Intro lecture

Beautiful Soup

```
from bs4 import BeautifulSoup

html="<!DOCTYPE html><html><head><title>Page
Title</title></head><body><h3><b id='boldest'>Lebron
James</b></h3><p> Salary: $ 92,000,000 </p><h3> Stephen
Curry</h3><p> Salary: $85,000, 000 </p><h3> Kevin Durant </h3><p>
Salary: $73,200, 000</p></body></html>"


soup = BeautifulSoup(html, 'html5lib')
```

Python iterable

```
table_row=table.find_all(name='tr')


table_row:
```

Pizza Place	Orders	Slices
Domino's Pizza	10	100
Little Caesars	12	144

```
[<tr><td>Pizza Place</td><td>Orders</td><td>Slices</td></tr>,
 <tr><td>Domino's Pizza</td><td>10</td><td>100</td></tr>,
 <tr><td>Little Caesars</td><td>12</td><td>144</td></tr>,
```

Tag object

```
first_row =table_row[0]
first_row:

<tr><td>Pizza Place</td><td>Orders</td> <td>Slices </td></tr>


first_row.td :


<td>Pizza Place</td>
```

Variable row

Pizza Place	Orders	Slices
Domino's Pizza	10	100
Little Caesars	12	144
Papa John's	15	166

```
for i,row in enumerate(table_rows):

    print("row", i)
    cells=row.find_all("td")

    for j,cell in enumerate(cells):

        print("column", j , "cell", cell)
```

Full scrapins

Python program for web scraping

```
import requests
from bs4 import BeautifulSoup

page = requests.get("http://EnterWebsiteURL...").text

#Creates a BeautifulSoup object
soup = BeautifulSoup(page, "html.parser")

# Pulls all instances of <a> tag
artists = soup.find_all('a')

#Clears data of all tags
for artist in artists:
    names = artist.contents[0]
    fullLink = artist.get('href')
    print(names)
    print(fullLink)
```

Request

Requests

```
import requests
url='https://www.ibm.com/'
r=requests.get(url)
r.status_code:200
r.request.headers
```

```
{'User-Agent': 'python-requests/2.24.0', 'Accept-Encoding': 'gzip, deflate',
'Accept': '*/*', 'Connection': 'keep-alive', 'Cookie':
'_abck=A5C90067E0241F8BBD3ECC70ECDB1EC0~-
1~YAAQLc2U0ZaliKB1AQAAwucY4QT77mnX/GJQJvRGOV48PDVR70euYZ9FTBsFbF3z4kwEcKgttV+Ot6Sz
P+sjsPDIFni428WfQn/yUYVddQ2e1ejPilhGVE+eNJl4xbIRXxE1U59jQLvi0jd/Q8e6ClE7mw27Qzb5nC
gdMcd4ZnULKeu6U5QSVLFRUPnhMLvOM40+1WFW1CNH36WrccSUtoKkTsExTltvbsvxIbthin3znfL1rackK
tV9WJFJlAEdqajkt0qX/frGBULGLK/r47j78DuZHabdMhdalss9J1MpfXj6kXCFe~-1~-1~-1;
bm_sz=564D34683F98DB203E92B73A05A3324C~YAAQLc2U0ZWliKB1AQAAwucY4Qn7jwZFH9NO7QY5q55
JvdADHcL95RCSPsAGLlx2RAr+iIH6hEt/frpWT8RXkjWBcCtS29lu91pIEz6zjed+o0yyTH7aL/qkC9nzV
mdZfvAhOjFKnFzewR7xguYLBc7XlAAeG8GrkPFHq0vMTo38GwTMCC+xGYQYpM6y'}
```

