Important Libraries Do a check to gain more information about requests module: https://www.w3schools.com/python/module_requests.asp

Know More about Python List: https://www.w3schools.com/python/python_lists.asp, https://www.w3schools.com/python/python_lists_access.asp

If you want to know more about BeautifulSoup : https://beautiful-soup-4.readthedocs.io/en/latest/

Not possible to cover every concept today,but I can give you the basic understanding about it: 🙂

You can always remove the '#' and run the following commands , also '#' is used to make a comment in python

```
import requests
from bs4 import BeautifulSoup as bsp
import pandas as pd
```

```
requests.get('https://internshala.com/internships/','html.parser')
```

```
<Response [200]>
```

```
requests.get('https://internshala.com/internships/','html.parser').content
```

```
name="email" autofocus="true" placeholder="Enter your e-mail address">\n
<input type="hidden" name="src_url" value="/internships/%3Fhtml.parser">\n
<input type="hidden" id="subscription_location" name="subscription_location"
value="subscription_popup_new">\n                                <span class="input-
group-btn"><button type="submit" class="btn btn-primary btn-input"
id="but_subscription_popup_form">Subscribe</button></span>\n
</div>\n                    </div>\n                </form>\n                <div
class="subscription_alert_footer">\n                    <div
class="link_container">\n                        <a href="#"
id="already_subscribed">I\'m already subscribed</a>\n                    </div>\n
<div class="link_container">\n                        <a href="#"
id="no_thanks">No, thanks</a>\n                    </div>\n
</div>\n            </div>\n        </div>\n    </div>\n\n\n    <script
nonce="YyqjQ7b/o4iVffOzkJ9I0A==">\n        if
(document.getElementById("subs_timer") != null) {\n            var distance = new
Date(\'2024/06/01 00:00:00\').getTime() - new Date(\'2024/05/31
01:23:46\').getTime();\n\n            var x = setInterval(function () {\n
if (distance < 0) {\n                clearInterval(x);\n
document.getElementById("subs_timer").innerHTML = "";\n                }\n\n
var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));\n
var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));\n
var seconds = Math.floor((distance % (1000 * 60)) / 1000);\n\n                if
(hours == 0 && minutes == 0 && seconds == 0) {\n
clearInterval(x);\n
document.getElementById("subs_timer").innerHTML = "";\n                }\n\n
if (hours < 10) {\n                hours = "0" + hours\n                }\n
if (minutes < 10) {\n                minutes = "0" + minutes\n
}\n                if (seconds < 10) {\n                    seconds = "0" +
seconds\n                }\n\n                distance = distance - 1000;\n
```

```
url='https://internshala.com/internships/'
```

Getting Responses from the website

```
resp=requests.get(url,'html.parser')
print(resp.status_code)
```

```
200
```

```
fieldname=input("Enter your profile : ")
```

```
Enter your profile : web development
```

```
field=fieldname.replace(' ','-')
```

URL Handling

```
modified_url=url+str(field)+'-internship/'
print(f"Status :{requests.get(modified_url)}")
print(modified_url)
```

```
Status :<Response [200]>
https://internshala.com/internships/web-development-internship/
```

```
resp_new=requests.get(modified_url)


soup=bsp(resp_new.content,'html.parser')
#print(soup)


# type(soup)


pages=int(soup.find('span',id='total_pages').text)
# print(pages)


urlList = []
page = 1
while page <= pages:
  newUrl = modified_url+str(f"page-{page}/")
  urlList.append(newUrl)
  page +=1
# print(urlList)


soup2 = []
for url in urlList:
  resp_new=requests.get(url)
  soup3=bsp(resp_new.content,'html.parser')
  soup2.append(soup3)
# print(len(soup2))


# print(soup.prettify())
```

