DAY-28

Pomodoro GUI Application

In [2]:
```
# ---------------------------- CONSTANTS ------------------------------- #
PINK = "#e2979c"
RED = "#e7305b"
GREEN = "#9bdeac"
YELLOW = "#f7f5dd"
FONT_NAME = "Courier"
WORK_MIN = 25
SHORT_BREAK_MIN = 5
LONG_BREAK_MIN = 20

# ---------------------------- TIMER RESET ------------------------------- #

# ---------------------------- TIMER MECHANISM ------------------------------- #

# ---------------------------- COUNTDOWN MECHANISM ------------------------------- #

# ---------------------------- UI SETUP ------------------------------- #

import tkinter

window=tkinter.Tk()

window.title("Pomodoro")

#Adding background color in the window
window.config(padx=100,pady=50,bg=YELLOW)

#This canvas is used to add image in the gui and change the size and to do other things
#If we need to add the wigets into the image then we need to use canvas
#In order to add bg color in the image we add bg as yellow in canvas
canvas=tkinter.Canvas(width=200,height=224,bg=YELLOW,highlightthickness=0)#highlightthickness is used to r

#The canvas does not take the png format image so the PhotoImage takes the png image and convert it into r
tomato_png=tkinter.PhotoImage(file="tomato.png")

#Creating an image
canvas.create_image(100,112,image=tomato_png)

#To add text into the image we use create_text method in the canvas
canvas.create_text(100,130,text="00:00",fill="white",font=(FONT_NAME,35,"bold"))

#Adding the canvasimage into the window
canvas.pack()

window.mainloop()
```

In [5]:
```
# Challange:Adding  Start Reset Buttons Tick Mark and  Timer Text into the pomodoro GUI Application


# ---------------------------- CONSTANTS ------------------------------- #
PINK = "#e2979c"
RED = "#e7305b"
GREEN = "#9bdeac"
YELLOW = "#f7f5dd"
FONT_NAME = "Courier"
WORK_MIN = 25
SHORT_BREAK_MIN = 5
LONG_BREAK_MIN = 20

# ---------------------------- TIMER RESET ------------------------------- #

# ---------------------------- TIMER MECHANISM ------------------------------- #
```

```
# ---------------------------- COUNTDOWN MECHANISM ------------------------------- #

# ---------------------------- UI SETUP ------------------------------- #

import tkinter

window=tkinter.Tk()

window.title("Pomodoro")

window.config(padx=100,pady=50,bg=YELLOW)

canvas=tkinter.Canvas(width=200,height=224,bg=YELLOW,highlightthickness=0)

tomato_png=tkinter.PhotoImage(file="tomato.png")

canvas.create_image(100,112,image=tomato_png)

canvas.create_text(100,130,text="00:00",fill="white",font=(FONT_NAME,35,"bold"))

canvas.grid(column=1,row=1)
label1=tkinter.Label(text="Timer",fg=GREEN,font=(FONT_NAME,30,"bold"),bg=YELLOW)
label1.grid(column=1,row=0)

label2=tkinter.Label(text="✓",fg=GREEN,bg=YELLOW)
label2.grid(column=1,row=4)

button1=tkinter.Button(text="Start")
button1.grid(column=0,row=3)

button2=tkinter.Button(text="Reset")
button2.grid(column=2,row=3)

window.mainloop()
```

In [ ]:

In [2]:

```
#### After Function to sleep the tkinter

import tkinter

window=tkinter.Tk()
def say_something(a,b,c):
    print(a,b,c)
#Here it delays for 1 seconf(1000 mili second) and adding a function in the 2 parameter and in 3rd paramet
#Here we add arguments for say_something function
window.after(1000,say_something,1,2,3)

window.mainloop()
```

```
1 2 3
```

In [39]:

```
# Adding Countdown and tick marks
import tkinter
import math

# ---------------------------- CONSTANTS ------------------------------- #
PINK = "#e2979c"
RED = "#e7305b"
GREEN = "#9bdeac"
YELLOW = "#f7f5dd"
FONT_NAME = "Courier"
WORK_MIN = 25
SHORT_BREAK_MIN = 5
LONG_BREAK_MIN = 20
reps=0

# ---------------------------- TIMER RESET ------------------------------- #
```

```
# ---------------------------- TIMER MECHANISM ------------------------------- #

def start_timer():
    global reps
    reps+=1

    work_sec=WORK_MIN *60
    short_break_sex=SHORT_BREAK_MIN *60
    long_break_sec=LONG_BREAK_MIN * 60

    if reps % 8 == 0:
        count_down(long_break_sec)
        label1.config(text="Break",fg=RED)
    elif reps % 2 ==0:
        count_down(short_break_sec)
        label1.config(text="Break",fg=PINK)
    else:
        count_down(work_sec)
        label1.config(text="Work",fg=GREEN)


# ---------------------------- COUNTDOWN MECHANISM ------------------------------- #

def count_down(count):
    count_min=math.floor(count/60)
    count_sec=count%60
    if count_sec <10:
        count_sec="0"+str(count_sec)
    if count_sec==0:
        count_sec="00"


    canvas.itemconfig(timer_text,text=f"{count_min}:{count_sec}")
    if count >0:
        window.after(1000,count_down,count-1)

    else:
        start_timer()
        marks = ""
        work_sessions = math.floor(reps/2)
        for _ in range(work_sessions):
            marks += "✓"
        check_marks.config(text=marks)


# ---------------------------- UI SETUP ------------------------------- #

import tkinter

window=tkinter.Tk()

window.title("Pomodoro")

window.config(padx=100,pady=50,bg=YELLOW)



canvas=tkinter.Canvas(width=200,height=224,bg=YELLOW,highlightthickness=0)

tomato_png=tkinter.PhotoImage(file="tomato.png")

canvas.create_image(100,112,image=tomato_png)
timer_text=canvas.create_text(100,130,text="00:00",fill="white",font=(FONT_NAME,35,"bold"))

canvas.grid(column=1,row=1)
label1=tkinter.Label(text="Timer",fg=GREEN,font=(FONT_NAME,30,"bold"),bg=YELLOW)
label1.grid(column=1,row=0)

label2=tkinter.Label(text="✓",fg=GREEN,bg=YELLOW)
label2.grid(column=1,row=4)

button1=tkinter.Button(text="Start",command=start_timer)
```

```
button1.grid(column=0,row=3)

button2=tkinter.Button(text="Reset")
button2.grid(column=2,row=3)



window.mainloop()
```

Pomodoro Final Code

In [2]:

```
# Resetting timer
import tkinter
import math

# ---------------------------- CONSTANTS ------------------------------- #
PINK = "#e2979c"
RED = "#e7305b"
GREEN = "#9bdeac"
YELLOW = "#f7f5dd"
FONT_NAME = "Courier"
WORK_MIN = 1
SHORT_BREAK_MIN = 25
LONG_BREAK_MIN = 20
reps=0
timer=None

# ---------------------------- TIMER RESET ------------------------------- #
def reset_timer():
    window.after_cancel(timer)
    label1.config(text="Timer",fg=GREEN)
    canvas.itemconfig(timer_text,text="00:00")
    label_2.config(text="")
# ---------------------------- TIMER MECHANISM ------------------------------- #

def start_timer():
    global reps
    reps+=1

    work_sec=WORK_MIN *60
    short_break_sex=SHORT_BREAK_MIN *60
    long_break_sec=LONG_BREAK_MIN * 60

    if reps % 8 == 0:
        count_down(long_break_sec)
        label1.config(text="Break",fg=RED)
    elif reps % 2 ==0:
        count_down(short_break_sec)
        label1.config(text="Break",fg=PINK)
    else:
        count_down(work_sec)
        label1.config(text="Work",fg=GREEN)


# ---------------------------- COUNTDOWN MECHANISM ------------------------------- #

def count_down(count):
    count_min=math.floor(count/60)
    count_sec=count%60
    if count_sec <10:
        count_sec="0"+str(count_sec)
    if count_sec==0:
        count_sec="00"


    canvas.itemconfig(timer_text,text=f"{count_min}:{count_sec}")
    if count >0:
        global timer
        timer=window.after(1000,count_down,count-1)
```

```
    else:
        start_timer()
        marks = ""
        work_sessions = math.floor(reps/2)
        for _ in range(work_sessions):
            marks += "✓"
        label2.config(text=marks)


# ---------------------------- UI SETUP ------------------------------- #

import tkinter

window=tkinter.Tk()

window.title("Pomodoro")

window.config(padx=100,pady=50,bg=YELLOW)



canvas=tkinter.Canvas(width=200,height=224,bg=YELLOW,highlightthickness=0)

tomato_png=tkinter.PhotoImage(file="tomato.png")

canvas.create_image(100,112,image=tomato_png)
timer_text=canvas.create_text(100,130,text="00:00",fill="white",font=(FONT_NAME,35,"bold"))

canvas.grid(column=1,row=1)
label1=tkinter.Label(text="Timer",fg=GREEN,font=(FONT_NAME,30,"bold"),bg=YELLOW)
label1.grid(column=1,row=0)

label2=tkinter.Label(text="",fg=GREEN,bg=YELLOW)
label2.grid(column=1,row=4)

button1=tkinter.Button(text="Start",command=start_timer)
button1.grid(column=0,row=3)

button2=tkinter.Button(text="Reset",command=reset_timer)
button2.grid(column=2,row=3)



window.mainloop()
```

DAY-29

Building a Password Manager GUI App with Tkinter

In [97]:

```
#Challenge-1  Working with Images and Setting up the Canvas

import tkinter

# ---------------------------- PASSWORD GENERATOR ------------------------------- #

# ---------------------------- SAVE PASSWORD ------------------------------- #

# ---------------------------- UI SETUP ------------------------------- #
window=tkinter.Tk()
window.title("Password Manager")
window.config(padx=20,pady=20)
canvas=tkinter.Canvas(width=200,height=200)
lock_image=tkinter.PhotoImage(file="logo.png")
canvas.create_image(100,112,image=lock_image)
canvas.pack()

window.mainloop()
```

In [178…

```
#Challenge:2  Use grid() and columnspan to Complete the User Interface

import tkinter
YELLOW = "#f7f5dd"
# ---------------------------- PASSWORD GENERATOR ------------------------------- #

# ---------------------------- SAVE PASSWORD ------------------------------- #

# ---------------------------- UI SETUP ------------------------------- #
window=tkinter.Tk()
window.title("Password Manager")
window.config(padx=50,pady=50,bg="white")
canvas=tkinter.Canvas(width=200,height=200,bg="white",highlightthickness=0)
lock_image=tkinter.PhotoImage(file="logo.png")
canvas.create_image(100,100,image=lock_image)
canvas.grid(column=1,row=0)

website=tkinter.Label(text="Website:  ",fg="black",font=("Arial",10,"bold"),bg="white")
website.grid(column=0,row=1)

website_input=tkinter.Entry(width=35)
website_input.grid(column=1,row=1,columnspan=2)

mail=tkinter.Label(text="Email/Username:  ",fg="black",font=("Arial",10,"bold"),bg="white")
mail.grid(column=0,row=2)

mail_input=tkinter.Entry(width=35)
mail_input.grid(column=1,row=2,columnspan=2)

password=tkinter.Label(text="Password:  ",fg="black",font=("Arial",10,"bold"),bg="white")
password.grid(column=0,row=3)

password_input=tkinter.Entry(width=21)
password_input.grid(column=1,row=3,columnspan=1)

generate_button=tkinter.Button(text="Generate Passowrd",bg="white",highlightthickness=0)
generate_button.grid(column=2,row=3,columnspan=1)

add_button=tkinter.Button(text="Add",bg="white",width=35)
add_button.grid(column=1,row=4,columnspan=13)


window.mainloop()
```

In [15]:

```
#Challenge 3 - Saving Data to File


import tkinter
YELLOW = "#f7f5dd"
# ---------------------------- PASSWORD GENERATOR ------------------------------- #

# ---------------------------- SAVE PASSWORD ------------------------------- #

def save():
    with open("file3.txt","a") as file:
        file.write(f"\n{website_input.get()} | {mail_input.get()} | {password_input.get()} ")
        website_input.delete(0,"end")
        password_input.delete(0,"end")

# ---------------------------- UI SETUP ------------------------------- #
window=tkinter.Tk()
window.title("Password Manager")
window.config(padx=50,pady=50,bg="white")
canvas=tkinter.Canvas(width=200,height=200,bg="white",highlightthickness=0)
lock_image=tkinter.PhotoImage(file="logo.png")
canvas.create_image(100,100,image=lock_image)
canvas.grid(column=1,row=0)

website=tkinter.Label(text="Website:  ",fg="black",font=("Arial",10,"bold"),bg="white")
website.grid(column=0,row=1)
```

```
website_input=tkinter.Entry(width=35)
website_input.grid(column=1,row=1,columnspan=2)
#Adding the corsor to the input box on startup
website_input.focus()


mail=tkinter.Label(text="Email/Username:  ",fg="black",font=("Arial",10,"bold"),bg="white")
mail.grid(column=0,row=2)

mail_input=tkinter.Entry(width=35)
mail_input.grid(column=1,row=2,columnspan=2)
mail_input.insert(0,"srenath@gmail.com")


password=tkinter.Label(text="Password:  ",fg="black",font=("Arial",10,"bold"),bg="white")
password.grid(column=0,row=3)

password_input=tkinter.Entry(width=21)
password_input.grid(column=1,row=3,columnspan=1)

generate_button=tkinter.Button(text="Generate Passowrd",bg="white",highlightthickness=0)
generate_button.grid(column=2,row=3,columnspan=1)

add_button=tkinter.Button(text="Add",bg="white",width=35,command=save)
add_button.grid(column=1,row=4,columnspan=13)


window.mainloop()
```

