Experiment 2 - Parsing using Python.

Roll No.	37
Name	Mikil Lalwani
Class	D20 B
Subject	Data Science Lab
LO Mapped	LO2: Explore use cases of Cognitive Computing.

Aim: To Build a Simple Parsing using Python.

Parsing:

Document parsing is the process of recognizing/examining data in a document and extracting useful information from it. For instance, data from PDF and Word documents can be extracted using document parser APIs and stored in a JSON file. How so? Thanks to Optical Character Recognition (OCR) and Named Entity Recognition (NER) technologies, Document Parser APIs are built in a way to extract textual content first, then locate and classify named entities into categories such as names, locations, quantities, percentages, etc.
Document Parsing can be found in various industries, to automate manual processes and improve data entry efficiency or to help with the digitalization of companies and eliminate paperwork for good.

Python Library Function Used:

Selenium is a popular open-source framework for automating web browsers. It provides a Python library that allows developers to control and interact with web browsers programmatically. Selenium is widely used for web testing, web scraping, and automating repetitive web-related tasks. Here are some key features and use cases of Selenium:

1. Web Testing: Selenium is commonly used for automated testing of web applications. Testers can write scripts to simulate user interactions with a web application, such as clicking buttons, filling out forms, and verifying web page content. This helps ensure that web applications function correctly and meet quality standards.

2. Web Scraping: Selenium can be used to scrape data from websites by automating the process of navigating web pages, interacting with elements, and extracting information. It is especially useful when websites use JavaScript to load dynamic content, as Selenium can wait for the page to fully load before extracting data.

3. Browser Automation: Selenium allows you to automate repetitive tasks in web browsers. For example, you can create scripts to perform tasks like filling out web forms, downloading files, or navigating through a series of web pages automatically.

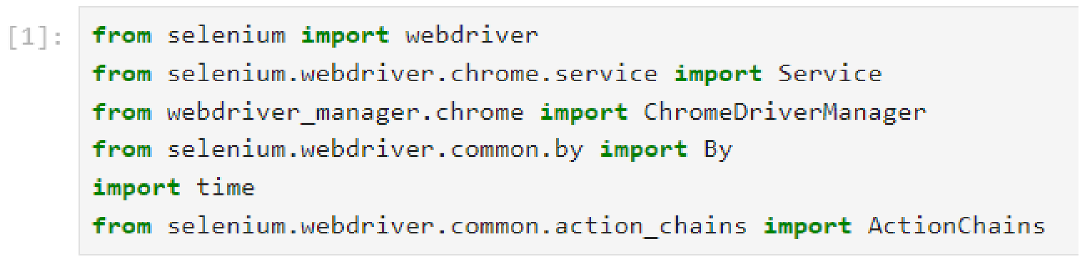
4. Cross-browser Testing: Selenium supports multiple web browsers, including Chrome, Firefox, Safari, and Internet Explorer. This makes it suitable for testing web applications across different browsers and ensuring cross-browser compatibility.

5. Integration with Testing Frameworks: Selenium can be integrated with various testing frameworks like pytest and unittest, allowing you to incorporate web testing into your automated testing workflow.

6. Headless Browsing: Selenium can run browsers in "headless" mode, meaning that the browser runs without a graphical user interface. This is useful for running tests or web scraping tasks in a background environment without displaying the browser window.

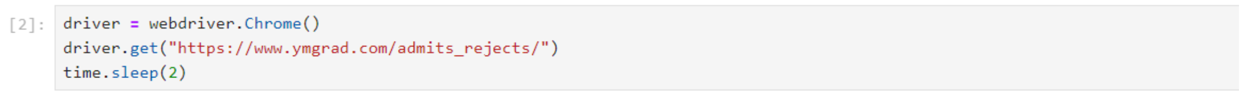
<u>Code and Observation</u>:

