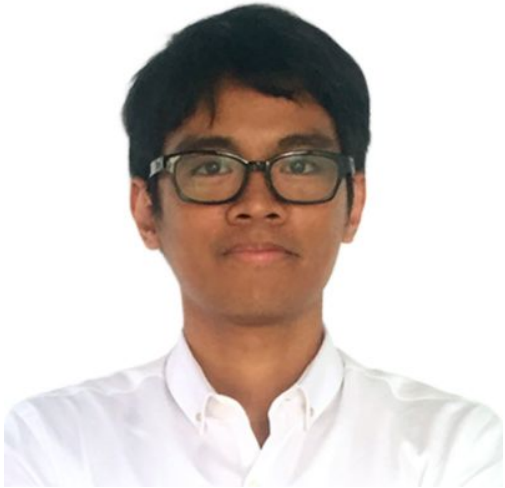
Kaizend: Web Scraping & Automation

April 5, 2020

Who am I?



Joshua “ARVS” Lat

➢ **Chief Technology Officer** of CBO, Insites, and Jepto

➢ **AWS Machine Learning Hero**

Goals

- Extract data from webpages
- Learn how to properly use delays to prevent the web scraping jobs from getting blocked or throttled
- Maintain an acceptable level of code quality of the web scraping script(s) as the complexity of the requirement increases

Process

Step # 1: Identify the **target URL**

Step # 2: Download the HTML Page and store the **HTML string** in a variable

Step # 3: Parse the HTML string and **extract the relevant values of the target elements**

DELAY

```python
from time import sleep

def delay(seconds):
    print(f"Sleeping for {seconds} second(s)")
    sleep(seconds)


def main():
    print('a')
    delay(seconds=3)
    print('b')


if __name__ == "__main__":
    main()
```

```
ubuntu:~/environment/labs $ python3 1.py
a
Sleeping for 3 second(s)
b
```

CHALLENGE # 1

GOAL:

- Loop 5 times
- For each iteration of the loop, sleep for 2 seconds

```
ubuntu:~/environment/labs $ python3 2.py
0
Sleeping for 2 second(s)
1
Sleeping for 2 second(s)
2
Sleeping for 2 second(s)
3
Sleeping for 2 second(s)
4
Sleeping for 2 second(s)
```

EXTRA CHALLENGE

```python
from time import sleep
from functools import wraps


def delay(seconds):
    # insert code here
    def inner_function(function):
        # insert code here
        return function

    return inner_function


@delay(seconds=2)
def say_something(word):
    print(word)


def main():
    say_something("hello")


if __name__ == "__main__":
    main()
```

```
ubuntu:~/environment/medium $ python3 1.py
Sleeping for 2 second(s)
[START]
hello
[END]
```

RANDOM NUMBER

```python
import random

def get_random_number():
    return random.randint(1, 5)

def main():
    print(get_random_number())
    print(get_random_number())
    print(get_random_number())
    print(get_random_number())
    print(get_random_number())


if __name__ == "__main__":
    main()
```

```
ubuntu:~/environment/labs $ python3 3.py
1
1
4
4
2
```

CHALLENGE # 2

GOAL:

- Loop 5 times
- For each iteration of the loop, sleep for a random number of seconds

```
ubuntu:~/environment/labs $ python3 4.py
1
Sleeping for 2 second(s)
2
Sleeping for 2 second(s)
3
Sleeping for 3 second(s)
4
Sleeping for 1 second(s)
```

IPYTHON EMBED

```python
from IPython import embed


def main():
    a = 5
    b = 3

    embed()

    print(a)
    print(b)



if __name__ == "__main__":
    main()
```

```
ubuntu:~/environment/labs $ python3 5.py
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: a
Out[1]: 5

In [2]: b
Out[2]: 3

In [3]:
```

