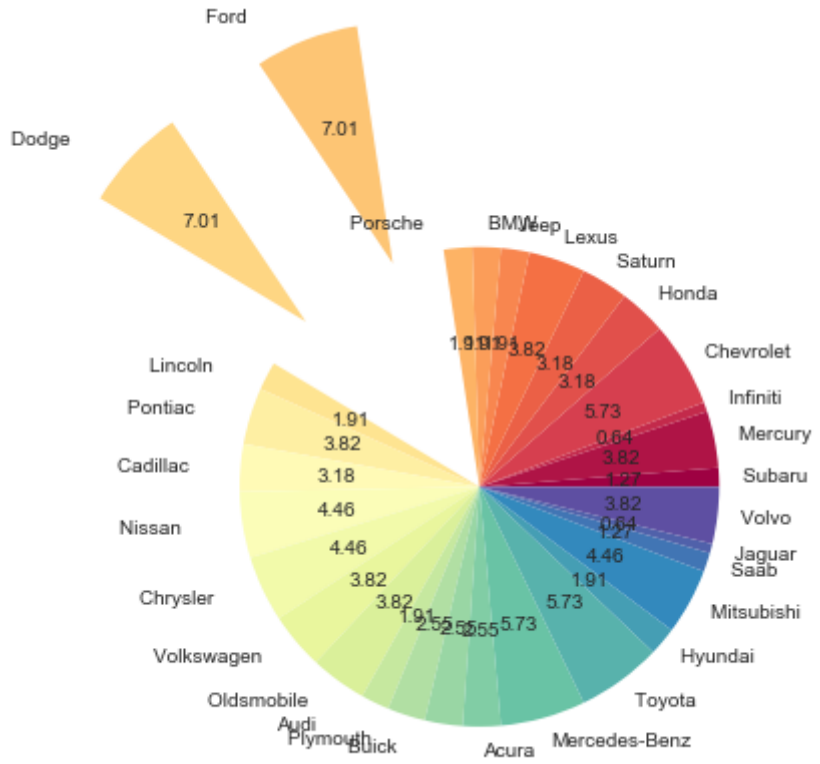


28. Create a custom color list and generate the pie chart again using new colors.

[**Hint**: Using linspace generate a custom color list]

In [45]:

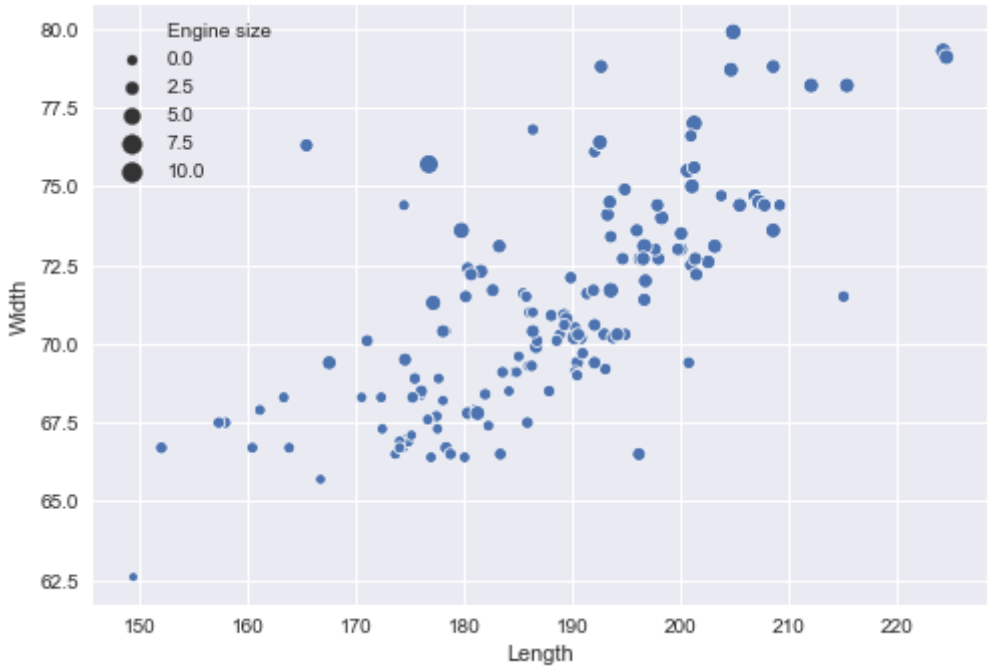
```
cmap = plt.get_cmap('Spectral')
colors = [cmap(i) for i in np.linspace(0,1,len(set(car.Manufacturer)))]
plt.pie(val,labels=list(set(car.Manufacturer)),autopct='%0.2f',explode=explode_max,colo
rs=colors)
plt.show()
```



29. Generate scatterplot comparing Length and Width of the car and use Engine size to determine to size of points.

In [46]:

```
sns.scatterplot(car.Length,car.Width,size=car['Engine size'])
plt.show()
```



30. Generate a heatmap using columns : 'Sales in thousands','Price in thousands','4-year resale value'.

In [47]:

```
sns.heatmap([car['Sales in thousands'],car['Price in thousands'],car['4-year resale value'],])
```

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23dbb13af60>
```