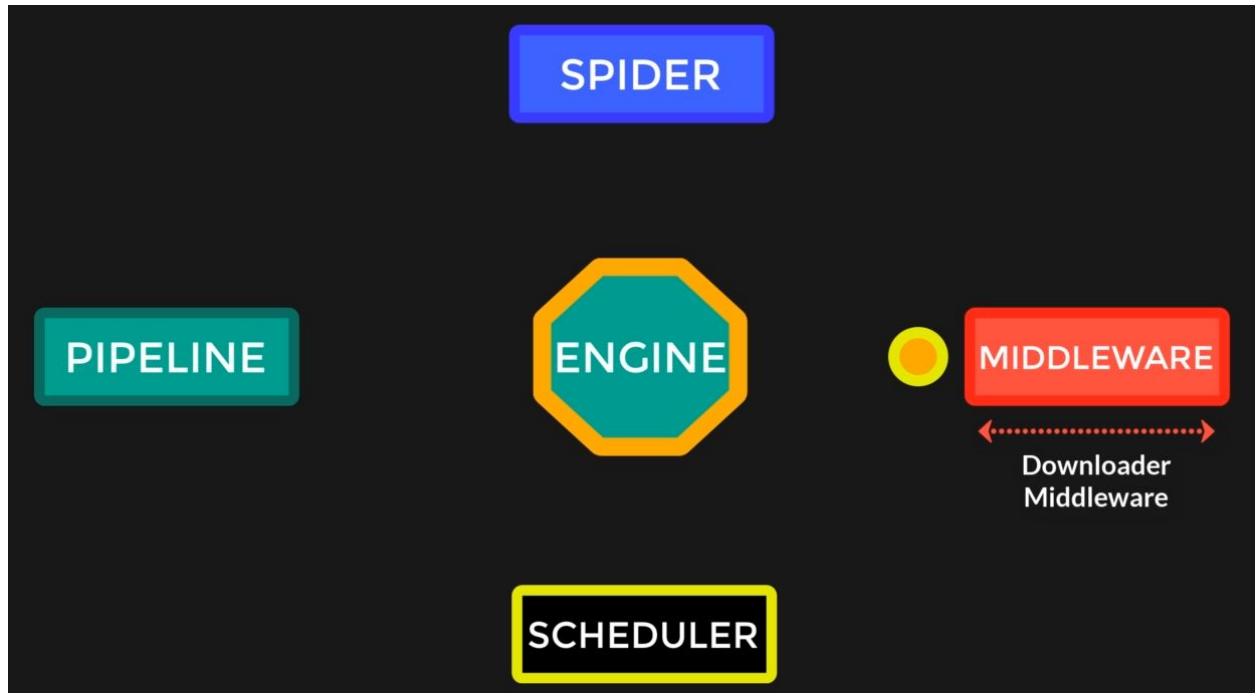
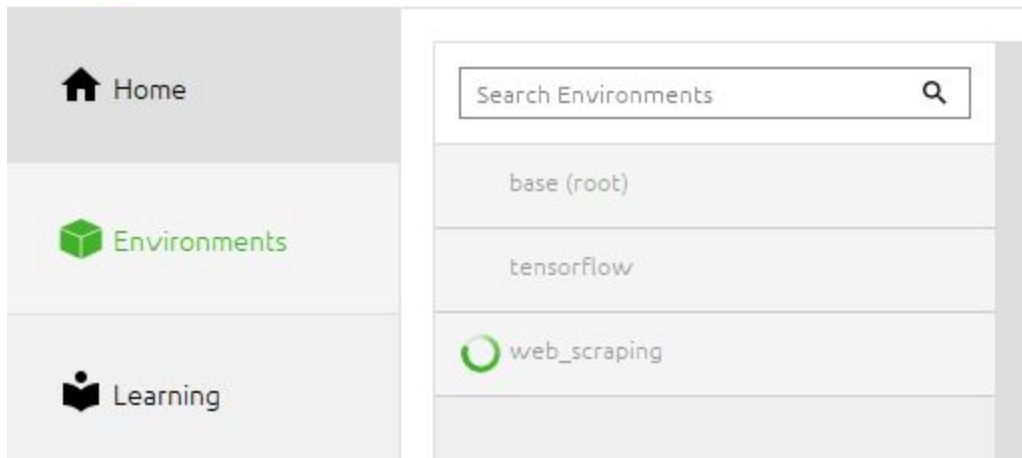


