Python Training Certification Course

FULL STACK

Shell Scripting and Django

simplilearn

Learning Objectives

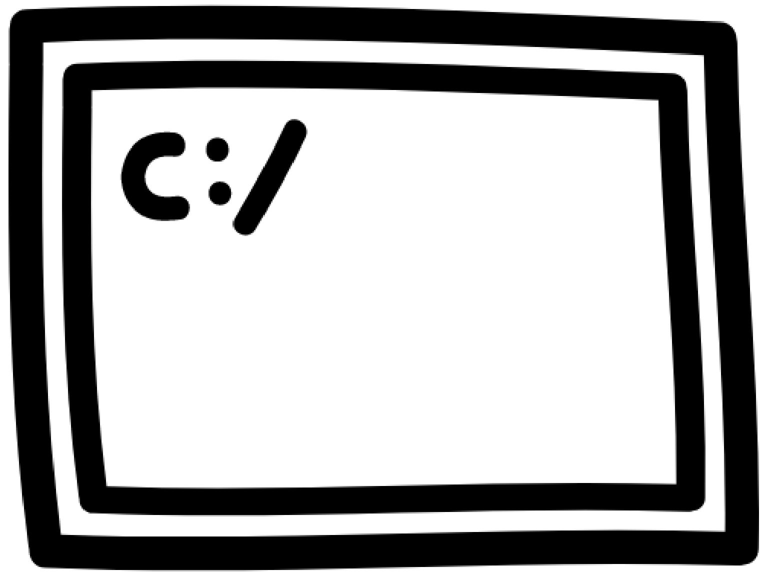
By the end of this lesson, you will be able to:

- Demonstrate Shell Scripting
- Demonstrate Web Scraping
- Explain Django

Shell Scripting

Shell Scripting

- A Shell script is a computer program designed to be run by the Unix shell.
- A Shell is a program which provides a user interface for operating system services.
- If you are using any operating system, you are interacting with a shell.
- The Shell is initiated when the user logs in or starts the terminal.
- A special program called Terminal in Linux or MacOS and Command Prompt in Windows OS is provided to type in the human readable commands to be executed.

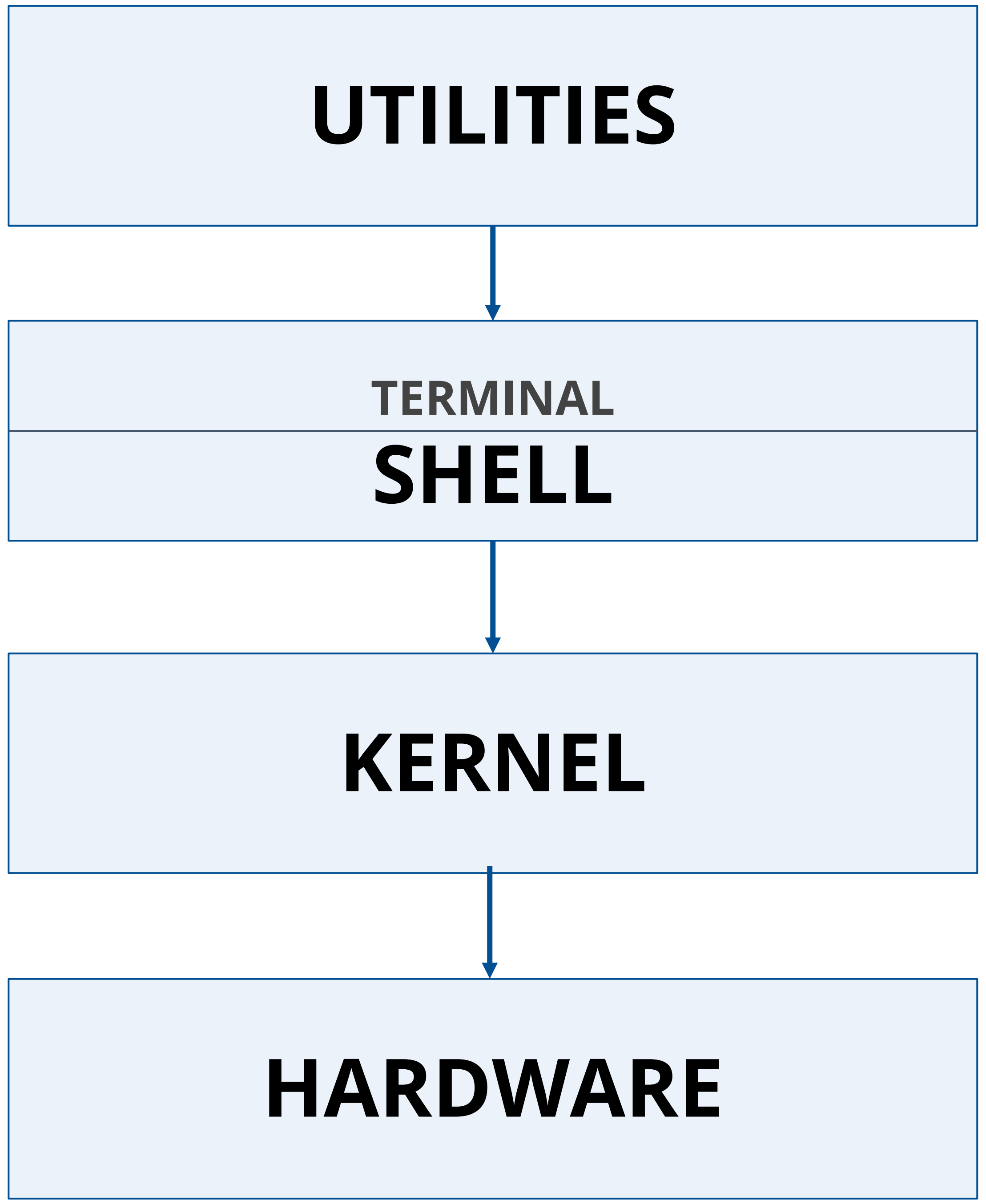


Shell Scripting

- Shell accepts human readable commands and converts them into something the kernel can understand.
- The kernel is a computer program at the core of a computer’s operating system and controls everything in the system.
- The kernel is responsible for the following in a Linux environment:
  - File management
  - Process management
  - I/O management
  - Memory management
  - Device management

Kernel, Shell, and Terminal

Figure shows the relationship between kernel, Shell, and terminal:



Shell Scripting

- Shells usually accept command as input from users and execute them.
- When we have a group of commands to be executed routinely, we can write these commands in a file that can be read and executed by the Shell.
- These files are called Shell scripts or Shell programs.
- Each Shell script is saved with .sh file extension .
  - Example: myscript.sh

Elements of Shell Scripting

- Shell keywords (example: break)
- Shell commands (example: cd, ls, echo, pwd, and touch)
- Functions
- Control flow statements (example: if..then..else, case, and Shell loops)

Applications of Shell Scripting

- Automating repetitive tasks
- Making routine backups
- Monitoring system
- Adding new functionality to the Shell

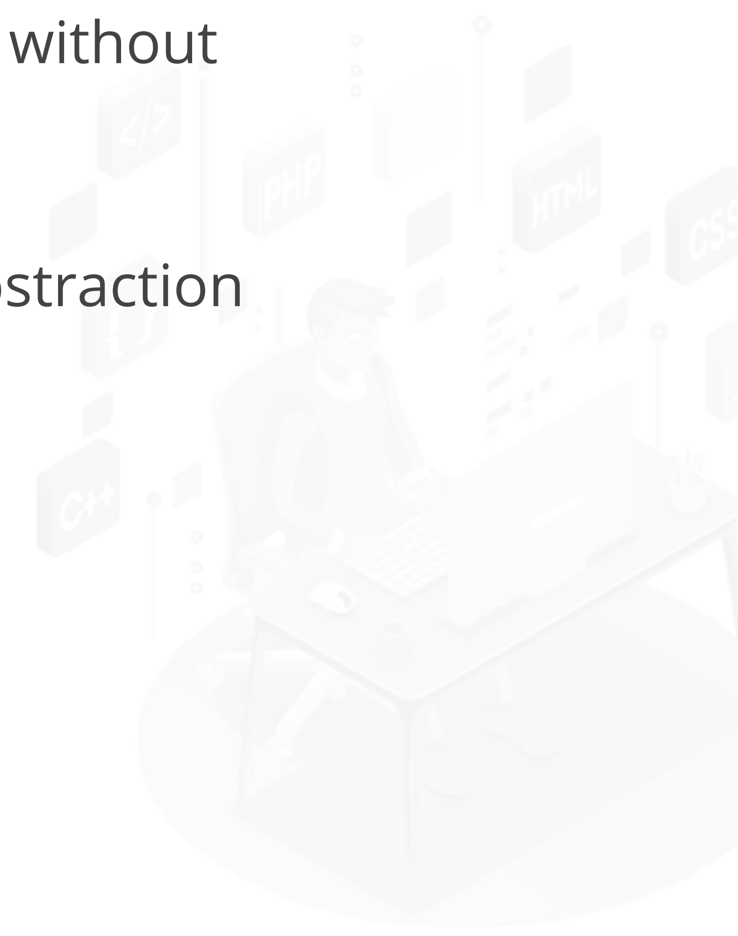
Features of Shell Script

- A primitive programming environment
- Complex syntax
- Difficult to test
- Variables tend to be global
- String is the only data structure of the Shell language
- Perl, Bash, ksh, or Ruby

Shell Scripting with Python

Shell Scripting with Python

- Python is well suited for system programming and platform independent system programming.
- Python offers smart data structures and lets you work with numbers and dates without complexity.
- Python lets you write unit tests.
- System programming with the aid of the sys and the os module serves as an abstraction layer between the application and the operating system.
- The general advantages of Python in system programming are:
  - Simple and clear
  - Well structured
  - Highly flexible



sys Module

The sys module provides:

- Information on the Python interpreter
- Access to command line arguments
- Access to standard data streams
- Input or output redirections
- Functionality to change the output behavior of the Shell

sys Module

The table below gives the name and description of methods under the sys module:

Name	Description
dir(system)	Gives a summary of the available constants, functions, and methods
sys.getrecursionlimit()	Gives the maximum recursion depth
sys.setrecursionlimit()	Provides the possibility to change the recursion depth
sys.version()	Gives the current version number of Python
sys.argv	Contains the command-line arguments passed to the script
sys.displayhook	Changes the way the interpreter prints interactively entered expressions
sys.stdin sys.stdout sys.stderr	Gives access to the standard input, standard output, and standard error data streams

sys Module

Name	Description
sys.byteorder	Indicator of the native byte order
sys.executable	String containing the name of the executable binary (path and executable file name) for the Python interpreter.
sys.maxint	Attribute containing the largest positive integer supported by Python’s regular integer type
sys.maxsize	Reports the platform's pointer size that limits the size of Python's data structures such as strings and lists
sys.maxunicode	Integer giving the largest supported code point for a Unicode character
sys.modules	A dictionary mapping module names to modules which have already been loaded

sys Module

Name	Description
sys.path	Contains the search path, where Python is looking for modules
sys.platform	Name of the platform on which Python is running
sys.version_info	A tuple containing the five components of the version number: major, minor, micro, release-level, and serial. The values of this tuple are integers except the value for the release level, which is one of the following: 'alpha', 'beta', 'candidate', or 'final'
sys.__stdin__ sys.__stdout__ sys.__stderr__	Contains the original values of stdin, stderr, and stdout at the start of the program

os Module

- Most important module for interacting with the operating system
- Allows platform independent programming by providing abstract methods
- Provides various methods to access the file system and execute Shell scripts

The table below shows methods under the os module:

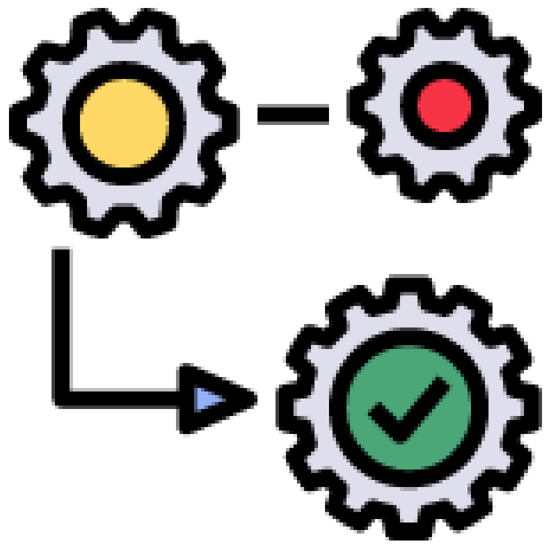
Function	Description
os.getcwd()	Returns a string with the path of the current working directory
os.chdir(path)	Changes the current working directory to path
os.getcwdu()	Like getcwd(), but outputs unicode
os.listdir(path)	A list with the content of the directory defined by path, that is, subdirectories and file names

os Module

Function	**Description**
os.mkdir(path[, mode=0755])	Creates a directory named path with numeric mode if it is not existing. The default mode is 0777 (octal)
os.makedirs(name[, mode=511])	Recursive directory creation function
os.rename(old, new)	The *old* file or directory is renamed to *new*
os.renames(old, new)	Works like rename(), except that it creates recursively any intermediate directories needed to make the *new* pathname
os.rmdir(path)	Removes the directory path

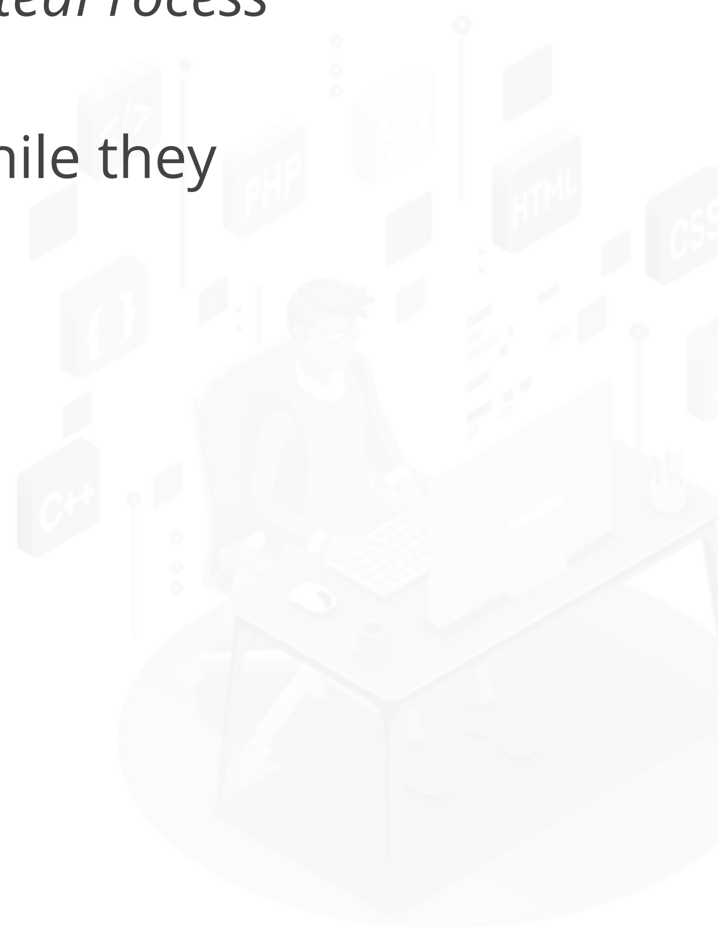
subprocess Module

- The subprocess module is available from Python 2.4.
- It lets you create spawn processes, connect to their input, output, and error pipes, and obtain their return codes.
- The module subprocess was created to replace the following modules:
  - os.system
  - os.spawn*
  - os.popen*
  - popen2.*
  - commands.*



subprocess Module

- Instead of using the system method of the os module, os.system('touch xyz'), we can use the popen() command of the subprocess module.
- subprocess.run starts a process, waits for it to finish, and then returns a *CompletedProcess* instance that has information about what happened.
- If you want processes to run in the background or need to interact with them while they continue to run, you need the the Popen constructor.



subprocess Module

- The subprocess module is safe from injection by default, unless shell=True is used.
- Programs like SSH give arguments to a Shell after they have started causing injection vulnerabilities.
- shlex.quote will ensure that any spaces or shell metacharacters are properly escaped.

The code below shows how to use shlex.quote

```
#Without shlex
>>> sp.run(['ssh', 'user@host', 'ls', path])

#With shlex
>>> import shlex
>>> sp.run(['ssh', 'user@host', 'ls', shlex.quote(path)])
```

Reading and Writing Files

- Traditionally, we use coreutils like grep, sed, awk, tr, sort to go over text files line-by-line.
- In Python, we need to turn a path into a file object before processing.
- The open() function takes a path and returns a file object.

The code below shows how to read files in Python:

```
with open('file.txt') as new_file:
    for line in new_file:
        #do stuff
```

Reading and Writing Files

The code below shows how to write to files in Python:

```
with open('file.txt', 'w') as new_file:
    new_file.write('some text\n')
    new_file.writelines(['a\n', 'b\n', 'c\n'])
    print('another line', file=new_file)
```

Replacing Miscellaneous File Operations

The shutil library provides utility functions for copying and archiving files and directory trees. The code below shows how shutil performs various file operations in Python:

```
import shutil
# $ mv src dest
shutil.move('src', 'dest')
# $ cp src dest
shutil.copy2('src', 'dest')
# $ cp -r src dest
shutil.copytree('src', 'dest')
# $ rm a_file
os.remove('a_file') # ok, that's not shutil
# $ rm -r a_dir
shutil.rmtree('a_dir')
# $ tar caf 'my_archive.tar.gz' 'my_folder'
shutil.make_archive('my_archive.tar.gz', 'gztar', 'my_folder')
```

Replacing Miscellaneous File Operations

```
  # $ tar xaf 'my_archive.tar.gz'
  shutil.unpack_archive('my_archive.tar.gz')
  # chown user:ninjaaron a_file.txt
  shutil.chown('a_file.txt', 'ninjaaron', 'user')
  # info about disk usage, a bit like `df`, but not exactly.
  shutil.disk_usage('.')
usage(total=123008450560, used=86878904320, free=36129546240)
  #  ^ sizes in bytes
  # which vi
  shutil.which('vi')
'/usr/bin/vi'
  # info about the terminal you're running in.
  shutil.get_terminal_size()
os.terminal_size(columns=138, lines=30)
```

Replacing sed, grep, and awk

- Shell commands like sed, grep, and awk can be replaced with regular expressions in Python.
- The regex functionality is encapsulated in the re module, in Python.
- grep is the Unix utility that goes through each line of a file, tests if it contains a certain pattern, and then prints the lines that match.

The code below shows how to do this in Python with and without regex:

```
>>> 'substring' in 'string containing substring'
True

>>> import re
>>> re.search(r'a pattern', r'string containing a pattern')
<_sre.SRE_Match object; span=(18, 27), match='a pattern'>
>>> re.search(r'a pattern', r'string without the pattern')
>>> # Returns None, which isn't printed in the Python REPL
```

Replacing sed, grep, and awk

- sed can be thought of as a text editor without a UI.
- Instead of editing text manually, you give sed instructions about changes to apply to lines.

The code below shows an example of replacing sed with Python:

```
>>> # sed 's/a string/another string/g' -- i.e. doesn't regex
>>> replaced = (s.replace('a string', 'another string') for s in ics)
>>> # sed 's/pattern/replacement/g' -- needs regex
>>> replaced = (re.sub(r'pattern', r'replacement', s) for s in ics)
```

Replacing sed, grep, and awk

- AWK is a Turing-complete text or table processing language.
- Inside the Shell scripts, it is frequently used to extract fields from tabular data which is mostly splitting strings.