1. Import libraries.

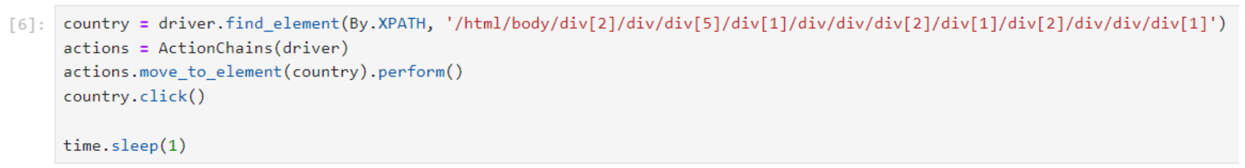


```
[1]: from selenium import webdriver
     from selenium.webdriver.chrome.service import Service
     from webdriver_manager.chrome import ChromeDriverManager
     from selenium.webdriver.common.by import By
     import time
     from selenium.webdriver.common.action_chains import ActionChains
```

2. Navigate to the webpage for scraping.

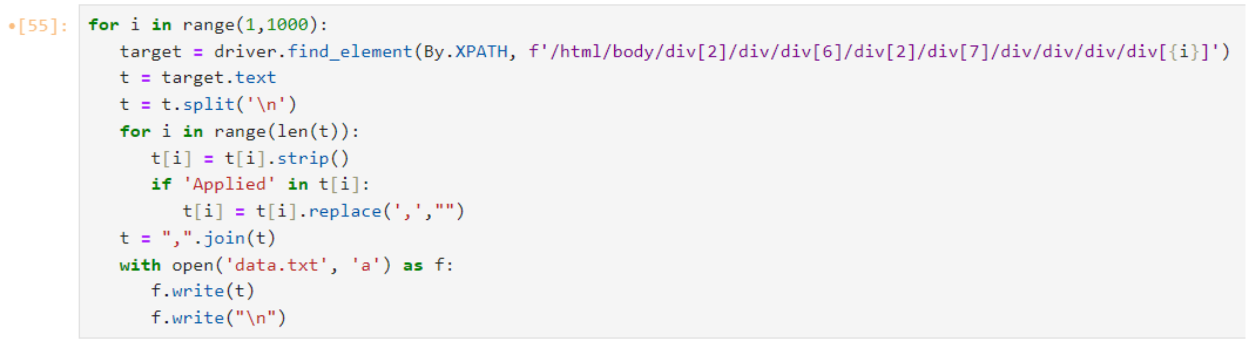


```
[2]: driver = webdriver.Chrome()
     driver.get("https://www.ymgrad.com/admits_rejects/")
     time.sleep(2)
```



