Day-Month 2019

## Table of Contents



Student Names (Student IDs)

Teacher Name

# 1. Introduction:

Write a paragraph (less than 100 words) to describe what is project all about

## 1.1. Objectives if Projects

Write a paragraph/list (100-150 words) to describe the Objective of the project

## 1.2. Deliverables

Write a paragraph/list (100-150 words) to describe What are the deliverables of the project.

# 2. Design a scraper to scrape information from a website

Web scraping is a simple means of collecting data from different websites. It allows the Users to collect and manage data as per their requirements. It has wide applications in domains such as price monitoring and collecting huge datasets for various machine learning tasks.

In this component of the project, a Python based scraping tool Scrapy is used to scrape information from the first page of Amazon website's search on earphones. The information extracted includes product name, the link to the product image and the ratings of the product. The output is displayed on the terminal along with being stored in a JSON file.

## 2.1. Coding a scraper

The code for the **website_spider.py** of the scraper along with it's **items.py** is as follows:

**website_spider.py**

```
import scrapy
import csv
from ..items import WebsitescraperItem
```

```
class WebsiteSpiderSpider(scrapy.Spider):
   name = 'website_spider'
                                    start_urls = ['https://www.amazon.in/s?
k=earphones+with+mic&i=electronics&rh=p_72%3A1318476031&dc&crid=2N58U
5A6XCGT2&qid=1586958396&rnid=1318475031&sprefix=ear%2Celectronics
%2C355&ref=sr_nr_p_72_1']

   def parse(self, response):
         product = WebsitescraperItem()

                                    product_name = response.css('.a-color-base.a-text-
normal').css('::text').extract()
                         product_image_link = response.css('.s-image-fixed-height .s-
image::attr(src)').extract()
         product_rating = response.css('.a-icon-alt::text').extract()


         product['product_name'] =  product_name
         product['product_image_link'] =  product_image_link
         product['product_rating'] = product_rating[4:]


         yield product

   pass
```

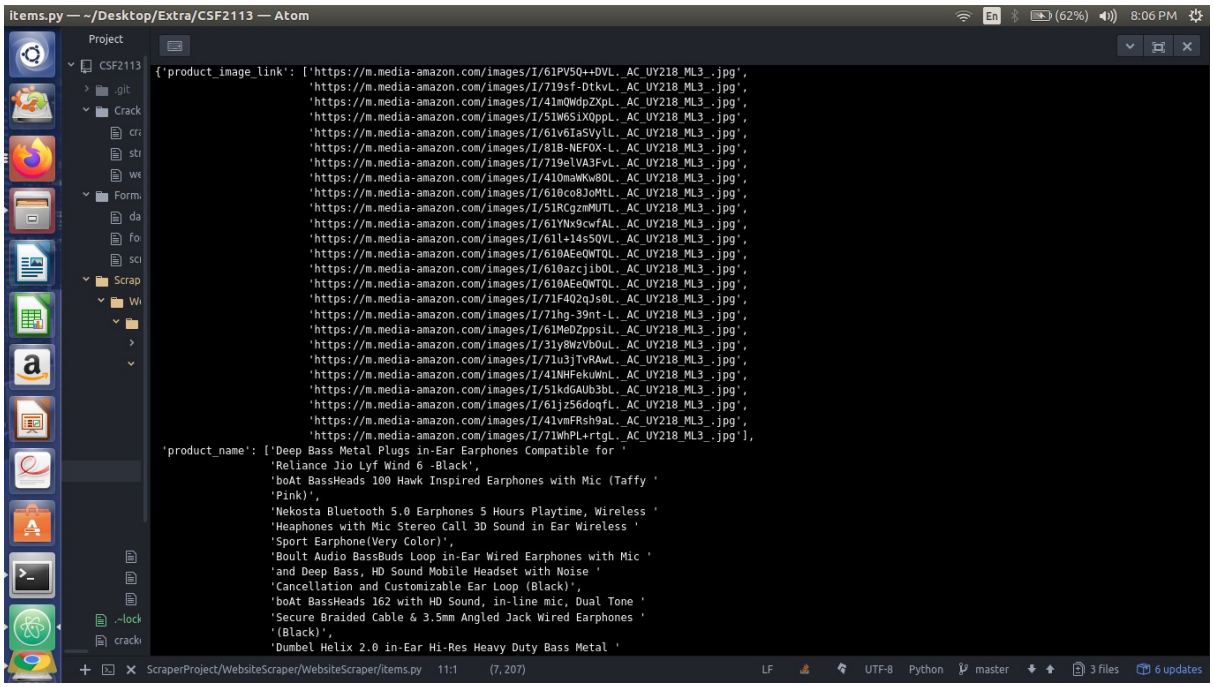
**items.py**

```
import scrapy


class WebsitescraperItem(scrapy.Item):
   # define the fields for your item here like:
    product_name = scrapy.Field()
    product_image_link = scrapy.Field()
    product_rating = scrapy.Field()

pass
```