Scraping

```
name=[]
for soup in soup2:
  names=soup.find_all('div',class_='individual_internship_header')
  for i in names:
    name.append(i)
# print(len(name))
#print(name)


# print(type(name))
# profile=name.find_all('h3',class_='heading_4_5 profile')
```

```
# for i in name:
#   p=i.find('h3',class_='heading_4_5 profile')
  # print(p)
  # print(p.text)
  # print(p.text.strip())
  # break


profile=[]
for i in name:
  p=i.find('h3',class_='heading_4_5 profile')
  # print(p)
  # print(p.text)
  # print(p.text.strip())
  a=p.text.strip()
  profile.append(a)
  #break
# print(len(profile))
print(f"All profiles available are : {profile}")
```

```
All profiles available are : ['Web Development', 'PHP Development', 'Demo Post', 'Flu
```

```
company=[]
for i in name:
  com=i.find('p').text.strip()
  #print(com)
  company.append(com)
# print(len(company))
print(company)
```

```
['Stirring Minds', 'UI TECH LAB LLP', 'Seven Arc Info Systems LLP', 'AppyHigh Technol
```

```
detail=[]
for soup in soup2:
  detailList=soup.find_all('div',class_='individual_internship_internship')
  for i in detailList:
    detail.append(i)
# len(detail)


#print(detail[0])


location=[]
for i in detail:
  loc=i.find('a').text
  location.append(loc)
  #print(loc)
# print(len(location))
print(f"Locations are : {location}")
```

```
Locations are : ['Delhi', 'Patna', 'Gurgaon', 'Gurgaon', 'Jaipur', 'Work from home',
```

```
duration_detail1=[]
for soup in soup2:
  duraList=soup.find_all('div',class_='item_body')
  for i in duraList:
    duration_detail1.append(i)
duration=[]
i = 1
while i < len(duration_detail1):
  duration.append(duration_detail1[i].text.strip()[0])
  i +=3
# print(len(duration))
print(duration)
```

```
['6', '6', '1', '6', '6', '6', '6', '3', '6', '6', '3', '2', '3', '6', '6', '6', '3',
```

```
stipend=[]
for soup in soup2:
  stipList=soup.find_all('span',class_='stipend')
  for i in stipList:
    val=i.text
    stipend.append(val)
# print(len(stipend))
print(f"Stipend is : {stipend}")
```

```
Stipend is : ['₹ 7,000 /month', '₹ 5,000 /month', '₹ 10,000 /month', '₹ 15,000-18,000
```

```
cont=[]
for soup in soup2:
  coutList=soup.find_all('div',class_='cta_container')
  for i in coutList:
    cont.append(i)


application_link=[]
for i in cont:
  anc=i.find('a')
  link=anc.get('href')
  #print(link)
  updated_link='https://internshala.com/'+link
  #print(updated_link)
  application_link.append(updated_link)
# print(len(application_link))
print(f"Application Link is : {application_link}")
```

```
Application Link is : ['https://internshala.com//internship/details/web-development-i
```

```
dataTable = {
    'profile': profile,
    "company": company,
    "location": location,
    "stipend":stipend,
    "duration": duration,
    "application Link": application_link
    }

df = pd.DataFrame(dataTable)
#print(f"""{len(profile)} {len(company)} {len(location)} {len(duration)} {len(stipend)} {
```

```
filename='internship_data_'+str(fieldname.replace(' ','_'))+'.csv'
print(filename)
```

```
internship_data_web_development.csv
```

```
df.to_csv(filename, index=False)
```

New Section

Find Duration of each individual internship All the information u have got till now create a dataframe out of it using pandas Save the dataframe in CSV Create a general Code to fetch data from any number of page : 😉

Remember : https://internshala.com/internships/analytics-internship/page-1/ and https://internshala.com/internships/analytics-internship/ both are same

```
from google.colab import drive
drive.mount('/content/drive')
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
df = pd.read_csv("/content/drive/MyDrive/Ecommerce Customers.csv")
df.head(10)
```

```
df.shape
```

```
df.dtypes
```

```
Email                   object
Address                 object
Avatar                  object
Avg. Session Length    float64
Time on App            float64
Time on Website        float64
Length of Membership   float64
Yearly Amount Spent    float64
dtype: object
```

