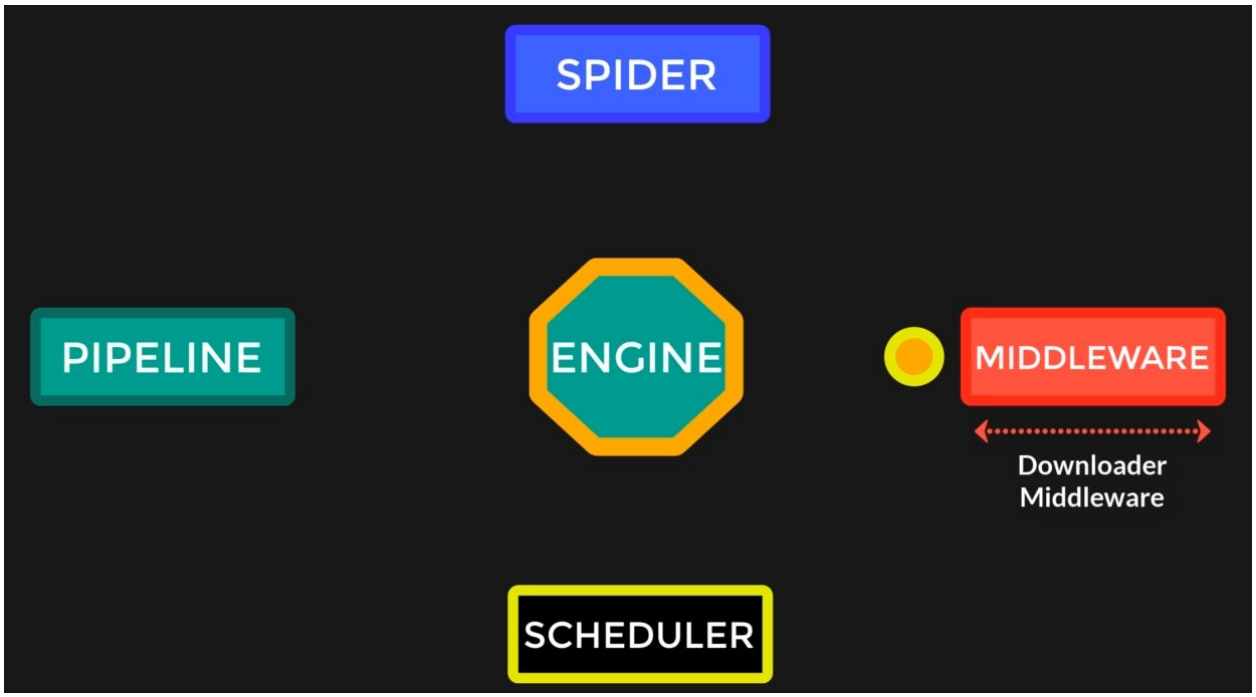
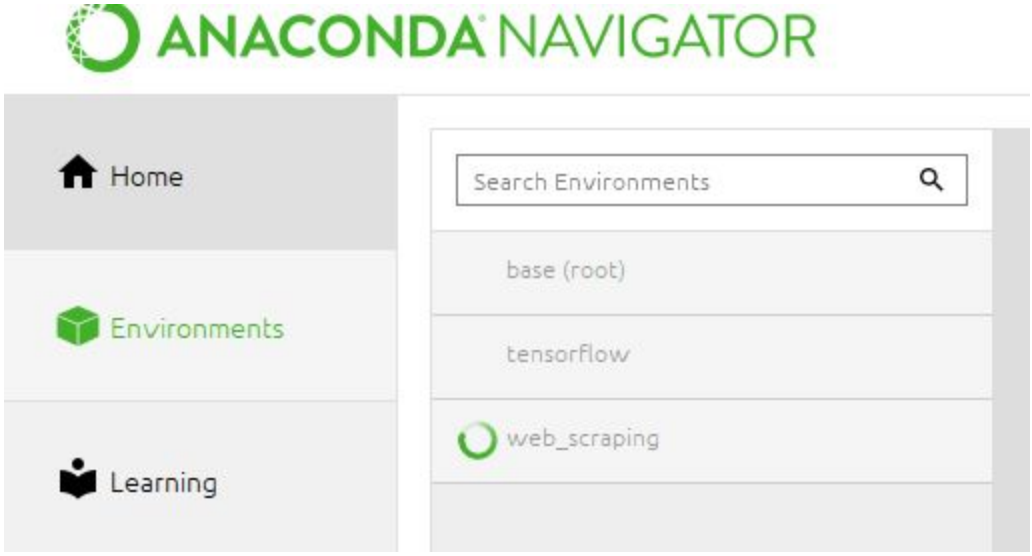