In [11]:

```
# Adding Dialog Boxes and Pop-Ups in Tkinter


import tkinter
from tkinter import messagebox
YELLOW = "#f7f5dd"
# ---------------------------- PASSWORD GENERATOR ------------------------------- #

# ---------------------------- SAVE PASSWORD ------------------------------- #
def save():
    website=website_input.get()
    mailid=mail_input.get()
    password=password_input.get()

    if len(website)==0 or len(mailid)==0 or len(password)==0:
        messagebox.showinfo("Oops","Don't Leave the Spaces Empty")

    else:
        is_save=messagebox.askokcancel(title=website,message="Given details are\n" f"Email/Username:{maili
        if is_save:
            with open("file3.txt","a") as file:
                file.write(f"\n{website_input.get()} | {mail_input.get()} | {password_input.get()} ")
                website_input.delete(0,"end")
                password_input.delete(0,"end")

# ---------------------------- UI SETUP ------------------------------- #
window=tkinter.Tk()
window.title("Password Manager")
window.config(padx=50,pady=50,bg="white")
canvas=tkinter.Canvas(width=200,height=200,bg="white",highlightthickness=0)
lock_image=tkinter.PhotoImage(file="logo.png")
canvas.create_image(100,100,image=lock_image)
canvas.grid(column=1,row=0)

website=tkinter.Label(text="Website:  ",fg="black",font=("Arial",10,"bold"),bg="white")
website.grid(column=0,row=1)

website_input=tkinter.Entry(width=35)
website_input.grid(column=1,row=1,columnspan=2)
#Adding the corsor to the input box on startup
website_input.focus()
```

```
mail=tkinter.Label(text="Email/Username:  ",fg="black",font=("Arial",10,"bold"),bg="white")
mail.grid(column=0,row=2)

mail_input=tkinter.Entry(width=35)
mail_input.grid(column=1,row=2,columnspan=2)
mail_input.insert(0,"srenath@gmail.com")


password=tkinter.Label(text="Password:  ",fg="black",font=("Arial",10,"bold"),bg="white")
password.grid(column=0,row=3)

password_input=tkinter.Entry(width=21)
password_input.grid(column=1,row=3,columnspan=1)

generate_button=tkinter.Button(text="Generate Passowrd",bg="white",highlightthickness=0)
generate_button.grid(column=2,row=3,columnspan=1)

add_button=tkinter.Button(text="Add",bg="white",width=35,command=save)
add_button.grid(column=1,row=4,columnspan=13)


window.mainloop()
```

Building a Password Manager GUI App with Tkinter-->Final Code

In [55]:

```
# Generate a Password & Copy it to the Clipboard
import tkinter
from tkinter import messagebox
YELLOW = "#f7f5dd"
# ---------------------------- PASSWORD GENERATOR ------------------------------- #
def generate_password():
    letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', '
               'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', '
               'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
    numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    symbols = ['!', '#', '$', '%', '&', '(', ')', '*', '+']

    password_letters = [choice(letters) for _ in range(randint(8, 10))]
    password_symbols = [choice(symbols) for _ in range(randint(2, 4))]
    password_numbers = [choice(numbers) for _ in range(randint(2, 4))]

    password_list = password_letters + password_symbols + password_numbers
    shuffle(password_list)

    password = "".join(password_list)
    #Adding into the list
    password_input.insert(0, password)
    pyperclip.copy(password)

# ---------------------------- SAVE PASSWORD ------------------------------- #
def save():
    website=website_input.get()
    mailid=mail_input.get()
    password=password_input.get()

    if len(website)==0 or len(mailid)==0 or len(password)==0:
        messagebox.showinfo("Oops","Don't Leave the Spaces Empty")

    else:
        is_save=messagebox.askokcancel(title=website,message="Given details are\n" f"Email/Username:{maili
        if is_save:
            with open("file3.txt","a") as file:
                file.write(f"\n{website_input.get()} | {mail_input.get()} | {password_input.get()} ")
                website_input.delete(0,"end")
                password_input.delete(0,"end")

# ---------------------------- UI SETUP ------------------------------- #
window=tkinter.Tk()
window.title("Password Manager")
```

```
window.config(padx=50,pady=50,bg="white")
canvas=tkinter.Canvas(width=200,height=200,bg="white",highlightthickness=0)
lock_image=tkinter.PhotoImage(file="logo.png")
canvas.create_image(100,100,image=lock_image)
canvas.grid(column=1,row=0)

website=tkinter.Label(text="Website:   ",fg="black",font=("Arial",10,"bold"),bg="white")
website.grid(column=0,row=1)

website_input=tkinter.Entry(width=35)
website_input.grid(column=1,row=1,columnspan=2)
#Adding the corsor to the input box on startup
website_input.focus()


mail=tkinter.Label(text="Email/Username:   ",fg="black",font=("Arial",10,"bold"),bg="white")
mail.grid(column=0,row=2)

mail_input=tkinter.Entry(width=35)
mail_input.grid(column=1,row=2,columnspan=2)
mail_input.insert(0,"srenath@gmail.com")


password=tkinter.Label(text="Password:   ",fg="black",font=("Arial",10,"bold"),bg="white")
password.grid(column=0,row=3)

password_input=tkinter.Entry(width=21)
password_input.grid(column=1,row=3,columnspan=1)

generate_button=tkinter.Button(text="Generate Passowrd",bg="white",highlightthickness=0,command=generate_p
generate_button.grid(column=2,row=3,columnspan=1)

add_button=tkinter.Button(text="Add",bg="white",width=35,command=save)
add_button.grid(column=1,row=4,columnspan=13)


window.mainloop()
```

DAY-30

The try catch except finally Pattern

In [7]:

```
#This is a try and except error
#Here it try some of the code if it gives an error it goes to except statement
try:
    open(t.txt)
except:
    print("There is a error")
```

```
There is a error
```

In [16]:

```
#Here we can add a particular Error like FileNotFoundError or IndexError
try:
    list1=[2,3,2]
    print(list1[3])
except IndexError:
    print("Index Error is popped")
#What ever try or except popped it will print the finally statement codes
finally:
    print("Code runned")
```

```
Index Error is popped
Code runned
```

Raising own exception

In [18]:

```
#we can rasie error with raise keyword
try:
```

```
    list1=[2,3,2]
    print(list1[3])
except IndexError:
    print("Index Error is popped")
#What ever try or except popped it will print the finally statement codes
finally:
    print("Code runned")
    raise KeyError
```

```
Index Error is popped
Code runned
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-18-3a22626a25b7> in <module>
      8 finally:
      9     print("Code runned")
---> 10     raise KeyError

KeyError:
```

In [19]:

```
#we can rasie error with raise keyword
try:
    list1=[2,3,2]
    print(list1[3])
except IndexError:
    print("Index Error is popped")
#What ever try or except popped it will print the finally statement codes
finally:
    print("Code runned")
    raise TypeError("This is created Error")
```

```
Index Error is popped
Code runned
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-81cf5d72a51f> in <module>
      8 finally:
      9     print("Code runned")
---> 10     raise TypeError("This is created Error")

TypeError: This is created Error
```

In [21]:

```
height = float(input("Height: "))
weight = int(input("Weight: "))

if height > 3:
    raise ValueError("Human Height should not be over 3 meters.")

bmi = weight / height ** 2
print(bmi)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-21-9c1cb3a2f032> in <module>
      3
      4 if height > 3:
----> 5     raise ValueError("Human Height should not be over 3 meters.")
      6
      7 bmi = weight / height ** 2

ValueError: Human Height should not be over 3 meters.
```

[Interactive Coding Exercise] IndexError Handling

In [22]:

```
fruits = ["Apple", "Pear", "Orange"]

#TODO: Catch the exception and make sure the code runs without crashing.
def make_pie(index):
    fruit = fruits[index]
    print(fruit + " pie")
```

```
try:
    make_pie(4)
except IndexError:
    print("List is out of the range")
```

```
List is out of the range
```

Code Exercise: Exception Handling in the NATO Phonetic Alphabet Project

In [5]:

```
# We need to accept only alphabets alone and not numbers or symbols
import pandas

data = pandas.read_csv("nato_phonetic_alphabet.csv")
phonetic_dict = {row.letter: row.code for (index, row) in data.iterrows()}
print(phonetic_dict)
while True:
    try:
        word = input("Enter a word: ").upper()
        output_list = [phonetic_dict[letter] for letter in word]
        print(output_list)
        break
    except KeyError:
        print("Only Alphabets are allowed not numbers or any other string")
```

```
{'A': 'Alfa', 'B': 'Bravo', 'C': 'Charlie', 'D': 'Delta', 'E': 'Echo', 'F': 'Foxtrot', 'G': 'Golf', 'H':
'Hotel', 'I': 'India', 'J': 'Juliet', 'K': 'Kilo', 'L': 'Lima', 'M': 'Mike', 'N': 'November', 'O': 'Osca
r', 'P': 'Papa', 'Q': 'Quebec', 'R': 'Romeo', 'S': 'Sierra', 'T': 'Tango', 'U': 'Uniform', 'V': 'Victor',
'W': 'Whiskey', 'X': 'X-ray', 'Y': 'Yankee', 'Z': 'Zulu'}

Only Alphabets are allowed not numbers or any other string

['Sierra', 'Delta']
```

Write, read and update JSON data in the Password Manager

In [10]:

```
from tkinter import *
from tkinter import messagebox
from random import choice, randint, shuffle
import pyperclip
import json

# ---------------------------- PASSWORD GENERATOR ------------------------------- #

#Password Generator Project
def generate_password():
    letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', '
    numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    symbols = ['!', '#', '$', '%', '&', '(', ')', '*', '+']

    password_letters = [choice(letters) for _ in range(randint(8, 10))]
    password_symbols = [choice(symbols) for _ in range(randint(2, 4))]
    password_numbers = [choice(numbers) for _ in range(randint(2, 4))]

    password_list = password_letters + password_symbols + password_numbers
    shuffle(password_list)

    password = "".join(password_list)
    password_entry.insert(0, password)
    pyperclip.copy(password)

# ---------------------------- SAVE PASSWORD ------------------------------- #
def save():

    website = website_entry.get()
    email = email_entry.get()
    password = password_entry.get()
    new_data = {
        website: {
```

```
            "email": email,
            "password": password,
        }
    }

    if len(website) == 0 or len(password) == 0:
        messagebox.showinfo(title="Oops", message="Please make sure you haven't left any fields empty.")
    else:
        try:
            with open("data.json", "r") as data_file:
                #Reading old data
                data = json.load(data_file)
        except FileNotFoundError:
            with open("data.json", "w") as data_file:
                json.dump(new_data, data_file, indent=4)
        else:
            #Updating old data with new data
            data.update(new_data)

            with open("data.json", "w") as data_file:
                #Saving updated data
                json.dump(data, data_file, indent=4)
        finally:
            website_entry.delete(0, END)
            password_entry.delete(0, END)


# ---------------------------- FIND PASSWORD ------------------------------- #
def find_password():
    website = website_entry.get()
    try:
        with open("data.json") as data_file:
            data = json.load(data_file)
    except FileNotFoundError:
        messagebox.showinfo(title="Error", message="No Data File Found.")
    else:
        if website in data:
            email = data[website]["email"]
            password = data[website]["password"]
            messagebox.showinfo(title=website, message=f"Email: {email}\nPassword: {password}")
        else:
            messagebox.showinfo(title="Error", message=f"No details for {website} exists.")


# ---------------------------- UI SETUP ------------------------------- #

window = Tk()
window.title("Password Manager")
window.config(padx=50, pady=50)

canvas = Canvas(height=200, width=200)
logo_img = PhotoImage(file="logo.png")
canvas.create_image(100, 100, image=logo_img)
canvas.grid(row=0, column=1)

#Labels
website_label = Label(text="Website:")
website_label.grid(row=1, column=0)
email_label = Label(text="Email/Username:")
email_label.grid(row=2, column=0)
password_label = Label(text="Password:")
password_label.grid(row=3, column=0)

#Entries
website_entry = Entry(width=21)
website_entry.grid(row=1, column=1)
website_entry.focus()
email_entry = Entry(width=35)
email_entry.grid(row=2, column=1, columnspan=2)
email_entry.insert(0, "angela@gmail.com")
password_entry = Entry(width=21)
password_entry.grid(row=3, column=1)
```

```
# Buttons
search_button = Button(text="Search", width=13, command=find_password)
search_button.grid(row=1, column=2)
generate_password_button = Button(text="Generate Password", command=generate_password)
generate_password_button.grid(row=3, column=2)
add_button = Button(text="Add", width=36, command=save)
add_button.grid(row=4, column=1, columnspan=2)

window.mainloop()
```

DAY-31

Creating n Flash Card App Capstone Project

In [1]:

```
from tkinter import *
import pandas
import random

BACKGROUND_COLOR = "#B1DDC6"
current_card = {}
to_learn = {}

try:
    data = pandas.read_csv("data/words_to_learn.csv")
except FileNotFoundError:
    original_data = pandas.read_csv("french_words.csv")
    print(original_data)
    to_learn = original_data.to_dict(orient="records")
else:
    to_learn = data.to_dict(orient="records")


def next_card():
    global current_card, flip_timer
    window.after_cancel(flip_timer)
    current_card = random.choice(to_learn)
    canvas.itemconfig(card_title, text="French", fill="black")
    canvas.itemconfig(card_word, text=current_card["French"], fill="black")
    canvas.itemconfig(card_background, image=card_front_img)
    flip_timer = window.after(3000, func=flip_card)


def flip_card():
    canvas.itemconfig(card_title, text="English", fill="white")
    canvas.itemconfig(card_word, text=current_card["English"], fill="white")
    canvas.itemconfig(card_background, image=card_back_img)


def is_known():
    to_learn.remove(current_card)
    print(len(to_learn))
    data = pandas.DataFrame(to_learn)
    data.to_csv("data/words_to_learn.csv", index=False)
    next_card()


window = Tk()
window.title("Flashy")
window.config(padx=50, pady=50, bg=BACKGROUND_COLOR)

flip_timer = window.after(3000, func=flip_card)

canvas = Canvas(width=800, height=526)
card_front_img = PhotoImage(file="card_front.png")
card_back_img = PhotoImage(file="card_back.png")
card_background = canvas.create_image(400, 263, image=card_front_img)
card_title = canvas.create_text(400, 150, text="", font=("Ariel", 40, "italic"))
card_word = canvas.create_text(400, 263, text="", font=("Ariel", 60, "bold"))
```

```
canvas.config(bg=BACKGROUND_COLOR, highlightthickness=0)
canvas.grid(row=0, column=0, columnspan=2)

cross_image = PhotoImage(file="wrong.png")
unknown_button = Button(image=cross_image, highlightthickness=0, command=next_card)
unknown_button.grid(row=1, column=0)

check_image = PhotoImage(file="right.png")
known_button = Button(image=check_image, highlightthickness=0, command=is_known)
known_button.grid(row=1, column=1)

next_card()

window.mainloop()
```

```
        French    English
0       partie       part
1     histoire    history
2     chercher     search
3    seulement       only
4       police     police
..         ...        ...
96      plutôt     rather
97        aura  will have
98      filles      girls
99       jouer    to play
100     bureau     office

[101 rows x 2 columns]
100
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\srena\anaconda3\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "<ipython-input-1-02edbff1969c>", line 39, in is_known
    data.to_csv("data/words_to_learn.csv", index=False)
  File "C:\Users\srena\anaconda3\lib\site-packages\pandas\core\generic.py", line 3387, in to_csv
    return DataFrameRenderer(formatter).to_csv(
  File "C:\Users\srena\anaconda3\lib\site-packages\pandas\io\formats\format.py", line 1083, in to_csv
    csv_formatter.save()
  File "C:\Users\srena\anaconda3\lib\site-packages\pandas\io\formats\csvs.py", line 228, in save
    with get_handle(
  File "C:\Users\srena\anaconda3\lib\site-packages\pandas\io\common.py", line 642, in get_handle
    handle = open(
FileNotFoundError: [Errno 2] No such file or directory: 'data/words_to_learn.csv'
```

