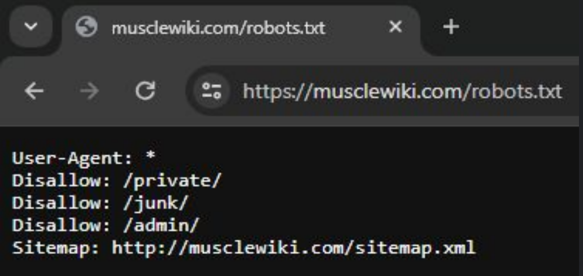
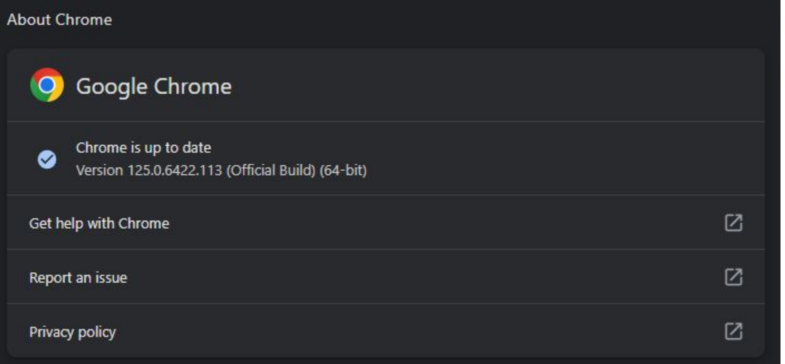
Introduction

Slide 2


musclewiki.com/robots.txt

https://musclewiki.com/robots.txt

```
User-Agent: *
Disallow: /private/
Disallow: /junk/
Disallow: /admin/
Sitemap: http://musclewiki.com/sitemap.xml
```


About Chrome

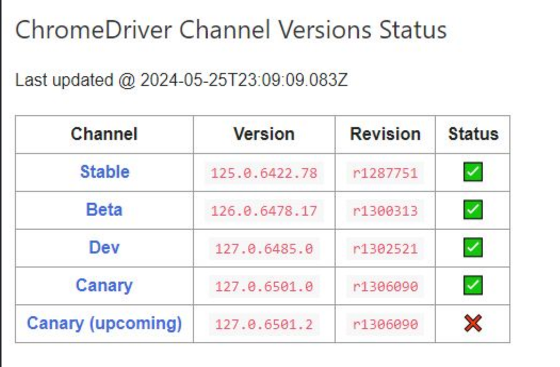
Google Chrome

Chrome is up to date
Version 125.0.6422.113 (Official Build) (64-bit)

Get help with Chrome

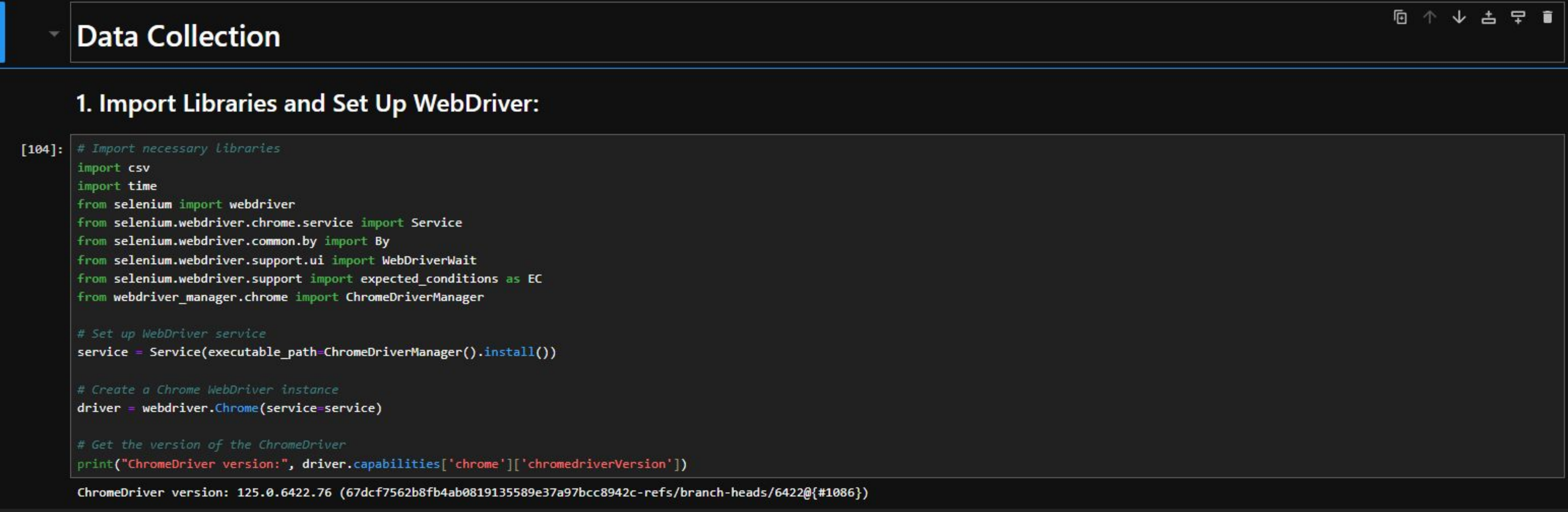
Report an issue

Privacy policy


ChromeDriver Channel Versions Status

Last updated @ 2024-05-25T23:09:09.083Z

Channel	Version	Revision	Status
Stable	125.0.6422.78	r1287751	✅
Beta	126.0.6478.17	r1300313	✅
Dev	127.0.6485.0	r1302521	✅
Canary	127.0.6501.0	r1306090	✅
Canary (upcoming)	127.0.6501.2	r1306090	❌

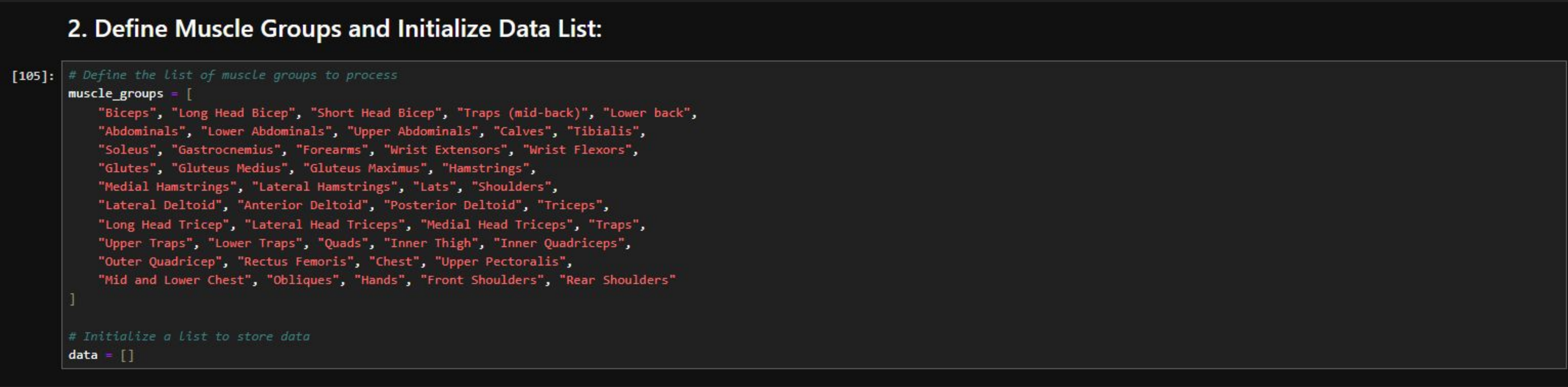

Data Collection

1. Import Libraries and Set Up WebDriver:

[104]:
```python
# Import necessary libraries
import csv
import time
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager

# Set up WebDriver service
service = Service(executable_path=ChromeDriverManager().install())

# Create a Chrome WebDriver instance
driver = webdriver.Chrome(service=service)

# Get the version of the ChromeDriver
print("ChromeDriver version:", driver.capabilities['chrome']['chromedriverVersion'])
```

```
ChromeDriver version: 125.0.6422.76 (67dcf7562b8fb4ab0819135589e37a97bcc8942c-refs/branch-heads/6422@{#1086})
```

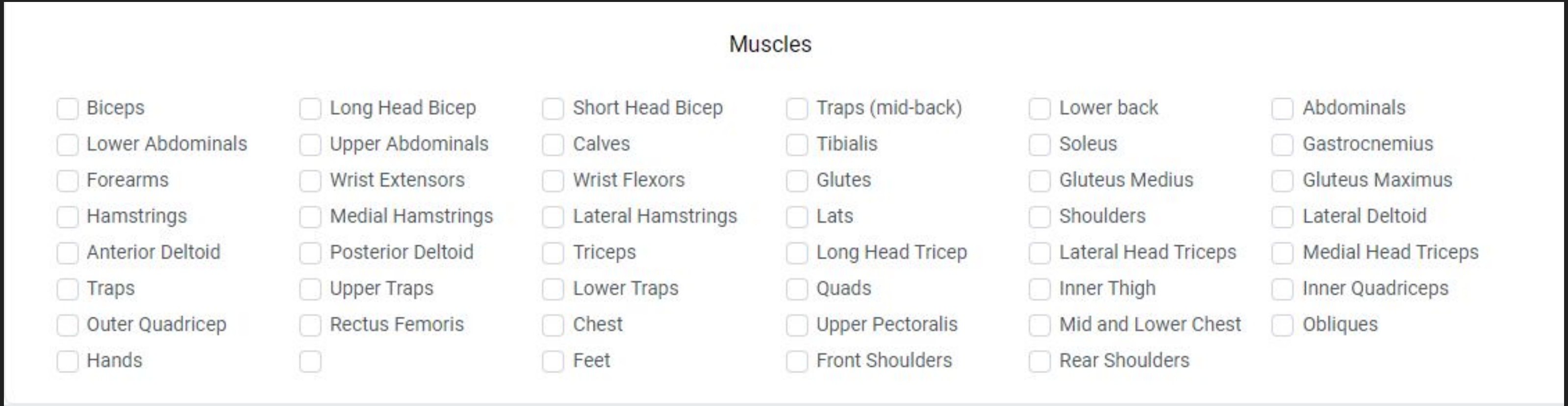
Slide 3



2. Define Muscle Groups and Initialize Data List:

```
[105]:
# Define the list of muscle groups to process
muscle_groups = [
    "Biceps", "Long Head Bicep", "Short Head Bicep", "Traps (mid-back)", "Lower back",
    "Abdominals", "Lower Abdominals", "Upper Abdominals", "Calves", "Tibialis",
    "Soleus", "Gastrocnemius", "Forearms", "Wrist Extensors", "Wrist Flexors",
    "Glutes", "Gluteus Medius", "Gluteus Maximus", "Hamstrings",
    "Medial Hamstrings", "Lateral Hamstrings", "Lats", "Shoulders",
    "Lateral Deltoid", "Anterior Deltoid", "Posterior Deltoid", "Triceps",
    "Long Head Tricep", "Lateral Head Triceps", "Medial Head Triceps", "Traps",
    "Upper Traps", "Lower Traps", "Quads", "Inner Thigh", "Inner Quadriceps",
    "Outer Quadricep", "Rectus Femoris", "Chest", "Upper Pectoralis",
    "Mid and Lower Chest", "Obliques", "Hands", "Front Shoulders", "Rear Shoulders"
]

# Initialize a list to store data
data = []
```