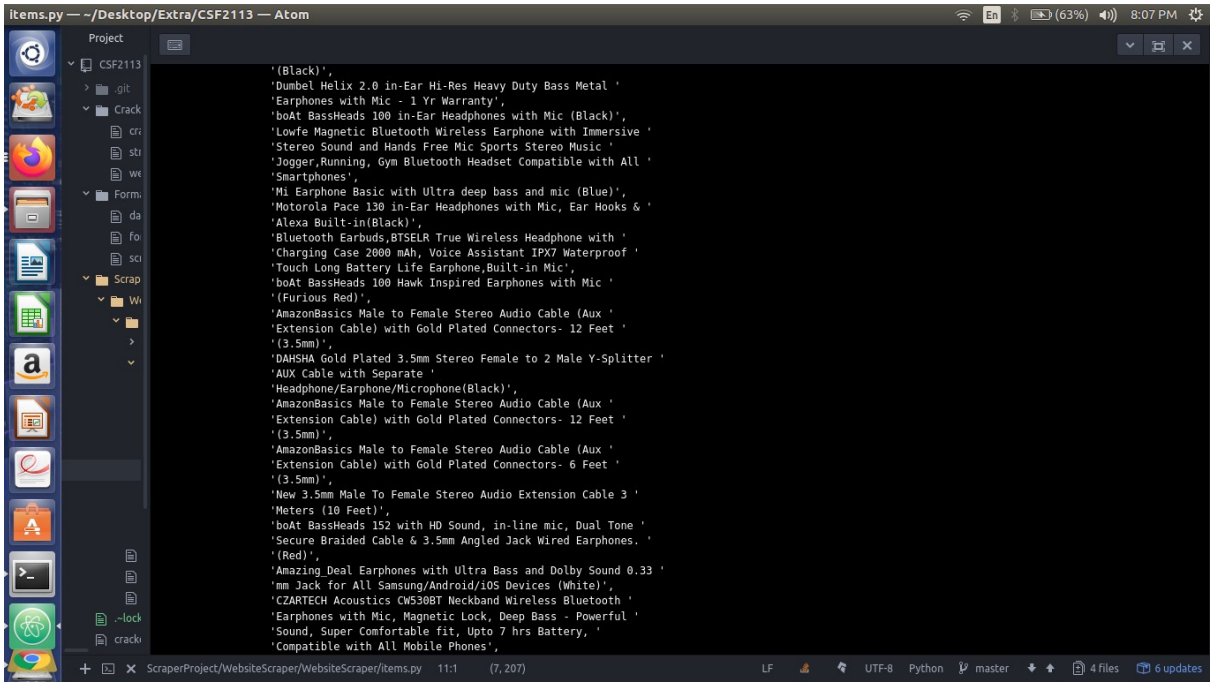
For the entire working code, refer to the code submitted.

## 2.2. **Testing a scraper**

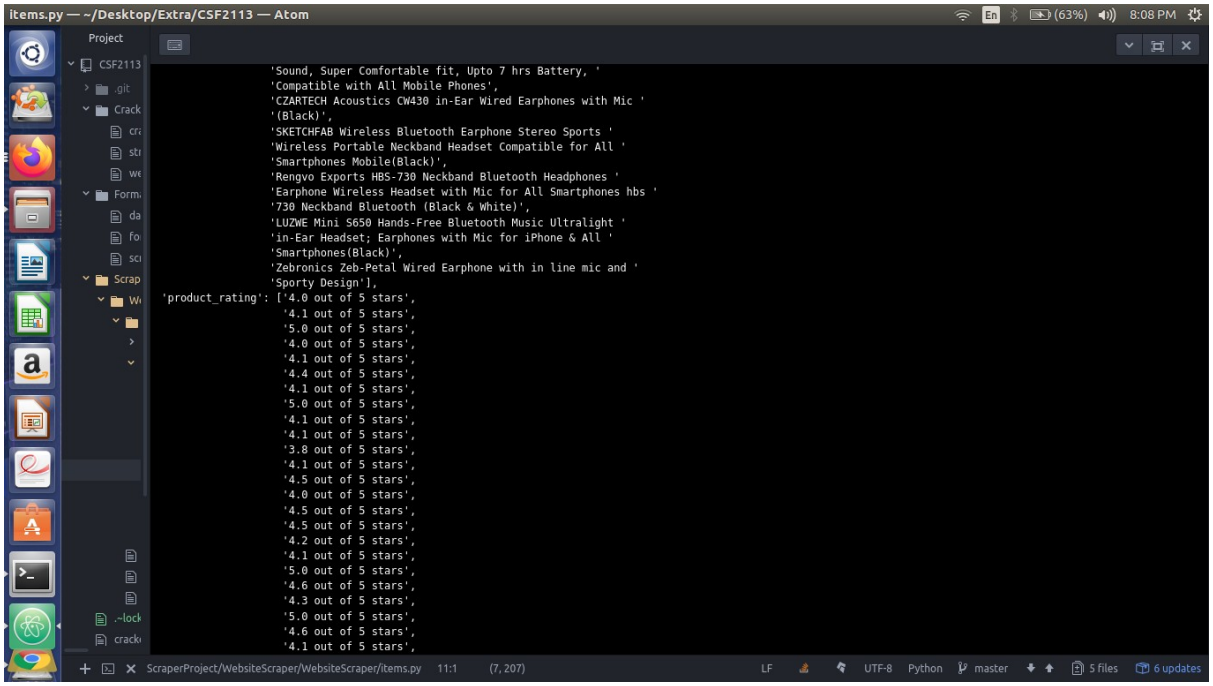
The output displayed on the terminal is as follows:




Terminal Output Part 1

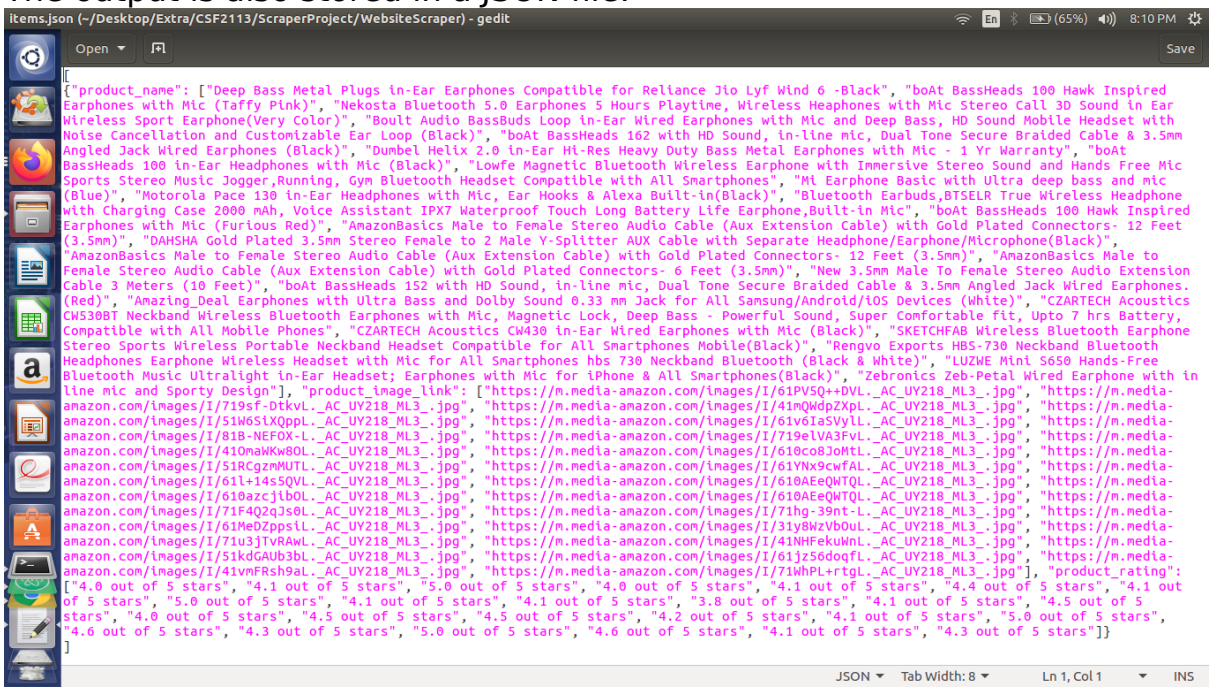



Terminal Output Part 2




Terminal Output Part 3

The output is also stored in a JSON file:




Output JSON file

# 3. Design a formatter to format output data in CSV file

The formatter takes the “json” file generated by the scrapper as its input and returns the data in a “tsv” file named “scrapped_data.tsv”, in a human friendly format. The json file obtained through scrapping does not have data in a human friendly format. Hence, the data needs to be formatted according to the various attributes that it represents. For our present project, we have decided to use scrap “amazon.in” and we obtain data corresponding to three different tags, which are :

1. Product name
2. Product image url
3. Product rating

The formatter parses the json file and creates a data-frame using the python library pandas. The columns in the data-frame correspond to the above mentioned attributes. All scrapped data-instances are arranged according to the same and then stored in a tab-seperated-value manner (as the instances contained strings which contain commas, we could not use csv file format for seperation). The formatter saves the dataframe in the “scrapped_data.tsv”.

## 3.1. Coding a formatter

```
import sys
import pandas as pd

""" The script takes the input in the following format :
-> python formatter.py <path for the json file obtained through scrapper>
and the script produces the output in a file named scrapped_data.tsv """

# the following piece checks for the availability of the input
if len(sys.argv) == 1:
        print("Need Filename!")
        sys.exit(-1)
else:
        path = sys.argv[1]

# opening the file in read_only mode
try :
        file = open(path, "r")
except :
        print("File failed to open! Please verify if the path-name is correctly specified")
        quit()

# the input is expected as a json file and hence, the following piece of code is to extract all the relevant
data from a json file
tag = 0
for each in file:
        if tag == 1 :
                data = each[:-1]
```

```
                    break
            tag = tag + 1
file.close()

# formatting the data
i = 0
columns = []
dataset = []
while i < len(data):
        if data[i] == ":":
                if data[i-1] == "":
                name = ""
                j = -2
                while data[i+j] != "":
                        name = name + data[i+j]
                        j = j - 1
                columns.append(name[::-1])
                aux = ""
                tag = 0
                while data[i] != "]":
                        if data[i] == "[":
                                tag = 1
                                i = i + 1
                        if tag == 1:
                                aux = aux + data[i]
                        i = i + 1
                dataset.append(aux)
        i = i + 1
column_data = [[] for i in range(len(columns))]

tag = 0
for each in dataset:
        i = 0
        while i < len(each):
                if each[i] == "":
                        word = ""
                        i = i + 1
                        while each[i] != "":
                                word = word + each[i]
                                i = i + 1
                        column_data[tag].append(word)
                i = i + 1
        tag = tag + 1

# encoding the data into a pandas data_frame
dataset = [[] for i in range(len(column_data[0]))]
i = 0
while i < len(column_data[0]):
```

```
        j = 0
        aux = []
        while j < len(columns):
                aux.append(column_data[j][i])
                j = j + 1
        dataset[i] = aux
        i = i + 1

# saving the data as a tab-seperated file using pandas
df = pd.DataFrame(dataset, columns = columns)
df.to_csv("scrapped_data.tsv", sep = "\t", index = False)
```

### 3.2. Testing a formatter

* The screenshots have also been added in the “Images” folder.
* The file generated as the output has also been included in the ‘FormatterProject’ folder.