DAY-32

Send Email (smtplib) and Manage Dates (datetime)

In [12]:

```
import smtplib

username="jarvisai215@gmail.com"
password="srenath2002"

#Accessing to gmail server
with smtplib.SMTP("smtp.gmail.com",587) as connection:
    #For secured and encrypted messages
    connection.starttls()
    #Login to the account
    connection.login(user=username,password=password)
    connection.sendmail(from_addr=username,to_addrs="srenath.e0120006@sret.edu.in",msg="Subject:Hello\n\nT
    connection.close()
```

Working with the datetime Module

In [2]:

```
import datetime as dt
#Here dt is a module datetime is a class and now is a method(function)
now=dt.datetime.now()
now
```

```
year=now.year

month=now.month
#It returns an integer type
year
```

Out[2]: 2021

In [21]:
```
month
```

Out[21]: 6

In [25]:
```
week=now.weekday()
#Mon=0 Tue=1 Wed=2 Thu=3 Fri=4 Sat=5 Sun=6
#Computer starts from 0 according to its calculation
week
```

Out[25]: 3

Creating own(self) datetime object for setting

In [3]:
```
date_of_birth=dt.datetime(year=1995,month=12,day=12)
```

In [4]:
```
#Creating date of birth object with default 12 am time
date_of_birth
```

Out[4]: datetime.datetime(1995, 12, 12, 0, 0)

Send Motivational Quotes on Mondays via Email

In [27]:
```
import datetime as dt
import smtplib
import random
now=dt.datetime.now()

username="jarvisai215@gmail.com"
password="srenath2002"
if now.weekday() == 3:
    with smtplib.SMTP("smtp.gmail.com",587) as connection:
        #Opening the file
        file=open("quotes.txt","r")
        #Reading the lines
        words=file.readlines()
        #Chosing the random lines
        choice=random.choice(words)
        #And sending
        connection.starttls()
        connection.login(user=username,password=password)
        connection.sendmail(from_addr=username,to_addrs="srenath.e0120006@sret.edu.in",msg=choice)
        connection.close()
        print(choice)
        file.close()
```

Reading and writing into a file through a string

In [36]:
```
#Reading and closing
fin = open("untitled.txt", "rt")
data = fin.read()
data = data.replace('name', 'python')
fin.close()

#Writing into same read file and closing by replacing the required text change
```

```
fin = open("untitled.txt", "wt")
fin.write(data)
fin.close()
```

Automated Birthday Wisher Project Challenge

In [105…

```
from datetime import datetime
import pandas
import random
import smtplib

MY_EMAIL = "jarvisai215@gmail.com"
MY_PASSWORD = "srenath2002"

today = datetime.now()
today_tuple = (today.month, today.day)

data = pandas.read_csv("birthdays.csv")
#this iterrow helps to iterate the dictionary
#Converting the dictionary into {month,date:email,name} for this we use dictionary
birthdays_dict = {(data_row["month"], data_row["day"]): data_row for (index, data_row) in data.iterrows()}
if today_tuple in birthdays_dict:
    birthday_person = birthdays_dict[today_tuple]
    file_path = f"letter_{random.randint(1,3)}.txt"
    with open(file_path) as letter_file:
        contents = letter_file.read()
        contents = contents.replace("[NAME]", birthday_person["name"])

    with smtplib.SMTP("smtp.gmail.com",587) as connection:
        connection.starttls()
        connection.login(MY_EMAIL, MY_PASSWORD)
        connection.sendmail(
            from_addr=MY_EMAIL,
            to_addrs=birthday_person["email"],
            msg=f"Subject:Happy Birthday!\n\n{contents}"
        )
```

DAY-33

API-->Application Programming Interface

Learning about API

In [107…

```
import requests

response=requests.get("http://api.open-notify.org/iss-now.json")

response
```

Out[107… <Response [200]>

Response codes: 1xx Hold On there is some issue in the website 2xx Data's Shown successful 3xx Go Away there is no access 4xx Does not exist 5xx server not working or down or other issue

In [109…

```
response.status_code  #Success
```

Out[109… 200

In [112…

```
#If there is a error in the url the response ode is 404 error
response=requests.get("http://api.open-notify.org/is-now.json")
#This url "http://api.open-notify.org/is-now.json" is the endpoint of the api
response
```

Out[112… <Response [404]>

In [113…
```
### Raising an error function in the requests module
#This raise_for_status will raise the error if the error found
response=requests.get("http://api.open-notify.org/is-now.json")
response.raise_for_status()
```

```
---------------------------------------------------------------------------
HTTPError                                 Traceback (most recent call last)
<ipython-input-113-191e00f2298d> in <module>
      1 ### Raising an error function in the requests module
      2 response=requests.get("http://api.open-notify.org/is-now.json")
----> 3 response.raise_for_status()

~\anaconda3\lib\site-packages\requests\models.py in raise_for_status(self)
    941
    942         if http_error_msg:
--> 943             raise HTTPError(http_error_msg, response=self)
    944
    945     def close(self):

HTTPError: 404 Client Error: Not Found for url: http://api.open-notify.org/is-now.json
```

In [125…
```
import requests

response=requests.get("http://api.open-notify.org/iss-now.json")
#Getting the json data from the api url
data=response.json()
#this json file is like a dictionary
data
```

Out[125…
```
{'message': 'success',
 'timestamp': 1623348625,
 'iss_position': {'longitude': '-178.6483', 'latitude': '-24.5953'}}
```

In [126…
```
data["iss_position"]
```

Out[126…
```
{'longitude': '-178.6483', 'latitude': '-24.5953'}
```

In [127…
```
data["iss_position"]["longitude"]
```

Out[127…
```
'-178.6483'
```

In [128…
```
longitude=data["iss_position"]["longitude"]
latitude=data["iss_position"]["latitude"]
```

In [129…
```
#WE are making it for our requirement
iss_position=(longitude,latitude)
iss_position
```

Out[129…
```
('-178.6483', '-24.5953')
```

Challenge - Build a Kanye Quotes App using the Kanye Rest API

In [141…
```
from tkinter import *
import requests

def get_quote():
    response=requests.get("https://api.kanye.rest/")
    data_json=response.json()["quote"]
    canvas.itemconfig(quote_text,text=data_json)



window = Tk()
window.title("Kanye Says...")
```

```
window.config(padx=50, pady=50)

canvas = Canvas(width=300, height=414)
background_img = PhotoImage(file="background.png")
canvas.create_image(150, 207, image=background_img)
quote_text = canvas.create_text(150, 207, text="Kanye Quote Goes HERE", width=250, font=("Arial", 30, "bol
canvas.grid(row=0, column=0)

kanye_img = PhotoImage(file="kanye.png")
kanye_button = Button(image=kanye_img, highlightthickness=0, command=get_quote)
kanye_button.grid(row=1, column=0)



window.mainloop()
```

API Parameters

In [150…

```
#This type of api parametes will give response result when parameters are given

LON=79.941330
LAT=12.973780

parameters={
    "lat":LAT,
    "lng":LON
}

import requests
#This is the format of giving parameters to the api
#First we need to create dicionary and add it into the api requests
response=requests.get(url="https://api.sunrise-sunset.org/json",params=parameters)
response.raise_for_status()
sunrise=response.json()["results"]["sunrise"]
sunset=response.json()["results"]["sunset"]

print(sunrise,sunset)
```

```
12:13:29 AM 1:06:07 PM
```

In [153…

```
import datetime as dt
now=dt.datetime.now()
now
```

Out[153… `datetime.datetime(2021, 6, 11, 0, 7, 5, 711551)`

In [154…

```
#Here in the above code we can se the sunrise sunset are in different format
# sunrise and sunset are in 12 hrs format
# datetime module time is in 24 hrs format
```

In [161…

```
#So in order to change the time to 24 hrs clock in api we can change formatted to 0 to api

LON=79.941330
LAT=12.973780

parameters={
    "lat":LAT,
    "lng":LON,
    "formatted":0
}

import requests
#This is the format of giving parameters to the api
#First we need to create dicionary and add it into the api requests
response=requests.get(url="https://api.sunrise-sunset.org/json",params=parameters)
response.raise_for_status()
sunrise=response.json()["results"]["sunrise"]
```

```
sunset=response.json()["results"]["sunset"]

#Next step is spliting the time
#Getting hours
sunrise_list=sunrise.split("T")[1].split(":")[0]
sunset_list=sunset.split("T")[1].split(":")[0]

sunset_list
```

Out[161… '13'

In [163…

```
sunrise_list
```

Out[163… '00'

ISS Overhead Notifier Project - Challenge

In [ ]:

```
import requests
from datetime import datetime
import smtplib
import time

MY_EMAIL = "jarvisai215@gmail.com"
MY_PASSWORD = "srenath2002"
MY_LAT = 51.507351 # Your Latitude
MY_LONG = -0.127758 # Your Longitude


def is_iss_overhead():
    response = requests.get(url="http://api.open-notify.org/iss-now.json")
    response.raise_for_status()
    data = response.json()

    iss_latitude = float(data["iss_position"]["latitude"])
    iss_longitude = float(data["iss_position"]["longitude"])

    #Your position is within +5 or -5 degrees of the iss position.
    if MY_LAT-5 <= iss_latitude <= MY_LAT+5 and MY_LONG-5 <= iss_longitude <= MY_LONG+5:
        return True


def is_night():
    parameters = {
        "lat": MY_LAT,
        "lng": MY_LONG,
        "formatted": 0,
    }
    response = requests.get("https://api.sunrise-sunset.org/json", params=parameters)
    response.raise_for_status()
    data = response.json()
    sunrise = int(data["results"]["sunrise"].split("T")[1].split(":")[0])
    sunset = int(data["results"]["sunset"].split("T")[1].split(":")[0])

    time_now = datetime.now().hour

    if time_now >= sunset or time_now <= sunrise:
        return True


while True:
    #This will run every 60 seconds
    time.sleep(60)
    if is_iss_overhead() and is_night():
        connection = smtplib.SMTP("__YOUR_SMTP_ADDRESS_HERE___")
        connection.starttls()
        connection.login(MY_EMAIL, MY_PASSWORD)
        connection.sendmail(
            from_addr=MY_EMAIL,
            to_addrs=MY_EMAIL,
```

```
            msg="Subject:Look Up👆\n\nThe ISS is above you in the sky."
        )
```

DAY-34

Trivia Question API Challenge

In [8]:

```
#Changing day-17 quiz challange to api link for questions

import requests

question_data=requests.get("https://opentdb.com/api.php?amount=10&category=18&type=boolean")
question_data=question_data.json()["results"]



THEME_COLOR = "#375362"

class QuizBrain:

    def __init__(self, q_list):
        self.question_number = 0
        self.score = 0
        self.question_list = q_list
        self.current_question = None

    def still_has_questions(self):
        return self.question_number < len(self.question_list)

    def next_question(self):
        self.current_question = self.question_list[self.question_number]
        self.question_number += 1
        user_answer = input(f"Q.{self.question_number}: {self.current_question.text} (True/False): ")
        self.check_answer(user_answer)

    def check_answer(self, user_answer):
        correct_answer = self.current_question.answer
        if user_answer.lower() == correct_answer.lower():
            self.score += 1
            print("You got it right!")
        else:
            print("That's wrong.")

        print(f"Your current score is: {self.score}/{self.question_number}")
        print("\n")


class Question:

    def __init__(self, q_text, q_answer):
        self.text = q_text
        self.answer = q_answer







question_bank = []
for question in question_data:
    question_text = question["question"]
    question_answer = question["correct_answer"]
    new_question = Question(question_text, question_answer)
    question_bank.append(new_question)


quiz = QuizBrain(question_bank)

while quiz.still_has_questions():
```

```
    quiz.next_question()

print("You've completed the quiz")
print(f"Your final score was: {quiz.score}/{quiz.question_number}")
```

```
That's wrong.
Your current score is: 0/1


That's wrong.
Your current score is: 0/2


That's wrong.
Your current score is: 0/3


You got it right!
Your current score is: 1/4


You got it right!
Your current score is: 2/5


That's wrong.
Your current score is: 2/6


That's wrong.
Your current score is: 2/7


That's wrong.
Your current score is: 2/8


That's wrong.
Your current score is: 2/9


That's wrong.
Your current score is: 2/10


You've completed the quiz
Your final score was: 2/10
```

In [ ]:

```
# We notice that we have some bug  in question api &quot this is html bug
# These are html entities (escape characters)

#For more Referece : https://www.w3schools.com/html/html_entities.asp

#There is a model called html in that there is a method call upescape which converts the entits to origina
```

In [ ]:

```
#Changing day-17 quiz challange to api link for questions

import requests
import html
question_data=requests.get("https://opentdb.com/api.php?amount=10&category=18&type=boolean")
question_data=question_data.json()["results"]



THEME_COLOR = "#375362"
```

```
class QuizBrain:

    def __init__(self, q_list):
        self.question_number = 0
        self.score = 0
        self.question_list = q_list
        self.current_question = None

    def still_has_questions(self):
        return self.question_number < len(self.question_list)

    def next_question(self):
        self.current_question = self.question_list[self.question_number]
        self.question_number += 1
#########################################################################################################
        #Changing the html entities to original characters
        q_text=html.unescape(self.current_question.text)
        user_answer = input(f"Q.{self.question_number}: {q_text} (True/False): ")
        self.check_answer(user_answer)
#########################################################################################################

    def check_answer(self, user_answer):
        correct_answer = self.current_question.answer
        if user_answer.lower() == correct_answer.lower():
            self.score += 1
            print("You got it right!")
        else:
            print("That's wrong.")

        print(f"Your current score is: {self.score}/{self.question_number}")
        print("\n")


class Question:

    def __init__(self, q_text, q_answer):
        self.text = q_text
        self.answer = q_answer







question_bank = []
for question in question_data:
    question_text = question["question"]
    question_answer = question["correct_answer"]
    new_question = Question(question_text, question_answer)
    question_bank.append(new_question)


quiz = QuizBrain(question_bank)

while quiz.still_has_questions():
    quiz.next_question()

print("You've completed the quiz")
print(f"Your final score was: {quiz.score}/{quiz.question_number}")

#So we can see the code is edited and the question inputted works perfectly
```

```
That's wrong.
Your current score is: 0/1
```