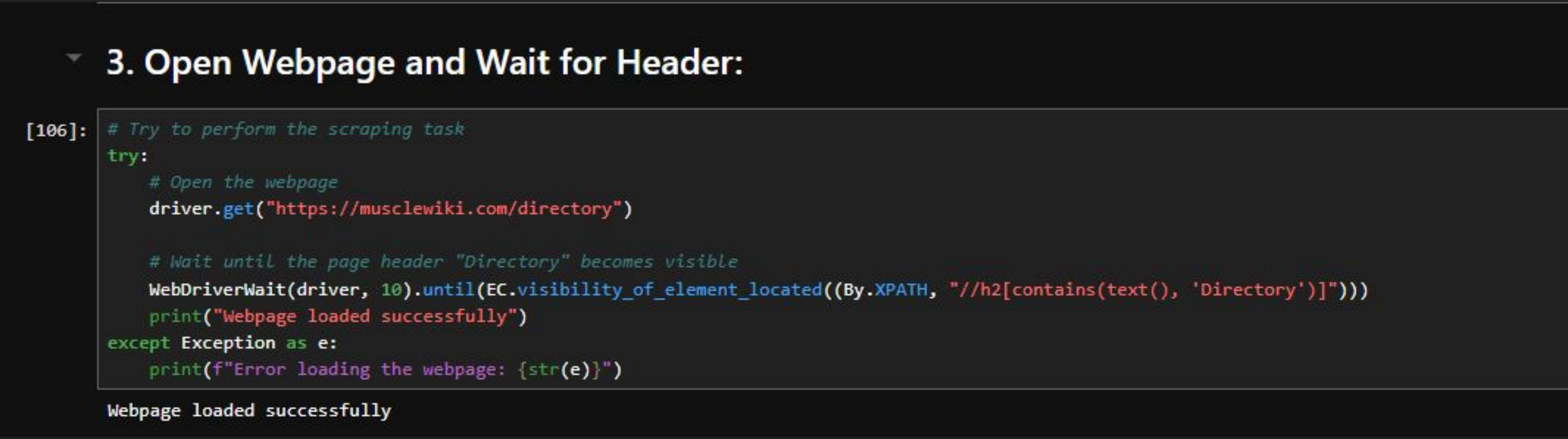
MuscleWiki.com's Muscle Directory



Muscles

☐ Biceps	☐ Long Head Bicep	☐ Short Head Bicep	☐ Traps (mid-back)	☐ Lower back	☐ Abdominals
☐ Lower Abdominals	☐ Upper Abdominals	☐ Calves	☐ Tibialis	☐ Soleus	☐ Gastrocnemius
☐ Forearms	☐ Wrist Extensors	☐ Wrist Flexors	☐ Glutes	☐ Gluteus Medius	☐ Gluteus Maximus
☐ Hamstrings	☐ Medial Hamstrings	☐ Lateral Hamstrings	☐ Lats	☐ Shoulders	☐ Lateral Deltoid
☐ Anterior Deltoid	☐ Posterior Deltoid	☐ Triceps	☐ Long Head Tricep	☐ Lateral Head Triceps	☐ Medial Head Triceps
☐ Traps	☐ Upper Traps	☐ Lower Traps	☐ Quads	☐ Inner Thigh	☐ Inner Quadriceps
☐ Outer Quadricep	☐ Rectus Femoris	☐ Chest	☐ Upper Pectoralis	☐ Mid and Lower Chest	☐ Obliques
☐ Hands	☐	☐ Feet	☐ Front Shoulders	☐ Rear Shoulders	

Slide 4

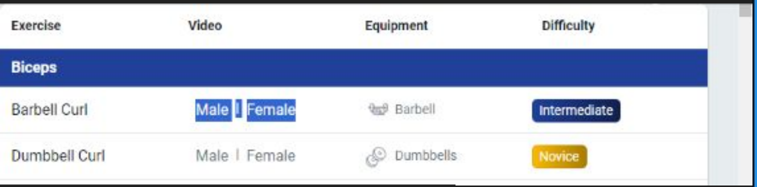


3. Open Webpage and Wait for Header:

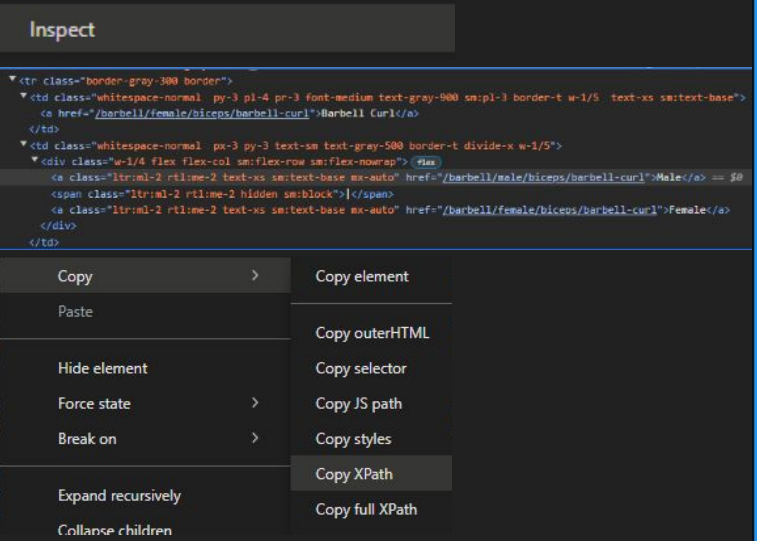
```
[106]: # Try to perform the scraping task
try:
    # Open the webpage
    driver.get("https://musclewiki.com/directory")

    # Wait until the page header "Directory" becomes visible
    WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.XPATH, "//h2[contains(text(), 'Directory')]")))
    print("Webpage loaded successfully")
except Exception as e:
    print(f"Error loading the webpage: {str(e)}")
```

```
Webpage loaded successfully
```

Slide 5

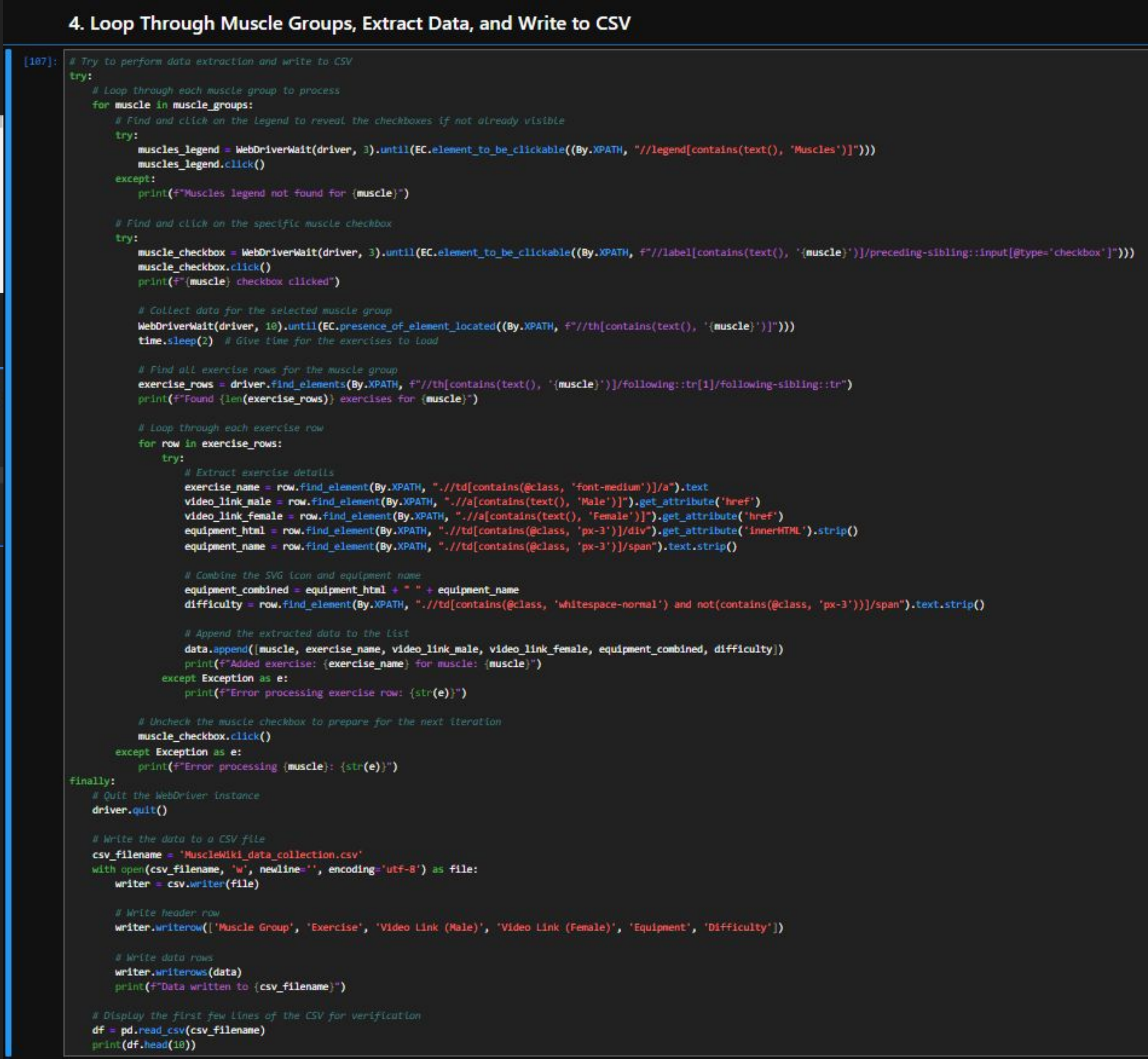


Exercise	Video	Equipment	Difficulty
Biceps			
Barbell Curl	Male | Female	Barbell	Intermediate
Dumbbell Curl	Male | Female	Dumbbells	Novice