The code below shows an example of replacing awk with Python to split strings:

```
>>> # awk '{print $1}'
>>> field1 = (f[0] for f in (s.split() for s in ics))
>>> # awk -F : '{print $1}'
>>> field1 = (f[0] for f in (s.split(':') for s in ics))
>>> # awk -F '[^a-zA-Z]' '{print $1}'
>>> field1 = (f[0] for f in (re.split(r'[^a-zA-Z]', s) for s in ics))
```

Dealing with Exit Codes

- A process that fails doesn't raise an exception by default in Python and Shell.
- A non-zero exit code indicates something other than an error and hence can be used in an **if** condition to check for process failures.
- If you want a non-zero exit code to crash the program, especially during development, you can use the check parameter.

The code below shows an example of dealing with exit codes in Python:

```
>>> if proc.returncode != 0:
...     # do something else

>>> subprocess.run(['ls', '-lh', 'foo bar baz'], check=True)
```

Redirecting Process IO

- We can use stdin and stdout to redirect input and output to files respectively.
- To do something with input and output text inside the script, we need to use the special constant, subprocess.PIPE

The code below shows an example of using pipes:

```
>>> proc = subprocess.run(['ls'], stdout=subprocess.PIPE,
universal_newlines=True)
>>> print(proc.stdout)
foo
out.html
README.rst
```

Dealing with Time

- In a shell script, you just use the output of date for time.
- Python has two libraries for dealing with time: time and datetime

The code below shows an example of using time and datetime in Python:

```
>>> import time
>>> time.strftime('%Y.%m.%d')
'2019.08.19'

>>> import datetime
>>> # get the current time as a datetime object
>>> datetime.datetime.now()
datetime.datetime(2018, 8, 18, 10, 5, 56, 518515)
>>> now = _
>>> str(now)
'2018-08-18 10:05:56.518515'
>>> now.strftime('%Y.%m.%d')
'2018.08.18'
```

Shell Scripting in Python

Objective: You are given a project to write a Python program that performs shell operations.

Steps to a Python program that performs shell operations:

1. Open the code editor.
2. Import the Python modules like os, datetime, and sys.
3. Code methods for reading files, managing processes, performing date arithmetic and other shell operations.
4. Execute the code.

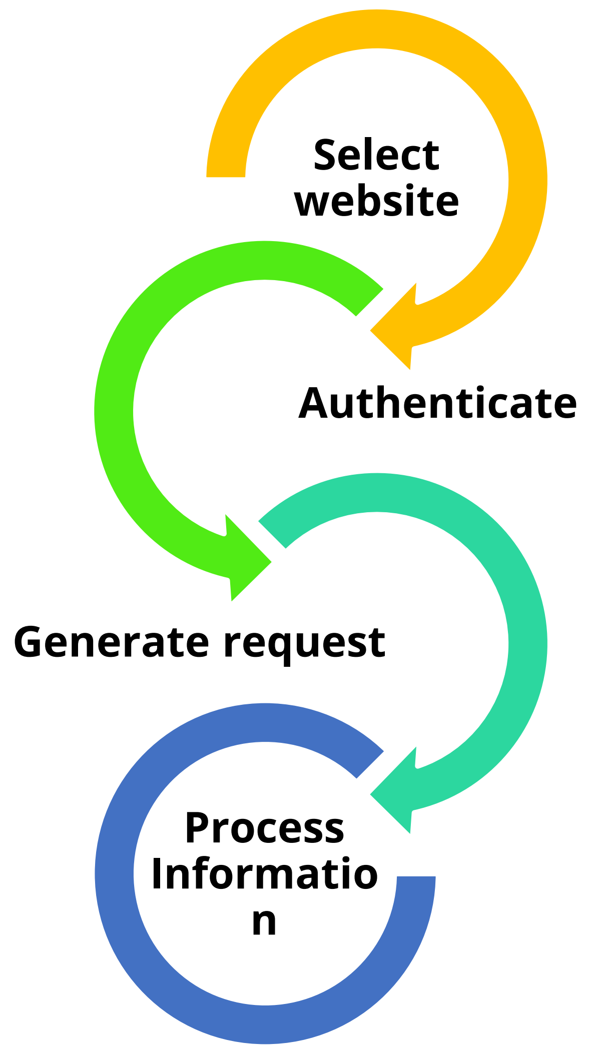
ASSISTED PRACTICE

Web Scraping

Web Scraping

Web Scraping is another process usually done with Shell scripts. It is the process of extracting information from a website or internet. Web scraping is one of the most important techniques of data extraction from internet. It allows the extraction of unstructured data from websites and convert it into structured data.

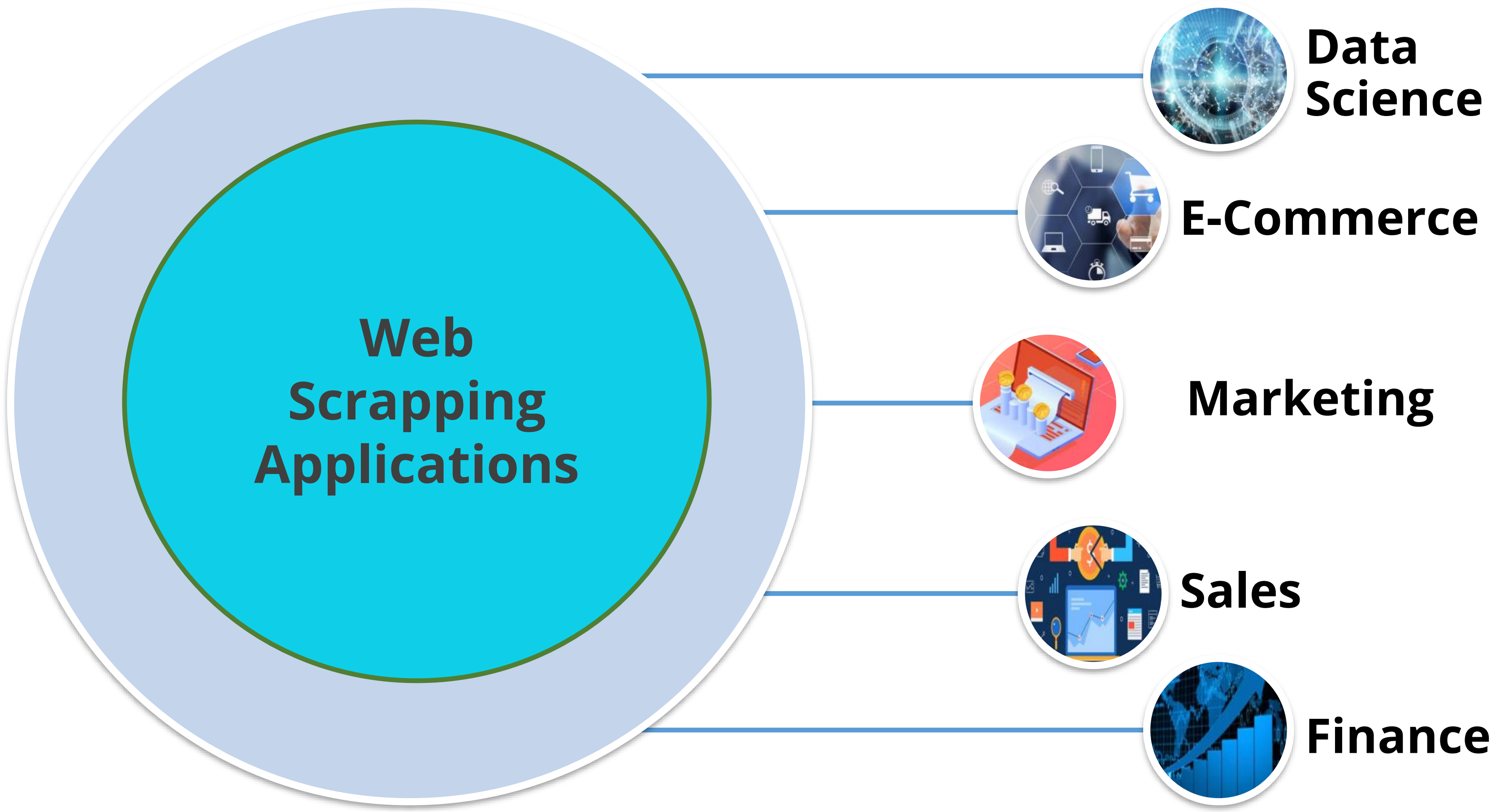
BASIC STEPS FOR WEB SCRAPING



Web Scraping Applications

Web Scraping plays a major role in data extraction that helps in business Improvements. At present, a website to any business is mandatory. This explains the importance of web scraping in information extraction