Class based Tkinter UI

In [ ]:

```
import requests
```

```
parameters = {
    "amount": 10,
    "type": "boolean",
}

response = requests.get("https://opentdb.com/api.php", params=parameters)
response.raise_for_status()
data = response.json()
question_data = data["results"]

class Question:

    def __init__(self, q_text, q_answer):
        self.text = q_text
        self.answer = q_answer


import html


class QuizBrain:

    def __init__(self, q_list):
        self.question_number = 0
        self.score = 0
        self.question_list = q_list
        self.current_question = None

    def still_has_questions(self):
        return self.question_number < len(self.question_list)

    def next_question(self):
        self.current_question = self.question_list[self.question_number]
        self.question_number += 1
        q_text = html.unescape(self.current_question.text)
        return f"Q.{self.question_number}: {q_text}"

    def check_answer(self, user_answer):
        correct_answer = self.current_question.answer
        if user_answer.lower() == correct_answer.lower():
            self.score += 1
            return True
        else:
            return False




from tkinter import *
from quiz_brain import QuizBrain

THEME_COLOR = "#375362"


class QuizInterface:

    def __init__(self, quiz_brain: QuizBrain):
        self.quiz = quiz_brain

        self.window = Tk()
        self.window.title("Quizzler")
        self.window.config(padx=20, pady=20, bg=THEME_COLOR)

        self.score_label = Label(text="Score: 0", fg="white", bg=THEME_COLOR)
        self.score_label.grid(row=0, column=1)

        self.canvas = Canvas(width=300, height=250, bg="white")
        self.question_text = self.canvas.create_text(
            150,
            125,
            width=280,
```

```
            text="Some Question Text",
            fill=THEME_COLOR,
            font=("Arial", 20, "italic")
        )
        self.canvas.grid(row=1, column=0, columnspan=2, pady=50)

        true_image = PhotoImage(file="true.png")
        self.true_button = Button(image=true_image, highlightthickness=0, command=self.true_pressed)
        self.true_button.grid(row=2, column=0)

        false_image = PhotoImage(file="false.png")
        self.false_button = Button(image=false_image, highlightthickness=0, command=self.false_pressed)
        self.false_button.grid(row=2, column=1)

        self.get_next_question()

        self.window.mainloop()

    def get_next_question(self):
        self.canvas.config(bg="white")
        if self.quiz.still_has_questions():
            self.score_label.config(text=f"Score: {self.quiz.score}")
            q_text = self.quiz.next_question()
            self.canvas.itemconfig(self.question_text, text=q_text)
        else:
            self.canvas.itemconfig(self.question_text, text="You've reached the end of the quiz.")
            self.true_button.config(state="disabled")
            self.false_button.config(state="disabled")

    def true_pressed(self):
        self.give_feedback(self.quiz.check_answer("True"))

    def false_pressed(self):
        is_right = self.quiz.check_answer("False")
        self.give_feedback(is_right)

    def give_feedback(self, is_right):
        if is_right:
            self.canvas.config(bg="green")
        else:
            self.canvas.config(bg="red")
        self.window.after(1000, self.get_next_question)




question_bank = []
for question in question_data:
    question_text = question["question"]
    question_answer = question["correct_answer"]
    new_question = Question(question_text, question_answer)
    question_bank.append(new_question)


quiz = QuizBrain(question_bank)
quiz_ui = QuizInterface(quiz)
```

```
That's wrong.
Your current score is: 0/1


You got it right!
Your current score is: 1/1
```

Assigining a variable to a data type

In [5]:

```
age:int
```

```
age=12
print(age)
```

```
12
```

Type Hints

In [7]:
```
#This ->data type will only return oen function run
#If ->str is given then it will only return string only
def greeting(name:str)->str:
    return "Hello" +name

greeting("Srenath")
```

Out[7]:
```
'HelloSrenath'
```

DAY-35

API AUTHENTATICATION

In [ ]:
```
API AUTHENTATICATION-->It means that we need to get permission (acesss) for an api data

For this authentication we use api key
This api key is the key which they can able to track grant and also deniying the process
```

Using API Keys to Authenticate and Get the Weather from OpenWeatherMap

Jarvis AI Weather API Key:3a6f8ab0baa30a8b0daeee22bf8fc927

CHALLANGE API CALL FOR ONE CALL API AND GET HOURLY DATA

In [ ]:
```
import requests

url="https://api.openweathermap.org/data/2.5/onecall"

parameters={
    "lat":12.974780,
    "lon":79.956863,
    "appid":"3a6f8ab0baa30a8b0daeee22bf8fc927"
}
data=requests.get(url,params=parameters)
data.json()["hourly"]
```

Challenge - Check if it Will Rain in the Next 12 Hours

In [ ]:
```
There is a parameter called exclude which removes the unrequired data on api call
```

In [19]:
```
import requests

url="https://api.openweathermap.org/data/2.5/onecall"

parameters={
    "lat":12.974780,
    "lon":79.956863,
    "appid":"3a6f8ab0baa30a8b0daeee22bf8fc927",
    "exclude":"current,minutely,daily"
}
response=requests.get(url,params=parameters)
twelve_hourly_weather_data=response.json()["hourly"][:12]
will_rain=False
for i in twelve_hourly_weather_data:
    condition_code=i["weather"][0]["id"]
    if int(condition_code) <700:
        will_rain=True
```

```
if will_rain:
    print("Bring an umbrella")
```

In [20]:
```
will_rain
```

Out[20]: False

Sending SMS via the Twilio API if its rain

In [114…
```
import requests
import os
from twilio.rest import Client
url="https://api.openweathermap.org/data/2.5/onecall"
################################################################
#               TWILIO AUTHENTATION DETAILS

account_sid ='AC2d4861b014392ed4ecae3129dcdb2926'
auth_token ='cd59e944190ab673edc22dc7413aceb9'



parameters={
    "lat":45.646820,
    "lon":-84.474854,
    "appid":"3a6f8ab0baa30a8b0daeee22bf8fc927",
    "exclude":"current,minutely,daily"
}
response=requests.get(url,params=parameters)
twelve_hourly_weather_data=response.json()["hourly"][:12]
will_rain=False
for i in twelve_hourly_weather_data:
    condition_code=i["weather"][0]["id"]
    if int(condition_code) <700:
        will_rain=True
if will_rain:

    client = Client(account_sid, auth_token)
    message = client.messages \
                .create(
                     body="It's Going to rain today.Remember to bring an ☂.",
                     from_='+19123485222',
                     to='+919894395228'
                 )
    print(message.status)
```

queued

Understanding Environment Variables and Hiding API Keys

Environment variables are used to dtore apikeys as a variable in the terminal console

In [26]:
```
# i,e instead of storing the value in a variable in a file we can store as a variable in the console or te
```

THESE ARE USED TO HIDE THE KEYS FROM OUTSIDE WORLD

COMMANDS TO RUN IN CONSOLE: Method 1: 1)export ACCOUNT_SID=AC2d4861b014392ed4ecae3129dcdb2926 2)export AUTH_TOKEN=cd59e944190ab673edc22dc7413aceb9 3)export OWN_API_KEY=3a6f8ab0baa30a8b0daeee22bf8fc927 Method 2: export ACCOUNT_SID=AC2d4861b014392ed4ecae3129dcdb2926;export AUTH_TOKEN=cd59e944190ab673edc22dc7413aceb9;python3 main.py

In [ ]:
```
#Note! For the code to work you need to replace all the placeholders with
#Your own details. e.g. account_sid, lat/lon, from/to phone numbers.

import requests
import os
from twilio.rest import Client
from twilio.http.http_client import TwilioHttpClient
```

```
OWM_Endpoint = "https://api.openweathermap.org/data/2.5/onecall"
api_key = os.environ.get("OWM_API_KEY")
account_sid = "YOUR ACCOUNT SID"
auth_token = os.environ.get("AUTH_TOKEN")

weather_params = {
    "lat": "YOUR LATITUDE",
    "lon": "YOUR LONGITUDE",
    "appid": api_key,
    "exclude": "current,minutely,daily"
}

response = requests.get(OWM_Endpoint, params=weather_params)
response.raise_for_status()
weather_data = response.json()
weather_slice = weather_data["hourly"][:12]

will_rain = False

for hour_data in weather_slice:
    condition_code = hour_data["weather"][0]["id"]
    if int(condition_code) < 700:
        will_rain = True

if will_rain:
    proxy_client = TwilioHttpClient()
    proxy_client.session.proxies = {'https': os.environ['https_proxy']}

    client = Client(account_sid, auth_token, http_client=proxy_client)
    message = client.messages \
        .create(
        body="It's going to rain today. Remember to bring an ☂",
        from_="YOUR TWILIO VIRTUAL NUMBER",
        to="YOUR TWILIO VERIFIED REAL NUMBER"
    )
    print(message.status)
```

DAY-36

Stock Trading News Alert Project

In [122…

```
STOCK = "TSLA"
COMPANY_NAME = "Tesla Inc"

## STEP 1: Use https://www.alphavantage.co
# When STOCK price increase/decreases by 5% between yesterday and the day before yesterday then print("Get
import requests
stock_url="https://www.alphavantage.co/query"
stock_parameters={
    "function":"TIME_SERIES_DAILY",
    "symbol":"TSLA",
    "apikey":"80TJZ6TV8A34S0E7"
}

response=requests.get(stock_url,stock_parameters)
response.raise_for_status
data=response.json()["Time Series (Daily)"]
data_list = [value for (key, value) in data.items()]
yesterday_data=data_list[0]["4. close"]
day_before_yesterday_data=data_list[1]["4. close"]
diff=float(yesterday_data)-float(day_before_yesterday_data)
if diff > 0:
    up_down="▲"
else:
    up_down="▼"

percentage=(diff/float(yesterday_data))*100
percentage_round=round(percentage,2)
percen_symbol=str(up_down)+str(percentage_round)
```

```
## STEP 2: Use https://newsapi.org
# Instead of printing ("Get News"), actually get the first 3 news pieces for the COMPANY_NAME.
from newsapi import NewsApiClient
NEWS_ENDPOINT = "https://newsapi.org/v2/everything"

if abs(percentage_round) > 1:
    news_params = {
        "apiKey": "bfb3a28daf6443e5a3be9dd315d118bb",
        "qInTitle": "tesla",
    }

    news_response = requests.get(NEWS_ENDPOINT, params=news_params)
    articles = news_response.json()["articles"]

    #Use Python slice operator to create a list that contains the first 3 articles. Hint: https://stackove
    ## STEP 3: Use https://www.twilio.com
    # Send a seperate message with the percentage change and each article's title and description to your
    from twilio.rest import Client
    account_sid ='AC2d4861b014392ed4ecae3129dcdb2926'
    auth_token ='cd59e944190ab673edc22dc7413aceb9'
    client = Client(account_sid, auth_token)

    for i in range(0,3):
        headlines=articles[i]["title"]
        briefs=articles[i]["description"]

        final_data="Bitcoin: "+percen_symbol+"\n\n"+"Headline: "+headlines+"\n\n"+"Brief:"+briefs+"\n\n"
        message = client.messages \
                .create(
                     body=final_data,
                     from_='+19123485222',
                     to='+919894100731'
                 )
        print(message.status)

#Optional: Format the SMS message like this:
#"""
#TSLA: ▲2%
#Headline: Were Hedge Funds Right About Piling Into Tesla Inc. (TSLA)?.
#Brief: We at Insider Monkey have gone over 821 13F filings that hedge funds and prominent investors are r
#or
#"TSLA: ▼5%
#Headline: Were Hedge Funds Right About Piling Into Tesla Inc. (TSLA)?.
#Brief: We at Insider Monkey have gone over 821 13F filings that hedge funds and prominent investors are r
#"""
```

```
queued
queued
queued
```

DAY-37

In [1]:
```
# Get request means we are getting data from the api server
```

In [2]:
```
# Post request means we are giving data to the api server
```

In [3]:
```
# Put request is to update the existing data in the api server data
```

In [4]:
```
# Delete request is to delete a piece of data from the api server
```

HTTP POST REQUEST

In [ ]:
```
```

```
## Creating an Application called pixela which helps to track our daily life study or coding hours

#We use pixela api for this

#All below steps are done according to api documentation

#All the codes are interlinked i,e first we need to create an account with that we need to create a graph
# Like that it goes on
```

Creating an account

In [21]:
```
import requests

TOKEN="dsfsdf343fdsfsdgfds"
USERNAME="jarvis215"


url="https://pixe.la/v1/users"
user_parameters={
    "token":TOKEN, #ramdom token characters
    "username":USERNAME,
    "agreeTermsOfService":"yes",
    "notMinor":"yes"
}
response=requests.post(url=url,json=user_parameters)  #Here in post requests we are giving our data in jso
                                                      #  format so we are giving parameters into json vari

print(response.text)
```

```
{"message":"Success. Let's visit https://pixe.la/@jarvis215 , it is your profile page!","isSuccess":true}
```

In [16]:
```
#So more in above we have created an account for us

#It's like we are requesting to open a html page by giving our details
```

Creating an graph

In [23]:
```
graph_url=f"https://pixe.la/v1/users/{USERNAME}/graphs"

header={
    "X-USER-TOKEN":TOKEN         #This x-user-token is a encrypted and helps to hidea api key from the out
                                 #It is recommended to use this header api authentication
}
graph_parameters={
    "id":"coding122",
    "name":"Coding Tracker",
    "unit":"hrs",
    "type":"int",
    "color":"sora"
}

response=requests.post(url=graph_url,json=graph_parameters,headers=header)
print(response.text)
```

```
{"message":"Success.","isSuccess":true}
```

Challange:Add a Pixel to the Habit Tracker using a Post Request

In [ ]:
```
import requests


pixel_url=f"https://pixe.l/v1/users/{USERNAME}/graphs/coding122"



pixel_parameters={
    "date":"20210611",
    "quantity":"6",
}
```

```
response=requests.post(url=pixel_url,json=pixel_parameters,headers=header)
print(response.raise_for_status)
```

Modifing the date with datatime module

In [28]:
```
import requests
from datetime import datetime

pixel_url=f"https://pixe.l/v1/users/{USERNAME}/graphs/coding122"

today=datetime(year=2021,month=6,day=12)

pixel_parameters={
    "date":today.strftime("%Y%m%d"),
    "quantity":"6",
}
print(pixel_parameters)

#response=requests.post(url=pixel_url,json=pixel_parameters,headers=header)
#print(response.raise_for_status)
```

```
{'date': '20210612', 'quantity': '6'}
```

In [29]:
```
#Now we can see that we get it in formatted way
```

HTTP PUT

In [ ]:
```
## PUT
put_pixel_url=f"https://pixe.l/v1/users/{USERNAME}/graphs/coding122/{today.strftime('%Y%m%d')}"
new_put_pixel_data={
    quantity="10"
}
response = requests.put(url=put_pixel_url, json=new_put_pixel_data, headers=headers)
print(response.text)
```

