***HTML, CSS and Web Scraping***

<!DOCTYPE html>

<html lang="en">

<!-- This is the Header element -->

<h1>Python Practical Project <h5>by </h5> </h1> <h3> Mr. A. Awoseyi </h3>

<h2>Milestones</h2>

<h3>Day 41-44 \_\_ Web Foundations</h3>

<h4> • HTML-Header Elements</h4>

<h4> • HTML-Paragraph Elements</h4>

<h4> • HTML-Void Elements</h4>

<h4> ••Milestone Project</h4>

<h4> • HTML Boilerplate </h4>

<h4> • List, Nesting and Indentation</h4>

<h4> • Anchor elements, Image elements</h4>

<h4> ••Milestone Project 2</h4>

<h3>Day 45-53 \_\_ Intermediate Projects with Python</h3>

<h4> • Web Scraping with Beautiful soup</h4>

<h4> • Playlist</h4>

<h4></h4>

<h4></h4>

<h4></h4>

<h4></h4>

<h4></h4>

<!-- This is the Paragraph element -->

<p>The concept of this course is to understand the basis of computer programming in Computer science which follows

from the onset course taken last semester(Computer Programming I) - CSC 201. </p>

<p>This course is CSC-202 wihch is a contination of the earlier one stated above in which we learn the application

of python programming in todays world.</p>

<!--This is the void Element-->

<br>

<hr />

<p>Name of Student: Akinbayo Bolaji Boluwaji <br />

125/22/1/0027<br />

Male <br />

Team Kakashi team leader <br />

Intermediate Python Programmer <br />

</p>

<hr />

<!--This is the major milestone project-->

<h1>JhaeARCH Top movies in the last 3 years</h1>

<h2>My Top 3 in a list of few</h2>

<hr />

<h3>Dune 1 and 2</h3>

<p>Coolest unrealistic movie I've watched lately</p>

<h3>SEAL TEAM</h3>

<p>All things Seal only means one thing which is US-NAVY special forces</p>

<h3>WhoamI-No system is safe</h3>

<p>Although I've watched this before but i just enjoyed watching it again</p>

<!--HTML Boilerplate; this is the part that gives the browser information about the HTML script

I tells what type of HTML in use and also gives more infoo to the browser -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width", initial-scale="1.0">

<title>My sample Boilerplate</title>

</head>

<body>

This place includes all document the browser will display

</body>

</html>

<!--List, Nesting and Indentation-->

<h3>For the dough:</h3>

<ul>

<li>¾ cup warm milk</li>

<li>2 ¼ teaspoons yeast </li>

<li>¼ cup granulated sugar</li>

<li>1 egg plus 1 egg yolk</li>

<li>¼ cup butter</li>

<li>3 cups bread flour</li>

</ul>

<h3>For the filling:</h3>

<ol>

<li>2/3 cup dark brown sugar</li>

<li>1 ½ tablespoons ground cinnamon</li>

<li>¼ cup butter</li>

</ol>

<ul>

<li>A</li>

<li>

B

<ol>

<li>B1</li>

<li>B2

<ul>

<li>B2a

<ul>

<li>B2aa</li>

<li>B2ab</li>

</ul>

</li>

<li>B2b</li>

<li>B2c</li>

</ul>

</li>

<li>B3

<ol>

<li>B31</li>

<li>B32</li>

</ol>

</li>

</ol>

</li>

<li>C</li>

</ul>

<!--Anchor and Image Elements-->

<h1>My top 5 Favourite Websites</h1>

<ol>

<li><a href="https://www.gemini.google.com/">Gemini</a></li>

<li><a href="https://github.com/">Github</a></li>

<li><a href="https://www.facebook.com/">Facebook</a></li>

<li><a href="https://hackertyper.com/">Hacker Typer</a></li>

<li><a href="https://mail.google.com/">Gmail</a></li>

</ol>

<h1>I am a Cat Person</h1>

<img src="https://raw.githubusercontent.com/appbrewery/webdev/main/kitten.jpeg" alt="kitten held in hand" />