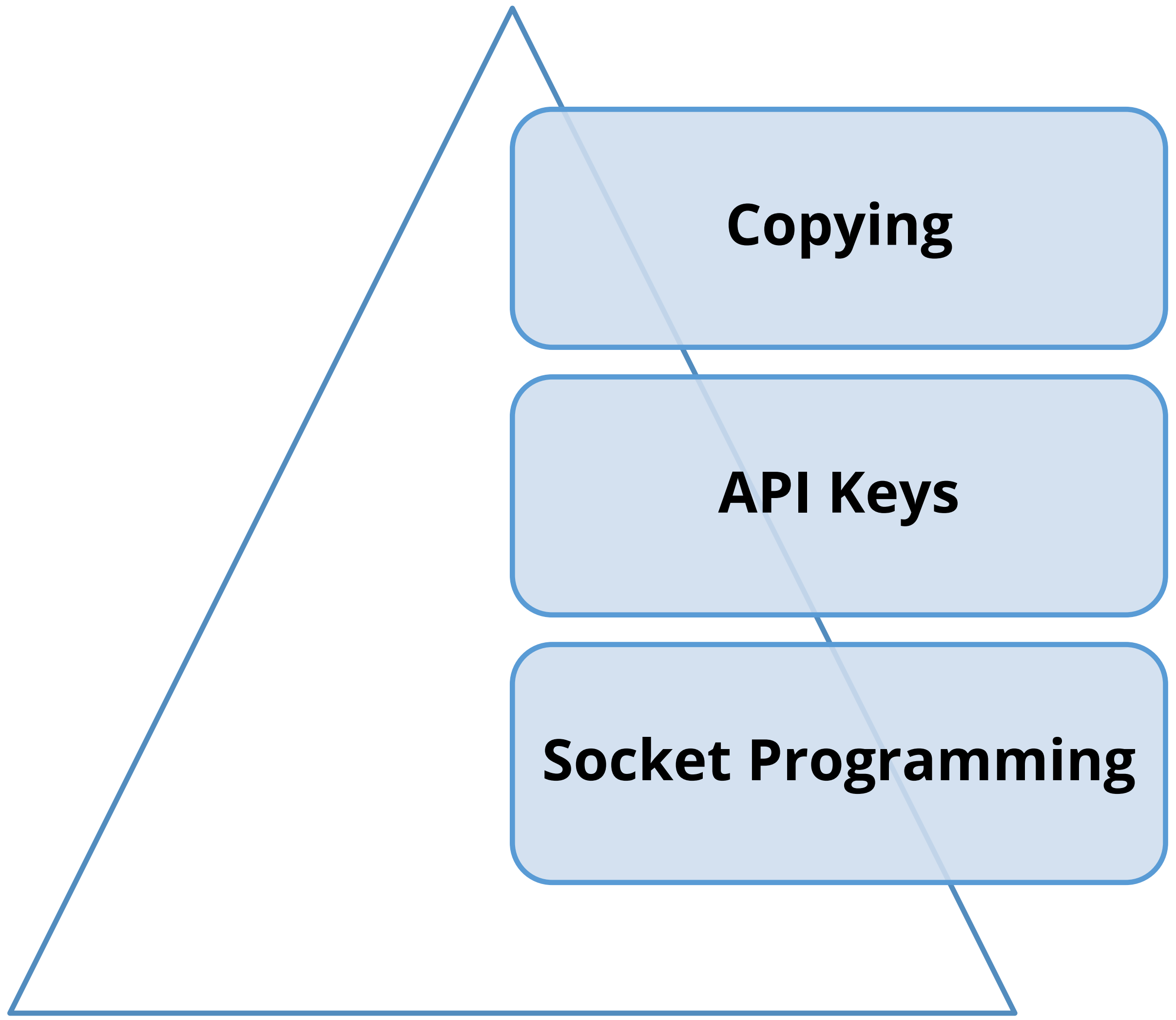
Let’s see some of the applications of web scraping.



Different Methods of Web Scraping

There are different methods to extract information from websites. Authentication is an important aspect for web scraping and every website has some restrictions for their content extraction.

Web scraping focuses on extracting data such as product costs, weather data, pollution check, criminal data, stock price movements etc,. in our local database for analysis.



Web Scraping in Python

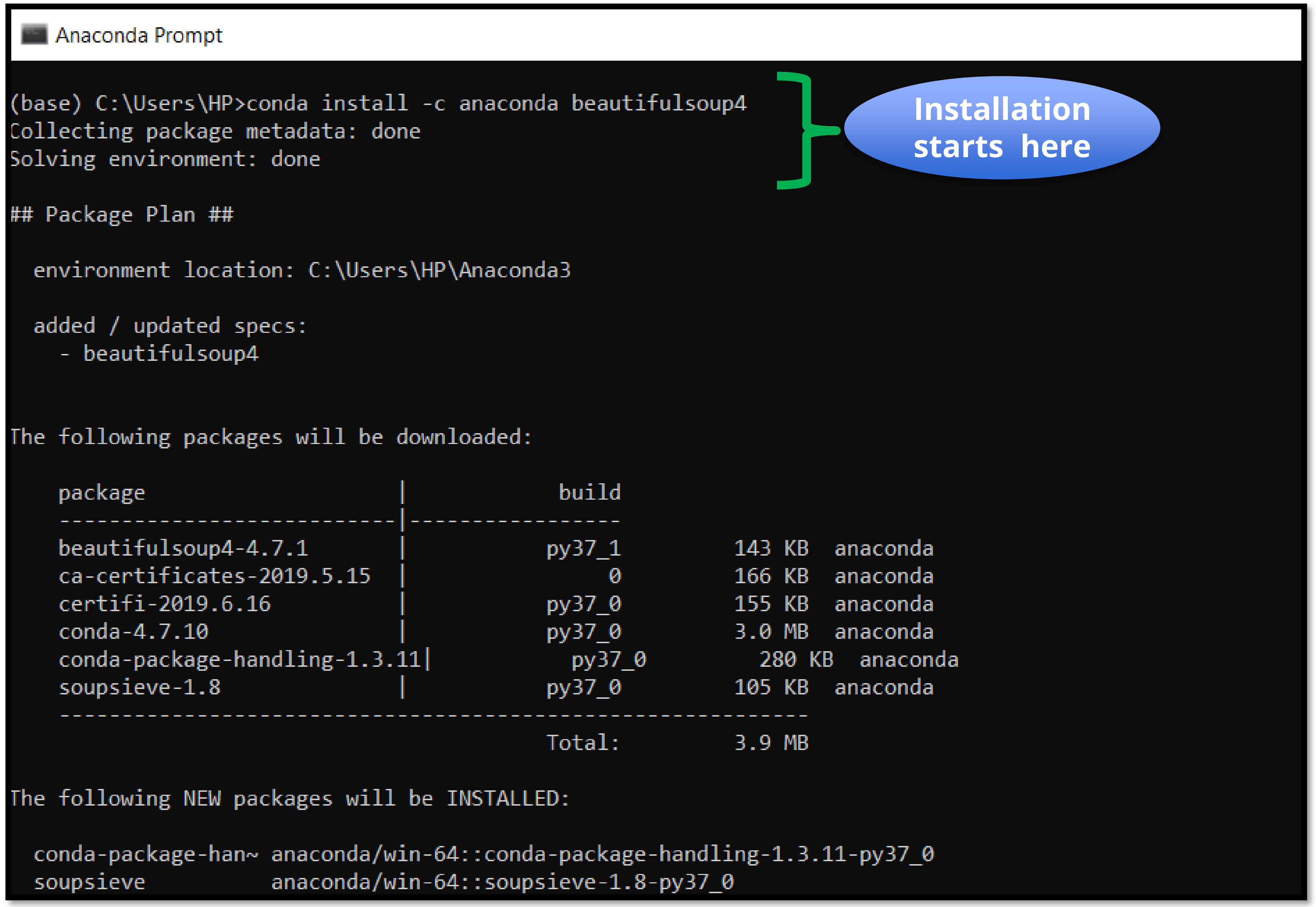
Python is one of the favorite languages for web scraping. Web scraping can be used for data analysis when we have to analyze information from a website

The important libraries in Python that assists us in web scraping are:

- **Beautiful Soup** — **Allows to scrape information from website in simple steps.**
- **Mechanize** — **Web scraping and automation tool**