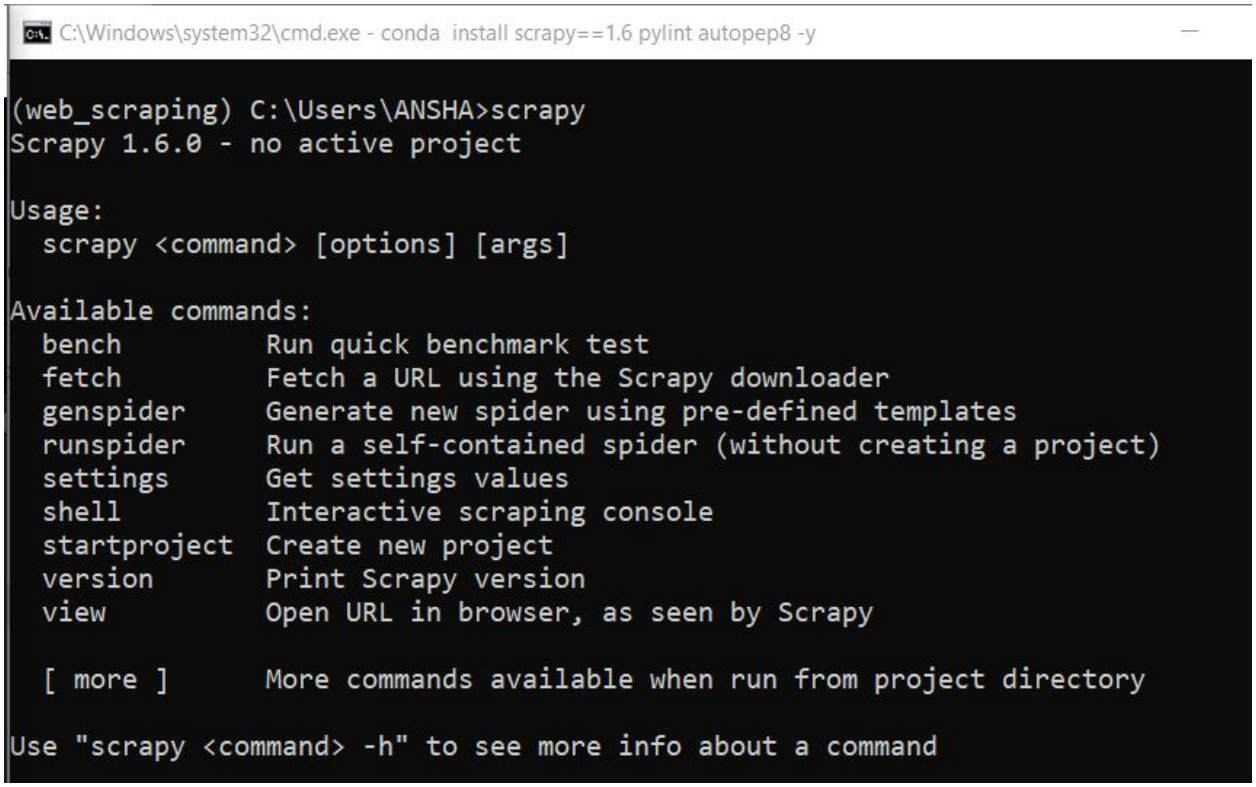
WEB SCRAPING