HTTP DELETE

In [ ]:
```
delete_endpoint = f"https://pixe.l/v1/users/{USERNAME}/graphs/coding122/{today.strftime('%Y%m%d')}"


## DELETE
response = requests.delete(url=delete_endpoint, headers=headers)
print(response.text)
```

DAY-38

Workout Tracking Using Google Sheets

Step 1 - Setup API Credentials and Google Spreadsheet

In [10]:
```
APP_ID="62697aef"
API_KEY="1f848e5a96335450ce5f1b9969f7ee38"
```

Step 2 - Get Exercise Stats with Natural Language Queries

In [13]:
```
import requests
header={
    "x-app-id":APP_ID,
    "x-app-key":API_KEY
}
```

```
GENDER = "MALE"
WEIGHT_KG = 80
HEIGHT_CM = 180
AGE = 18


exercise_endpoint = "https://trackapi.nutritionix.com/v2/natural/exercise"

exercise_text = input("Tell me which exercises you did: ")



parameters = {
    "query": exercise_text,
    "gender": GENDER,
    "weight_kg": WEIGHT_KG,
    "height_cm": HEIGHT_CM,
    "age": AGE
}

response = requests.post(exercise_endpoint, json=parameters, headers=header)
result = response.json()
```

Step 3 - Setup Your Google Sheet with Sheety

Step 4 - Saving Data into Google Sheets

In [15]:

```
from datetime import datetime
today_date = datetime.now().strftime("%d/%m/%Y")
now_time = datetime.now().strftime("%X")
sheet_endpoint="https://api.sheety.co/bbd43d5608ed875fd2340b7cab160474/workoutSpreadsheet/workouts"
for exercise in result["exercises"]:
    sheet_inputs = {
        "workout": {
            "date": today_date,
            "time": now_time,
            "exercise": exercise["name"].title(),
            "duration": exercise["duration_min"],
            "calories": exercise["nf_calories"]
        }
    }

    sheet_response = requests.post(sheet_endpoint, json=sheet_inputs)

    print(sheet_response.text)
```

DAY-39 & DAY-40

In [2]:

```
import requests


TEQUILA_ENDPOINT = "https://tequila-api.kiwi.com"
TEQUILA_API_KEY = "L_Cdkzbgd2JNuBlv7AoI-FQdW9EKSnv5"


class FlightSearch:

    def get_destination_code(self, city_name):
        location_endpoint = f"{TEQUILA_ENDPOINT}/locations/query"
        headers = {"apikey": TEQUILA_API_KEY}
        query = {"term": city_name, "location_types": "city"}
        response = requests.get(url=location_endpoint, headers=headers, params=query)
        results = response.json()["locations"]
        code = results[0]["code"]
        return code
```

```
    def check_flights(self, origin_city_code, destination_city_code, from_time, to_time):
        headers = {"apikey": TEQUILA_API_KEY}
        query = {
            "fly_from": origin_city_code,
            "fly_to": destination_city_code,
            "date_from": from_time.strftime("%d/%m/%Y"),
            "date_to": to_time.strftime("%d/%m/%Y"),
            "nights_in_dst_from": 7,
            "nights_in_dst_to": 28,
            "flight_type": "round",
            "one_for_city": 1,
            "max_stopovers": 0,
            "curr": "GBP"
        }

        response = requests.get(
            url=f"{TEQUILA_ENDPOINT}/v2/search",
            headers=headers,
            params=query,
        )

        try:
            data = response.json()["data"][0]
        except IndexError:
            print(f"No flights found for {destination_city_code}.")
            return None

        flight_data = FlightData(
            price=data["price"],
            origin_city=data["route"][0]["cityFrom"],
            origin_airport=data["route"][0]["flyFrom"],
            destination_city=data["route"][0]["cityTo"],
            destination_airport=data["route"][0]["flyTo"],
            out_date=data["route"][0]["local_departure"].split("T")[0],
            return_date=data["route"][1]["local_departure"].split("T")[0]
        )
        print(f"{flight_data.destination_city}: £{flight_data.price}")
        return flight_data
```

In [3]:

```
from pprint import pprint
import requests

SHEETY_PRICES_ENDPOINT = "https://api.sheety.co/bbd43d5608ed875fd2340b7cab160474/flightDeal/prices"


class DataManager:

    def __init__(self):
        self.destination_data = {}

    def get_destination_data(self):
        response = requests.get(url=SHEETY_PRICES_ENDPOINT)
        data = response.json()
        self.destination_data = data["prices"]
        return self.destination_data

    def update_destination_codes(self):
        for city in self.destination_data:
            new_data = {
                "price": {
                    "iataCode": city["iataCode"]
                }
            }
            response = requests.put(
                url=f"{SHEETY_PRICES_ENDPOINT}/{city['id']}",
                json=new_data
            )
            print(response.text)
```

In [4]:
```
class FlightData:

    def __init__(self, price, origin_city, origin_airport, destination_city, destination_airport, out_date
        self.price = price
        self.origin_city = origin_city
        self.origin_airport = origin_airport
        self.destination_city = destination_city
        self.destination_airport = destination_airport
        self.out_date = out_date
        self.return_date = return_date
```

In [5]:
```
from twilio.rest import Client

TWILIO_SID = "AC2d4861b014392ed4ecae3129dcdb2926"
TWILIO_AUTH_TOKEN = "cd59e944190ab673edc22dc7413aceb9"
TWILIO_VIRTUAL_NUMBER = "+19123485222"
TWILIO_VERIFIED_NUMBER = "+919894100731"


class NotificationManager:

    def __init__(self):
        self.client = Client(TWILIO_SID, TWILIO_AUTH_TOKEN)

    def send_sms(self, message):
        message = self.client.messages.create(
            body=message,
            from_=TWILIO_VIRTUAL_NUMBER,
            to=TWILIO_VERIFIED_NUMBER,
        )
        # Prints if successfully sent.
        print(message.sid)
```

In [ ]:
```
import datetime
import requests
data_manager = DataManager()
sheet_data = data_manager.get_destination_data()
flight_search = FlightSearch()
notification_manager = NotificationManager()

ORIGIN_CITY_IATA = "LON"

if sheet_data[0]["iataCode"] == "":
    for row in sheet_data:
        row["iataCode"] = flight_search.get_destination_code(row["city"])
    data_manager.destination_data = sheet_data
    data_manager.update_destination_codes()

tomorrow = datetime.datetime.now() + datetime.timedelta(days=1)
six_month_from_today = datetime.datetime.now() + datetime.timedelta(days=(6 * 30))

#for destination in sheet_data:
#    flight = flight_search.check_flights(
#        ORIGIN_CITY_IATA,
#        destination["iataCode"],
#        from_time=tomorrow,
#        to_time=six_month_from_today
#    )
#    if flight.price < destination["lowestPrice"]:
#        notification_manager.send_sms(
#            message=f"Low price alert! Only £{flight.price} to fly from {flight.origin_city}-{flight.orig
#        )

def inputs_to_excel():
    print("Weclome to Srenath's Flight Club")
    print("We find best deals for you to reduce your spending savings money")
```

```
    first_name=input("Enter your First Name:")
    last_name=input("Enter your Last Name:")
    email_1=input("Enter your Email:")
    email_2=input("Enter your Email Again")
    print("You are in the Club")
    print("Your email has been added")
    url4="https://api.sheety.co/bbd43d5608ed875fd2340b7cab160474/user1/sheet1"
    user_data={
        "First Name":first_name,
        "Last Name":last_name,
        "Email":email_1
    }
    response4=requests.post(url=url4,json=user_data)
    print(response4.text)

inputs_to_excel()
```

DAY --> 41 - 42 --> HTML

DAY --> 43 - 44 --> CSS

DAY-45

Extracting details from html webpage with Beautiful Soup in Python

In [ ]:
```
#Basically this Beautiful soup is used to get the datas from a html file through python code
```

website.html

Angela Yu

Founder of The App Brewery.
I am an iOS and Web Developer. I ♡ coffee and motorcycles.

Books and Teaching

- The Complete iOS App Development Bootcamp
- The Complete Web Development Bootcamp
- 100 Days of Code - The Complete Python Bootcamp

Other Pages

My Hobbies Contact Me

In [9]:
```
#This is the module we use for beautiful soup
from bs4 import BeautifulSoup
with open("website.html",encoding="utf8") as file:
    contents=file.read()

#Creating a object from a class BeautifulSoup
#The first parameter is the html file content and the second parameter is the type of the file weather it
#If it is html we need to put it as html.parser
#If it is lxml we need to put it as lxml in the parameter
soup=BeautifulSoup(contents,"html.parser")
#Here we are taking title from the html file
soup.title

#This isone of the way to understand beautiful soup
```

Out[9]:
```
<title>Angela's Personal Site</title>
```

In [6]:
```
#In order to get the name of the tag used in the html file,

print(soup.title.name)
#Here we can see that they have used title tag
```

```
title
```

In [10]:
```
#In order to get the string of the tag("What name is added in the title tag")
soup.title.string
```

Out[10]:
```
"Angela's Personal Site"
```

In [13]:
```
#In order to view the html file
soup
```

Out[13]:
```
<!DOCTYPE html>

<html>
<head>
<meta charset="utf-8"/>
<title>Angela's Personal Site</title>
</head>
<body>
<h1 id="name">Angela Yu</h1>
<p><em>Founder of <strong><a href="https://www.appbrewery.co/">The App Brewery</a></strong>.</em></p>
<p>I am an iOS and Web Developer. I ♡ coffee and motorcycles.</p>
<hr/>
<h3 class="heading">Books and Teaching</h3>
<ul>
<li>The Complete iOS App Development Bootcamp</li>
<li>The Complete Web Development Bootcamp</li>
<li>100 Days of Code - The Complete Python Bootcamp</li>
</ul>
<hr/>
<h3 class="heading">Other Pages</h3>
<a href="https://angelabauer.github.io/cv/hobbies.html">My Hobbies</a>
<a href="https://angelabauer.github.io/cv/contact-me.html">Contact Me</a>
</body>
</html>
```

In [14]:
```
#This will pop up first tag i,e(if a is given it will pop up first a tag)
soup.a
```

Out[14]:
```
<a href="https://www.appbrewery.co/">The App Brewery</a>
```

In [15]:
```
soup.li
```

Out[15]:
```
<li>The Complete iOS App Development Bootcamp</li>
```

In order to get all the particular tag

In [17]:
```
#For this we use find_all function with get what tag you want
getting_all_a_tags= soup.find_all(name="a")
print(getting_all_a_tags)

#Remember that it will return in a list
```

```
[<a href="https://www.appbrewery.co/">The App Brewery</a>, <a href="https://angelabauer.github.io/cv/hobbi
es.html">My Hobbies</a>, <a href="https://angelabauer.github.io/cv/contact-me.html">Contact Me</a>]
```

Remember the function find_all will return a list

In [18]:
```
getting_all_li=soup.find_all(name="li")
```

```
getting_all_li
```

Out[18]:
```
[<li>The Complete iOS App Development Bootcamp</li>,
 <li>The Complete Web Development Bootcamp</li>,
 <li>100 Days of Code - The Complete Python Bootcamp</li>]
```

In [21]:
```
#In order to get only value first we need to create a for loop and use getText function to get the text al

for tag in getting_all_li:
    print(tag.getText())
#We can see that we get only text from the tags
```

```
The Complete iOS App Development Bootcamp
The Complete Web Development Bootcamp
100 Days of Code - The Complete Python Bootcamp
```

In [23]:
```
# In order to get all the links from (a) tags

for tag in getting_all_a_tags:
    print(tag.get("href"))
```

```
https://www.appbrewery.co/
https://angelabauer.github.io/cv/hobbies.html
https://angelabauer.github.io/cv/contact-me.html
```

In [24]:
```
#If the given string is not found in the tag it will return none
for tag in getting_all_a_tags:
    print(tag.get("li"))
```

```
None
None
None
```

In [ ]:
```
#Find all is to find all the details from the html file whereas
#Find is used to find only first tag the tag specified
```

In [26]:
```
# We can also find the details through attribute
#So we can consider we want  h1 tag with id specified
#If there is h1 tag and the tag id should be name then only it will return
heading=soup.find(name="h1",id="name")
heading
```

Out[26]:
```
<h1 id="name">Angela Yu</h1>
```

In [33]:
```
# We can also do it with class attribute here we use it as class_
section_heading=soup.find(name="h3",class_="heading")
section_heading
```

Out[33]:
```
<h3 class="heading">Books and Teaching</h3>
```

In [35]:
```
#Getting class value
section_heading.get("class")
#Getting h3 text
section_heading.string
```

Out[35]:
```
'Books and Teaching'
```

In [36]:
```
# In order to get a particular tag through looped p tag or a tag
#This select_one is a function with selector parameter which returns the tag and values ,we need to specif
#Here in html file a tag is inside p tag so we put it as p a it is like file inside file
company_url=soup.select_one(selector="p a")
company_url
```

Out[36]:
```
<a href="https://www.appbrewery.co/">The App Brewery</a>
```

In [37]:
```
# Select one is used to select first tag details
#Select is used to get all the details
```

In [40]:
```
#We can also get the details through the ids too
#If you are selecting ids use #-->Pound sign
name=soup.select_one(selector="#name")
name
```

Out[40]:
```
<h1 id="name">Angela Yu</h1>
```

In [41]:
```
#If we are getting details through class we use . symbol
names=soup.select_one(".heading")
```

In [42]:
```
names
```

Out[42]:
```
<h3 class="heading">Books and Teaching</h3>
```

Scraping a Live Website

In [142…
```
#So the first step is so get data from live website,we use request to get the data from website
import requests

response=requests.get("https://news.ycombinator.com/")
#You will all html file data
#print(response.text)
data=response.text
```

In [143…
```
# What if i want to get all the links and heaading from the live website through code
#So let get started
from bs4 import BeautifulSoup

soup=BeautifulSoup(data,"html.parser")
#So perfectly rendered
soup.title.string
```

Out[143…
```
'Hacker News'
```

In [144…
```
#This process below is to get data from the website and convert it into a list for access
articles=soup.find_all("a",class_="storylink")
article_text=[]
article_link=[]
for article_tag in articles:
    text=article_tag.getText()
    article_text.append(text)
    link=article_tag.get("href")
    article_link.append(link)
```

In [179…
```
article_upvotes=[score.getText() for score in soup.find_all("span",class_="score")]
```