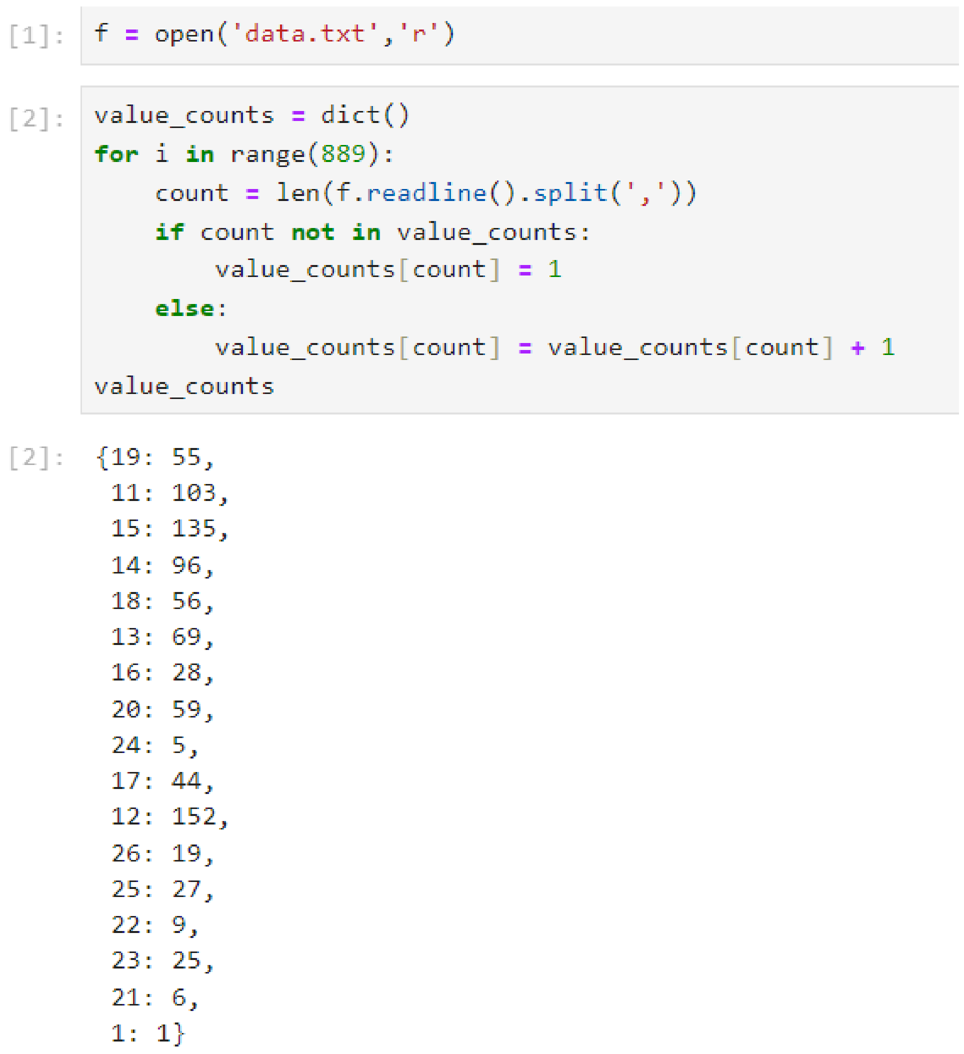
```
[6]: country = driver.find_element(By.XPATH, '/html/body/div[2]/div/div[5]/div[1]/div/div/div[2]/div[1]/div[2]/div/div/div[1]')
     actions = ActionChains(driver)
     actions.move_to_element(country).perform()
     country.click()

     time.sleep(1)
```

3. Scrape the target webpage.



```
•[55]: for i in range(1,1000):
           target = driver.find_element(By.XPATH, f'/html/body/div[2]/div/div[6]/div[2]/div[7]/div/div/div/div[{i}]')
           t = target.text
           t = t.split('\n')
           for i in range(len(t)):
               t[i] = t[i].strip()
               if 'Applied' in t[i]:
                   t[i] = t[i].replace(',',"")
           t = ",".join(t)
           with open('data.txt', 'a') as f:
               f.write(t)
               f.write("\n")
```

4. Now read the scraped data for segregation.

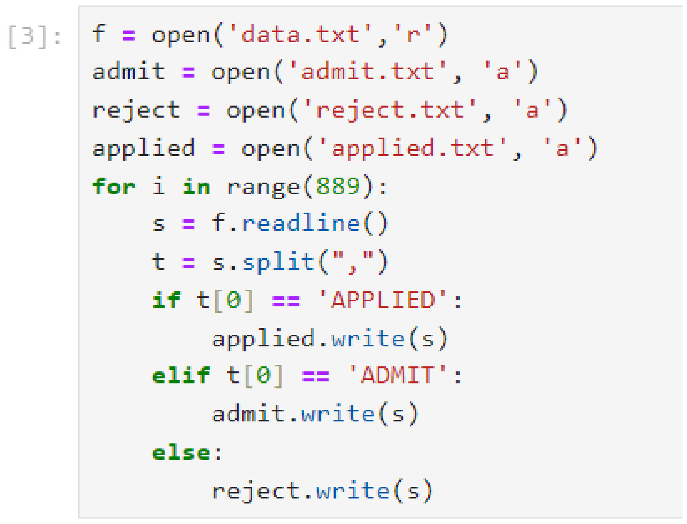


```
[1]: f = open('data.txt','r')
```

```
[2]: value_counts = dict()
     for i in range(889):
         count = len(f.readline().split(','))
         if count not in value_counts:
             value_counts[count] = 1
         else:
             value_counts[count] = value_counts[count] + 1
     value_counts
```