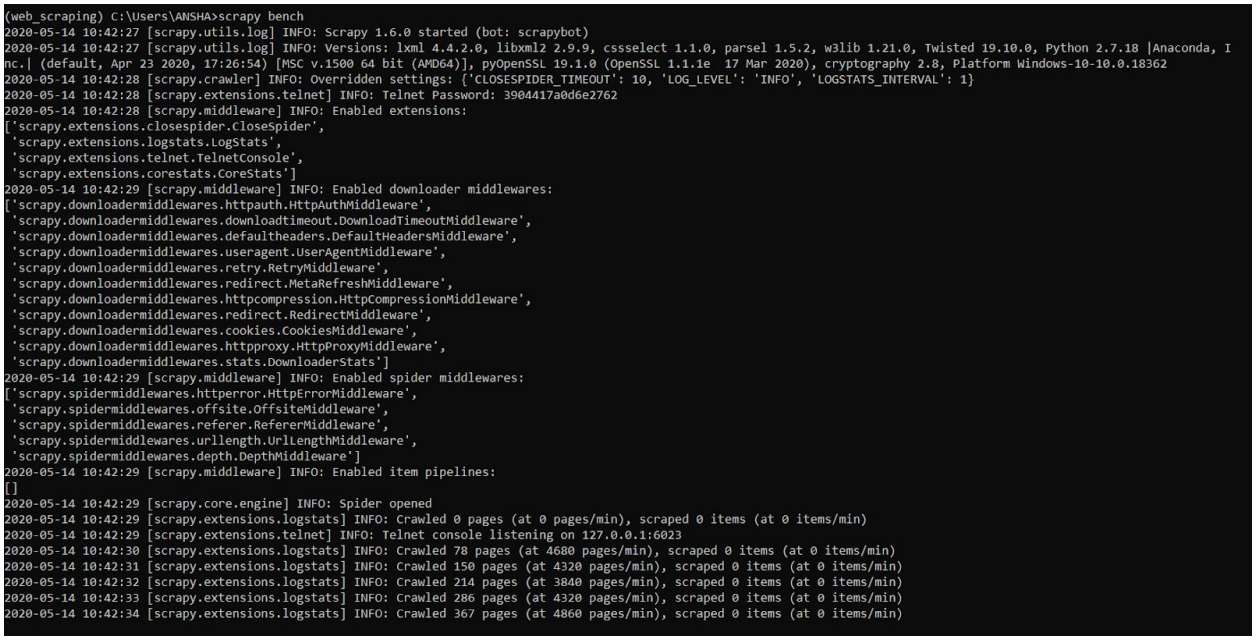
- Add new environment for web scraping



- Install scrapy using “conda install scrapy==1.6”
- Install pylint & autopep8 for better development experience.
- “conda install scrapy==1.6 pylint autopep8 -y” to download all.



=> When we write scrapy we get the list of all available commands



=> Scrapy banch is going to be show like this. Depend upon system to System.

=> scrapy fetch http://google.com is going to fetch markup of website.

STEPS:

★ Now we are going to start a project.
- Use “scrapy startproject worldometers”
- cd worldometers
- code . (to open VsCode)

★ Now add a new spider
- scrapy genspider countries www.worldometers.info/world-population/population-by-country

★ conda install ipython

❖ One project can contain different spiders with unique names.

❖ Now scrapy shell is used to open shell

```
[s] Available Scrapy objects:
[s]   scrapy     scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s]   crawler    <scrapy.crawler.Crawler object at 0x00000000050078C8>
[s]   item       {}
[s]   settings   <scrapy.settings.Settings object at 0x0000000005007348>
[s] Useful shortcuts:
[s]   fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are foll
owed)
[s]   fetch(req)                  Fetch a scrapy.Request and update local objects
[s]   shelp()           Shell help (print this help)
[s]   view(response)    View response in a browser
In [1]:
```

```
In [1]: fetch("https://www.worldometers.info/world-population/population-by-country/")
```