Inspect

```
▼<tr class="border-gray-300 border">
  ▼<td class="whitespace-normal  py-3 pl-4 pr-3 font-medium text-gray-900 sm:pl-3 border-t w-1/5  text-xs sm:text-base">
     <a href="/barbell/female/biceps/barbell-curl">Barbell Curl</a>
   </td>
  ▼<td class="whitespace-normal  px-3 py-3 text-sm text-gray-500 border-t divide-x w-1/5">
    ▼<div class="w-1/4 flex flex-col sm:flex-row sm:flex-nowrap"> flex
       <a class="ltr:ml-2 rtl:me-2 text-xs sm:text-base mx-auto" href="/barbell/male/biceps/barbell-curl">Male</a> == $0
       <span class="ltr:ml-2 rtl:me-2 hidden sm:block">|</span>
       <a class="ltr:ml-2 rtl:me-2 text-xs sm:text-base mx-auto" href="/barbell/female/biceps/barbell-curl">Female</a>
     </div>
   </td>
```

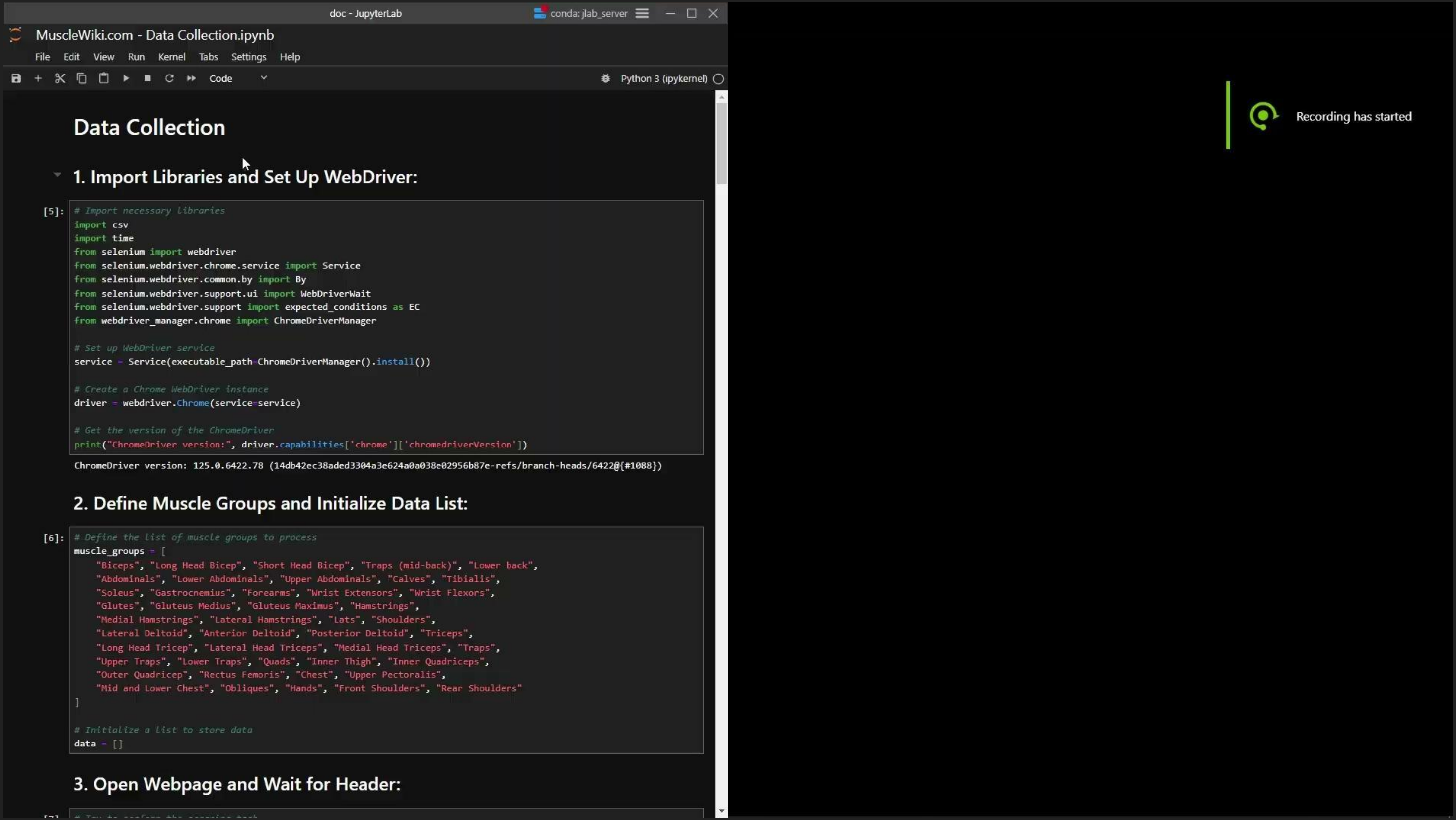


4. Loop Through Muscle Groups, Extract Data, and Write to CSV

[107]:
```python
# Try to perform data extraction and write to CSV
try:
    # Loop through each muscle group to process
    for muscle in muscle_groups:
        # Find and click on the legend to reveal the checkboxes if not already visible
        try:
            muscles_legend = WebDriverWait(driver, 3).until(EC.element_to_be_clickable((By.XPATH, "//legend[contains(text(), 'Muscles')]")))
            muscles_legend.click()
        except:
            print(f"Muscles legend not found for {muscle}")

        # Find and click on the specific muscle checkbox
        try:
            muscle_checkbox = WebDriverWait(driver, 3).until(EC.element_to_be_clickable((By.XPATH, f"//label[contains(text(), '{muscle}')]/preceding-sibling::input[@type='checkbox']")))
            muscle_checkbox.click()
            print(f"{muscle} checkbox clicked")

            # Collect data for the selected muscle group
            WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.XPATH, f"//th[contains(text(), '{muscle}')]")))
            time.sleep(2)  # Give time for the exercises to load

            # Find all exercise rows for the muscle group
            exercise_rows = driver.find_elements(By.XPATH, f"//th[contains(text(), '{muscle}')]/following::tr[1]/following-sibling::tr")
            print(f"Found {len(exercise_rows)} exercises for {muscle}")

            # Loop through each exercise row
            for row in exercise_rows:
                try:
                    # Extract exercise details
                    exercise_name = row.find_element(By.XPATH, ".//td[contains(@class, 'font-medium')]/a").text
                    video_link_male = row.find_element(By.XPATH, ".//a[contains(text(), 'Male')]").get_attribute('href')
                    video_link_female = row.find_element(By.XPATH, ".//a[contains(text(), 'Female')]").get_attribute('href')
                    equipment_html = row.find_element(By.XPATH, ".//td[contains(@class, 'px-3')]/div").get_attribute('innerHTML').strip()
                    equipment_name = row.find_element(By.XPATH, ".//td[contains(@class, 'px-3')]/span").text.strip()

                    # Combine the SVG icon and equipment name
                    equipment_combined = equipment_html + " " + equipment_name
                    difficulty = row.find_element(By.XPATH, ".//td[contains(@class, 'whitespace-normal') and not(contains(@class, 'px-3'))]/span").text.strip()

                    # Append the extracted data to the list
                    data.append([muscle, exercise_name, video_link_male, video_link_female, equipment_combined, difficulty])
                    print(f"Added exercise: {exercise_name} for muscle: {muscle}")
                except Exception as e:
                    print(f"Error processing exercise row: {str(e)}")

            # Uncheck the muscle checkbox to prepare for the next iteration
            muscle_checkbox.click()
        except Exception as e:
            print(f"Error processing {muscle}: {str(e)}")
finally:
    # Quit the WebDriver instance
    driver.quit()

    # Write the data to a CSV file
    csv_filename = 'MuscleWiki_data_collection.csv'
    with open(csv_filename, 'w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)

        # Write header row
        writer.writerow(['Muscle Group', 'Exercise', 'Video Link (Male)', 'Video Link (Female)', 'Equipment', 'Difficulty'])

        # Write data rows
        writer.writerows(data)
        print(f"Data written to {csv_filename}")

    # Display the first few lines of the CSV for verification
    df = pd.read_csv(csv_filename)
    print(df.head(10))
```

Slide 6

Brief preview of Selenium and ChromeDriver being used


doc - JupyterLab
conda: jlab_server
MuscleWiki.com - Data Collection.ipynb
File Edit View Run Kernel Tabs Settings Help
Code
Python 3 (ipykernel)
Recording has started

Data Collection

1. Import Libraries and Set Up WebDriver:

```
[5]:
# Import necessary libraries
import csv
import time
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager

# Set up WebDriver service
service = Service(executable_path=ChromeDriverManager().install())

# Create a Chrome WebDriver instance
driver = webdriver.Chrome(service=service)

# Get the version of the ChromeDriver
print("ChromeDriver version:", driver.capabilities['chrome']['chromedriverVersion'])
```

```
ChromeDriver version: 125.0.6422.78 (14db42ec38aded3304a3e624a0a038e02956b87e-refs/branch-heads/6422@{#1088})
```