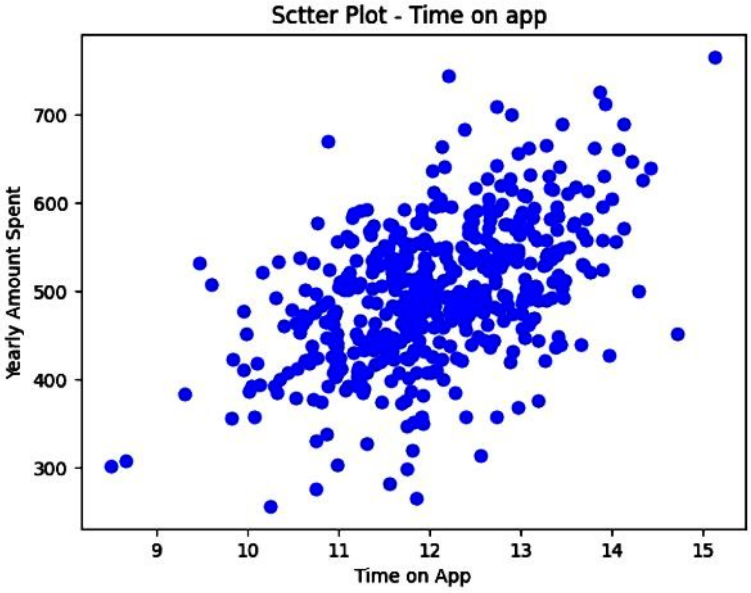
```
df.isnull().sum()
```

```
Email                  0
Address                0
Avatar                 0
Avg. Session Length    0
Time on App            0
Time on Website        0
Length of Membership   0
Yearly Amount Spent    0
dtype: int64
```

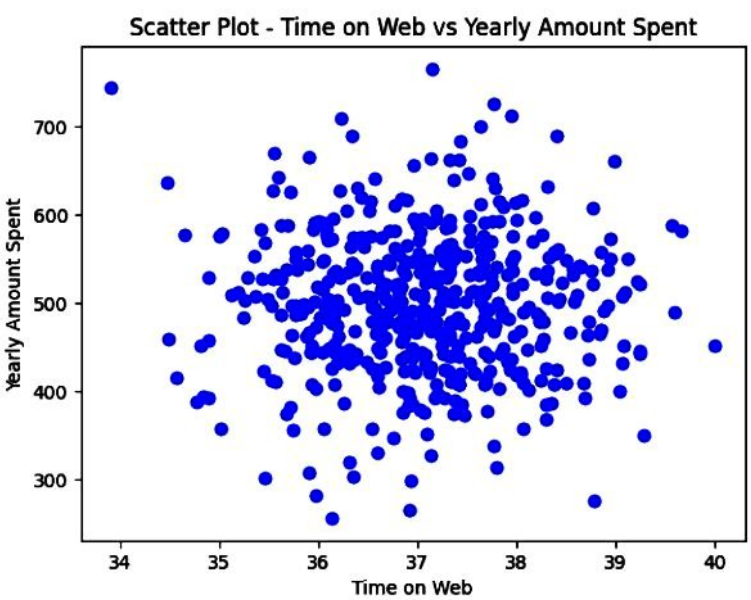
```
df.describe()
```

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	33.053194	12.052488	37.060445	3.533462	499.314038
std	0.992563	0.994216	1.010489	0.999278	79.314782
min	29.532429	8.508152	33.913847	0.269901	256.670582
25%	32.341822	11.388153	36.349257	2.930450	445.038277
50%	33.082008	11.983231	37.069367	3.533975	498.887875
75%	33.711985	12.753850	37.716432	4.126502	549.313828
max	36.139662	15.126994	40.005182	6.922689	765.518462

```
x = df['Time on App']
y = df['Yearly Amount Spent']

plt.scatter(x, y, color='blue', marker='o')

plt.title('Sctter Plot - Time on app ')
plt.xlabel('Time on App')
plt.ylabel('Yearly Amount Spent')

plt.show()
```