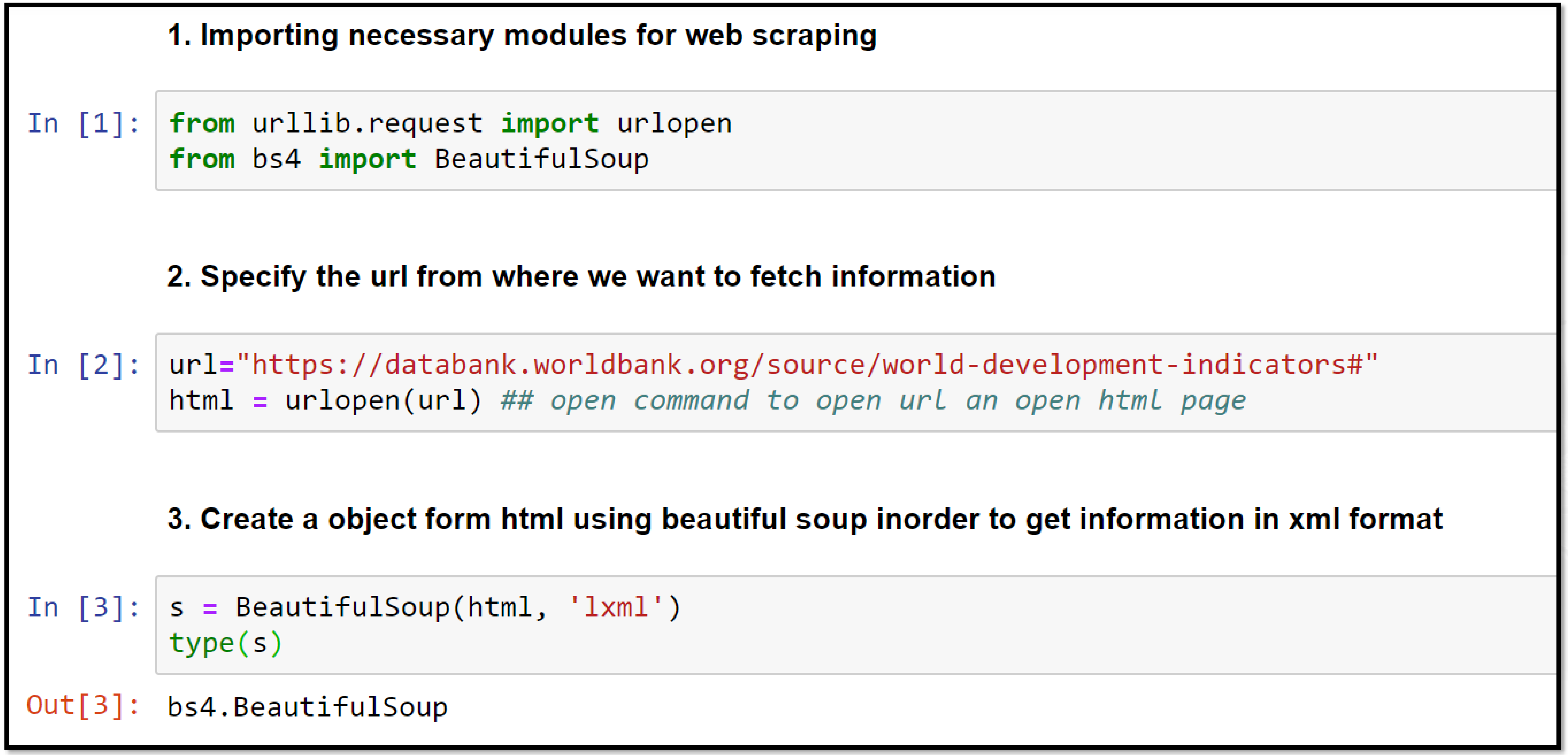
Beautiful Soup Installation Steps

Write conda install –c anaconda beautifulsoup4 in anaconda prompt



```
(base) C:\Users\HP>conda install -c anaconda beautifulsoup4
Collecting package metadata: done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\HP\Anaconda3

  added / updated specs:
    - beautifulsoup4


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    beautifulsoup4-4.7.1       |           py37_1         143 KB  anaconda
    ca-certificates-2019.5.15  |                0         166 KB  anaconda
    certifi-2019.6.16          |           py37_0         155 KB  anaconda
    conda-4.7.10               |           py37_0         3.0 MB  anaconda
    conda-package-handling-1.3.11|           py37_0         280 KB  anaconda
    soupsieve-1.8              |           py37_0         105 KB  anaconda
    ------------------------------------------------------------
                                           Total:         3.9 MB

The following NEW packages will be INSTALLED:

  conda-package-han~ anaconda/win-64::conda-package-handling-1.3.11-py37_0
  soupsieve          anaconda/win-64::soupsieve-1.8-py37_0
```

Demo: Web Scraping Using Beautiful Soup



1. Importing necessary modules for web scraping

```
In [1]: from urllib.request import urlopen
        from bs4 import BeautifulSoup
```

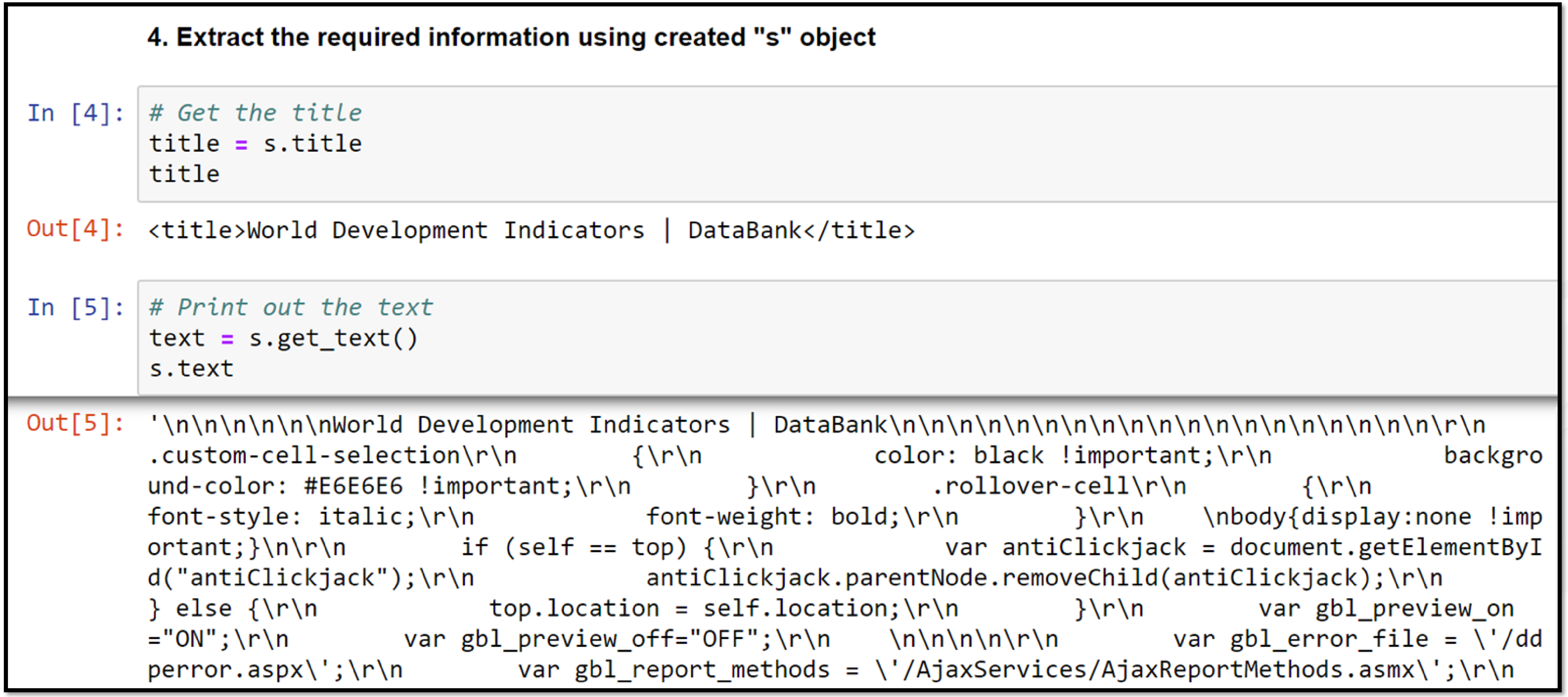
2. Specify the url from where we want to fetch information

```
In [2]: url="https://databank.worldbank.org/source/world-development-indicators#"
        html = urlopen(url) ## open command to open url an open html page
```

3. Create a object form html using beautiful soup inorder to get information in xml format

```
In [3]: s = BeautifulSoup(html, 'lxml')
        type(s)

Out[3]: bs4.BeautifulSoup
```

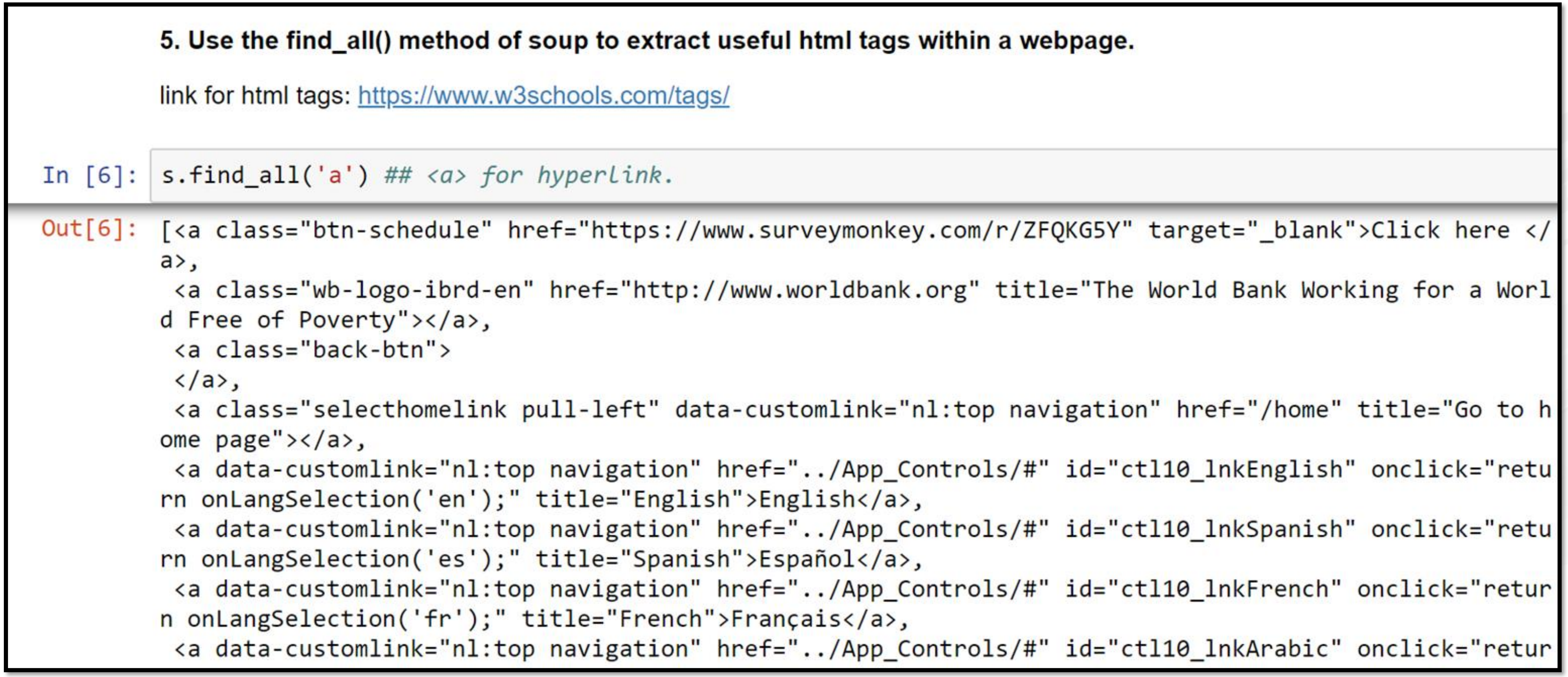
Demo: Web Scraping Using Beautiful Soup



4. Extract the required information using created "s" object

```
In [4]: # Get the title
        title = s.title
        title

Out[4]: <title>World Development Indicators | DataBank</title>

In [5]: # Print out the text
        text = s.get_text()
        s.text

Out[5]: '\n\n\n\n\n\nWorld Development Indicators | DataBank\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\r\n
        .custom-cell-selection\r\n        {\r\n            color: black !important;\r\n            backgro
        und-color: #E6E6E6 !important;\r\n        }\r\n        .rollover-cell\r\n        {\r\n
        font-style: italic;\r\n            font-weight: bold;\r\n        }\r\n    \nbody{display:none !imp
        ortant;}\n\r\n        if (self == top) {\r\n            var antiClickjack = document.getElementByI
        d("antiClickjack");\r\n            antiClickjack.parentNode.removeChild(antiClickjack);\r\n
        } else {\r\n            top.location = self.location;\r\n        }\r\n        var gbl_preview_on
        ="ON";\r\n        var gbl_preview_off="OFF";\r\n    \n\n\n\n\r\n        var gbl_error_file = \'/dd
        perror.aspx\';\r\n        var gbl_report_methods = \'/AjaxServices/AjaxReportMethods.asmx\';\r\n
```