In [146…
```
article_text
```

Out[146…
```
['The 88x31 GIF Collection',
 'Amazon is blocking Google’s FLoC',
 'The rise of E Ink Tablets and Note Takers: reMarkable 2 vs Onyx Boox Note Air',
 'On Comments in Code',
 'Richard Feynman’s Integral Trick (2018)',
 'Emacs Love Tale by sdp',
```

```
 'The home computer as a cultural object is physically vanishing',
 'How do I opt my access point out of Google Location services?',
 'Launch HN: Axolo (YC W21) – Faster pull requests and code reviews',
 'Tensil (YC S19) Is Hiring FPGA Engineers',
 'Next.js Live: Code in the Browser with ESM, ServiceWorkers, Replicache, and WASM',
 'Finish Your Stuff (2015)',
 'J Concepts in SC (SuperCollider)',
 'What We Learned Doing Fast Grants',
 'Future from a16z',
 'IBM APL2: Software withdrawal and support discontinuance',
 'A New Era for Mechanical CAD',
 'DraftKings: A $21B SPAC Betting It Can Hide Its Black Market Operations',
 'Exponential Backoff and Jitter (2015)',
 'Modelplace, the AI Model Marketplace by OpenCV',
 'Next.js 11',
 'Tailwindo: Convert Your Bootstrap CSS to Tailwind CSS',
 'The Tinkerings of Robert Noyce (1983)',
 'U.S. workers are among the most stressed in the world, new Gallup report finds',
 "Most hospitals aren't complying with price transparency rule",
 'A Timeline of CIA Atrocities (1993)',
 'Universities have formed a company that looks a lot like a patent troll',
 'Why bugs might feel “impossible”',
 "Europe's Software Problem",
 'Gravitational Lensing by Spinning Black Holes in Astrophysics, and Interstellar']
```

In [147…

```
article_link
```

Out[147…

```
['http://cyber.dabamos.de/88x31/',
 'https://digiday.com/media/amazon-is-blocking-googles-floc-and-that-could-seriously-weaken-the-fledgling-tracking-system/',
 'https://www.hanselman.com/blog/the-quiet-rise-of-e-ink-tablets-and-infinite-paper-note-takers-remarkable-2-vs-onyx-boox-note-air',
 'https://henrikwarne.com/2021/06/15/on-comments-in-code/',
 'https://www.cantorsparadise.com/richard-feynmans-integral-trick-e7afae85e25c',
 'https://emacs.love/tales/emacs-love-tale-by-sdp.html',
 'http://contemporary-home-computing.org/where-did-the-computer-go/',
 'https://support.google.com/maps/answer/1725632#how_opt_out&zippy=',
 'item?id=27515468',
 'https://www.ycombinator.com/companies/tensil/jobs/PulzAJpgj-lead-fpga-engineer',
 'http://nextjs.org/live',
 'https://250bpm.com/blog:50/',
 'https://doc.sccode.org/Guides/J-concepts-in-SC.html',
 'https://future.com/what-we-learned-doing-fast-grants/',
 'https://future.a16z.com/',
 'https://www-01.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_ca/1/897/ENUS921-031/index.html&request_locale=en',
 'https://queue.acm.org/detail.cfm?id=3469844',
 'https://hindenburgresearch.com/draftkings/',
 'https://aws.amazon.com/blogs/architecture/exponential-backoff-and-jitter/',
 'https://modelplace.ai',
 'https://nextjs.org/blog/next-11',
 'https://awssat.com/opensource/tailwindo/1_introduction.html',
 'https://web.stanford.edu/class/e145/2007_fall/materials/noyce.html',
 'https://www.cnbc.com/2021/06/15/gallup-us-workers-are-among-the-most-stressed-in-the-world.html',
 'https://www.axios.com/hospitals-price-transparency-costs-regulations-noncompliance-ebf6bd21-5709-4298-b67a-74c8a90a1ec1.html',
 'http://www.huppi.com/kangaroo/CIAtimeline.html',
 'https://www.eff.org/deeplinks/2021/06/15-universities-have-formed-company-looks-lot-patent-troll',
 'https://jvns.ca/blog/2021/06/08/reasons-why-bugs-might-feel-impossible/',
 'https://berthub.eu/articles/posts/europes-software-problem/',
 'https://arxiv.org/abs/1502.03808']
```

In [181…

```
article_upvotes
```

Out[181…

```
['143 points',
 '322 points',
 '419 points',
 '25 points',
 '158 points',
 '106 points',
 '18 points',
 '148 points',
 '59 points',
 '63 points',
```

```
 '97 points',
 '9 points',
 '124 points',
 '95 points',
 '49 points',
 '37 points',
 '98 points',
 '41 points',
 '101 points',
 '196 points',
 '46 points',
 '12 points',
 '111 points',
 '242 points',
 '123 points',
 '732 points',
 '111 points',
 '243 points',
 '33 points']
```

In [182…
```
#Therefor all the details are converted into lists
```

In [183…
```
#Now we are removing the points word and convert the string numbers into int numbers
article_upvotes=[int(score.getText().split()[0]) for score in soup.find_all(name="span",class_="score")]
```

In [184…
```
article_upvotes
```

Out[184…
```
[143,
 322,
 419,
 25,
 158,
 106,
 18,
 148,
 59,
 63,
 97,
 9,
 124,
 95,
 49,
 37,
 98,
 41,
 101,
 196,
 46,
 12,
 111,
 242,
 123,
 732,
 111,
 243,
 33]
```

In [185…
```
#To find the details of max upvote got

max_value=max(article_upvotes)
index=article_upvotes.index(max_value)

print(article_text[index])
print(article_link[index])
```

```
A Timeline of CIA Atrocities (1993)
http://www.huppi.com/kangaroo/CIAtimeline.html
```

100 Movies that You Must Watch Challange

Scraping with Python

In [212…
```
from bs4 import BeautifulSoup
import requests

response=requests.get("https://www2.bfi.org.uk/greatest-films-all-time")
data=response.text
```

In [213…
```
soup=BeautifulSoup(data,"html.parser")
soup.title.string
```

Out[213… 'The top 100 Greatest Films of All Time | Sight & Sound | BFI'

In [251…
```
all_movies = soup.select(selector="h3 a")
```

In [254…
```
num=1
with open("m.txt","w") as file1:
    for i in all_movies:
        file1.write(f"{num} {i.getText()}\n")
        num += 1
```

In [257…
```
with open("m.txt","r") as file1:
    content=file1.read()
    file1.close()

print(content)
```

```
1 Vertigo
2 Citizen Kane
3 Tokyo Story
4 La Règle du jeu
5 Sunrise: A Song of Two Humans
6 2001: A Space Odyssey
7 The Searchers
8 Man with a Movie Camera
9 The Passion of Joan of Arc
10 8½
11 Battleship Potemkin
12 L’Atalante
13 Breathless
14 Apocalypse Now
15 Late Spring
16 Au hasard Balthazar
17 Seven Samurai
18 Persona
19 Mirror
20 Singin’ in the Rain
21 L’Avventura
22 Le Mépris
23 The Godfather
24 Ordet
25 In the Mood for Love
26 Rashomon
27 Andrei Rublev
28 Mulholland Dr.
29 Stalker
30 Shoah
31 The Godfather Part II
32 Taxi Driver
33 Bicycle Thieves
34 The General
35 Metropolis
36 Psycho
37 Jeanne Dielman, 23 quai du Commerce 1080 Bruxelles
38 Sátántangó
39 The 400 Blows
40 La dolce vita
41 Journey to Italy
```

```
42 Pather Panchali
43 Some Like It Hot
44 Gertrud
45 Pierrot le fou
46 Play Time
47 Close-Up
48 The Battle of Algiers
49 Histoire(s) du cinéma
50 City Lights
51 Ugetsu monogatari
52 La Jetée
53 North by Northwest
54 Rear Window
55 Raging Bull
56 M
57 The Leopard
58 Touch of Evil
59 Sherlock Jr
60 Barry Lyndon
61 La Maman et la putain
62 Sansho Dayu
63 Wild Strawberries
64 Modern Times
65 Sunset Blvd.
66 Night of the Hunter, The
67 Pickpocket
68 Rio Bravo
69 Blade Runner
70 Blue Velvet
71 Sans Soleil
72 A Man Escaped
73 The Third Man
74 L’eclisse
75 Les enfants du paradis
76 La grande illusion
77 Nashville
78 Chinatown
79 Beau Travail
80 Once Upon a Time in the West
81 The Magnificent Ambersons
82 Lawrence of Arabia
83 The Spirit of the Beehive
84 Fanny and Alexander
85 Casablanca
86 The Colour of Pomegranates
87 Greed
88 A Brighter Summer Day
89 The Wild Bunch
90 Partie de campagne
91 Aguirre, Wrath of God
92 A Matter of Life and Death
93 The Seventh Seal
94 Un chien andalou
95 Intolerance
96 A One and a Two
97 The Life and Death of Colonel Blimp
98 Touki Bouki
99 Fear Eats the Soul
100 Imitation of Life
101 Madame de…
```

DAY-46

Create a Spotify Playlist using the Musical Time Machine

Step 1 - Scraping the Billboard Hot 100

In [6]:

```
from bs4 import BeautifulSoup
import requests

date=input("What year you would like to travel to?Type the date in this format YYYY-MM-DD:")
```

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

In [7]:
```
soup=BeautifulSoup(data,"html.parser")
```

In [8]:
```
top_100_songs=soup.find_all(name="span",class_="chart-element__information__song text--truncate color--pri
```

In [9]:
```
list_100_songs=[]
for i in top_100_songs:
    text=i.getText()
    list_100_songs.append(text)
```

In [10]:
```
list_100_songs
```

Out[10]:
```
['Cheap Thrills',
 'One Dance',
 'This Is What You Came For',
 "Can't Stop The Feeling!",
 "Don't Let Me Down",
 'Ride',
 'Needed Me',
 "Don't Mind",
 'Panda',
 'Send My Love (To Your New Lover)',
 'Rise',
 'Just Like Fire',
 'Work From Home',
 'Me Too',
 'H.O.L.Y.',
 'I Took A Pill In Ibiza',
 'Make Me...',
 'Controlla',
 'For Free',
 'Too Good',
 'Treat You Better',
 'Heathens',
 'Work',
 'Into You',
 'Never Be Like You',
 '7 Years',
 "We Don't Talk Anymore",
 'All In My Head (Flex)',
 'Stressed Out',
 'I Got The Keys',
 'Sorry',
 'Let It Go',
 'Low Life',
 'Broccoli',
 'I Hate U I Love U',
 'Dangerous Woman',
 'Love Yourself',
 'My House',
 'Cake By The Ocean',
 'Gold',
 'Close',
 'THat Part',
 'Lost Boy',
 'Wicked',
 'Sorry',
 'All The Way Up',
 'Vice',
 'From The Ground Up',
 'Never Forget You',
 'Kill Em With Kindness',
 'Hymn For The Weekend',
 'Church Bells',
```

```
'Record Year',
'Toothbrush',
'Head Over Boots',
'Wherever I Go',
'My PYT',
'Lights Come On',
'Peter Pan',
'Make You Miss Me',
'Unsteady',
'Luv',
'No Limit',
'Sit Still, Look Pretty',
'Fix',
'Different For Girls',
'American Country Love Song',
'Girls Talk Boys',
'Uber Everywhere',
'Wake Up',
'Famous',
'Pop Style',
'Lockjaw',
'Money Longer',
"She's Got A Way With Words",
'Wasted Time',
'With You Tonight / Hasta El Amanecer',
"Huntin', Fishin' And Lovin' Every Day",
'Sucker For Pain',
'You & Me',
'Mama Said',
'Brand New',
'Duele El Corazon',
'Body Say',
'No',
'M.I.L.F. $',
'Why You Always Hatin?',
"Night's On Fire",
'No Shopping',
'A Little More Summertime',
'You Was Right',
'Light It Up',
'No Problem',
'Ophelia',
'Childs Play',
'Wat U Mean (Aye, Aye, Aye)',
'Sex With Me',
'Come And See Me',
'Body',
'Lush Life']
```

Step 2 - Authentication with Spotify

In [25]:

```
CLIENT_ID="8e03ff5c6e964323b780e3424b4911df"
CLIENT_SECRET="4f38aba2edbf4d9b8b41a1f957e2e4f7"

import spotipy
from spotipy.oauth2 import SpotifyOAuth

sp = spotipy.Spotify(
    auth_manager=SpotifyOAuth(
        scope="playlist-modify-private",
        redirect_uri="http://example.com",
        client_id=CLIENT_ID,
        client_secret=CLIENT_SECRET,
        show_dialog=True,
        cache_path="token.txt"
    )
)
user_id = sp.current_user()["id"]
```

Step 3 - Search Spotify for the Songs from Step 1

In [26]:

```
song_uris = []
```

```
year = date.split("-")[0]
for song in list_100_songs:
    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.")
        pass
```

Step 4 - Creating and Adding to Spotify Playlist

In [27]:
```
playlist = sp.user_playlist_create(user=user_id, name=f"{date} Billboard 100", public=False)
# print(playlist)

sp.playlist_add_items(playlist_id=playlist["id"], items=song_uris)
```

Out[27]:
```
{'snapshot_id': 'Miw0OGJiNjlkZjQ0NTU0M2U3ZTMwNGFmYmVhNTI0YjZiMWZiYjJiZmQ5'}
```

DAY-47

Create an Automated Amazon Price Tracker

Step 1 - Use BeautifulSoup to Scrape the Product Price

In [29]:
```
from bs4 import BeautifulSoup
import requests
import lxml

headers={
    "Accept-Language":"en-US,en;q=0.9",
    "User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
}
response=requests.get("https://www.amazon.in/New-Apple-iPhone-12-128GB/dp/B08L5VJWCV/ref=sr_1_3?dchild=1&k
data=response.text
```

In [32]:
```
soup=BeautifulSoup(data,"lxml")
```

In [34]:
```
soup.title
```

Out[34]:
```
<title>New Apple iPhone 12 (128GB) - Green: Amazon.in</title>
```

In [86]:
```
price=soup.find(name="span",class_="a-size-medium a-color-price priceBlockBuyingPriceString").getText()
```

In [87]:
```
price=price.split("\xa0")[1]
price=price.split(",")
price=price[0]+price[1]
price=float(price)
price
```

Out[87]:
```
81500.0
```

In [84]:
```
type(price)
```

Out[84]:
```
float
```

Step 2 - Email Alert When Price Below Preset Value

In [88]:
```
import smtplib
username="jarvisai215@gmail.com"
password="srenath2002"

if price <= 81500:
    with smtplib.SMTP("smtp.gmail.com",587) as connection:
        #For secured and encrypted messages
        connection.starttls()
        #Login to the account
        connection.login(user=username,password=password)
        connection.sendmail(from_addr=username,to_addrs="srenath.e0120006@sret.edu.in",msg="Price Drop in
        connection.close()
```

DAY-48

In [1]:
```
#Learning about Selenium
```

Selenium Webdriver Browser and Game Playing Bot

In [2]:
```
# The selenium is to interact with html files in intermediate level
```

In [8]:
```
# The difference between selenium and beautiful soup is we can type what ever we want in the website, we c
#click particular button
```