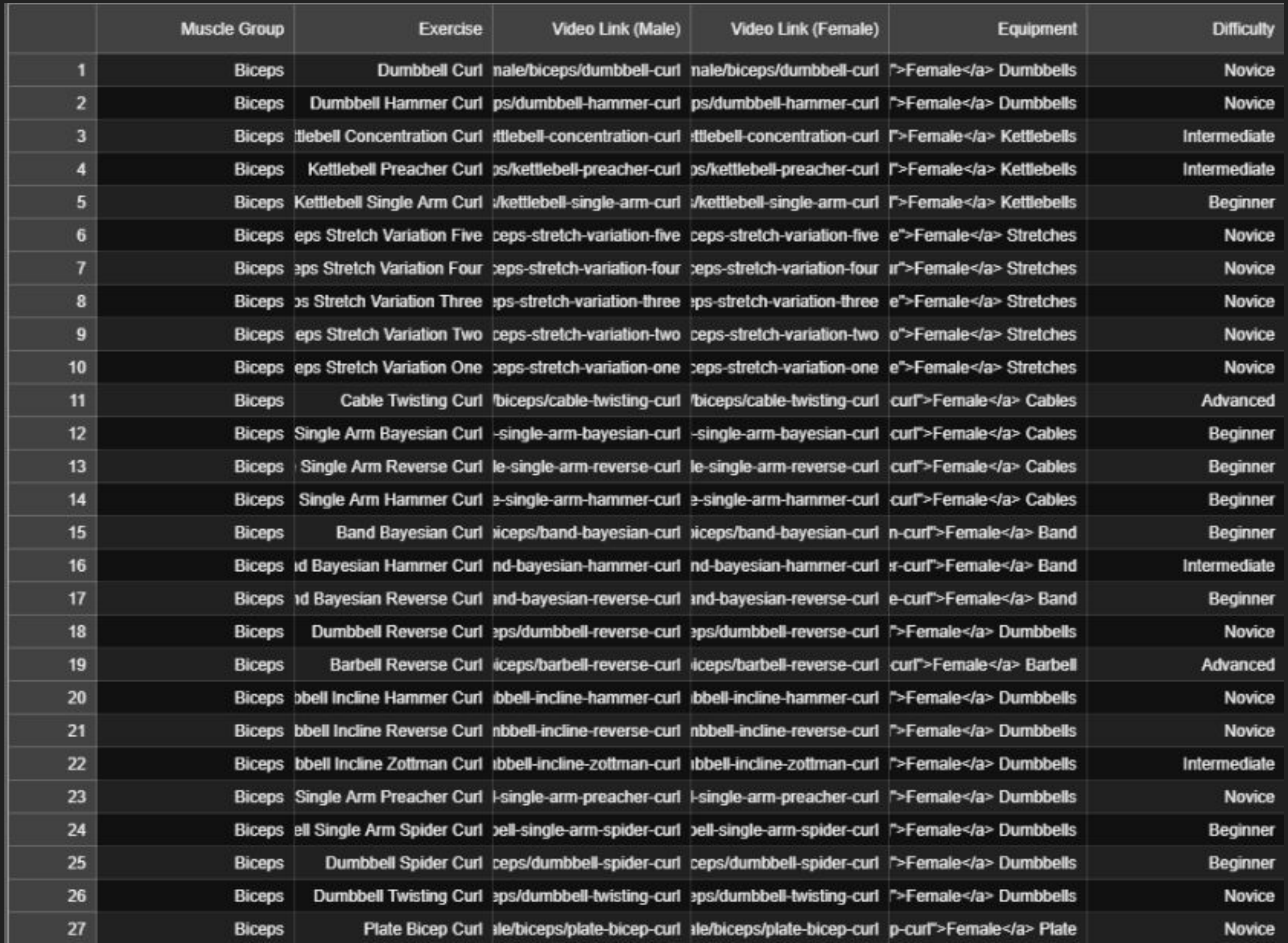
2. Define Muscle Groups and Initialize Data List:

```
[6]:
# Define the list of muscle groups to process
muscle_groups = [
    "Biceps", "Long Head Bicep", "Short Head Bicep", "Traps (mid-back)", "Lower back",
    "Abdominals", "Lower Abdominals", "Upper Abdominals", "Calves", "Tibialis",
    "Soleus", "Gastrocnemius", "Forearms", "Wrist Extensors", "Wrist Flexors",
    "Glutes", "Gluteus Medius", "Gluteus Maximus", "Hamstrings",
    "Medial Hamstrings", "Lateral Hamstrings", "Lats", "Shoulders",
    "Lateral Deltoid", "Anterior Deltoid", "Posterior Deltoid", "Triceps",
    "Long Head Tricep", "Lateral Head Triceps", "Medial Head Triceps", "Traps",
    "Upper Traps", "Lower Traps", "Quads", "Inner Thigh", "Inner Quadriceps",
    "Outer Quadricep", "Rectus Femoris", "Chest", "Upper Pectoralis",
    "Mid and Lower Chest", "Obliques", "Hands", "Front Shoulders", "Rear Shoulders"
]

# Initialize a list to store data
data = []
```

3. Open Webpage and Wait for Header:

Slide 7

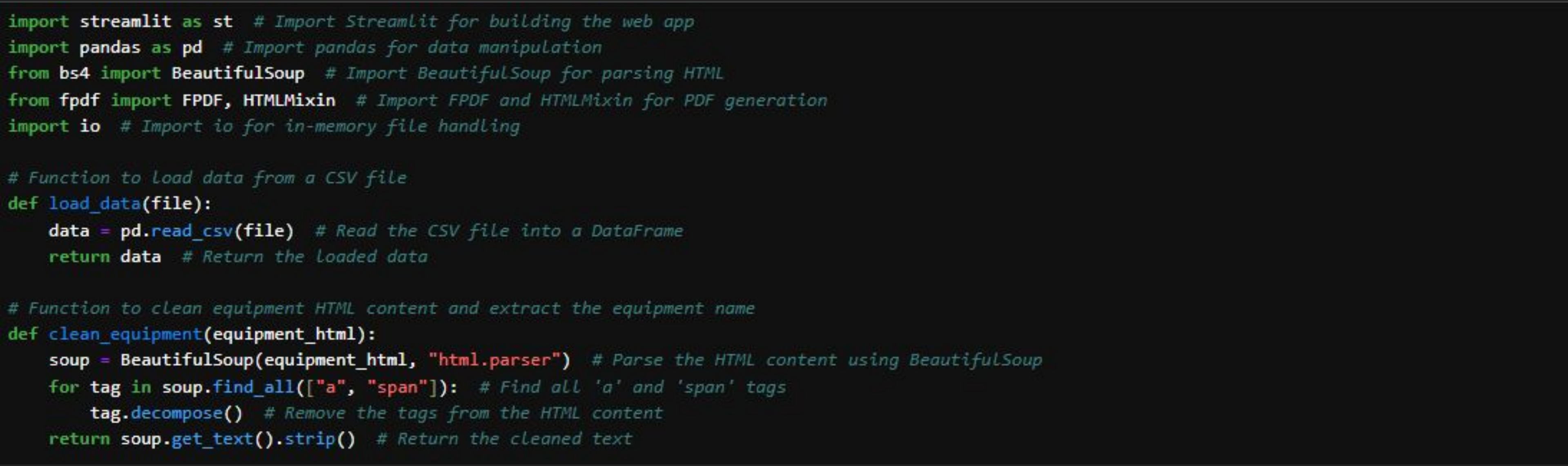
Sample of the collected data



	Muscle Group	Exercise	Video Link (Male)	Video Link (Female)	Equipment	Difficulty
1	Biceps	Dumbbell Curl	nale/biceps/dumbbell-curl	nale/biceps/dumbbell-curl	">Female</a> Dumbbells	Novice
2	Biceps	Dumbbell Hammer Curl	ps/dumbbell-hammer-curl	ps/dumbbell-hammer-curl	">Female</a> Dumbbells	Novice
3	Biceps	tlebell Concentration Curl	ttlebell-concentration-curl	ttlebell-concentration-curl	">Female</a> Kettlebells	Intermediate
4	Biceps	Kettlebell Preacher Curl	ps/kettlebell-preacher-curl	ps/kettlebell-preacher-curl	">Female</a> Kettlebells	Intermediate
5	Biceps	Kettlebell Single Arm Curl	/kettlebell-single-arm-curl	/kettlebell-single-arm-curl	">Female</a> Kettlebells	Beginner
6	Biceps	eps Stretch Variation Five	ceps-stretch-variation-five	ceps-stretch-variation-five	e">Female</a> Stretches	Novice
7	Biceps	eps Stretch Variation Four	ceps-stretch-variation-four	ceps-stretch-variation-four	ır">Female</a> Stretches	Novice
8	Biceps	ps Stretch Variation Three	eps-stretch-variation-three	eps-stretch-variation-three	e">Female</a> Stretches	Novice
9	Biceps	eps Stretch Variation Two	ceps-stretch-variation-two	ceps-stretch-variation-two	o">Female</a> Stretches	Novice
10	Biceps	eps Stretch Variation One	ceps-stretch-variation-one	ceps-stretch-variation-one	e">Female</a> Stretches	Novice
11	Biceps	Cable Twisting Curl	/biceps/cable-twisting-curl	/biceps/cable-twisting-curl	curl">Female</a> Cables	Advanced
12	Biceps	Single Arm Bayesian Curl	-single-arm-bayesian-curl	-single-arm-bayesian-curl	curl">Female</a> Cables	Beginner
13	Biceps	Single Arm Reverse Curl	le-single-arm-reverse-curl	le-single-arm-reverse-curl	curl">Female</a> Cables	Beginner
14	Biceps	Single Arm Hammer Curl	e-single-arm-hammer-curl	e-single-arm-hammer-curl	curl">Female</a> Cables	Beginner
15	Biceps	Band Bayesian Curl	iceps/band-bayesian-curl	iceps/band-bayesian-curl	n-curl">Female</a> Band	Beginner
16	Biceps	ıd Bayesian Hammer Curl	nd-bayesian-hammer-curl	nd-bayesian-hammer-curl	r-curl">Female</a> Band	Intermediate
17	Biceps	ıd Bayesian Reverse Curl	and-bayesian-reverse-curl	and-bayesian-reverse-curl	e-curl">Female</a> Band	Beginner
18	Biceps	Dumbbell Reverse Curl	eps/dumbbell-reverse-curl	eps/dumbbell-reverse-curl	">Female</a> Dumbbells	Novice
19	Biceps	Barbell Reverse Curl	iceps/barbell-reverse-curl	iceps/barbell-reverse-curl	curl">Female</a> Barbell	Advanced
20	Biceps	bbell Incline Hammer Curl	bbell-incline-hammer-curl	bbell-incline-hammer-curl	">Female</a> Dumbbells	Novice
21	Biceps	bbell Incline Reverse Curl	nbbell-incline-reverse-curl	nbbell-incline-reverse-curl	">Female</a> Dumbbells	Novice
22	Biceps	bbell Incline Zottman Curl	ıbbell-incline-zottman-curl	ıbbell-incline-zottman-curl	">Female</a> Dumbbells	Intermediate
23	Biceps	Single Arm Preacher Curl	l-single-arm-preacher-curl	l-single-arm-preacher-curl	">Female</a> Dumbbells	Novice
24	Biceps	ell Single Arm Spider Curl	bell-single-arm-spider-curl	bell-single-arm-spider-curl	">Female</a> Dumbbells	Beginner
25	Biceps	Dumbbell Spider Curl	ceps/dumbbell-spider-curl	ceps/dumbbell-spider-curl	">Female</a> Dumbbells	Beginner
26	Biceps	Dumbbell Twisting Curl	eps/dumbbell-twisting-curl	eps/dumbbell-twisting-curl	">Female</a> Dumbbells	Novice
27	Biceps	Plate Bicep Curl	ale/biceps/plate-bicep-curl	ale/biceps/plate-bicep-curl	p-curl">Female</a> Plate	Novice