Demo: Web Scraping Using Beautiful Soup

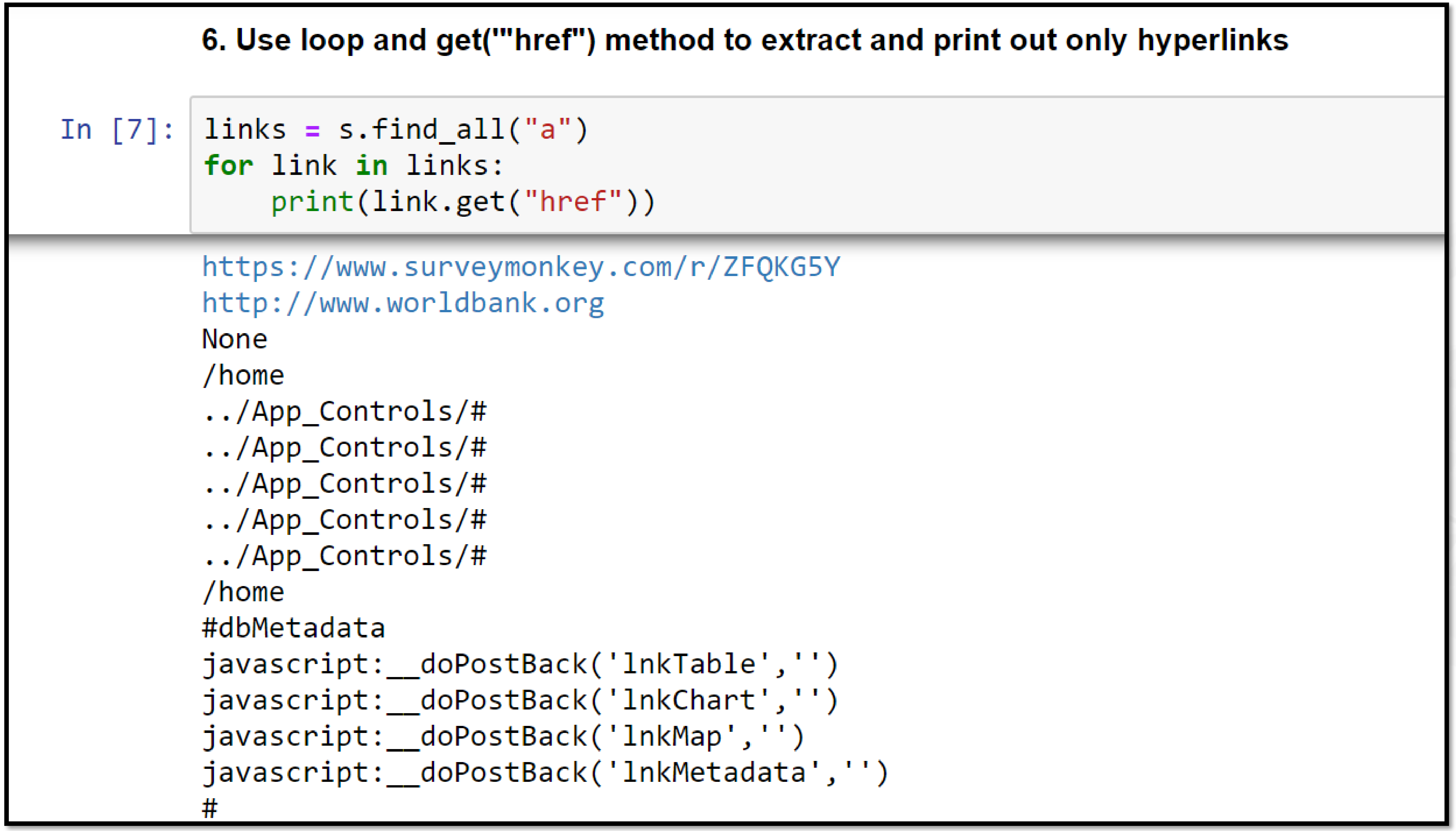


5. Use the find_all() method of soup to extract useful html tags within a webpage.

link for html tags: https://www.w3schools.com/tags/

```
In [6]: s.find_all('a') ## <a> for hyperlink.

Out[6]: [<a class="btn-schedule" href="https://www.surveymonkey.com/r/ZFQKG5Y" target="_blank">Click here </
        a>,
         <a class="wb-logo-ibrd-en" href="http://www.worldbank.org" title="The World Bank Working for a Worl
        d Free of Poverty"></a>,
         <a class="back-btn">
         </a>,
         <a class="selecthomelink pull-left" data-customlink="nl:top navigation" href="/home" title="Go to h
        ome page"></a>,
         <a data-customlink="nl:top navigation" href="../App_Controls/#" id="ctl10_lnkEnglish" onclick="retu
        rn onLangSelection('en');" title="English">English</a>,
         <a data-customlink="nl:top navigation" href="../App_Controls/#" id="ctl10_lnkSpanish" onclick="retu
        rn onLangSelection('es');" title="Spanish">Español</a>,
         <a data-customlink="nl:top navigation" href="../App_Controls/#" id="ctl10_lnkFrench" onclick="retur
        n onLangSelection('fr');" title="French">Français</a>,
         <a data-customlink="nl:top navigation" href="../App_Controls/#" id="ctl10_lnkArabic" onclick="retur
```

Demo: Web Scraping Using Beautiful Soup



6. Use loop and get("href") method to extract and print out only hyperlinks

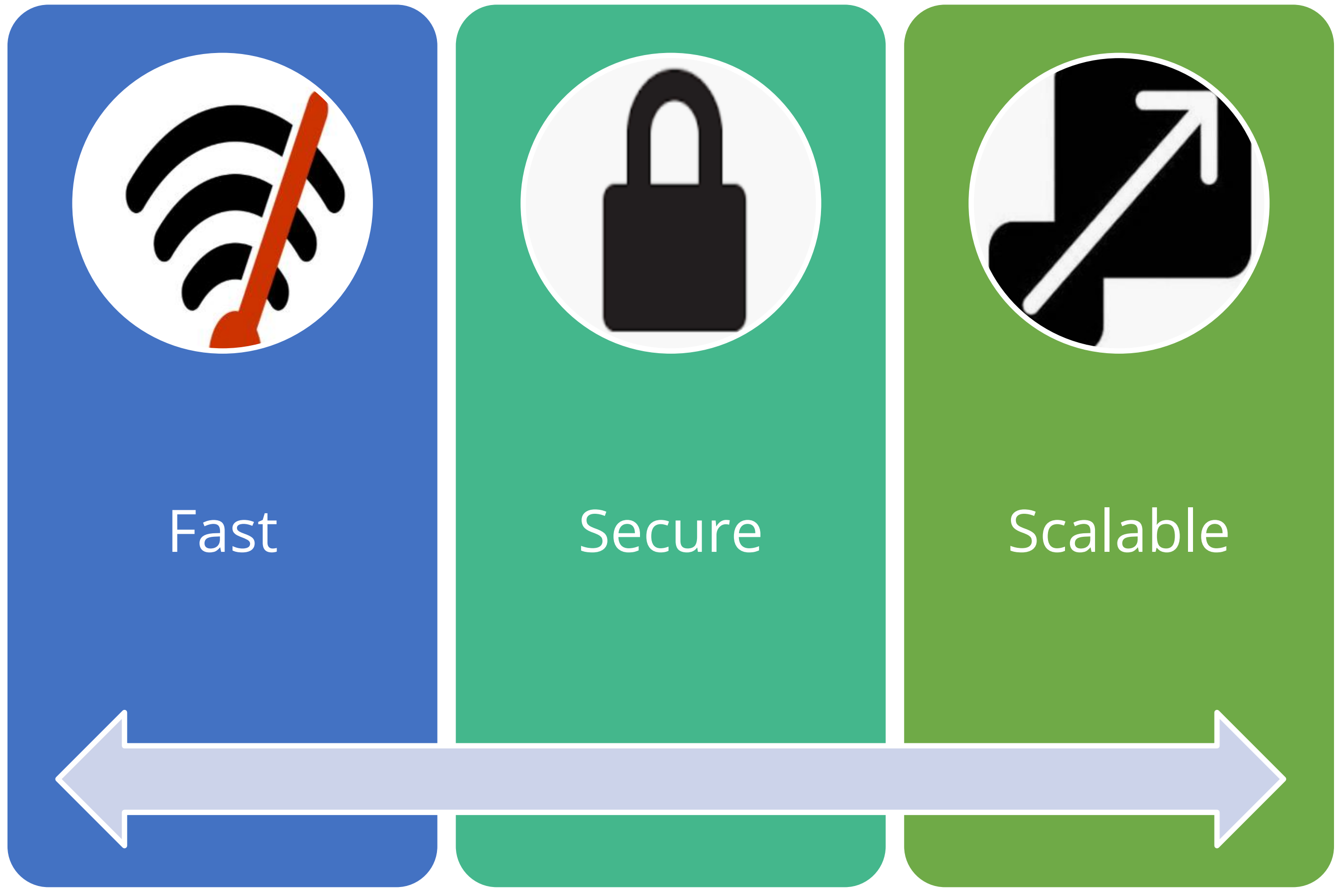
In [7]:
```
links = s.find_all("a")
for link in links:
    print(link.get("href"))
```

```
https://www.surveymonkey.com/r/ZFQKG5Y
http://www.worldbank.org
None
/home
../App_Controls/#
../App_Controls/#
../App_Controls/#
../App_Controls/#
../App_Controls/#
/home
#dbMetadata
javascript:__doPostBack('lnkTable','')
javascript:__doPostBack('lnkChart','')
javascript:__doPostBack('lnkMap','')
javascript:__doPostBack('lnkMetadata','')
#
```

Django

Django

Django is a high-level, popular Python framework for web development. Access to Django is free & open source. Django is open-source and web apps can be created with less code. As a framework, it is used for backend and front-end web development.