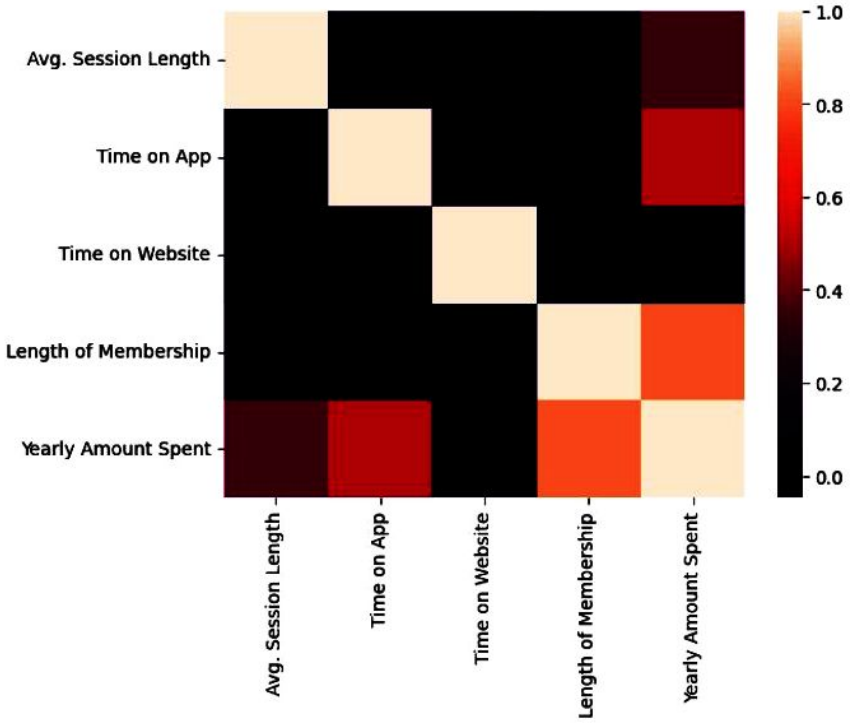



```
x = df['Time on Website']
y = df['Yearly Amount Spent']

plt.scatter(x, y, color='blue', marker='o')

plt.title('Scatter Plot - Time on Web vs Yearly Amount Spent')
plt.xlabel('Time on Web')
plt.ylabel('Yearly Amount Spent')

plt.show()
```




```
corr = df.select_dtypes('number').corr()

sns.heatmap(corr)
```

```
<Axes: >
```




```
df.drop(['Email', 'Address', 'Avatar'],axis=1 ,inplace=True)
df.head(10)
```

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	34.497268	12.655651	39.577668	4.082621	587.951054
1	31.926272	11.109461	37.268959	2.664034	392.204933
2	33.000915	11.330278	37.110597	4.104543	487.547505
3	34.305557	13.717514	36.721283	3.120179	581.852344
4	33.330673	12.795189	37.536653	4.446308	599.406092
5	33.871038	12.026925	34.476878	5.493507	637.102448
6	32.021596	11.366348	36.683776	4.685017	521.572175
7	32.739143	12.351959	37.373359	4.434273	549.904146
8	33.987773	13.386235	37.534497	3.273434	570.200409
9	31.936549	11.814128	37.145168	3.202806	427.199385

```
from sklearn.model_selection import train_test_split

y = df[['Yearly Amount Spent']]
X = df.drop(columns=['Yearly Amount Spent'])
```

```
y.head(5)
```

	Yearly Amount Spent
0	587.951054
1	392.204933
2	487.547505
3	581.852344
4	599.406092

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

```
X_test.shape
```

```
(100, 4)
```

```
X_train.shape
```

```
(400, 4)
```

```
from sklearn.preprocessing import StandardScaler
```

```
scl = StandardScaler()
scaled_X_train = scl.fit_transform(X_train)

scaled_X_test = scl.fit_transform(X_test)
```

```
pd.DataFrame(scaled_X_train)
```

	0	1	2	3
0	-1.822160	1.205863	-0.125495	-1.272752
1	-0.921078	-2.041998	0.081056	0.002643
2	0.765303	-0.892446	-0.785963	0.774123
3	-0.216300	-0.241272	-0.279913	-0.060903
4	0.333489	0.716997	-1.520914	-0.305554
...	...	...	...	...
395	0.080599	1.835929	2.084425	-0.629094
396	-0.680505	-1.409122	0.891667	-0.131719
397	2.723773	-1.192048	-1.506509	2.598134
398	0.121726	-0.452630	-1.110151	0.102238
399	-0.194598	-0.087798	-0.053617	-0.080550

400 rows × 4 columns

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
lr = LinearRegression()
lr.fit(scaled_X_train, y_train)

y_pred = lr.predict(scaled_X_test)
```

```
y_pred
```

```
       [578.83036909],
       [529.78443183],
       [487.40146045],
       [669.09605192],
       [579.48540182],
       [544.35890877],
       [288.31426732],
       [512.82731915],
       [601.56260052],
       [433.69746917],
       [471.39921528],
       [516.84171225],
       [418.48851112],
       [498.66453583],
       [552.7273268 ],
       [451.39693179],
       [520.4980463 ],
       [630.32492152],
       [517.1108975 ],
       [515.53327877],
       [526.69976729],
       [590.57903691],
       [466.98137317]])
```

```
y_test
```

	Yearly Amount Spent
304	494.687156
340	501.122492
47	563.672873
67	469.310861
479	402.167122
...	...
11	522.337405
192	505.119638
92	515.828815
221	591.437736
110	459.285123

100 rows × 1 columns