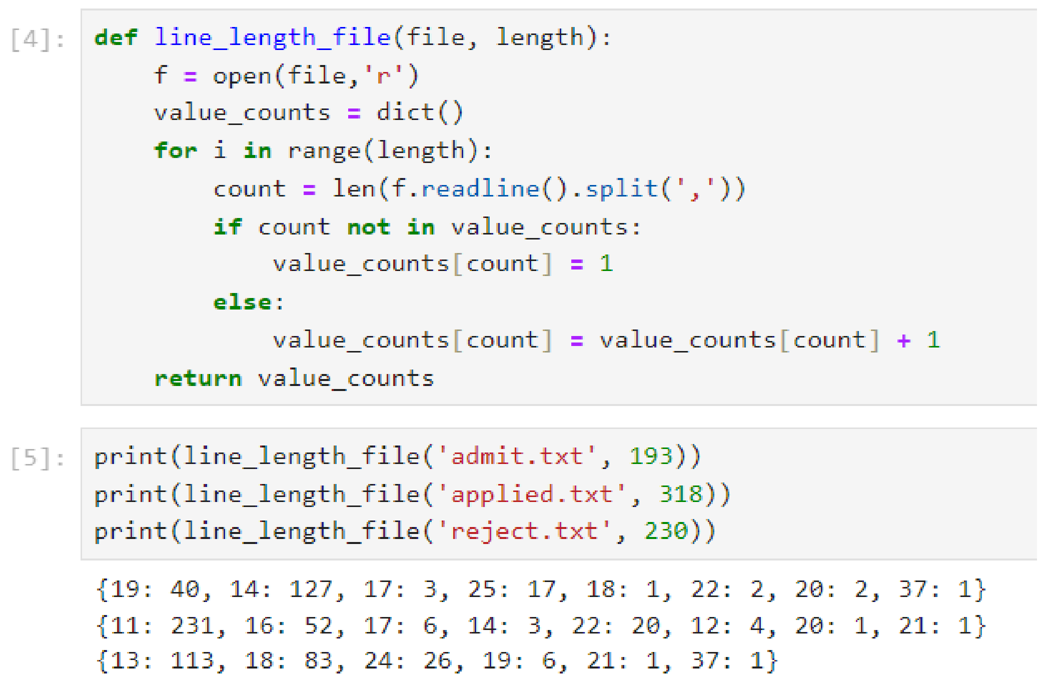
```
[2]: {19: 55,
      11: 103,
      15: 135,
      14: 96,
      18: 56,
      13: 69,
      16: 28,
      20: 59,
      24: 5,
      17: 44,
      12: 152,
      26: 19,
      25: 27,
      22: 9,
      23: 25,
      21: 6,
      1: 1}
```

5. Now segregate the data.



```
[3]: f = open('data.txt','r')
     admit = open('admit.txt', 'a')
     reject = open('reject.txt', 'a')
     applied = open('applied.txt', 'a')
     for i in range(889):
         s = f.readline()
         t = s.split(",")
         if t[0] == 'APPLIED':
             applied.write(s)
         elif t[0] == 'ADMIT':
             admit.write(s)
         else:
             reject.write(s)
```

6. Now parse the data.



[4]:
```python
def line_length_file(file, length):
    f = open(file,'r')
    value_counts = dict()
    for i in range(length):
        count = len(f.readline().split(','))
        if count not in value_counts:
            value_counts[count] = 1
        else:
            value_counts[count] = value_counts[count] + 1
    return value_counts
```

[5]:
```python
print(line_length_file('admit.txt', 193))
print(line_length_file('applied.txt', 318))
print(line_length_file('reject.txt', 230))
```

```
{19: 40, 14: 127, 17: 3, 25: 17, 18: 1, 22: 2, 20: 2, 37: 1}
{11: 231, 16: 52, 17: 6, 14: 3, 22: 20, 12: 4, 20: 1, 21: 1}
{13: 113, 18: 83, 24: 26, 19: 6, 21: 1, 37: 1}
```

Conclusion:
Thus we have built a Simple Parsing program using Python.