In [4]:
```
# In order to work with selenium we need to install chrome_driver and copy the file path
chrome_driver_path="New folder (5)/chromedriver.exe"
```

In [5]:
```
# This chrome driver acts like a bridge which connects to chrome browser through selenium
```

In [5]:
```
from selenium import webdriver

# we create a object called driver from webdriver class
# As we are runing in chrome we use chrome method for execution
driver=webdriver.Chrome(executable_path=chrome_driver_path)
#Getting a request to open the page
driver.get("https://www.amazon.com/Apple-iPhone-512GB-Midnight-Green/dp/B08BHGY8M1/ref=sr_1_3?dchild=1&key
# This driver.close helps to close the browser when the work is over
#This close funcion will close only one tab
driver.close()
```

In [ ]:
```
# This quit close entire tabs in the chrome browser
driver.quit()
```

In [ ]:
```
# The first three lines are important to run each function
```

Search the website with find_element_by_id in Selenium

In [11]:
```
# In order to get price from the amazon wesite
driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://www.amazon.com/Apple-iPhone-512GB-Midnight-Green/dp/B08BHGY8M1/ref=sr_1_3?dchild=1&key
price=driver.find_element_by_id("priceblock_ourprice")
print(price.text)
driver.close()
```

```
$937.00
```

In [ ]:
```
# As we can see that it prints price from the website
```

In [12]:
```
# BeautifulSoup will get slower if the website is made up of angular or react
```

Seach the website with find_element_by_name in Selenium

In [13]:
```
driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://www.python.org/")
search_bar=driver.find_element_by_name("q")
print(search_bar)

driver.close()
```

```
<selenium.webdriver.remote.webelement.WebElement (session="0065638a7456d4a44a531995dbd87228", element="b2b
e9ce3-3a4a-4a6d-809e-dd7fa9e9fda1")>
```

In [14]:
```
#When selenium locates the element it wont print print the html tag ,it will return selenium element like
```

In [15]:
```
driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://www.python.org/")
search_bar=driver.find_element_by_name("q")
# It prints what kind it is
print(search_bar.get_attribute("placeholder"))

driver.close()
```

```
Search
```

Search the website with find_element_by_class_name

In [7]:
```
from selenium import webdriver
driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://www.python.org/")
logo=driver.find_element_by_class_name("python-logo")
#We can also print the size of the logo with this
print(logo.size)
driver.close()
```

```
{'height': 72, 'width': 255}
```

In [14]:
```
#### Search the webstie with find_element_by_css_selector
driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://www.python.org/")
#Trying to get documentation url
documentation_url=driver.find_element_by_css_selector(".documentation-widget a")
print(documentation_url.text)
driver.close()
```

```
docs.python.org
```

In [15]:
```
# If there is a situation that all the find_elements funcitons cant find the required tag
#Then we use xpath  which always work
```

In [17]:
```
driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://www.python.org/")
#We are tracking the tag with xpath
bug_link=driver.find_element_by_xpath('//*[@id="site-map"]/div[2]/div/ul/li[3]/a')
print(bug_link.text)
driver.close()
```

```
Submit Website Bug
```

Challenge: Use Selenium to Scrape Website Data

In [49]:
```
# Task is to get upcomming event in python.org with date and title
from selenium import webdriver

chrome_driver_path="New folder (5)/chromedriver.exe"
driver=webdriver.Chrome(executable_path=chrome_driver_path)

driver.get("https://www.python.org/")

mon_date_dict={}
for i in range(1,6):
    mon_date=driver.find_element_by_xpath(f'//*[@id="content"]/div/section/div[2]/div[2]/div/ul/li[{i}]/ti
    name=driver.find_element_by_xpath(f'//*[@id="content"]/div/section/div[2]/div[2]/div/ul/li[{i}]/a')
    mon_date_dict.update({i:{mon_date.text:name.text}})
driver.close()
```

In [50]:
```
mon_date_dict
```

Out[50]:
```
{1: {'06-18': 'PyCon Namibia 2021'},
 2: {'06-30': 'Pyladies Amsterdam - Bringing ML Models into Production Bootcamp - Lesson 1'},
 3: {'07-05': 'PyHEP 2021'},
 4: {'07-07': 'Pyladies Amsterdam - Bringing ML Models into Production Bootcamp - Lesson 2'},
 5: {'07-12': 'SciPy 2021'}}
```

Challenge: Use Selenium in a Blank Project & Scrape a Different Piece of Data

In [54]:
```
#Chalange is to get hte number of articles in wekipedia
from selenium import webdriver
chrome_driver_path="New folder (5)/chromedriver.exe"

driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://en.wikipedia.org/wiki/Main_Page")
number=driver.find_element_by_xpath('//*[@id="articlecount"]/a[1]')
print(number.text)
driver.close()
```

```
6,318,333
```

How to Automate Filling Out Forms and Clicking Buttons with Selenium

In [ ]:
```
# Click the link with Xpath
```

In [3]:
```
# With selenium we are going to click a particular link or a element link
#So let us take a previous example

from selenium import webdriver
chrome_driver_path="New folder (5)/chromedriver.exe"

driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://en.wikipedia.org/wiki/Main_Page")
number=driver.find_element_by_xpath('//*[@id="articlecount"]/a[1]')
number.click()
# So we can see that it automaticall clicks the button
```

In [ ]:
```
# Clicking the link with link text function
```

In [6]:
```
from selenium import webdriver
chrome_driver_path="New folder (5)/chromedriver.exe"

driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://en.wikipedia.org/wiki/Main_Page")
all_portals=driver.find_element_by_link_text("All portals")
all_portals.click()
```

Adding a input to the input bar tag in the website through Selenium

In [1]:
```
from selenium import webdriver

from selenium.webdriver.common.keys import Keys
chrome_driver_path="New folder (5)/chromedriver.exe"

driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://en.wikipedia.org/wiki/Main_Page")
#This below line helps to find the bar with name "search"
search=driver.find_element_by_name("search")
#Sending the input to the website as keys
search.send_keys("Python")
#After entering teh input it automatically press enter
search.send_keys(Keys.ENTER)
```

Challange: To fill the sigh up page with selenium

In [6]:
```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

chrome_driver_path="New folder (5)/chromedriver.exe"
driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("http://secure-retreat-92358.herokuapp.com/")

first_name=driver.find_element_by_name("fName")
first_name.send_keys("Srenath")

last_name=driver.find_element_by_name("lName")
last_name.send_keys("Kumar")

email=driver.find_element_by_name("email")
email.send_keys("sren@gmail.com")

signup=driver.find_element_by_css_selector(".form-signin button")
signup.click()
```

The Cookie Clicker Project

In [ ]:
```
from selenium import webdriver
import time

chrome_driver_path = "New folder (5)/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

Step 1 - Setup Your LinkedIn Account

In [2]:

```
# Account Created Successfully

email="jarvisai215@gmail.com"
password="srenath2002"
```

Step 2 - Automatically Login

In [9]:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from time import sleep
chrome_driver_path = "New folder (5)/chromedriver.exe"
driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://www.linkedin.com/checkpoint/rm/sign-in-another-account?fromSignIn=true&trk=guest_homep

email_id=driver.find_element_by_name("session_key")
email_id.send_keys(email)

passwords=driver.find_element_by_name("session_password")
passwords.send_keys(password)

button=driver.find_element_by_css_selector(".login__form_action_container button")
button.click()
```

Step 3 - Apply for a Job

In [20]:

```
from selenium import webdriver
```

```
from selenium.webdriver.common.keys import Keys
from time import sleep
chrome_driver_path = "New folder (5)/chromedriver.exe"
driver=webdriver.Chrome(executable_path=chrome_driver_path)
driver.get("https://www.linkedin.com/checkpoint/rm/sign-in-another-account?fromSignIn=true&trk=guest_homep

email_id=driver.find_element_by_name("session_key")
email_id.send_keys(email)

passwords=driver.find_element_by_name("session_password")
passwords.send_keys(password)

button=driver.find_element_by_css_selector(".login__form_action_container button")
button.click()
sleep(2)

search=driver.find_element_by_css_selector(".ember-view input")
search.send_keys("Python Development")
search.send_keys(Keys.ENTER)

first_job=driver.find_element_by_class_name("app-aware-link")
first_job.click()
```

DAY-50

In [ ]:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import ElementClickInterceptedException, NoSuchElementException
from time import sleep

FB_EMAIL = YOUR FACEBOOK LOGIN EMAIL
FB_PASSWORD = YOUR FACEBOOK PASSWORD

chrome_driver_path = YOUR CHROME DRIVER PATH
driver = webdriver.Chrome(executable_path=chrome_driver_path)

driver.get("http://www.tinder.com")

sleep(2)
login_button = driver.find_element_by_xpath('//*[@id="content"]/div/div[1]/div/main/div[1]/div/div/header/
login_button.click()

sleep(2)
fb_login = driver.find_element_by_xpath('//*[@id="modal-manager"]/div/div/div[1]/div/div[3]/span/div[2]/bu
fb_login.click()

sleep(2)
base_window = driver.window_handles[0]
fb_login_window = driver.window_handles[1]
driver.switch_to.window(fb_login_window)
print(driver.title)

email = driver.find_element_by_xpath('//*[@id="email"]')
password = driver.find_element_by_xpath('//*[@id="pass"]')

email.send_keys(FB_EMAIL)
password.send_keys(FB_PASSWORD)
password.send_keys(Keys.ENTER)

driver.switch_to.window(base_window)
print(driver.title)

sleep(5)
allow_location_button = driver.find_element_by_xpath('//*[@id="modal-manager"]/div/div/div/div/div[3]/butt
allow_location_button.click()
notifications_button = driver.find_element_by_xpath('//*[@id="modal-manager"]/div/div/div/div/div[3]/butto
notifications_button.click()
cookies = driver.find_element_by_xpath('//*[@id="content"]/div/div[2]/div/div/div[1]/button')
cookies.click()
```

```
for n in range(100):
    sleep(1)
    try:
        print("called")
        like_button = driver.find_element_by_xpath(
            '//*[@id="content"]/div/div[1]/div/main/div[1]/div/div/div[1]/div/div[2]/div[4]/button')
        like_button.click()
    except ElementClickInterceptedException:
        try:
            match_popup = driver.find_element_by_css_selector(".itsAMatch a")
            match_popup.click()
        except NoSuchElementException:
            sleep(2)

driver.quit()
```

DAY-51

Step 1 - Setup Your Twitter Account

In [34]:
```
PROMISED_DOWN=100
PROMISED_UP=100
chrome_driver_path = "New folder (5)/chromedriver.exe"
TWITTER_ID="9894100731"
TWITTER_PASS="srenath2002"
```

Step 2 - Create a Class

Step 3 - Get Internet Speeds

Step 4 - Building a Twitter Bot to Tweet at your Internet Provider

In [ ]:
```
from selenium import webdriver
from time import sleep
class InternetSpeedTwitterBot:
    def __init__(self):
        self.driver=webdriver.Chrome(executable_path=chrome_driver_path)
        self.down=0
        self.up=0

    def get_internet_speed(self):
        self.driver.get("https://www.speedtest.net/")
        self.driver.find_element_by_xpath('//*[@id="container"]/div/div[3]/div/div/div/div[2]/div[3]/div[1

        sleep(40)
        download_speed=self.driver.find_element_by_xpath('//*[@id="container"]/div/div[3]/div/div/div/div[
        upload_speed=self.driver.find_element_by_xpath('//*[@id="container"]/div/div[3]/div/div/div/div[2]
        return download_speed,upload_speed




    def tweet_at_provider(self,download,upload):
        text=f"Hello the internet speed is {download} d/s and {upload} u/s which is slower from the 100 d/
        self.driver.get("https://twitter.com/login")
        #sleep(10)
        self.driver.find_element_by_name("session[username_or_email]").send_keys(TWITTER_ID)
        self.driver.find_element_by_name("session[password]").send_keys(TWITTER_PASS)
        self.driver.find_element_by_xpath('//*[@id="react-root"]/div/div/div[2]/main/div/div/div[2]/form/d
        self.driver.find_element_by_xpath('//*[@id="react-root"]/div/div/div[2]/main/div/div/div/div[1]/di
        self.driver.find_element_by_xpath('//*[@id="react-root"]/div/div/div[2]/main/div/div/div/div[1]/di
```

```
tweet_process=InternetSpeedTwitterBot()
download,upload=tweet_process.get_internet_speed()

if float(download) < float(PROMISED_DOWN) or float(upload) < float(PROMISED_UP):
    tweet_process.tweet_at_provider(download,upload)
```

DAY-52

Intermediate+ Instagram Follower Bot

Step 1 - Get Your Instagram Credentials

In [6]:
```
USERNAME="jarvisai215@gmail.com"
PASSWORD="srenath2002"
CHROME_DRIVER_PATH = "New folder (5)/chromedriver.exe"
SIMILAR_ACCOUNT="chefsteps"
```

Step 2 - Create a Class

Step 3 - Login to Instagram

Step 4 - Find the followers of the target account

Step 5 - Follow all the followers

In [10]:
```
from selenium import webdriver
import time
from selenium.common.exceptions import ElementClickInterceptedException
class InstaFollower:
    def __init__(self):
        self.driver=webdriver.Chrome(executable_path=CHROME_DRIVER_PATH)

    def login(self):
        self.driver.get("https://www.instagram.com/")
        time.sleep(5)
        self.driver.find_element_by_name("username").send_keys(USERNAME)
        self.driver.find_element_by_name("password").send_keys(PASSWORD)
        self.driver.find_element_by_xpath('//*[@id="loginForm"]/div/div[3]/button/div').click()
        time.sleep(5)
        self.driver.find_element_by_xpath('//*[@id="react-root"]/section/main/div/div/div/section/div/butt
        time.sleep(5)
        self.driver.find_element_by_xpath('/html/body/div[4]/div/div/div/div[3]/button[2]').click()

    def find_followers(self):
        time.sleep(5)
        self.driver.get(f"https://www.instagram.com/{SIMILAR_ACCOUNT}")

        time.sleep(2)
        followers = self.driver.find_element_by_xpath('//*[@id="react-root"]/section/main/div/header/secti
        followers.click()

        time.sleep(2)
        modal = self.driver.find_element_by_xpath('/html/body/div[5]/div/div/div[2]')
        for i in range(10):
            #In this case we're executing some Javascript, that's what the execute_script() method does.
            #The method can accept the script as well as a HTML element.
            #The modal in this case, becomes the arguments[0] in the script.
            #Then we're using Javascript to say: "scroll the top of the modal (popup) element by the heigh
            self.driver.execute_script("arguments[0].scrollTop = arguments[0].scrollHeight", modal)
            time.sleep(2)

    def follow(self):
        all_buttons = self.driver.find_elements_by_css_selector("li button")
```