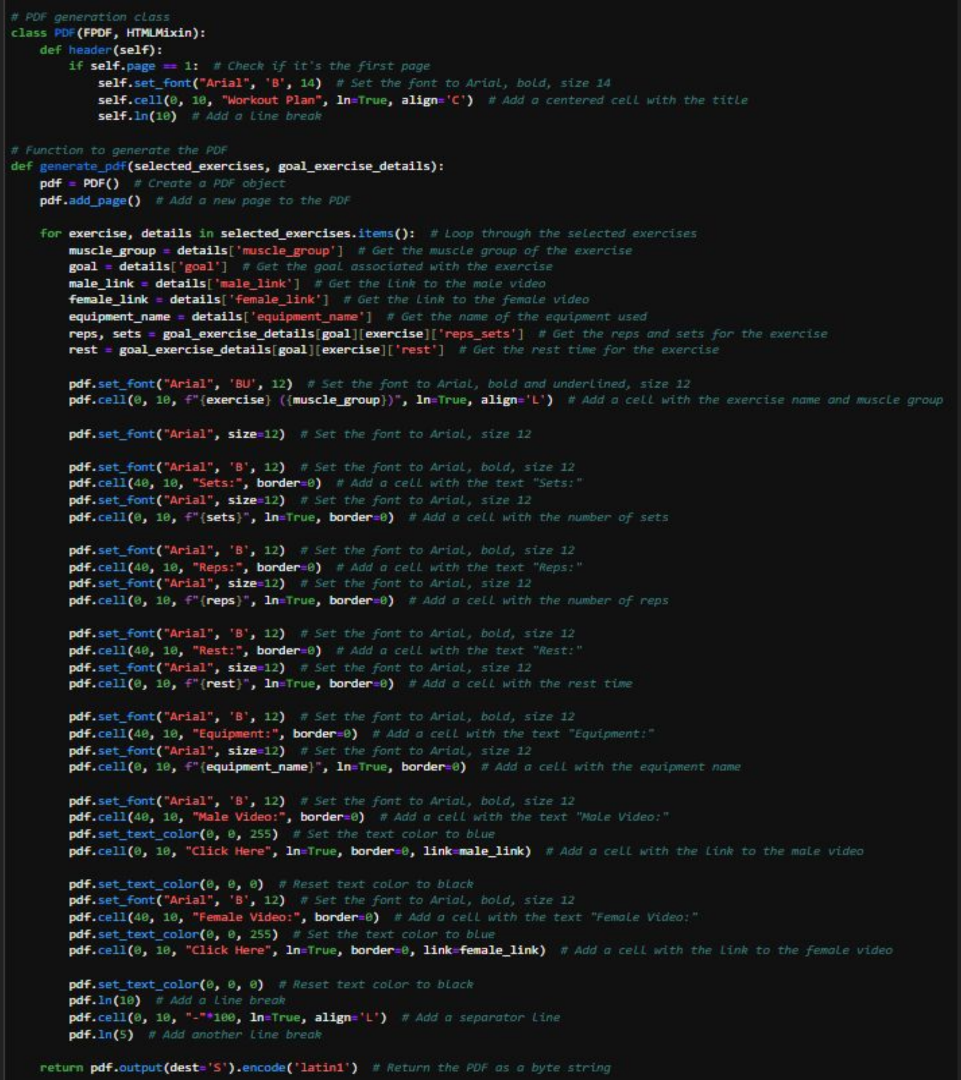
Streamlit script breakdown with detailed explanations

Slide 9



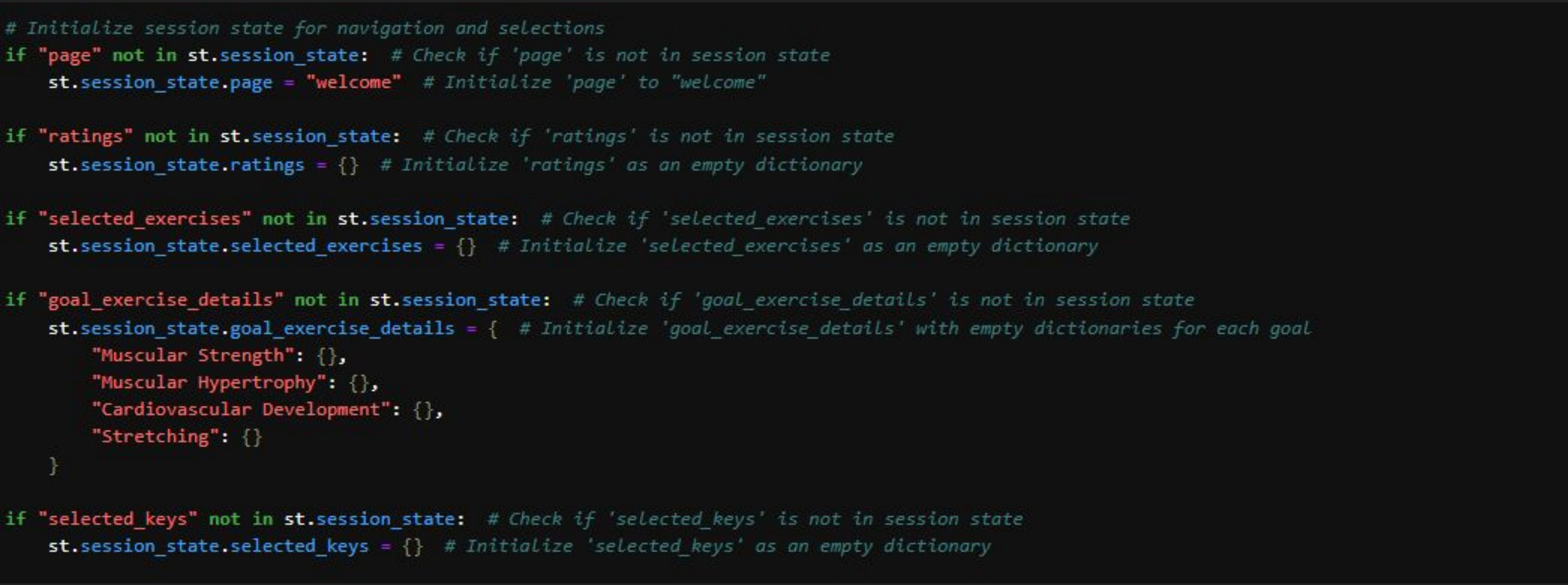
```python
import streamlit as st  # Import Streamlit for building the web app
import pandas as pd  # Import pandas for data manipulation
from bs4 import BeautifulSoup  # Import BeautifulSoup for parsing HTML
from fpdf import FPDF, HTMLMixin  # Import FPDF and HTMLMixin for PDF generation
import io  # Import io for in-memory file handling

# Function to load data from a CSV file
def load_data(file):
    data = pd.read_csv(file)  # Read the CSV file into a DataFrame
    return data  # Return the loaded data

# Function to clean equipment HTML content and extract the equipment name
def clean_equipment(equipment_html):
    soup = BeautifulSoup(equipment_html, "html.parser")  # Parse the HTML content using BeautifulSoup
    for tag in soup.find_all(["a", "span"]):  # Find all 'a' and 'span' tags
        tag.decompose()  # Remove the tags from the HTML content
    return soup.get_text().strip()  # Return the cleaned text
```