| | A | B | C |
|---|---|---|---|
| 1 | product_name | product_image_link | product_rating |
| 2 | Deep Bass Metal Plugs in-Ear Earphones Compatible for Reliance Jio Lyf Wind 6 -Black | https://m.media-amazon.com/images/I/61PV5Q++DVL._AC_UY218_ML3_.jpg | 4.0 out of 5 stars |
| 3 | boAt BassHeads 100 Hawk Inspired Earphones with Mic (Taffy Pink) | https://m.media-amazon.com/images/I/719sf-DtkvL._AC_UY218_ML3_.jpg | 4.1 out of 5 stars |
| 4 | Nekosta Bluetooth 5.0 Earphones 5 Hours Playtime, Wireless Headphones with Mic Stereo Call 3D Sound in Ear Wireless Sport Earphone(Very Color) | https://m.media-amazon.com/images/I/41mQWdpZXpL._AC_UY218_ML3_.jpg | 5.0 out of 5 stars |
| 5 | Boult Audio BassBuds Loop in-Ear Wired Earphones with Mic and Deep Bass, HD Sound Mobile Headset with Noise Cancellation and Customizable Ear Loop (Black) | https://m.media-amazon.com/images/I/51W8SiXQppL._AC_UY218_ML3_.jpg | 4.0 out of 5 stars |
| 6 | boAt BassHeads 162 with HD Sound, in-line mic, Dual Tone Secure Braided Cable & 3.5mm Angled Jack Wired Earphones (Black) | https://m.media-amazon.com/images/I/61v6laSVyiL._AC_UY218_ML3_.jpg | 4.1 out of 5 stars |
| 7 | Dumbel Helix 2.0 in-Ear Hi-Res Heavy Duty Bass Metal Earphones with Mic - 1 Yr Warranty | https://m.media-amazon.com/images/I/81B-NEFOX-L._AC_UY218_ML3_.jpg | 4.4 out of 5 stars |
| 8 | boAt BassHeads 100 in-Ear Headphones with Mic (Black) | https://m.media-amazon.com/images/I/719eIVA3FvL._AC_UY218_ML3_.jpg | 4.1 out of 5 stars |
| 9 | Lowfe Magnetic Bluetooth Wireless Earphone with Immersive Stereo Sound and Hands Free Mic Sports Stereo Music Jogger,Running, Gym Bluetooth Headset Compatible with All Smartphones | https://m.media-amazon.com/images/I/41OmaWKw8OL._AC_UY218_ML3_.jpg | 5.0 out of 5 stars |
| 10 | Mi Earphone Basic with Ultra deep bass and mic (Blue) | https://m.media-amazon.com/images/I/610co8JoMtL._AC_UY218_ML3_.jpg | 4.1 out of 5 stars |
| 11 | Motorola Pace 130 in-Ear Headphones with Mic, Ear Hooks & Alexa Built-in(Black) | https://m.media-amazon.com/images/I/51RCgzmMUTL._AC_UY218_ML3_.jpg | 4.1 out of 5 stars |
| 12 | Bluetooth Earbuds,BTSELR True Wireless Headphone with Charging Case 2000 mAh, Voice Assistant IPX7 Waterproof Touch Long Battery Life Earphone,Built-in Mic | https://m.media-amazon.com/images/I/61YNx9cwfAL._AC_UY218_ML3_.jpg | 3.8 out of 5 stars |
| 13 | boAt BassHeads 100 Hawk Inspired Earphones with Mic (Furious Red) | https://m.media-amazon.com/images/I/61I+14s5QVL._AC_UY218_ML3_.jpg | 4.1 out of 5 stars |
| 14 | AmazonBasics Male to Female Stereo Audio Cable (Aux Extension Cable) with Gold Plated Connectors- 12 Feet (3.5mm) | https://m.media-amazon.com/images/I/610AEeQWTQL._AC_UY218_ML3_.jpg | 4.5 out of 5 stars |
| 15 | DAHSHA Gold Plated 3.5mm Stereo Female to 2 Male Y-Splitter AUX Cable with Separate Headphone/Earphone/Microphone(Black) | https://m.media-amazon.com/images/I/610azcjbOL._AC_UY218_ML3_.jpg | 4.0 out of 5 stars |
| 16 | AmazonBasics Male to Female Stereo Audio Cable (Aux Extension Cable) with Gold Plated Connectors- 12 Feet (3.5mm) | https://m.media-amazon.com/images/I/610AEeQWTQL._AC_UY218_ML3_.jpg | 4.5 out of 5 stars |
| 17 | AmazonBasics Male to Female Stereo Audio Cable (Aux Extension Cable) with Gold Plated Connectors- 6 Feet (3.5mm) | https://m.media-amazon.com/images/I/71F4Q2qJs0L._AC_UY218_ML3_.jpg | 4.5 out of 5 stars |
| 18 | Now 3.5mm Male To Female Stereo Audio Extension Cable 3 Meters (10 Feet) | https://m.media-amazon.com/images/I/71hg-39nt-L._AC_UY218_ML3_.jpg | 4.2 out of 5 stars |
| 19 | boAt BassHeads 152 with HD Sound, in-line mic, Dual Tone Secure Braided Cable & 3.5mm Angled Jack Wired Earphones. (Red) | https://m.media-amazon.com/images/I/61MeDZppsiL._AC_UY218_ML3_.jpg | 4.1 out of 5 stars |
| 20 | Amazing_Deal Earphones with Ultra Bass and Dolby Sound 0.33 mm Jack for All Samsung/Android/IOS Devices (White) | https://m.media-amazon.com/images/I/31y8WzVbOuL._AC_UY218_ML3_.jpg | 5.0 out of 5 stars |
| 21 | CZARTECH Acoustics CW530BT Neckband Wireless Bluetooth Earphones with Mic, Magnetic Lock, Deep Bass - Powerful Sound, Super Comfortable fit, Upto 7 hrs Battery, Compatible with All Mobile Phones | https://m.media-amazon.com/images/I/71u3jTvRAwL._AC_UY218_ML3_.jpg | 4.6 out of 5 stars |
| 22 | CZARTECH Acoustics CW430 in-Ear Wired Earphones with Mic (Black) | https://m.media-amazon.com/images/I/41NHFekuWnL._AC_UY218_ML3_.jpg | 4.3 out of 5 stars |
| 23 | SKETCHFAB Wireless Bluetooth Earphone Stereo Sports Wireless Portable Neckband Headset Compatible for All Smartphones Mobile(Black) | https://m.media-amazon.com/images/I/51kdGAUb3bL._AC_UY218_ML3_.jpg | 5.0 out of 5 stars |
| 24 | Rengvo Exports HBS-730 Neckband Bluetooth Headphones Earphone Wireless Headset with Mic for All Smartphones hbs 730 Neckband Bluetooth (Black & White) | https://m.media-amazon.com/images/I/61jz56doqfL._AC_UY218_ML3_.jpg | 4.6 out of 5 stars |
| 25 | LUZWE Mini S650 Hands-Free Bluetooth Music Ultralight in-Ear Headset, Earphones with Mic for iPhone & All Smartphones(Black) | https://m.media-amazon.com/images/I/41vmFRsh9aL._AC_UY218_ML3_.jpg | 4.1 out of 5 stars |
| 26 | Zebronics Zeb-Petal Wired Earphone with in line mic and Sporty Design | https://m.media-amazon.com/images/I/71WhPL+rgL._AC_UY218_ML3_.jpg | 4.3 out of 5 stars |
| 27 | | | |
| 28 | | | |