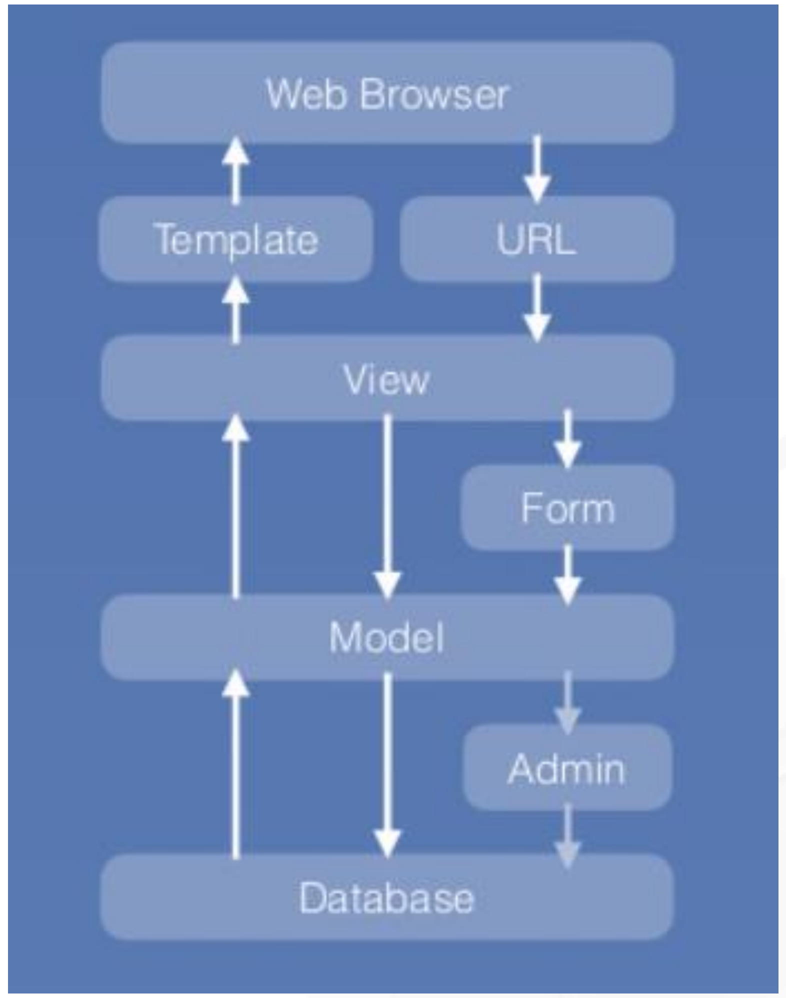


Companies Using Django

- Disqus
- Instagram
- YouTube
- Bitbucket
- Mozilla
- Spotify

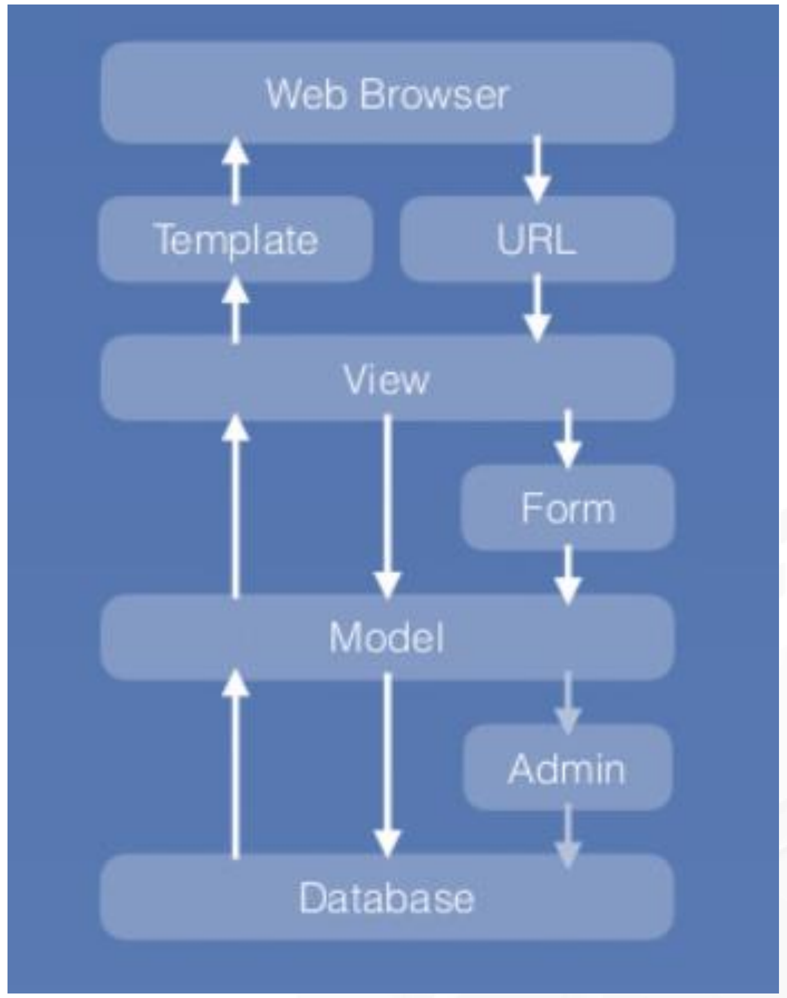
Important Attributes of Django

- A **web browser** is an interface for URL.
- A **URL** is the web address and the act of assigning functions to url is called mapping.
- Django **template** is simply a text document or a Python string marked-up using the Django template language. All the html files are stored in templates.
- Static folder is used to store other CSS files, java files , images etc.
- Functions related to web apps are written inside **view**. It also renders content to templates, puts information into model and gets information from databases.



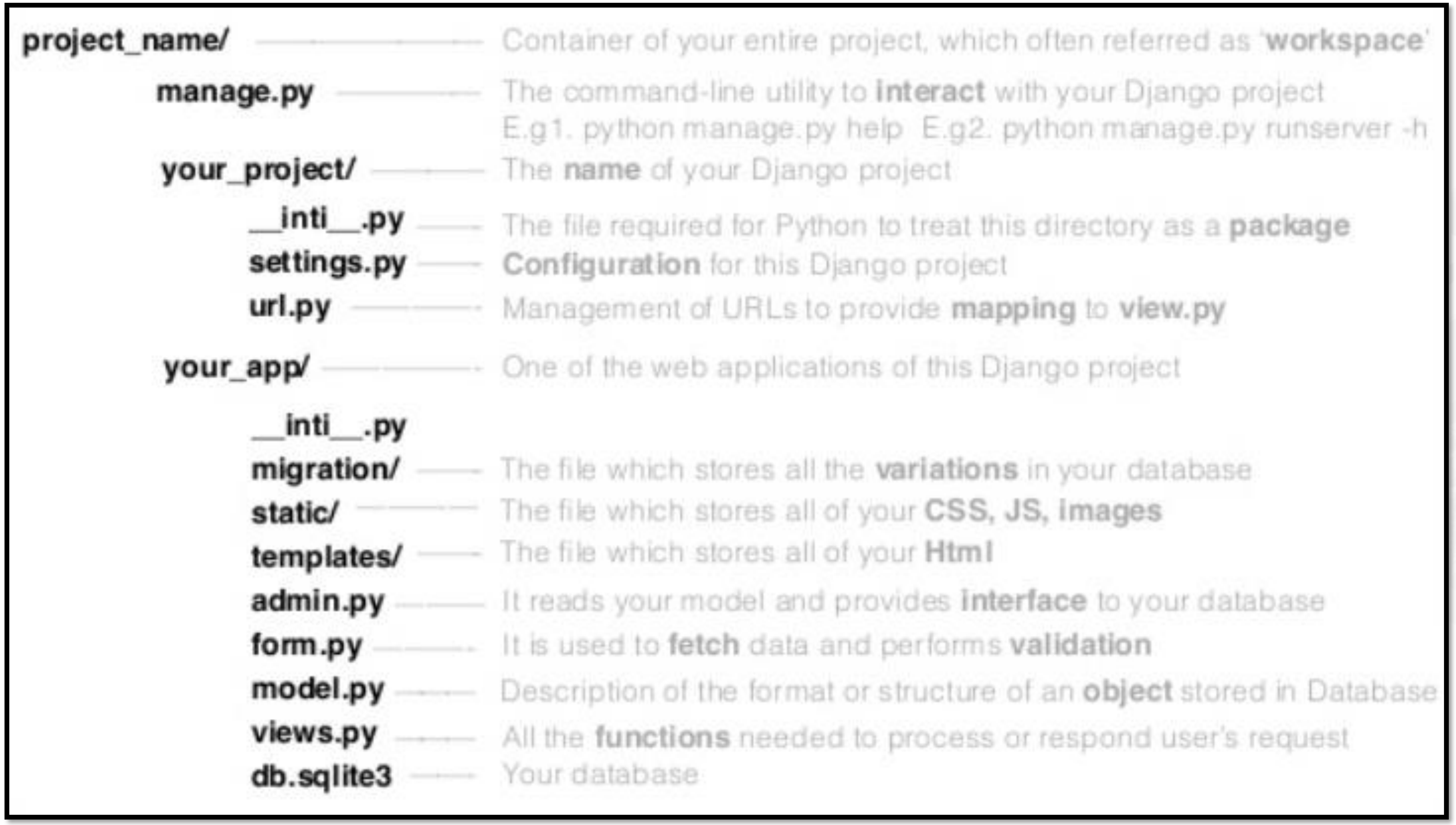
Important Attributes of Django

- **Form** fetches data from HTML form and helps connect to the model.
- **Model** is information about the object structure stored in a database. It contains essential fields and data behavior. Information can be directly edited in the database.
- **Django** automatically looks for an admin module in each application and imports it. Registration of object in model is done through admin, which is the mandatory first step for database management.
- **Database** is the collection of data at backend.