Slide 10



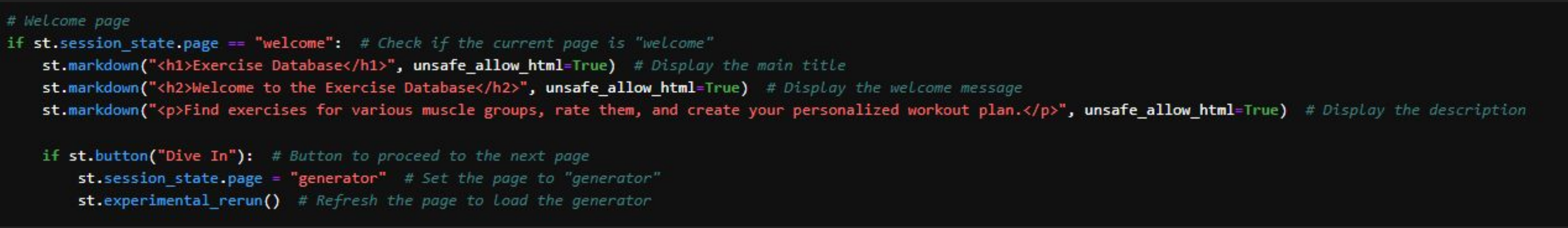
```python
# PDF generation class
class PDF(FPDF, HTMLMixin):
    def header(self):
        if self.page == 1:  # Check if it's the first page
            self.set_font("Arial", 'B', 14)  # Set the font to Arial, bold, size 14
            self.cell(0, 10, "Workout Plan", ln=True, align='C')  # Add a centered cell with the title
            self.ln(10)  # Add a line break

# Function to generate the PDF
def generate_pdf(selected_exercises, goal_exercise_details):
    pdf = PDF()  # Create a PDF object
    pdf.add_page()  # Add a new page to the PDF

    for exercise, details in selected_exercises.items():  # Loop through the selected exercises
        muscle_group = details['muscle_group']  # Get the muscle group of the exercise
        goal = details['goal']  # Get the goal associated with the exercise
        male_link = details['male_link']  # Get the link to the male video
        female_link = details['female_link']  # Get the link to the female video
        equipment_name = details['equipment_name']  # Get the name of the equipment used
        reps, sets = goal_exercise_details[goal][exercise]['reps_sets']  # Get the reps and sets for the exercise
        rest = goal_exercise_details[goal][exercise]['rest']  # Get the rest time for the exercise

        pdf.set_font("Arial", 'BU', 12)  # Set the font to Arial, bold and underlined, size 12
        pdf.cell(0, 10, f"{exercise} ({muscle_group})", ln=True, align='L')  # Add a cell with the exercise name and muscle group

        pdf.set_font("Arial", size=12)  # Set the font to Arial, size 12

        pdf.set_font("Arial", 'B', 12)  # Set the font to Arial, bold, size 12
        pdf.cell(40, 10, "Sets:", border=0)  # Add a cell with the text "Sets:"
        pdf.set_font("Arial", size=12)  # Set the font to Arial, size 12
        pdf.cell(0, 10, f"{sets}", ln=True, border=0)  # Add a cell with the number of sets

        pdf.set_font("Arial", 'B', 12)  # Set the font to Arial, bold, size 12
        pdf.cell(40, 10, "Reps:", border=0)  # Add a cell with the text "Reps:"
        pdf.set_font("Arial", size=12)  # Set the font to Arial, size 12
        pdf.cell(0, 10, f"{reps}", ln=True, border=0)  # Add a cell with the number of reps

        pdf.set_font("Arial", 'B', 12)  # Set the font to Arial, bold, size 12
        pdf.cell(40, 10, "Rest:", border=0)  # Add a cell with the text "Rest:"
        pdf.set_font("Arial", size=12)  # Set the font to Arial, size 12
        pdf.cell(0, 10, f"{rest}", ln=True, border=0)  # Add a cell with the rest time

        pdf.set_font("Arial", 'B', 12)  # Set the font to Arial, bold, size 12
        pdf.cell(40, 10, "Equipment:", border=0)  # Add a cell with the text "Equipment:"
        pdf.set_font("Arial", size=12)  # Set the font to Arial, size 12
        pdf.cell(0, 10, f"{equipment_name}", ln=True, border=0)  # Add a cell with the equipment name

        pdf.set_font("Arial", 'B', 12)  # Set the font to Arial, bold, size 12
        pdf.cell(40, 10, "Male Video:", border=0)  # Add a cell with the text "Male Video:"
        pdf.set_text_color(0, 0, 255)  # Set the text color to blue
        pdf.cell(0, 10, "Click Here", ln=True, border=0, link=male_link)  # Add a cell with the link to the male video

        pdf.set_text_color(0, 0, 0)  # Reset text color to black
        pdf.set_font("Arial", 'B', 12)  # Set the font to Arial, bold, size 12
        pdf.cell(40, 10, "Female Video:", border=0)  # Add a cell with the text "Female Video:"
        pdf.set_text_color(0, 0, 255)  # Set the text color to blue
        pdf.cell(0, 10, "Click Here", ln=True, border=0, link=female_link)  # Add a cell with the link to the female video

        pdf.set_text_color(0, 0, 0)  # Reset text color to black
        pdf.ln(10)  # Add a line break
        pdf.cell(0, 10, "-"*100, ln=True, align='L')  # Add a separator line
        pdf.ln(5)  # Add another line break

    return pdf.output(dest='S').encode('latin1')  # Return the PDF as a byte string
```

Slide 11



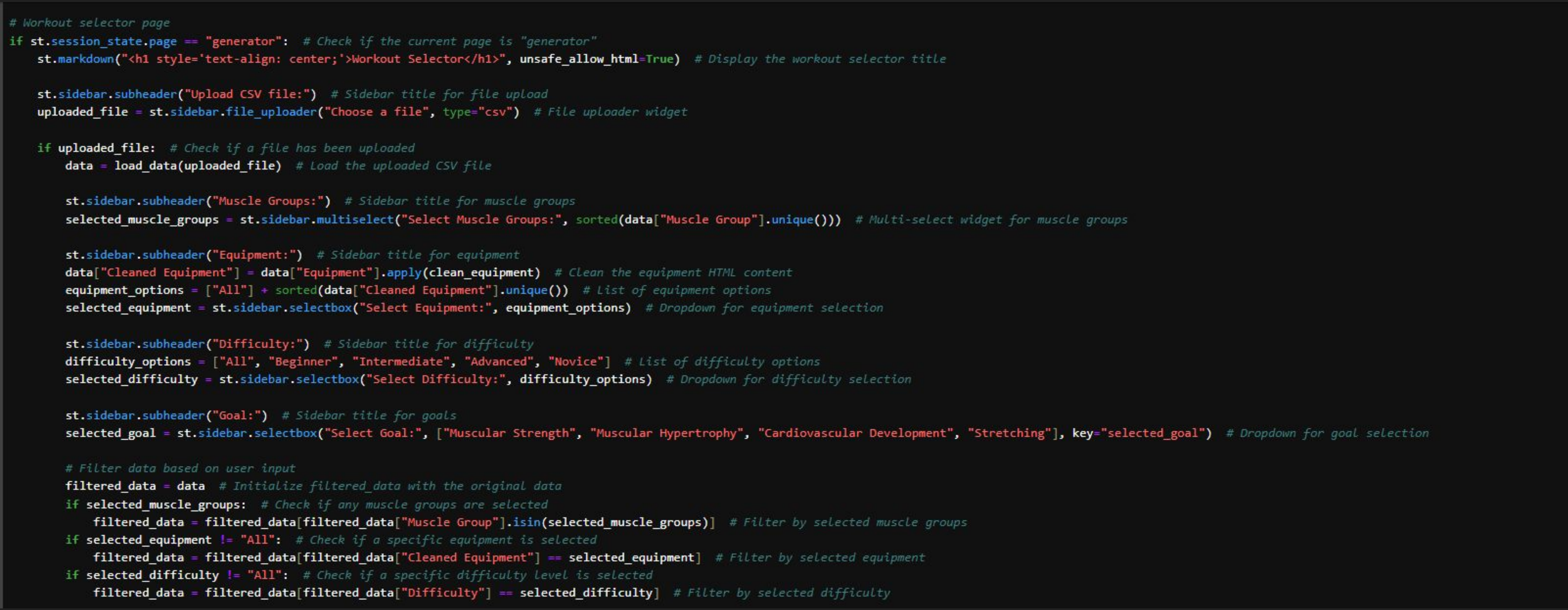
```python
# Initialize session state for navigation and selections
if "page" not in st.session_state:  # Check if 'page' is not in session state
    st.session_state.page = "welcome"  # Initialize 'page' to "welcome"

if "ratings" not in st.session_state:  # Check if 'ratings' is not in session state
    st.session_state.ratings = {}  # Initialize 'ratings' as an empty dictionary

if "selected_exercises" not in st.session_state:  # Check if 'selected_exercises' is not in session state
    st.session_state.selected_exercises = {}  # Initialize 'selected_exercises' as an empty dictionary

if "goal_exercise_details" not in st.session_state:  # Check if 'goal_exercise_details' is not in session state
    st.session_state.goal_exercise_details = {  # Initialize 'goal_exercise_details' with empty dictionaries for each goal
        "Muscular Strength": {},
        "Muscular Hypertrophy": {},
        "Cardiovascular Development": {},
        "Stretching": {}
    }

if "selected_keys" not in st.session_state:  # Check if 'selected_keys' is not in session state
    st.session_state.selected_keys = {}  # Initialize 'selected_keys' as an empty dictionary
```