**Fig: The above picture shows the snapshot of the ‘scrapped_data.tsv’ file**

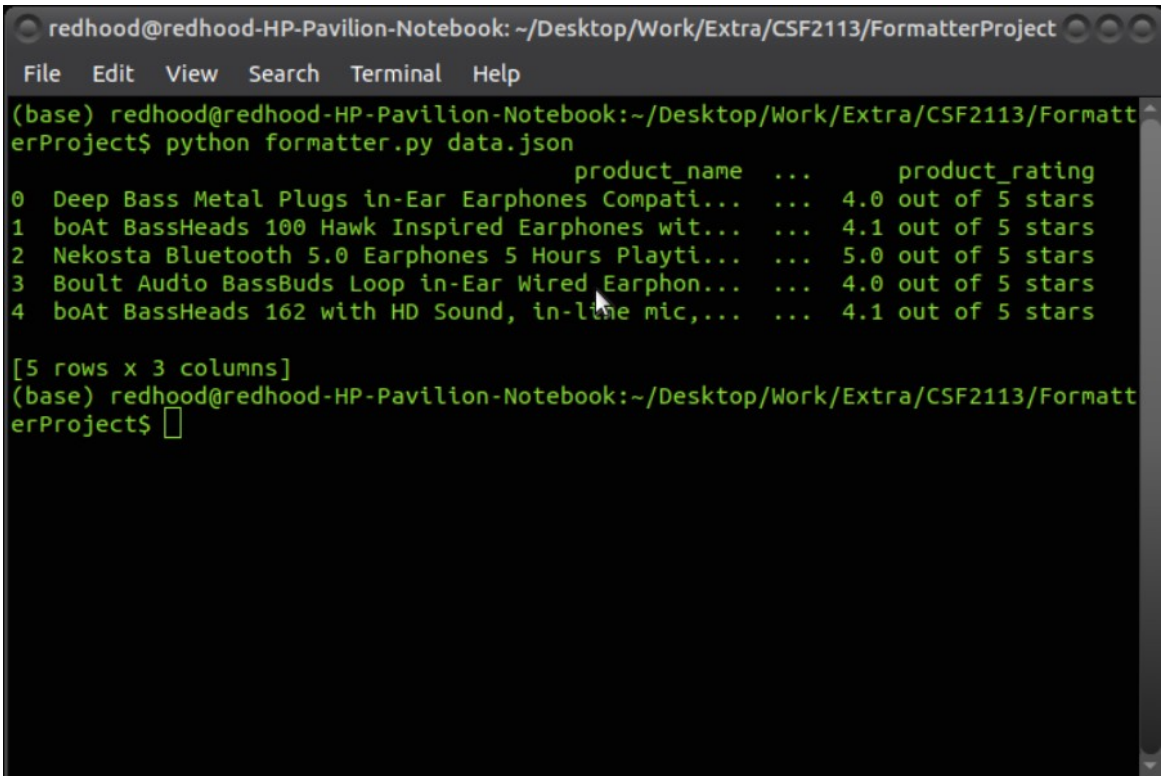



**Fig: The above figure shows the output in the terminal. The top 5 rows of the dataframe has been printed (dataframe.head())**

## 4. Design a cracker to crack a password

The cracker designed in the current project performs a variety of tasks.
To start with, we have defined passwords into two categories :
1. Strong – passwords which contain at least one small-alphabet, one capital-alphabet, one numeric instance and one special character.
2. Weak – all other strings consisting of valid characters but not satisfying at least one of the above criteria.
The cracker is then designed to generate passwords randomly. These can be either strong or weak and of any length that the user desires. For this, generate_strong_password() and generate_weak_password() functions can be used respectively.
The check_pass_word_type() function in the cracker takes a candidate password as its input and determines whether the password is valid or invalid, and if it is valid, whether it is strong or weak. The above mentioned conditions are utilized for the decision making process. The generate_password_dictionary() function takes in the number of strong passwords and the number of weak passwords to be generated, it generates these passwords randomly (all of length = 8) and stores in a python dictionary. The dictionary has keys as the string representing the password and the values as it's corresponding md5 hash. The save_dictionary() function takes in a password_dictionary (as described in the previous step) and saves all the passwords in the dictionary in two separate files, which are - 'strong_passwords.txt' and 'weak_passwords.txt'. All the passwords in the dictionary are classified on the run into the two types, and saved in the corresponding text files respectively.