Requests

```
r.request.body:None

header=r.headers
header:
```

Dictionary

```
{'Server': 'Apache', 'x-drupal-dynamic-cache': 'UNCACHEABLE', 'Link':
'<https://www.ibm.com/ca-en>; rel="canonical", <https://www.ibm.com/ca-en>; rel="revision",
<https://www.ibm.com/ca-en>; rel="revision", <//1.cms.s81c.com>; rel=preconnect; crossorigin,
<//1.cms.s81c.com>; rel=dns-prefetch', 'x-ua-compatible': 'IE=edge', 'Content-Language': 'en-
ca', 'x-generator': 'Drupal 8 (https://www.drupal.org)', 'x-dns-prefetch-control': 'on', 'x-
drupal-cache': 'MISS', 'Last-Modified': 'Thu, 19 Nov 2020 10:32:43 GMT', 'ETag':
'"1605781963"', 'Content-Type': 'text/html; charset=UTF-8', 'x-acquia-host': 'www.ibm.com',
'x-acquia-path': '/ca-en', 'x-acquia-site': '', 'x-acquia-purge-tags': '', 'x-varnish':
'18482279 12683819', 'x-cache-hits': '9', 'x-age': '7030', 'Accept-Ranges': 'bytes',
'Content-Encoding': 'gzip', 'Cache-Control': 'public, max-age=300', 'Expires': 'Thu, 19 Nov
2020 15:26:47 GMT', 'X-Akamai-Transformed': '9 11615 0 pmb=mTOE,1', 'Date': 'Thu, 19 Nov 2020
15:21:47 GMT', 'Content-Length': '11725', 'Connection': 'keep-alive', 'Vary': 'Accept-
Encoding', 'x-content-type-options': 'nosniff', 'X-XSS-Protection': '1; mode=block',
'Content-Security-Policy': 'upgrade-insecure-requests', 'Strict-Transport-Security': 'max-
age=31536000', 'x-ibm-trace': 'www-dipatcher: dynamic rule'}
```

Requests

```
header['date']:'Thu, 19 Nov 2020 15:21:47 GMT'

header['Content-Type']:'text/html; charset=UTF-8'

r.encoding:'UTF-8'

r.text[0:100]:

'<!DOCTYPE html>\n<html lang="en-ca" dir="ltr">\n
<head>\n <meta charset="utf-8" />\n<script>digitalD'
```

Create Query string

```
url_get='http://httpbin.org/get'
payload={"name":"Joseph","ID":"123"}
r=requests.get(url_get,params=payload)
r.url:'http://httpbin.org/get?name=Joseph&ID=123'
r.request.body : None
r.status_code: 200
```

Content-Type

```
r.text
```

{ "args": { "ID": "123", "name": "Joseph" }, "headers": { "Accept": "*/*", "Accept-Encoding": "gzip, deflate", "Host": "httpbin.org", "User-Agent": "python-requests/2.24.0", "X-Amzn-Trace-Id": "Root=1-5fbbd03e-106584ab42513a6d5cef39a9" }, "origin": "99.228.137.181", "url": "http://httpbin.org/get?name=Joseph&ID=123" }

```
r.headers['Content-Type']  'application/json'
```

Content-Type

```
r.json()
```

{'args': {'ID': '123', 'name': 'Joseph'}, 'headers': {'Accept': '*/*', 'Accept-Encoding': 'gzip, deflate', 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.24.0', 'X-Amzn-Trace-Id': 'Root=1-5fbbd03e-106584ab42513a6d5cef39a9'}, 'origin': '99.228.137.181', 'url': 'http://httpbin.org/get?name=Joseph&ID=123'}

```
r.json()['args']   {'ID': '123', 'name': 'Joseph'}
```

Post Request:

POST

```
url_post="http://httpbin.org/post"
payload={"name":"Joseph","ID":"123"}
r_post=requests.post(url_post,data=payload)
```

Compare POST and GET

```
print("POST request URL:",r_post.url )
print("GET request URL:",r.url)
```

```
POST request URL: http://httpbin.org/post
GET request URL: http://httpbin.org/get?name=Joseph&ID=123
```

Compare POST and GET

```
print("POST request body:",r_post.request.body)
print("GET request body:",r.request.body)

POST request body: name=Joseph&ID=123
GET request body: None

r_post.json()['form']

{'ID': '123', 'name': 'Joseph'}
```