EXTRA CHALLENGE

```python
def debug_input_and_output(func):
    # Insert code here

    return func


@debug_input_and_output
def say_something(word):
    print(word)


def main():
    say_something("hello")


if __name__ == "__main__":
    main()
```

```
ubuntu:~/environment/medium $ python3 2.py
[INPUT] ARGS: ('hello',)
[INPUT] KWARGS: {}
hello
[OUTPUT]: None
```

BEAUTIFUL SOUP

```python
from bs4 import BeautifulSoup
from IPython import embed


def generate_html():
    return """
        <html>
            <head></head>
            <body>
                <div id="target">Hello World</div>
            </body>
        </html>
    """


def main():
    html_doc = generate_html()
    soup = BeautifulSoup(html_doc, 'html.parser')
    target = soup.find(id="target")

    print(target.get_text())



if __name__ == "__main__":
    main()
```

```
ubuntu:~/environment/labs $ python3 6.py
Hello World
```

```python
from bs4 import BeautifulSoup
from IPython import embed


def generate_html():
    return """
        <html>
            <head></head>
            <body>
                <div>A</div>
                <div>B</div>
                <div>C</div>
                <div>D</div>
            </body>
        </html>
    """


def main():
    html_doc = generate_html()
    soup = BeautifulSoup(html_doc, 'html.parser')

    div_elements = soup.find_all('div')

    for div_element in div_elements:
        print(div_element.get_text())


if __name__ == "__main__":
    main()
```

```
ubuntu:~/environment/labs $ python3 7.py
A
B
C
D
```

CHALLENGE # 3

```
from bs4 import BeautifulSoup
from IPython import embed


def generate_html():
    return """
        <html>
            <head></head>
            <body>
                <a href="/a.html">A</a>
                <a href="/b.html">B</a>
                <a href="/c.html">C</a>
                <a href="/d.html">D</a>
            </body>
        </html>
    """


def main():
    html_doc = generate_html()
    embed()


if __name__ == "__main__":
    main()
```

EXTRA CHALLENGE

```
from bs4 import BeautifulSoup
from IPython import embed
import re


def generate_html():
    return """
        <html>
            <head></head>
            <body>
                <a href="/a.html">A</a>
                <a href="/b.html">B</a>
                <a href="/c.html">C</a>
                <a href="/d.html">D</a>

                <script>
                    var hello = "yoh";
                    alert(hello);
                </script>
            </body>
        </html>
    """


def main():
    html_doc = generate_html()
    soup = BeautifulSoup(html_doc, 'html.parser')


if __name__ == "__main__":
    main()
```

REQUESTS

```python
import requests

from IPython import embed


BASE_URL = "http://sample-target-bucket-with-html-pages.s3-website-ap-southeast-1.amazonaws.com"


def extract_html_content(target_url):
    response = requests.get(target_url)
    return response.text


def main():
    target_page = BASE_URL + "/index.html"
    print(target_page)

    html_content = extract_html_content(target_page)

    print(html_content)

    embed()


if __name__ == "__main__":
    main()
```

MANAGING COMPLEXITY

Best Practices

- Making debugging easier using verbose logging and usage of tools such as IPython.embed
- Set a limit to the number of lines for each function (e.g., 15 lines)
- Avoid the usage of try-catch statements
- Add a delay when downloading / extracting pages and files from online sources
- Make function and variable names clear and consistent

```python
import random

from IPython import embed
from time import sleep


BASE_URL = "http://sample-target-bucket-with-html-pages.s3-website-ap-southeast-1.amazonaws.com"

PAGES = ["1.html",
         "2.html",
         "3.html",
         "4.html",
         "5.html"]


def delay(seconds):
    print(f"Sleeping for {seconds} second(s)")
    sleep(seconds)


def get_random_number():
    return random.randint(1, 5)


def main():
    for page in PAGES:
        delay(get_random_number())
        print(BASE_URL + "/" + page)


if __name__ == "__main__":
    main()
```