Additionally, the cracker file also provides for :
1. Loading a dictionary from a text-file (where each line corresponds to a password string).
2. Using a dictionary to perform brute-password cracking (it uses a dictionary and attempts to crack a password(provided as an md5 hash input) by comparing the md5 hash values of all the passwords stored in the dictionary.

## 4.1. Coding to generate the combination of password and to classify the generated passwords

```
import random

import hashlib

import numpy as np


""" The passwords can be of two types, strong or weak.
Strong passwords are made up of a combination of at least one capital letter, small letter, number and a special character.
Weak passwords do not cater to all the above conditions. """

# 33 - 47 and 58 - 64 and 91 - 96 and 123 - 126: special characters
# 48 - 57 : numbers
# 65 - 90 : capital letters
# 97 - 122 : small letters

# declaring the various types of allowed characters in the password. If a password contains any other characters it is deemed invalid
special = [i for i in range(33,48)]
special.extend([i for i in range(58,65)])
special.extend([i for i in range(91,97)])
special.extend([i for i in range(123,127)])

numbers = [i for i in range(48,58)]
capital = [i for i in range(65,91)]
small = [i for i in range(97,123)]

# gives a random numric character
def get_random_number():
        return chr(numbers[np.random.randint(low = 0, high = len(numbers), size = 1)[0]])

# gives a random small-alphabet character
def get_random_small():
```

```
        return chr(small[np.random.randint(low = 0, high = len(small), size = 1)[0]])

# gives a random capital-alphabet character
def get_random_capital():
        return chr(capital[np.random.randint(low = 0, high = len(capital), size = 1)[0]])

# gives a random special character
def get_random_special():
        return chr(special[np.random.randint(low = 0, high = len(special), size = 1)[0]])

# function to print a dictionary in 'key : value' format
def print_dict(d):
        for each in d.keys():
                print(each, end = ": ")
                print(d[each])

# the following function generates a random strong password (the default length of the generated password
is 8)

def generate_strong_password(size = 8):
        if size < 4:
                print("Error! A strong password needs to be at least 4 characters in length")
                return "-1"

num = get_random_number()
small = get_random_small()
cap = get_random_capital()
spec = get_random_special()

out = np.random.randint(low = 33, high = 127, size = size-4)
pwd = num + small + cap + spec

for each in out:
        pwd = pwd + chr(each)

pwd = ''.join(random.sample(pwd, len(pwd)))
return pwd

# the following function generates a random weak password (the default length of the generated password
is 8)

def generate_weak_password(size = 8):
        rand = np.random.randint(low = 0, high = 4, size = 3)
        pwd = ""
        while len(pwd) < size:
                rand2 = np.random.randint(low = 0, high = 3, size = 1)
                if rand[rand2][0] == 0:
                        pwd = pwd + get_random_capital()
```

```
                elif rand[rand2][0] == 1:
                        pwd = pwd + get_random_number()
                elif rand[rand2][0] == 2:
                        pwd = pwd + get_random_small()
                else :
                        pwd = pwd + get_random_special()
        return pwd

# returns 0 for a weak password and returns 1 for a strong password and returns -1 if the password is
invalid

def check_pass_word_type(password):
        arr = [0 for i in range(4)]
        i = 0
        while i < len(password):
                if ord(password[i]) in special:
                        arr[0] = 1
                elif ord(password[i]) in numbers:
                        arr[1] = 1
                elif ord(password[i]) in capital:
                        arr[2] = 1
                elif ord(password[i]) in small:
                        arr[3] = 1
                else :
                        return -1
                i = i + 1
        for each in arr :
                if each == 0:
                        return 0
        return 1

# takes in the number of strong passwords and weak passwords to be generated and generates a dictionary
# the dictionary contains the passwords as its key value and it's md5 hash as its value

def generate_password_dict(num_strong = 10, num_weak = 10):
        passwords = {}
        i = 0
        while i < (num_strong):
                _ = generate_strong_password()
                if _ not in passwords.keys():
                        passwords[_] = hashlib.md5(_.encode("utf-8")).hexdigest()
                        i = i + 1
        i = 0
        while i < (num_weak):
                _ = generate_weak_password()
                if _ not in passwords.keys():
                        passwords[_] = hashlib.md5(_.encode("utf-8")).hexdigest()
                        i = i + 1
```

```
        return passwords

# takes a dictionary as the input and stores the passwords in the dictionary in two seperate files. The seperation is based on the basis of password type (strong and weak)

def save_dictionary(dic):
        f = open("weak_passwords.txt", "w")
        for each in dic.keys():
                if check_pass_word_type(each) == 0:
                        f.write(each)
                        f.write("\n")
        f.close()
        f = open("strong_passwords.txt", "w")
        for each in dic.keys():
                if check_pass_word_type(each) == 1:
                        f.write(each)
                        f.write("\n")
        f.close()
        return 1

# takes in path name as the input and generates a dictionary with all the passwords present in the path name

def load_dictionary(path):
        try :
                f = open(path, "r")
        except :
                print("Invalid path name!")
                quit()

        d = {}
        for each in f :
                each = each[:-1]
                if check_pass_word_type(each) in [0,1]:
                        d[each] = hashlib.md5(each.encode("utf-8")).hexdigest()
        f.close()
        return d

# takes in a dictionary and a password and adds the password to the dictionary

def add_password_to_dict(dic, password):
        if check_pass_word_type(password) == -1:
                print("Invalid password!")
                return -1
        if password in dic.keys():
                print("Password already exists!")
                return -1
        dic[password] = hashlib.md5(password.encode("utf-8")).hexdigest()
        return 1
```

```
# takes in a dictionary as the input and gives information regarding the passwords stored in the dictionary

def analyze_dic(dic):
	strong_pass = 0
	weak_pass = 0
	invalid_pass = 0
	for each in dic.keys():
		if check_pass_word_type(each) == 1:
			strong_pass = strong_pass + 1
		elif check_pass_word_type(each) == 0 :
			weak_pass = weak_pass + 1
		else :
			invalid_pass = invalid_pass + 1
	print("The dictionary contains: ")
	print(strong_pass, " strong passwords")
	print(weak_pass, " weak passwords")
	print(invalid_pass, " invalid passwords")
	return 1

# takes a dictionary of passwords and a md5 hash as the input and attempts to crack the password.
# if successful, it returns the password from the dictionary

def crack_pass_using_dic(dic, inp):
	for each in dic.keys():
		if dic[each] == inp:
			return (1, each)
	return (0, "No match found")
```