Slide 12



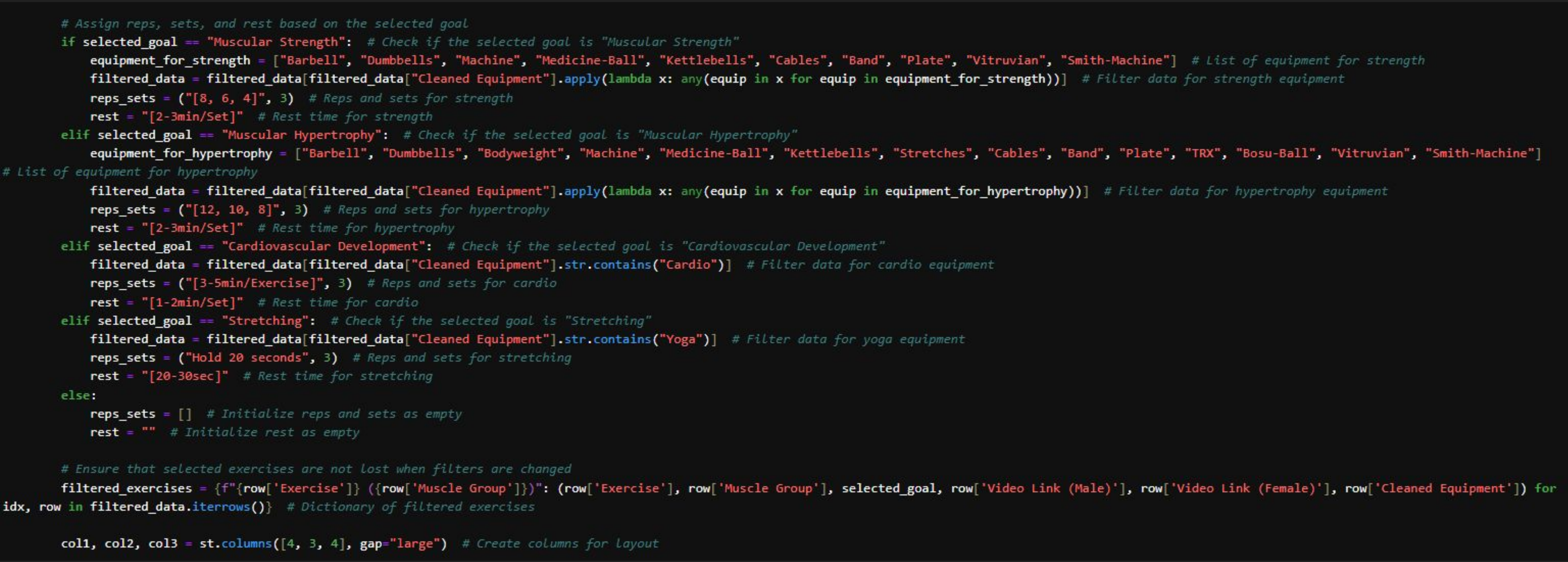
```python
# Welcome page
if st.session_state.page == "welcome":  # Check if the current page is "welcome"
    st.markdown("<h1>Exercise Database</h1>", unsafe_allow_html=True)  # Display the main title
    st.markdown("<h2>Welcome to the Exercise Database</h2>", unsafe_allow_html=True)  # Display the welcome message
    st.markdown("<p>Find exercises for various muscle groups, rate them, and create your personalized workout plan.</p>", unsafe_allow_html=True)  # Display the description

    if st.button("Dive In"):  # Button to proceed to the next page
        st.session_state.page = "generator"  # Set the page to "generator"
        st.experimental_rerun()  # Refresh the page to load the generator
```

Slide 13



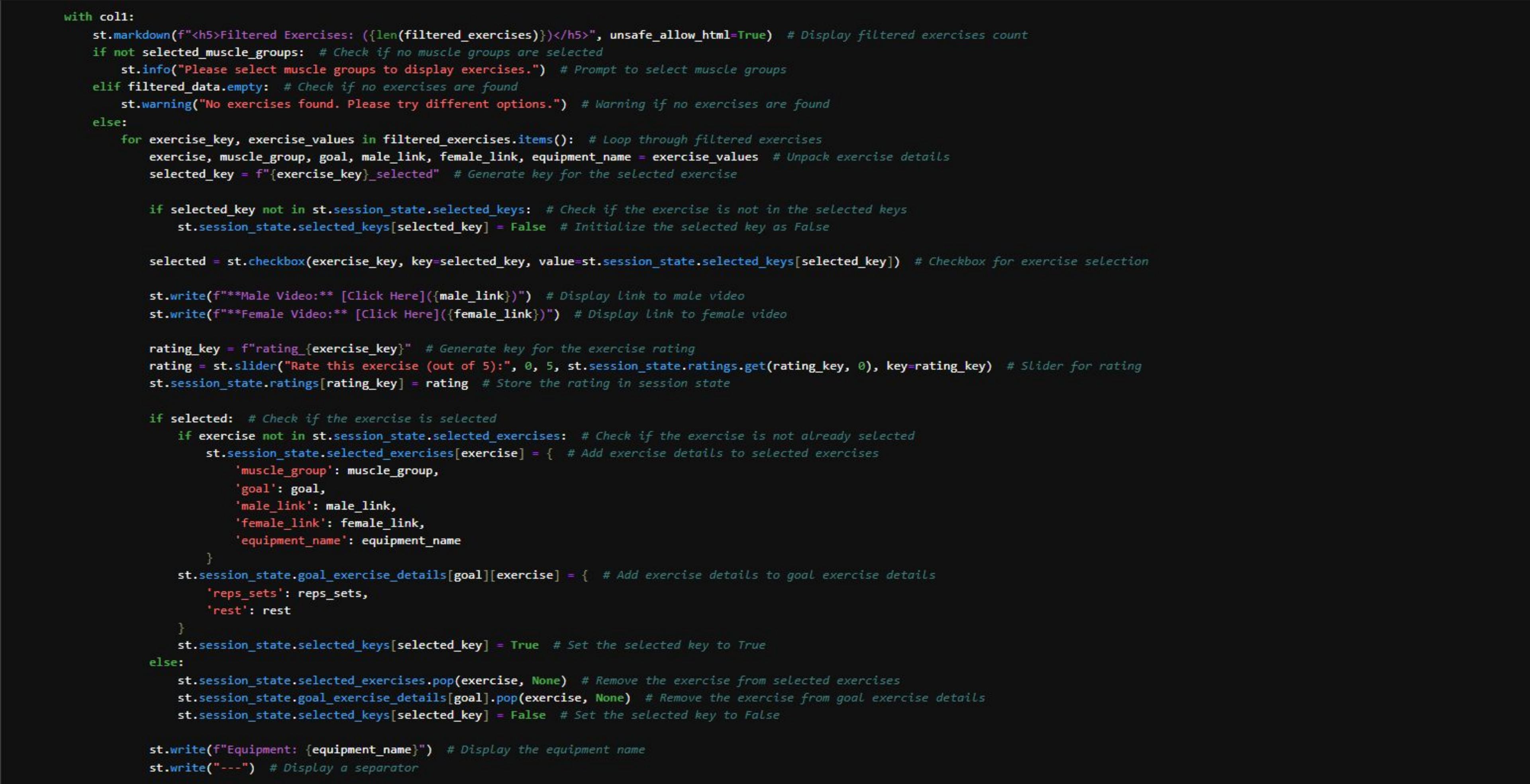
```python
# Workout selector page
if st.session_state.page == "generator":  # Check if the current page is "generator"
    st.markdown("<h1 style='text-align: center;'>Workout Selector</h1>", unsafe_allow_html=True)  # Display the workout selector title

    st.sidebar.subheader("Upload CSV file:")  # Sidebar title for file upload
    uploaded_file = st.sidebar.file_uploader("Choose a file", type="csv")  # File uploader widget

    if uploaded_file:  # Check if a file has been uploaded
        data = load_data(uploaded_file)  # Load the uploaded CSV file

        st.sidebar.subheader("Muscle Groups:")  # Sidebar title for muscle groups
        selected_muscle_groups = st.sidebar.multiselect("Select Muscle Groups:", sorted(data["Muscle Group"].unique()))  # Multi-select widget for muscle groups

        st.sidebar.subheader("Equipment:")  # Sidebar title for equipment
        data["Cleaned Equipment"] = data["Equipment"].apply(clean_equipment)  # Clean the equipment HTML content
        equipment_options = ["All"] + sorted(data["Cleaned Equipment"].unique())  # List of equipment options
        selected_equipment = st.sidebar.selectbox("Select Equipment:", equipment_options)  # Dropdown for equipment selection

        st.sidebar.subheader("Difficulty:")  # Sidebar title for difficulty
        difficulty_options = ["All", "Beginner", "Intermediate", "Advanced", "Novice"]  # List of difficulty options
        selected_difficulty = st.sidebar.selectbox("Select Difficulty:", difficulty_options)  # Dropdown for difficulty selection

        st.sidebar.subheader("Goal:")  # Sidebar title for goals
        selected_goal = st.sidebar.selectbox("Select Goal:", ["Muscular Strength", "Muscular Hypertrophy", "Cardiovascular Development", "Stretching"], key="selected_goal")  # Dropdown for goal selection

        # Filter data based on user input
        filtered_data = data  # Initialize filtered_data with the original data
        if selected_muscle_groups:  # Check if any muscle groups are selected
            filtered_data = filtered_data[filtered_data["Muscle Group"].isin(selected_muscle_groups)]  # Filter by selected muscle groups
        if selected_equipment != "All":  # Check if a specific equipment is selected
            filtered_data = filtered_data[filtered_data["Cleaned Equipment"] == selected_equipment]  # Filter by selected equipment
        if selected_difficulty != "All":  # Check if a specific difficulty level is selected
            filtered_data = filtered_data[filtered_data["Difficulty"] == selected_difficulty]  # Filter by selected difficulty
```