<h1>I am a Dog Person</h1>

<img src="https://raw.githubusercontent.com/appbrewery/webdev/main/puppy.gif" alt="puppy digging in the sand" />

<!--Milestone 2-->

<h1>It's My Birthday!</h1>

<h2>On the 13th of December, the existence of Legend happened</h2>

<img src="https://raw.githubusercontent.com/appbrewery/webdev/main/birthday-cake3.4.jpeg"

alt="No cakes just PCs" />

<h3>What to bring:</h3>

<ul>

<li>Bottle of German wine</li>

<li>Fast foods (trust me, you don't wanna miss all the fun)</li>

<li>Gadgets and Devices(Hacking gadgets and Samsung devices precisely for group photos)</li>

</ul>

<h3>The location is hidden beyond this image (Perform a reverse search and also digital footprint for the location)</h3>

<img src="https://raw.githubusercontent.com/appbrewery/webdev/main/birthday-cake3.4.jpeg" />

</html>

#Web Scraping with GeautifulSoup

import requests

from bs4 import BeautifulSoup

URL = "https://www.empireonline.com/movies/features/best-movies-2/"

response = requests.get(URL)

website\_html = report.text

soup = BeautifulSoup(website\_html, "html.parser")

#print(soup.prettify())

all\_movs = soup.find\_all(name="h3", class\_="title")

movie\_titles = [movie.getText() for movie in all\_movs]

**Birthday Invite Project**

<!-- This is one possible solution -->

<h1>It's My Birthday!</h1>

<h2>On the 12th May</h2>

<img src="https://raw.githubusercontent.com/appbrewery/webdev/main/birthday-cake3.4.jpeg"

  alt="purple birthday cake with candles" />

<h3>What to bring:</h3>

<ul>

  <li>Baloons (I love baloons)</li>

  <li>Cake (I'm really good at eating)</li>

  <li>An appetite (There will be lots of food)</li>

</ul>

<h3>This is where you need to go:</h3>

<a href="https://www.google.com/maps/@35.7040744,139.5577317,3a,75y,289.6h,87.01t,0.72r/data=!3m6!1e1!3m4!1sgT28ssf0BB2LxZ63JNcL1w!2e0!7i13312!8i6656">Google map link</a>

***Day 43***

**CSS Selectors**

**HTML**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=7">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>CSS Selectors</title>

  <link rel="stylesheet" href="./solution-style.css" />

</head>

<body>

  <h1>CSS Selectors</h1>

  <h2>Applying CSS to Different Parts of HTML</h2>

  <!-- TODO 1: Set the CSS for all paragraph tags to "color: red" -->

  <p class="note">1. The element selector targets elements based on their HTML tag name.</p>

  <ol>

    <!-- TODO 2: Set the CSS for all elements with a class of "note" to "font-size: 20px" -->

    <li class="note" value="2">Class selectors target elements based on the value of the class attribute.</li>

    <!-- TODO 3: Set the CSS for the element with an id of "id-selector-demo" to "color: green" -->

    <li class="note" id="id-selector-demo" value="3">ID selectors target elements based on the value of the id

      attribute.</li>

    <!-- TODO 4: Set the CSS for the li elements that have the "value" attribute set to "4" to have "color: blue" -->

    <li class="note" value="4">Attribute selectors target elements based on their attributes and values.</li>

    <!-- TODO 5: Set all elements to have "text-align: center" -->

    <li class="note" value="5">The universal selector targets all elements.</li>

  </ol>

</body>

</html>

**CSS**

ol {

  margin-left: -40px;

  margin-top: -20px;

  list-style-position: inside;

}

/\* Write your CSS below, don't change the rules above. \*/

/\* TODO 1: Set the CSS for all paragraph tags to "color: red;" \*/

p {

  color: red;

}

/\* TODO 2: Set the CSS for all elements with a class of "note" to "font-size: 20px;" \*/