CHALLENGE # 4

```python
import random
import requests

from IPython import embed
from time import sleep
from bs4 import BeautifulSoup


BASE_URL = "http://sample-target-bucket-with-html-pages.s3-website-ap-southeast-1.amazonaws.com"

PAGES = ["group1/1.html", "group1/2.html", "group1/3.html", "group1/4.html", "group1/5.html"]


def delay(seconds):
    print(f"Sleeping for {seconds} second(s)")
    sleep(seconds)


def get_random_number():
    return random.randint(1, 3)


def extract_html_content(target_url):
    return ""


def extract_target_value_from_page(html_string):
    return ""


def main():
    for page in PAGES:
        target_page = BASE_URL + "/" + page
        print(target_page)


if __name__ == "__main__":
    main()
```

EXTRA CHALLENGE

```
ubuntu:~/environment/medium $ python3 4.py
[START: extract_html_content]
Downloading HTML content of http://sample-target-bucket-with-html-pages.s3-website-ap-southeast-1.amazonaws.com/group1/1.html
[END: extract_html_content]]
<html>
    <body>
        <div id="target">Apple</div>
    </body>
</html>
[START: extract_target_value_from_page]
[END: extract_target_value_from_page]]
Apple
[START: delay]
Sleeping for 3 second(s)
[END: delay]]
[START: extract_html_content]
Downloading HTML content of http://sample-target-bucket-with-html-pages.s3-website-ap-southeast-1.amazonaws.com/group1/2.html
[END: extract_html_content]]
<html>
    <body>
        <div id="target">Banana</div>
    </body>
</html>
[START: extract_target_value_from_page]
[END: extract_target_value_from_page]]
Banana
[START: delay]
Sleeping for 3 second(s)
[END: delay]]
[START: extract_html_content]
```

CHALLENGE # 6

LINK: https://sample-target-bucket-with-html-pages.s3-ap-southeast-1.amazonaws.com/group2/index.html

```
<html>
    <body>
        <a href="/group2/oipuyouiyouo.html">1</a>
        <a href="/group2/qwerqasdfadsfa.html">2</a>
        <a href="/group2/zxvzcxvzxasdqasf.html">3</a>
    </body>
</html>
```

LINK: https://sample-target-bucket-with-html-pages.s3-ap-southeast-1.amazonaws.com/group2/oipuyouiyouo.html

```
<html>
    <body>
        <div id="target">42</div>
    </body>
</html>
```

FINAL CHALLENGE

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf` . See their respective man pages for detailed information.
- The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file apart of those located in `/var/www`, **public_html** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under /var/www. This is different to previous releases which provides better security out of the box.

Reporting Problems

Please use the `ubuntu-bug` tool to report bugs in the Apache2 package with Ubuntu. However, check **href="https://bugs.launchpad.net/ubuntu/+source/apache2" rel="nofollow">existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web

LINK: https://sample-target-bucket-with-html-pages.s3-ap-southeast-1.amazonaws.com/group3/target.html

FINAL CHALLENGE

```
ubuntu:~/environment/labs $ python3 final-challenge.py
Downloading HTML content of http://sample-target-bucket-with-html-pages.s3-website-ap-southeast-1.amazonaws.co
m/group3/target.html
apache2.conf is the main configuration file. It puts the pieces together by including all remaining configurat
ion files when starting up the web server.

ports.conf is always included from the main configuration file. It is used to determine the listening ports fo
r incoming connections, and this file can be customized anytime.

Configuration files in the mods-enabled/, conf-enabled/ and sites-enabled/ directories contain particular conf
iguration snippets which manage modules, global configuration fragments, or virtual host configurations, respe
ctively.

They are activated by symlinking available configuration files from their respective *-available/ counterparts
. These should be managed by using our helpers a2enmod, a2dismod, a2ensite, a2dissite, and a2enconf, a2disconf
 . See their respective man pages for detailed information.

The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 n
eeds to be started/stopped with /etc/init.d/apache2 or apache2ctl. Calling /usr/bin/apache2 directly will not
work with the default configuration.
```

Kaizend: Web Scraping & Automation

THE END