# WEB SCRAPING



- ❖ 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.

```
C:\Windows\system32\cmd.exe - conda install scrapy==1.6 pylint autopep8 -y

(web_scraping) C:\Users\ANSHA>scrapy
Scrapy 1.6.0 - no active project

Usage:
  scrapy <command> [options] [args]

Available commands:
  bench          Run quick benchmark test
  fetch          Fetch a URL using the Scrapy downloader
  genspider       Generate new spider using pre-defined templates
  runspider       Run a self-contained spider (without creating a project)
  settings        Get settings values
  shell           Interactive scraping console
  startproject    Create new project
  version         Print Scrapy version
  view           Open URL in browser, as seen by Scrapy

  [ more ]       More commands available when run from project directory

Use "scrapy <command> -h" to see more info about a command
```

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

```
(web_scraping) C:\Users\ANSHA>scrapy bench
2020-05-14 10:42:27 [scrapy.utils.log] INFO: Scrapy 1.6.0 started (bot: scrapybot)
2020-05-14 10:42:27 [scrapy.utils.log] INFO: Versions: lxml 4.4.2.0, libxml2 2.9.9, cssselect 1.1.0, parsel 1.5.2, w3lib 1.21.0, Twisted 19.10.0, Python 2.7.18 [Anaconda, I
nc.] (default, Apr 23 2020, 17:26:54) [MSC v.1500 64 bit (AMD64)], pyOpenSSL 19.1.0 (OpenSSL 1.1.1e 17 Mar 2020), cryptography 2.8, Platform Windows-10-10.0.18362
2020-05-14 10:42:28 [scrapy.crawler] INFO: Overridden settings: {'CLOSESPIDER_TIMEOUT': 10, 'LOG_LEVEL': 'INFO', 'LOGSTATS_INTERVAL': 1}
2020-05-14 10:42:28 [scrapy.extensions.telnet] INFO: Telnet Password: 3904417a0d6e2762
2020-05-14 10:42:28 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.closespider.CloseSpider',
 'scrapy.extensions.logstats.LogStats',
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.corestats.CoreStats']
2020-05-14 10:42:29 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2020-05-14 10:42:29 [scrapy.middleware] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referrer.ReferrerMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2020-05-14 10:42:29 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2020-05-14 10:42:29 [scrapy.core.engine] INFO: Spider opened
2020-05-14 10:42:29 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2020-05-14 10:42:29 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
2020-05-14 10:42:30 [scrapy.extensions.logstats] INFO: Crawled 78 pages (at 4680 pages/min), scraped 0 items (at 0 items/min)
2020-05-14 10:42:31 [scrapy.extensions.logstats] INFO: Crawled 150 pages (at 4320 pages/min), scraped 0 items (at 0 items/min)
2020-05-14 10:42:32 [scrapy.extensions.logstats] INFO: Crawled 214 pages (at 3840 pages/min), scraped 0 items (at 0 items/min)
2020-05-14 10:42:33 [scrapy.extensions.logstats] INFO: Crawled 286 pages (at 4320 pages/min), scraped 0 items (at 0 items/min)
2020-05-14 10:42:34 [scrapy.extensions.logstats] INFO: Crawled 367 pages (at 4860 pages/min), scraped 0 items (at 0 items/min)
```

=> Scrapy bench 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](http://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 0x0000000050078C8>
[s] item {}
[s] settings <scrapy.settings.Settings object at 0x000000005007348>
[s] Useful shortcuts:
[s] fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are followed)
[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-population/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)

>java

Panel Show JavaScript Profiler

Debugger Disable JavaScript

Sources Disable JavaScript source maps

❖ 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



V8  
Engine



Spider  
Monkey



Apple  
Webkit



Chakra

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 pymongo dnspython -y
```

❖ Sqlite3 db is already in python 3