.note {

  font-size: 20px;

}

/\* TODO 3: Set the CSS for the element with an id of "id-selector-demo" to "color: green;" \*/

#id-selector-demo {

  color: green;

}

/\* TODO 4: Set the CSS for the li elements that have the "value" attribute set to "4" to have "color: blue;" \*/

li[value="4"] {

  color: blue;

}

/\* TODO 5: Set all elements to have "font-family: sans-serif;" \*/

\* {

  text-align: center;

}

***Day 44***

**HTML**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Motivation Meme</title>

    <link rel="stylesheet" href="./style.css" />

    <link rel="preconnect" href="https://fonts.googleapis.com" />

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

    <link

      href="https://fonts.googleapis.com/css2?family=Libre+Baskerville&display=swap"

      rel="stylesheet"

    />

  </head>

  <body>

    <div class="poster">

      <img

        class="motivation-img"

        src="./assets/images/daenerys.jpeg"

        src="daenerys with egg"

        alt=""

      />

      <h1>That Special Moment</h1>

      <p>When you find the perfect avocado at the supermarket</p>

    </div>

  </body>

</html>

**CSS**

body {

  background-color: black;

}

h1 {

  text-transform: uppercase;

  font-size: 3rem;

}

.poster {

  width: 50%;

  margin-left: 25%;

  margin-top: 100px;

  color: white;

  font-family: "Libre Baskerville", serif;

  text-align: center;

}

.motivation-img {

  border: 5px solid white;

  width: 100%;

}

***Day 45***

**Movies to watch**

import requests

from bs4 import BeautifulSoup

URL = "https://web.archive.org/web/20200518073855/https://www.empireonline.com/movies/features/best-movies-2/"

response = requests.get(URL)

website\_html = response.text

soup = BeautifulSoup(website\_html, "html.parser")

all\_movies = soup.find\_all(name="h3", class\_="title")

movie\_titles = [movie.getText() for movie in all\_movies]

movies = movie\_titles[::-1]

with open("movies.txt", mode="w") as file:

    for movie in movies:

        file.write(f"{movie}\n")

***Day 46***

**A Spotify Playlist**

# step 1 - Scraping the Billboard Hot 100 for a particular date

from bs4 import BeautifulSoup

import requests

date = input("Which year do you want to travel to? Type the date in this format YYYY-MM-DD: ")

response = requests.get(f"https://www.billboard.com/charts/hot-100/{date}")

soup = BeautifulSoup(response.text, 'html.parser')

song\_names\_spans = soup.select("li ul li h3")

song\_names = [song.getText().strip() for song in song\_names\_spans]

# step 2 - Autentication with spotify

import spotipy

from spotipy.oauth2 import SpotifyOAuth

sp = spotipy.Spotify(

    auth\_manager = SpotifyOAuth(

        scope = "playlist-modify-private",

        redirect\_uri = "http://example.com",

        client\_id = "794a374bd6714699a17fefc70db654a9",

        client\_secret = "8e5b39af573d4a51a49f414f94c9285a",

        show\_dialog = True,

        cache\_path = "token.txt"

    )

)

user\_id = sp.current\_user()["id"]

print(user\_id)

#step 3 - Search spotify for the songs

song\_uris = []

year = date.split("-")[0]

for song in song\_names:

    result = sp.search(q=f"track:{song} year:{year}", type="track")

    print(result)

    try:

        uri = result["tracks"]["items"][0]["uri"]

        song\_uris.append(uri)

    except IndexError:

        print(f"{song} doesn't exist in Spotify. Skipped.")