Technical Architecture

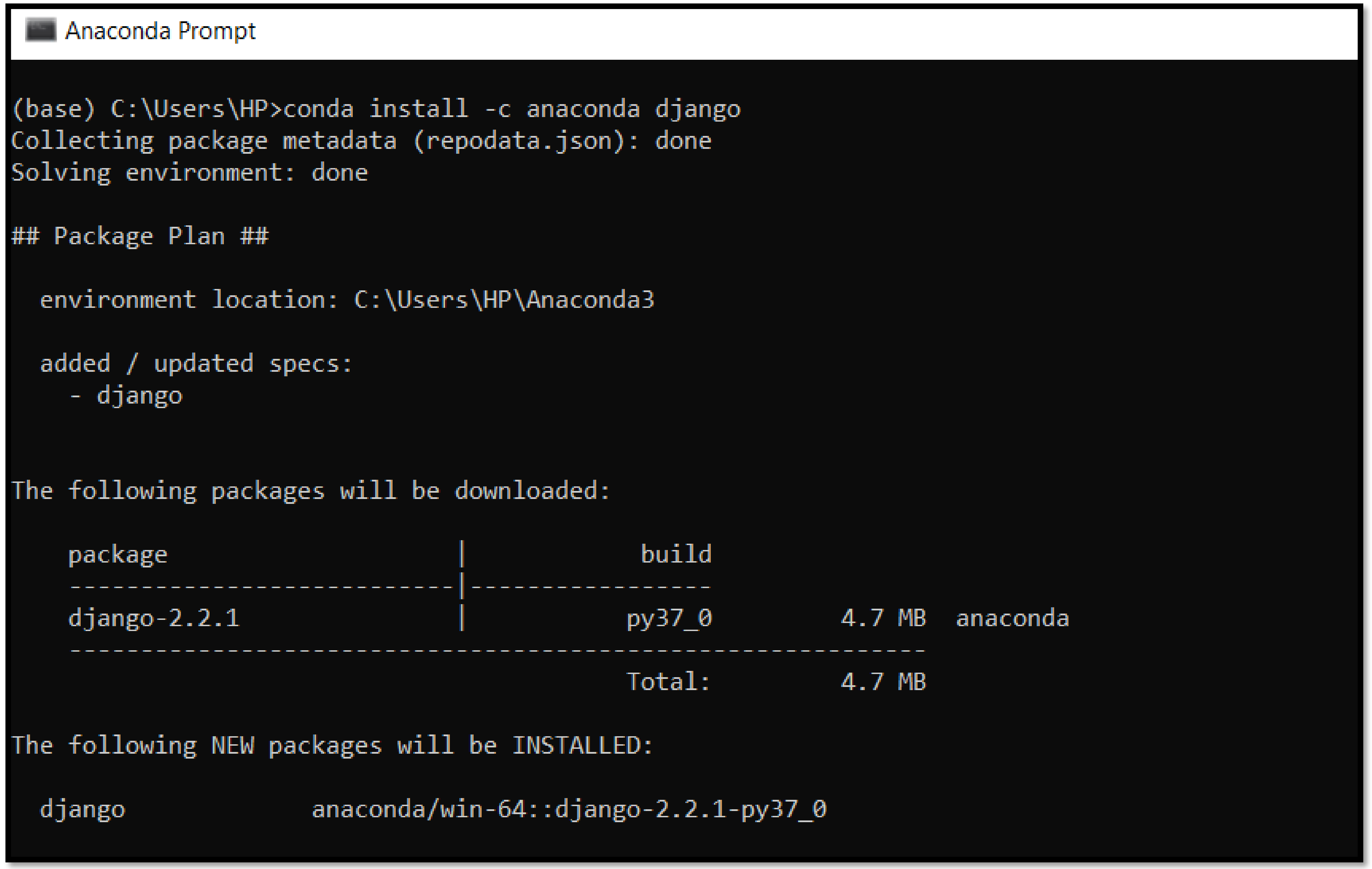
This is the technical architecture followed in Django:



project_name/	Container of your entire project, which often referred as '**workspace**'
manage.py	The command-line utility to **interact** with your Django project E.g1. python manage.py help E.g2. python manage.py runserver -h
your_project/	The **name** of your Django project
__inti__.py	The file required for Python to treat this directory as a **package**
settings.py	**Configuration** for this Django project
url.py	Management of URLs to provide **mapping** to **view.py**
your_app/	One of the web applications of this Django project
__inti__.py	
migration/	The file which stores all the **variations** in your database
static/	The file which stores all of your **CSS, JS, images**
templates/	The file which stores all of your **Html**
admin.py	It reads your model and provides **interface** to your database
form.py	It is used to **fetch** data and performs **validation**
model.py	Description of the format or structure of an **object** stored in Database
views.py	All the **functions** needed to process or respond user's request
db.sqlite3	Your database

Django Installation Steps

Write conda install -c anaconda django in anaconda prompt.



```
(base) C:\Users\HP>conda install -c anaconda django
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\HP\Anaconda3

  added / updated specs:
    - django


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    django-2.2.1               |           py37_0         4.7 MB  anaconda
    ------------------------------------------------------------
                                           Total:         4.7 MB

The following NEW packages will be INSTALLED:

  django             anaconda/win-64::django-2.2.1-py37_0
```

Web Scraping

Duration: 20 min.

Objective: Write a program using Python to demonstrate web scraping.

Steps to demonstrate web scraping:

1. Open Jupyter Notebook
2. Click on File New Notebook
3. Select Python (version 3)
4. Write your program
5. Save your program
6. Click on Run to execute program

Knowledge Check

Knowledge Check
1

Which of the following is a web scraping library in Python?

a. Beautiful Soup

b. Pandas

c. Numpy

d. None of the above

Knowledge Check
1

Which of the following is a web scraping library in Python?

a. Beautiful Soup

b. Pandas

c. Numpy

d. None of the above

The correct answer is **a**

Beautiful Soup is for web scraping, Pandas for data analysis, and Numpy for numerical analysis.

Knowledge Check

2

Data extraction is the most important aspect of web scraping.

a. False

b. True

Knowledge Check

2

Data extraction is the most important aspect of web scraping.

a. False

b. True

The correct answer is **b**

Web scraping means extracting information from a URL. So, data extraction is the most important aspect of web scraping.

Knowledge Check

3

What are the features available in Django web framework?

a. Web apps

b. Templates

c. Both a & b

d. None of the above

Knowledge Check

3

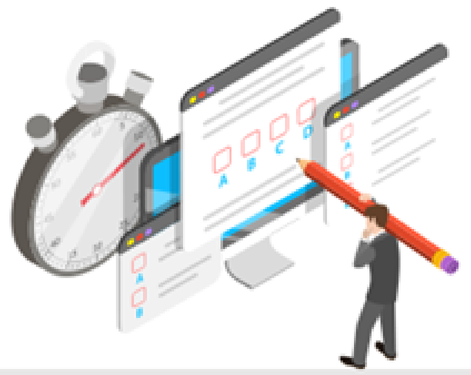
What are the features available in Django web framework?

a. Web apps

b. Templates

c. Both a & b

d. None of the above



The correct answer is **d**

Django framework is the simplest way to create web apps and templates using Python.

Knowledge Check

4

In Python, a=BeautifulSoup() is an expression, where a is ___________

a. A constructor

b. An object

c. A class

d. A value returning function

Knowledge Check

4

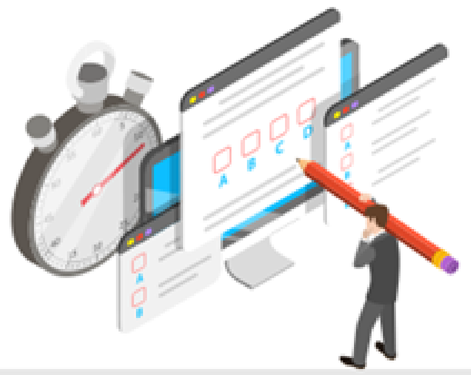
In Python, a=BeautifulSoup() is an expression, where a is ___________

a. A constructor

b. An object

c. A class

d. A value returning function



The correct answer is **b**

a is an object created using BeautifulSoup().

Knowledge Check

5

What is the role of render_to_response method in Django?

a. Generating web response

b. Rendering data from web

c. Rendering an HTML response

d. None of above

Knowledge Check

5

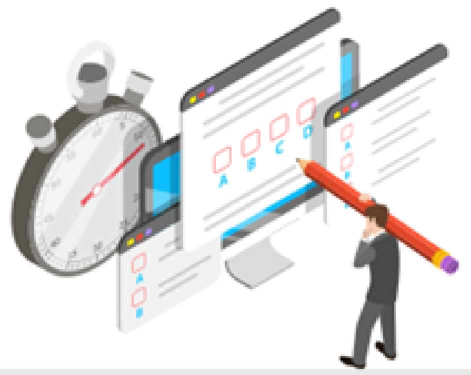
What is the role of render_to_response method in Django?

a. Generating web response

b. Rendering data from web

c. Rendering an HTML response

d. None of above



The correct answer is **c**

In Django, render_to_response method is used to easily render an HTML response.

Key Takeaways

- A Shell script is a computer program designed to be run by the Unix shell.
- Web scraping is a method of extracting information from a URL.
- Beautiful Soup is one of the simplest and most useful web scraping libraries in Python.
- Django is a high-level web framework used for web development in Python.

Salary Appraisal of Employees

Duration: 45 min.

Problem Statement: Create an Interface & Formulate Set of rules for Salary Appraisal of employees.

COURSE-END PROJECT