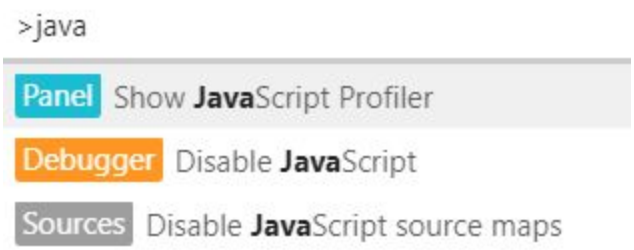
```
In [6]: r = scrapy.Request("https://www.worldometers.info/world-population/population-by-country/")
   ...:

In [7]: r = scrapy.Request("https://www.worldometers.info/world-population/population-by-country/")

In [8]: fetch(r)
2020-05-14 11:27:04 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.worldometers.info/world-popula
tion/population-by-country/> (referer: None)

In [9]: response.body
Out[9]: '\n<!DOCTYPE html><!--[if IE 8]> <html lang="en" class="ie8"> <![endif]--><!--[if IE 9]> <html lang="
en" class="ie9"> <![endif]--><!--[if !IE]><!--> <html lang="en"> <!--<![endif]--> <head> <meta charset="utf-8
"> <meta http-equiv="X-UA-Compatible" content="IE=edge"> <meta name="viewport" content="width=device-width, i
```

response.body will give the whole html page.

```
In [10]: view(response)
Out[10]: True
```

=> This will open page in browser

- ❖ View will open the website with JavaScript but spider view the website without JS.
  - ➢ To disable JS open inspect using “ctrl+shift+i”
  - ➢ After that open pellet using “ctrl+shift+p” and disable javascript.(Click on second)



- ❖ Xpath is used as:

```
In [11]: response.xpath('//h1')
Out[11]: [<Selector xpath='//h1' data=u'<h1>Countries in the world by populat...'>]
```

First try to find elements in inspection using “ctrl+f” and then proceed to selection using xpath in response.

```
In [12]: title = response.xpath('//h1')

In [13]: title = response.xpath('//h1/text()')

In [14]: title
Out[14]: [<Selector xpath='//h1/text()' data=u'Countries in the world by population ...'>]

In [15]: title.get()
Out[15]: u'Countries in the world by population (2020)'
```

- ❖ Now using css:

```
In [6]: title_css = response.css("h1::text")

In [7]: title_css
Out[7]: [<Selector xpath=u'descendant-or-self::h1/text()' data=u'Countries in the world by population ...'>]

In [8]: title_css.get()
Out[8]: u'Countries in the world by population (2020)'
```

❖ Now scrapping countries as:

```
In [9]: countries = response.xpath("//td/a/text()").getall()

In [10]: countries
Out[10]:
[u'China',
 u'India',
 u'United States',
```

=> Here **getall()** will fetch value as array in spite of string like **get()**

=> Using response.css

```
In [11]: countries_css = response.css("td a::text").getall()

In [12]: countries_css
Out[12]:
[u'China',
 u'India',
 u'United States',
 u'Indonesia',
 u'Pakistan',
 u'Brazil',
 u'Nigeria',
```

=> To run spider countries:

```
(web_scraping) G:\Web Scrapping\worldometers>scrapy crawl countries
```

❖ **XPATH :** XML Path Language

❖ **CSS :** Cascading Style Sheet

```
(web_scraping) G:\Web Scrapping\multiplePages>scrapy genspider -l
Available templates:
  basic
  crawl
  csvfeed
  xmlfeed
```

To use crawl template we use:

```
(web_scraping) G:\Web Scrapping\coin>scrapy genspider -t crawl coin_fetcher "www.livecoin.net/en"
```

- “scrapy genspider -l” lists the available templates
- Engines used by different browsers

Splash also uses “Apple Webkit”

❖ Selenium setup guide:

$ conda install selenium -y

$ conda install scrapy-selenium `scrapy-selenium`

❖ To use MongoDB in pype line we should install (*pymongo)

```
(web_scraping) G:\Web Scrapping\imdb>conda install pymango dnspython -y
```

❖ Sqlite3 db is already in python 3