# step 4 - Creating a new private playlist in Spotify

playlist = sp.user\_playlist\_create(user=user\_id, name=f"{date} Billboard 100", public=False)

print(playlist)

#Adding songs found into the new playlist

sp.playlist\_add\_items(playlist\_id=playlist["id"], items=song\_uris)

***Day 47***

**Create an Automated Amazon Price Tracker**

# Using beautifulsoup to scrap amazon website for pressure-cooking pot

import requests

import lxml

from bs4 import BeautifulSoup

url = "https://www.amazon.com/dp/B075CYMYK6?psc=1&ref\_=cm\_sw\_r\_cp\_ud\_ct\_FM9M699VKHTT47YD50Q6"

header = {

    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36 Edg/125.0.0.0",

    "Accept-Language": "en-US,en;q=0.9"

}

response = requests.get(url, headers=header)

soup = BeautifulSoup(response.content, "lxml")

# print(soup.prettify())

price = soup.find(class\_="a-offscreen").get\_text()

price\_without\_currency = price.split("$")[1]

price\_as\_float = float(price\_without\_currency)

print(price\_as\_float)

# Email alert when price below preset value

import smtplib

title = soup.find(id="productTitle").get\_text().strip()

print(title)

BUY\_PRICE = 200

if price\_as\_float < BUY\_PRICE:

    message = f"{title} is now {price}"

    with smtplib.SMTP("smtp.gmail.com", port=587) as connection:

        connection.starttls()

        result = connection.login("wareezakanmu249@gail.com", "??zir\_raw??")

        connection.sendmail(

            from\_addr="wareezakanmu249@gail.com",

            to\_addrs="wareezakanmu249@gail.com",

            msg=f"Subject:Amazon Price Alert!\n\n{message}\n{url}".encode("utf-8")

        )

***Day\_48***

**Cookie Clicker Project**

from selenium import webdriver

import time

chrome\_driver\_path = "C:\chromedriver-win64\chromedriver-win64\chromedriver.exe"

driver = webdriver.Chrome(chrome\_driver\_path)

driver.get("http://orteil.dashnet.org/experiments/cookie/")

#Get cookie to click on.

cookie = driver.find\_element\_by\_id("cookie")

#Get upgrade item ids.

items = driver.find\_elements\_by\_css\_selector("#store div")

item\_ids = [item.get\_attribute("id") for item in items]

timeout = time.time() + 5

five\_min = time.time() + 60\*5 # 5minutes

while True:

cookie.click()

#Every 5 seconds:

if time.time() > timeout:

#Get all upgrade <b> tags

all\_prices = driver.find\_elements\_by\_css\_selector("#store b")

item\_prices = []

#Convert <b> text into an integer price.

for price in all\_prices:

element\_text = price.text

if element\_text != "":

cost = int(element\_text.split("-")[1].strip().replace(",", ""))

item\_prices.append(cost)

#Create dictionary of store items and prices

cookie\_upgrades = {}

for n in range(len(item\_prices)):

cookie\_upgrades[item\_prices[n]] = item\_ids[n]

#Get current cookie count

money\_element = driver.find\_element\_by\_id("money").text

if "," in money\_element:

money\_element = money\_element.replace(",", "")

cookie\_count = int(money\_element)

#Find upgrades that we can currently afford

affordable\_upgrades = {}

for cost, id in cookie\_upgrades.items():

if cookie\_count > cost:

affordable\_upgrades[cost] = id

#Purchase the most expensive affordable upgrade

highest\_price\_affordable\_upgrade = max(affordable\_upgrades)

print(highest\_price\_affordable\_upgrade)

to\_purchase\_id = affordable\_upgrades[highest\_price\_affordable\_upgrade]

driver.find\_element\_by\_id(to\_purchase\_id).click()

#Add another 5 seconds until the next check

timeout = time.time() + 5

#After 5 minutes stop the bot and check the cookies per second count.

if time.time() > five\_min:

cookie\_per\_s = driver.find\_element\_by\_id("cps").text

print(cookie\_per\_s)

break

***Day\_49***

**LinkedIn Application Automation**

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from selenium.common.exceptions import NoSuchElementException

from selenium.webdriver.chrome.service import Service as ChromeService

from selenium.webdriver.common.by import By

import time

ACCOUNT\_EMAIL = "YOUR LOGIN EMAIL"