Slide 14



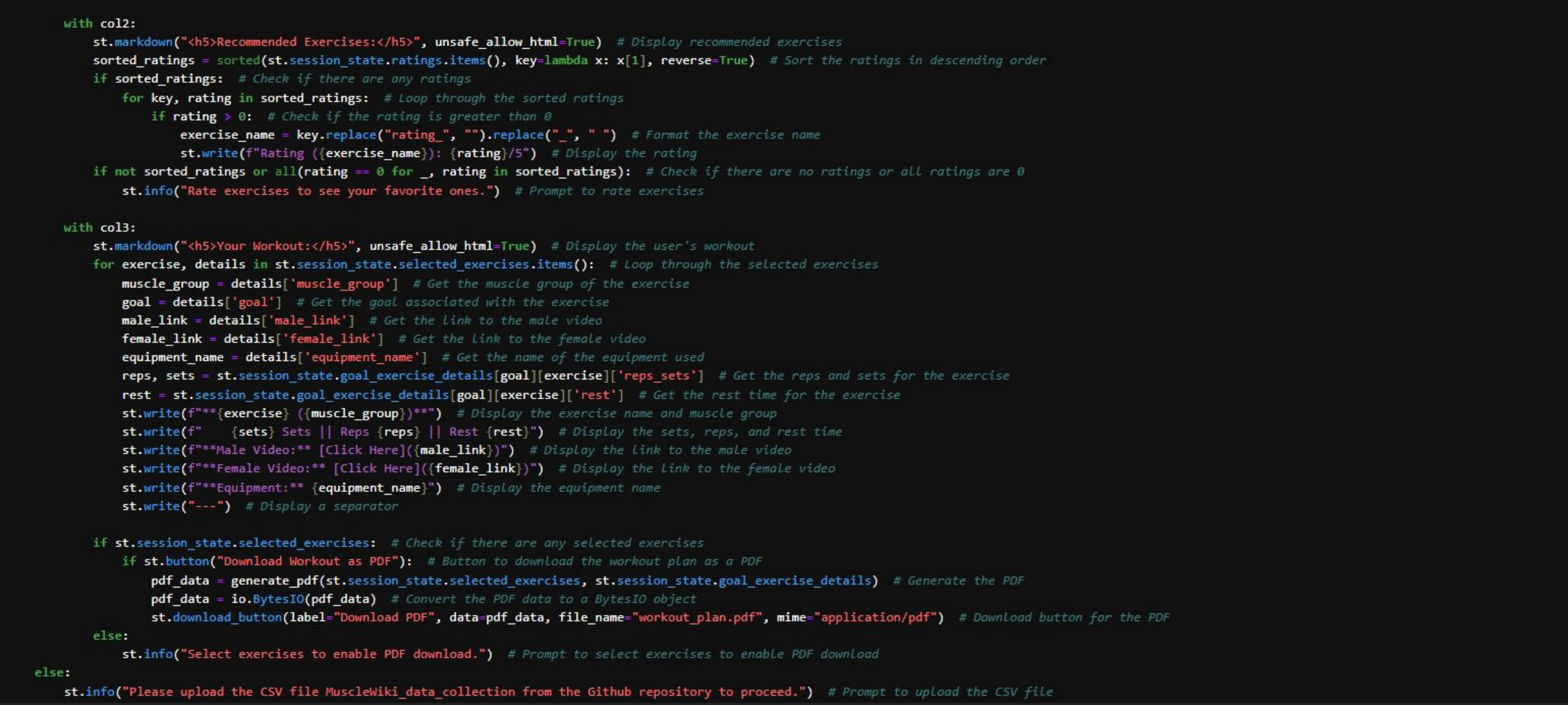
```
        # Assign reps, sets, and rest based on the selected goal
        if selected_goal == "Muscular Strength":  # Check if the selected goal is "Muscular Strength"
            equipment_for_strength = ["Barbell", "Dumbbells", "Machine", "Medicine-Ball", "Kettlebells", "Cables", "Band", "Plate", "Vitruvian", "Smith-Machine"]  # List of equipment for strength
            filtered_data = filtered_data[filtered_data["Cleaned Equipment"].apply(lambda x: any(equip in x for equip in equipment_for_strength))]  # Filter data for strength equipment
            reps_sets = ("[8, 6, 4]", 3)  # Reps and sets for strength
            rest = "[2-3min/Set]"  # Rest time for strength
        elif selected_goal == "Muscular Hypertrophy":  # Check if the selected goal is "Muscular Hypertrophy"
            equipment_for_hypertrophy = ["Barbell", "Dumbbells", "Bodyweight", "Machine", "Medicine-Ball", "Kettlebells", "Stretches", "Cables", "Band", "Plate", "TRX", "Bosu-Ball", "Vitruvian", "Smith-Machine"]
# List of equipment for hypertrophy
            filtered_data = filtered_data[filtered_data["Cleaned Equipment"].apply(lambda x: any(equip in x for equip in equipment_for_hypertrophy))]  # Filter data for hypertrophy equipment
            reps_sets = ("[12, 10, 8]", 3)  # Reps and sets for hypertrophy
            rest = "[2-3min/Set]"  # Rest time for hypertrophy
        elif selected_goal == "Cardiovascular Development":  # Check if the selected goal is "Cardiovascular Development"
            filtered_data = filtered_data[filtered_data["Cleaned Equipment"].str.contains("Cardio")]  # Filter data for cardio equipment
            reps_sets = ("[3-5min/Exercise]", 3)  # Reps and sets for cardio
            rest = "[1-2min/Set]"  # Rest time for cardio
        elif selected_goal == "Stretching":  # Check if the selected goal is "Stretching"
            filtered_data = filtered_data[filtered_data["Cleaned Equipment"].str.contains("Yoga")]  # Filter data for yoga equipment
            reps_sets = ("Hold 20 seconds", 3)  # Reps and sets for stretching
            rest = "[20-30sec]"  # Rest time for stretching
        else:
            reps_sets = []  # Initialize reps and sets as empty
            rest = ""  # Initialize rest as empty

        # Ensure that selected exercises are not lost when filters are changed
        filtered_exercises = {f"{row['Exercise']} ({row['Muscle Group']})": (row['Exercise'], row['Muscle Group'], selected_goal, row['Video Link (Male)'], row['Video Link (Female)'], row['Cleaned Equipment']) for
idx, row in filtered_data.iterrows()}  # Dictionary of filtered exercises

        col1, col2, col3 = st.columns([4, 3, 4], gap="large")  # Create columns for layout
```

Slide 15



```python
with col1:
    st.markdown(f"<h5>Filtered Exercises: ({len(filtered_exercises)})</h5>", unsafe_allow_html=True)  # Display filtered exercises count
    if not selected_muscle_groups:  # Check if no muscle groups are selected
        st.info("Please select muscle groups to display exercises.")  # Prompt to select muscle groups
    elif filtered_data.empty:  # Check if no exercises are found
        st.warning("No exercises found. Please try different options.")  # Warning if no exercises are found
    else:
        for exercise_key, exercise_values in filtered_exercises.items():  # Loop through filtered exercises
            exercise, muscle_group, goal, male_link, female_link, equipment_name = exercise_values  # Unpack exercise details
            selected_key = f"{exercise_key}_selected"  # Generate key for the selected exercise

            if selected_key not in st.session_state.selected_keys:  # Check if the exercise is not in the selected keys
                st.session_state.selected_keys[selected_key] = False  # Initialize the selected key as False

            selected = st.checkbox(exercise_key, key=selected_key, value=st.session_state.selected_keys[selected_key])  # Checkbox for exercise selection

            st.write(f"**Male Video:** [Click Here]({male_link})")  # Display link to male video
            st.write(f"**Female Video:** [Click Here]({female_link})")  # Display link to female video

            rating_key = f"rating_{exercise_key}"  # Generate key for the exercise rating
            rating = st.slider("Rate this exercise (out of 5):", 0, 5, st.session_state.ratings.get(rating_key, 0), key=rating_key)  # Slider for rating
            st.session_state.ratings[rating_key] = rating  # Store the rating in session state

            if selected:  # Check if the exercise is selected
                if exercise not in st.session_state.selected_exercises:  # Check if the exercise is not already selected
                    st.session_state.selected_exercises[exercise] = {  # Add exercise details to selected exercises
                        'muscle_group': muscle_group,
                        'goal': goal,
                        'male_link': male_link,
                        'female_link': female_link,
                        'equipment_name': equipment_name
                    }
                st.session_state.goal_exercise_details[goal][exercise] = {  # Add exercise details to goal exercise details
                    'reps_sets': reps_sets,
                    'rest': rest
                }
                st.session_state.selected_keys[selected_key] = True  # Set the selected key to True
            else:
                st.session_state.selected_exercises.pop(exercise, None)  # Remove the exercise from selected exercises
                st.session_state.goal_exercise_details[goal].pop(exercise, None)  # Remove the exercise from goal exercise details
                st.session_state.selected_keys[selected_key] = False  # Set the selected key to False

            st.write(f"Equipment: {equipment_name}")  # Display the equipment name
            st.write("---")  # Display a separator
```

Slide 16



```python
    with col2:
        st.markdown("<h5>Recommended Exercises:</h5>", unsafe_allow_html=True)  # Display recommended exercises
        sorted_ratings = sorted(st.session_state.ratings.items(), key=lambda x: x[1], reverse=True)  # Sort the ratings in descending order
        if sorted_ratings:  # Check if there are any ratings
            for key, rating in sorted_ratings:  # Loop through the sorted ratings
                if rating > 0:  # Check if the rating is greater than 0
                    exercise_name = key.replace("rating_", "").replace("_", " ")  # Format the exercise name
                    st.write(f"Rating ({exercise_name}): {rating}/5")  # Display the rating
        if not sorted_ratings or all(rating == 0 for _, rating in sorted_ratings):  # Check if there are no ratings or all ratings are 0
            st.info("Rate exercises to see your favorite ones.")  # Prompt to rate exercises

    with col3:
        st.markdown("<h5>Your Workout:</h5>", unsafe_allow_html=True)  # Display the user's workout
        for exercise, details in st.session_state.selected_exercises.items():  # Loop through the selected exercises
            muscle_group = details['muscle_group']  # Get the muscle group of the exercise
            goal = details['goal']  # Get the goal associated with the exercise
            male_link = details['male_link']  # Get the link to the male video
            female_link = details['female_link']  # Get the link to the female video
            equipment_name = details['equipment_name']  # Get the name of the equipment used
            reps, sets = st.session_state.goal_exercise_details[goal][exercise]['reps_sets']  # Get the reps and sets for the exercise
            rest = st.session_state.goal_exercise_details[goal][exercise]['rest']  # Get the rest time for the exercise
            st.write(f"**{exercise} ({muscle_group})**")  # Display the exercise name and muscle group
            st.write(f"    {sets} Sets || Reps {reps} || Rest {rest}")  # Display the sets, reps, and rest time
            st.write(f"**Male Video:** [Click Here]({male_link})")  # Display the link to the male video
            st.write(f"**Female Video:** [Click Here]({female_link})")  # Display the link to the female video
            st.write(f"**Equipment:** {equipment_name}")  # Display the equipment name
            st.write("---")  # Display a separator

        if st.session_state.selected_exercises:  # Check if there are any selected exercises
            if st.button("Download Workout as PDF"):  # Button to download the workout plan as a PDF
                pdf_data = generate_pdf(st.session_state.selected_exercises, st.session_state.goal_exercise_details)  # Generate the PDF
                pdf_data = io.BytesIO(pdf_data)  # Convert the PDF data to a BytesIO object
                st.download_button(label="Download PDF", data=pdf_data, file_name="workout_plan.pdf", mime="application/pdf")  # Download button for the PDF
        else:
            st.info("Select exercises to enable PDF download.")  # Prompt to select exercises to enable PDF download
else:
    st.info("Please upload the CSV file MuscleWiki_data_collection from the Github repository to proceed.")  # Prompt to upload the CSV file
```

Streamlit app

Click Here

Conclusion