## 4.2. **Testing password generation**

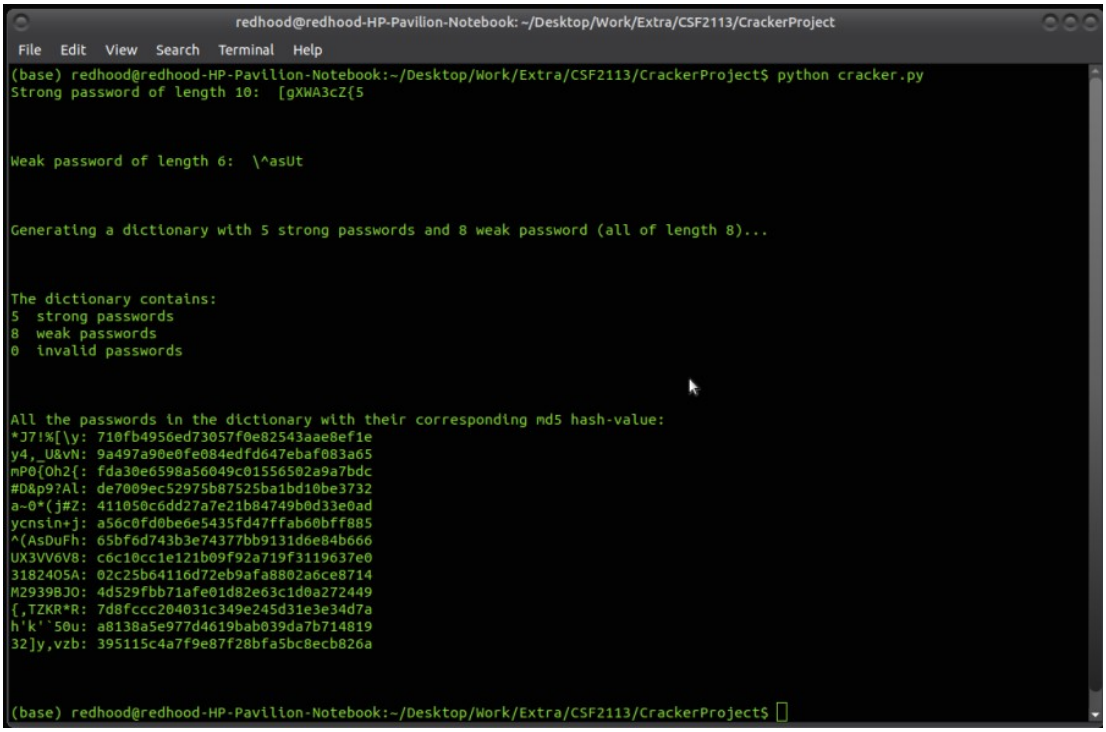



**Fig: The above figure shows the terminal output of the cracker.**

## 4.3. **Testing password classification**

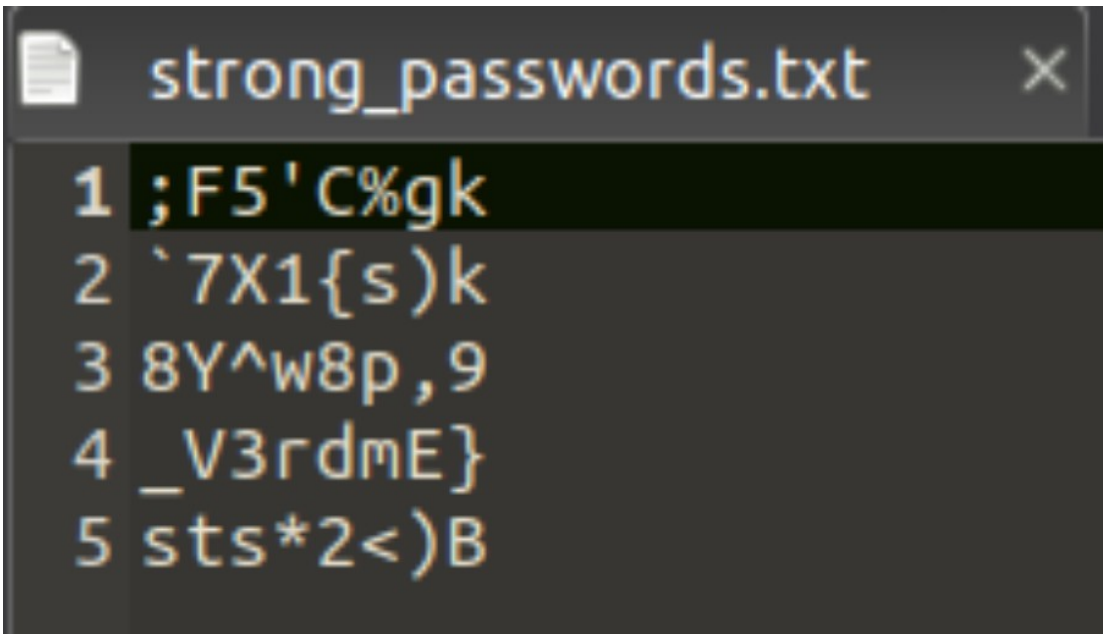



**Fig: The above figure shows the text-file where the generated strong passwords have been stored**

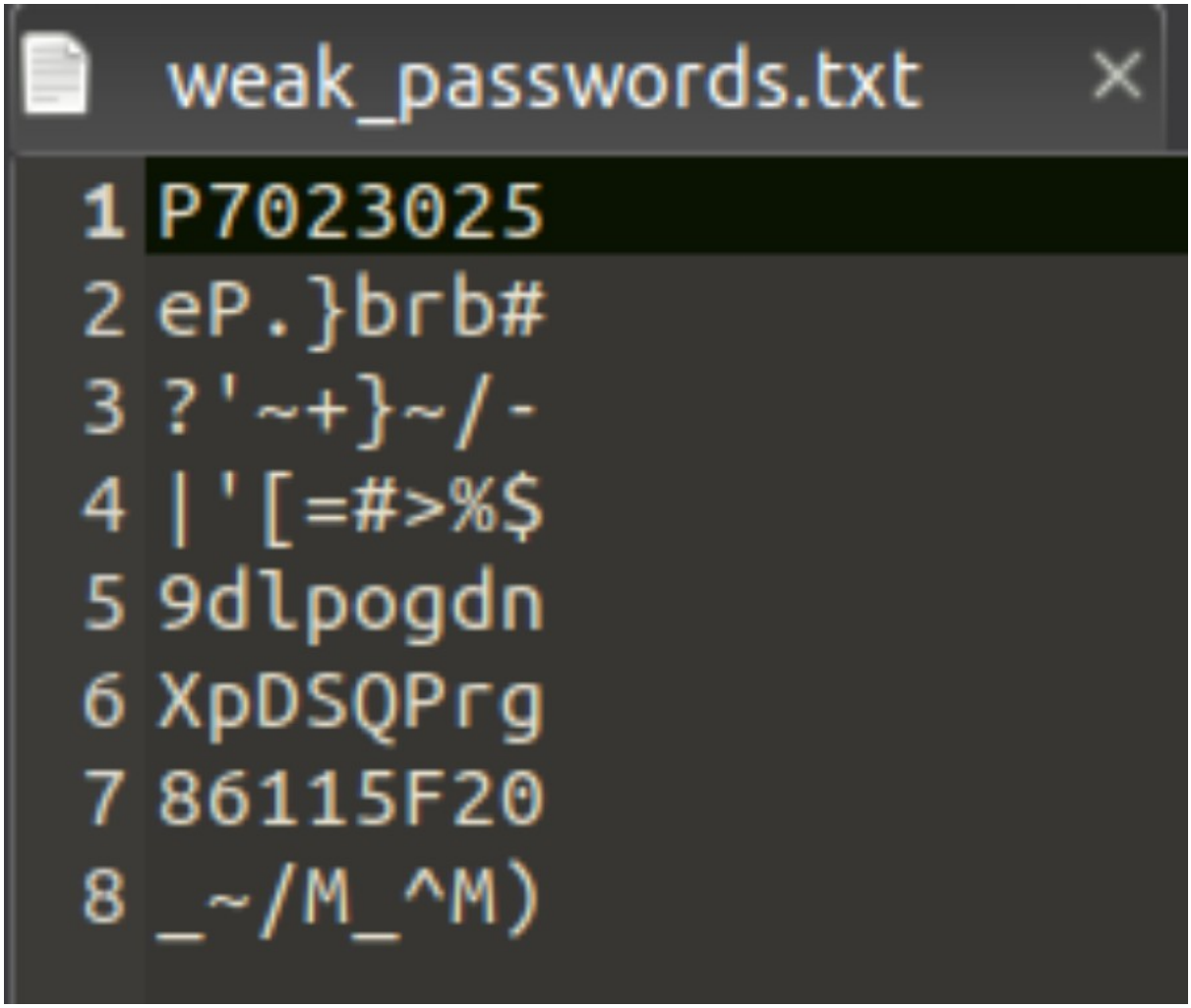



**Fig: The above figure shows the text-file where the generated weak passwords have been stored**

# 5. Reflection

## 5.1. Write the reflection of student1

Write down the reflection of student1

## 5.2. Write the reflection of student2

Write down the reflection of student1

## 5.3. Write the reflection of student3

Write down the reflection of student1

# 6. Appendix Page(s).

- All the images are present in the Images folder.