ACCOUNT\_PASSWORD = "YOUR LOGIN PASSWORD"

PHONE = "YOUR PHONE NUMBER"

def abort\_application():

*# Click Close Button*

    close\_button = driver.find\_element(*by*=By.CLASS\_NAME, *value*="artdeco-modal\_\_dismiss")

    close\_button.click()

    time.sleep(2)

*# Click Discard Button*

    discard\_button = driver.find\_elements(*by*=By.CLASS\_NAME, *value*="artdeco-modal\_\_confirm-dialog-btn")[1]

    discard\_button.click()

chrome\_driver\_path = " C:\chromedriver-win64\chromedriver-win64\chromedriver.exe "

*# Optional - Automatically keep your chromedriver up to date.*

from webdriver\_manager.chrome import ChromeDriverManager  *# pip install webdriver-manager*

chrome\_driver\_path = ChromeDriverManager(*path*="YOUR CHROME DRIVER FOLDER").install()

*# Optional - Keep the browser open if the script crashes.*

chrome\_options = webdriver.ChromeOptions()

chrome\_options.add\_experimental\_option("detach", True)

service = ChromeService(*executable\_path*=chrome\_driver\_path)

driver = webdriver.Chrome(*service*=service, *options*=chrome\_options)

driver.get("https://www.linkedin.com/jobs/search/?currentJobId=3586148395&f\_LF=f\_AL&geoId=101356765&"

           "keywords=python&location=London%2C%20England%2C%20United%20Kingdom&refresh=true")

*# Click Reject Cookies Button*

time.sleep(2)

reject\_button = driver.find\_element(*by*=By.CSS\_SELECTOR, *value*='button[action-type="DENY"]')

reject\_button.click()

*# Click Sign in Button*

time.sleep(2)

sign\_in\_button = driver.find\_element(*by*=By.LINK\_TEXT, *value*="Sign in")

sign\_in\_button.click()

*# Sign in*

time.sleep(5)

email\_field = driver.find\_element(*by*=By.ID, *value*="username")

email\_field.send\_keys(ACCOUNT\_EMAIL)

password\_field = driver.find\_element(*by*=By.ID, *value*="password")

password\_field.send\_keys(ACCOUNT\_PASSWORD)

password\_field.send\_keys(Keys.ENTER)

*# CAPTCHA - Solve Puzzle Manually*

input("Press Enter when you have solved the Captcha")

*# Get Listings*

time.sleep(5)

all\_listings = driver.find\_elements(*by*=By.CSS\_SELECTOR, *value*=".job-card-container--clickable")

*# Apply for Jobs*

for listing in all\_listings:

    print("Opening Listing")

    listing.click()

    time.sleep(2)

    try:

*# Click Apply Button*

        apply\_button = driver.find\_element(*by*=By.CSS\_SELECTOR, *value*=".jobs-s-apply button")

        apply\_button.click()

*# Insert Phone Number*

*# Find an <input> element where the id contains phoneNumber*

        time.sleep(5)

        phone = driver.find\_element(*by*=By.CSS\_SELECTOR, *value*="input[id\*=phoneNumber]")

        if phone.text == "":

            phone.send\_keys(PHONE)

*# Check the Submit Button*

        submit\_button = driver.find\_element(*by*=By.CSS\_SELECTOR, *value*="footer button")

        if submit\_button.get\_attribute("data-control-name") == "continue\_unify":

            abort\_application()

            print("Complex application, skipped.")

            continue

        else:

*# Click Submit Button*

            print("Submitting job application")

            submit\_button.click()

        time.sleep(2)

*# Click Close Button*

        close\_button = driver.find\_element(*by*=By.CLASS\_NAME, *value*="artdeco-modal\_\_dismiss")

        close\_button.click()

    except NoSuchElementException:

        abort\_application()

        print("No application button, skipped.")

        continue

time.sleep(5)

driver.quit()