```
        for button in all_buttons:
            time.sleep(1)
            try:
                button.click()
                time.sleep(1)
            except ElementClickInterceptedException:
                cancel_button = self.driver.find_element_by_xpath('/html/body/div[6]/div/div/div/div[3]/bu
                cancel_button.click()

bot=InstaFollower()
bot.login()
bot.find_followers()
bot.follow()
```

DAY-53

Capstone Project Program Requirements

In [ ]:

```
from bs4 import BeautifulSoup
from selenium import webdriver
import lxml
import requests
from time import sleep

header = {
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/537.36 (KHTML, like Gecko)
    "Accept-Language": "en-GB,en-US;q=0.9,en;q=0.8"
}


CHROME_DRIVER_PATH = "New folder (5)/chromedriver.exe"

response=requests.get("https://www.zillow.com/homes/for_rent/1-_beds/?searchQueryState=%7B%22pagination%22
data = response.text
soup = BeautifulSoup(data, "html.parser")


all_link_elements = soup.select(".list-card-top a")
all_links=[]

for i in all_link_elements:
    href=i["href"]

    if "http" not in href:
        all_links.append(f"https://www.zillow.com{href}")
    else:
        all_links.append(href)


all_prices=soup.select(".list-card-heading div")

prices=[]
for j in all_prices:
    prices.append(j.getText())



address=[]
all_address=soup.select(".list-card-info address")
for k in all_address:
    address.append(k.getText())



driver=webdriver.Chrome(executable_path=CHROME_DRIVER_PATH)
driver.get("https://docs.google.com/forms/d/e/1FAIpQLScorpUMXgE-TRlxsMatIOEg0xRApYn1geT9ILKCgWTqPIwcyg/vie

for i in range(0,len(all_links)):
    driver.find_element_by_xpath('//*[@id="mG61Hd"]/div[2]/div/div[2]/div[1]/div/div/div[2]/div/div[1]/div
```

```
    driver.find_element_by_xpath('//*[@id="mG61Hd"]/div[2]/div/div[2]/div[2]/div/div/div[2]/div/div[1]/div
    driver.find_element_by_xpath('//*[@id="mG61Hd"]/div[2]/div/div[2]/div[3]/div/div/div[2]/div/div[1]/div
    driver.find_element_by_xpath('//*[@id="mG61Hd"]/div[2]/div/div[3]/div[1]/div/div/span/span').click()
    sleep(1)
    driver.find_element_by_xpath('/html/body/div[1]/div[2]/div[1]/div/div[4]/a').click()
```

DAY-54

Introduction to Web Development with Flask

In [ ]:
```
# Basic Flask

from flask import Flask

app=Flask(__name__)

@app.route('/')

def hello_world():
    return "hello world"
```

In [ ]:
```
Terminal Commands:
    set FLASK_APP=hello.py
    flask run
```

Passing a function as a argument in another function

In [2]:
```
def add(n1,n2):
    return n1+n2

def calculate(add,n1,n2):
    print(add(n1,n2))


calculate(add,2,3)
```

```
5
```

Nested Functions

In [5]:
```
def outer_layer():
    print("outer")
    def inner_layer():
        print("inner")
    inner_layer()

outer_layer()
```

```
outer
inner
```

In [9]:
```
# Returning a function with another function
def outer_layer():
    print("outer")
    def inner_layer():
        print("inner")
    return inner_layer    #There should not be any paranthesis
```

```
#When you create a variable and assign it will execute only outer
inner_func=outer_layer()
```

```
outer
```

In [10]:

```
#On calling inner you will get inner function
# On returning inner layer the inner_func act as outer_layer
#The inner_func() with paranthesis gets inner function
inner_func()
```

```
inner
```

Creating a decerator

In [15]:

```
import time

#Creating a decerator which has another nested function and return wrapper to delay_decerator
#So now to call decerator we use @ symbol
#This delay_decerator function delays the time for 2 seconds
#We use decerators for adding  a single function functionalities to other functionalities with decerators
def delay_decerator(function):
    def wrapper():
        time.sleep(2)
        function()
    return wrapper

@delay_decerator
def say_hello():
    print("hello")


@delay_decerator
def exit_1():
    print("Bye")

def other():
    print("Other")


#Here we can see that it takes two seconds delay and printing it
say_hello()
exit_1()

# We can add anything into the decetator to run it in other function
#Do you know the @ symbol is called as syntatic sugar
```

```
hello
Bye
```

In [17]:

```
# Method 2 of passing a decerator

decerator_function=delay_decerator(other)
decerator_function()
```

```
Other
```

Creating a decerator in flask program

In [18]:

```
from flask import Flask

app=Flask(__name__)

@app.route('/')
def hello_world():
    return "hello world"

#This condition is that if there is by in the html url it will return to this function
@app.route("/bye")
def bye():
```

```
    return "Bye"

if __name__== " main ":
    app.run()
```

In [ ]:

```
Terminal Command:
    set FLASK_APP=hello.py
    flask run


    #On adding /bye in the html url above it will return bye function
```

[Interactive Coding Exercise] Create Your Own Python Decorator

In [21]:

```
import time

# Creating a decerator
#This function is to test the speed of the time taken for running each function
def speed_calc_decorator(function):
    def calculate():
        first_run=time.time()
        function()
        second_run=time.time()
        print(f"{function.__name__} run speed:{second_run-first_run}")
    return calculate


@ speed_calc_decorator
def fast_function():
    for i in range(10000000):
        i * i

@ speed_calc_decorator
def slow_function():
    for i in range(100000000):
        i * i

# Running first function
fast_function()
#Running second_function
slow_function()

#We can see that it takes some delay after changing to the function
```

```
fast_function run speed:1.1961455345153809
slow_function run speed:11.855688333511353
```

DAY-55

Adding a input function to a decerator in order to run it in python

In [23]:

```
from flask import Flask

app=Flask(__name__)

@app.route('/')
def hello_world():
    return "hello world"

#This condition is that if there is by in the html url it will return to this function
@app.route("/bye")
def bye():
    return "Bye"

@app.route("/<name>") #This <name> gets the input from the url and renter to the python code
def user(name):
    return f"Hello {name}"
```

```
if __name__== " main ":
    app.run(debug=True)           #We use debug to reload the server when we use ctrl S
```

In [ ]:

```
Terminal Command:
    set FLASK_APP=hello.py
    flask run
```

Adding Html tags into the return function in python

In [24]:

```
# Basic Flask

from flask import Flask

app=Flask(__name__)

@app.route('/')

def hello_world():
    return "<h1> Hello World </h1>"
```

In [ ]:

```
Terminal Command:
    set FLASK_APP=hello.py
    flask run
```

Allining the h1 to center

In [25]:

```
# Basic Flask

from flask import Flask

app=Flask(__name__)

@app.route('/')

def hello_world():
    return "<h1 style=text-align:center > Hello World </h1>"
```

Adding a image into the python code through html

In [ ]:

```
# Basic Flask

from flask import Flask

app=Flask(__name__)

@app.route('/')

def hello_world():
    return "<h1 style=text-align:center > Hello World </h1>"\
            # Adding a image and also we use \ to get the same string to next line
           "<img src='https://images.unsplash.com/photo-1578439231583-9eca0a363860?ixid=MnwxMjA3fDB8MHxwaG
```

Challenge: Use Python Decorators to Style HTML Tags

In [26]:

```
from flask import Flask

app=Flask(__name__)

def bolder(function):
    def user():
```

```
        return f"<b>{function()}</b>"
    return user

def italic(function):
    def user():
        return f"<em>{function()}</em>"
    return user


@app.route('/')
@bolder

def hello_world():
    return "hello world"



if __name__== " main ":
    app.run(debug=True)
    hello_world()
```

In [ ]:

```
Terminal Command:
    set FLASK_APP=hello.py
    flask run
```

Advanced Decorators with *args and **kwargs

In [28]:

```
## ********Day 55 Start**********

## Advanced Python Decorator Functions

#creating a class
class User:
    def __init__(self, name):
        self.name = name
        self.is_logged_in = False


#Creating a decerator with unknown *args,*kwargs to the wrapper function
#This args[0] will access the first parameter in wrapper function
def is_authenticated_decorator(function):
    def wrapper(*args, **kwargs):
        if args[0].is_logged_in == True:
            function(args[0])
    return wrapper

#Here the user parameter is send to wrapper as decerator in which user is args[0] and runs according to lo
@is_authenticated_decorator
def create_blog_post(user):
    print(f"This is {user.name}'s new blog post.")

new_user = User("Srenath")
new_user.is_logged_in = True
create_blog_post(new_user)
```

```
This is Srenath's new blog post.
```

[Interactive Coding Exercise] Advanced Decorators

In [29]:

```
# Create the logging_decorator() function 👇

def logging_decerator(function):
    def wrapper(*args):
        print(f"The function name is:{function.__name__}{args}")
        result=function(args[0],args[1],args[2])
        print(f"Result:{result}")
    return wrapper
```

```
@logging_decerator
def multiply(a,b,c):

    return a*b*c

multiply(2,2,2)



# Use the decorator 👇
```

```
The function name is:multiply(2, 2, 2)
Result:8
```

Final Project - Higher or Lower URLs

In [ ]:

```
# Basic Flask

from flask import Flask
import random
app=Flask(__name__)

random_num=random.randint(0,9)


@app.route('/')
def main_page():
    return '<h1>Guess a number between 0 and 9</h1>' \
           '<img src="https://media.giphy.com/media/3o7aCSPqXE5C6T8tBC/giphy.gif" width=500 height=500>'

@app.route('/<int:guess>')
def guess_num(guess):
    if guess < random_num:
        return '<h2 style="color:red;">Two Low,try again!</h2>' \
        '<img src="https://media.giphy.com/media/jD4DwBtqPXRXa/giphy.gif" width=500 height=500>'
    elif guess > random_num:
        return '<h2 style="color:purple;">Two High,try again!</h2>' \
        '<img src="https://media.giphy.com/media/3o6ZtaO9BZHcOjmErm/giphy.gif" width=500 height=500>'
    else:
        return '<h2 style="color:green;">You Found Me !</h2>' \
        '<img src="https://media.giphy.com/media/4T7e4DmcrP9du/giphy.gif" width=500 height=500>'

if __name__ == "__main__":
    app.run(debug=True)
```

DAY-56

Rendering a html and css file to a python file for running the output

In [ ]:

```
# For adding html files you need to compulsarily create a floder called templates
```

In [2]:

```
#index.html
#Remember that the html file should be in a floder only,then only it will work

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Srenath Webpage</title>
</head>
<body>
<h1>This is Srenath Kumar</h1>
</body>
</html>
```

In [ ]:

```
#server.py

from flask import Flask,render_template

app=Flask(__name__)

@app.route('/')
def hello():
    return render_template("index.html")


if __name__ =="__main__":
    app.run(debug=True)
```

Challange : To render your own cv page

In [ ]:

```
# For rendering an image into a html file we need to create a folder called static and
#copy the image to the folder and rename the html img src to the folder path

#cv.html in template folder

<html>
 <head>
<link type="text/css" rel="stylesheet" href="https://www.dropbox.com/s/trsldt0me90jzs8/resume.css"/>
<title></title>
</head>
<body style="background-color:powderblue;">
  <img src="static/srenath.jpg" alt="photo" width="138" height="200" align="right">
  <div id="header">
  <p id="name"><b>G S SRENATH KUMAR</b></p>
        <a href="mailto:srenath.e0120006@sret.edu.in" target="_blank"><p id="email"><i>srenath.e0120006@s
     </div>
     <div class="left">
     </div>
     <div class="right">
           <h3>Self CV</h3>
           <p>
           <ul>
               <li>Address: <br>No 38/177 Gandhi Road Sriperumbudur</li>
               <li>Phone Number: 9894100731</li>
               <li>Date of Birth: 03/08/2002</li>
           </ul>
           </p>

           <h3>Educational Qualifications</h3>
           <<table border = '5'>>
               <tr id="heading">
                   <td>Qualification</td>
                   <td>Board</td>
                   <td>Percentage / Grades</td>
                   <td>Year</td>
               </tr>
               <tr>
                   <td>10th Standard</td>
                   <td>CBSE</td>
                   <td>100%</td>
                   <td>2018</td>
               </tr>
               <tr>
                   <td>12th Standard</td>
                   <td>CBSE</td>
                   <td>100%</td>
                   <td>2020</td>
               </tr>
               <tr>
                   <td>B.Tech computer science engineering</td>
                   <td>Deemed university</td>
                   <td>100%</td>
                   <td>2023</td>
```

```
                </tr>
            </table>
            <h3>Independent Courses</h3>
            <p>
            <ul>
                <li>
                <span id="course-name">CS101: Introduction to Computer Science - Building a Web Crawler</s
                <li>
                <span id="course-name"> Introduction to Computer Science I</span> -edx.org</li>
                <li>
                <span id="course-name">Calculus 1</span>Coursera.org</li>
                <li>
                <span id="course-name">Python - Fundamentals and Dynamic Programming </span> - Codecademy.
                <li>
                <span id="course-name">HTML & CSS for Beginners - Web Fundamentals</span> - Codecademy.com
                <li>

                <span id="course-name">JavaScript - Programming Basics, JS Apps and Build Games </span> </
                <li>

                <span id="course-name">Introduction to Finance</span> - Coursera.org </li>
            </ul>
            <h3>Technical Skills</h3>
            <p>
            <ul>

                <li>
                <span id="course-name">Programming Skills:</span>HTML, CSS, Python, JavaScript, learning C
                <li>
                <span id="course-name">Operating Systems:</span> Windows 10, Linux</li>
                <li>
                <span id="course-name">Application Software:</span> Adobe Photoshop, Excel spreadsheets.</
            </p>
     </div>
     <div id="footer"></div>
    </body>
</html>
```

In [ ]:

```
# server.py

from flask import Flask,render_template

app=Flask(__name__)

@app.route('/')
def hello():
    return render_template("cv.html")


if __name__ =="__main__":
    app.run(debug=True)
```

Rendering Css file to html

In [ ]:

```
# Create a folder called static and store css

#Static folder is used to store css,images
#Template folder is used to store html files

#style.css

body{
background-color:purple
}
```

In [ ]:

```
#index.html

<!DOCTYPE html>
```

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Srenath Webpage</title>
    <link rel="stylesheet" href="static/style.css">
</head>
<body>
<h1>This is Srenath Kumar</h1>
</body>
</html>
```

In [ ]:

```
#server.py

from flask import Flask,render_template

app=Flask(__name__)

@app.route('/')
def hello():
    return render_template("index.html")


if __name__ =="__main__":
    app.run(debug=True)
```

DAY-57

In [ ]:

```
# Taking an html tag as a python code with with {{}} from jinja language

#index.html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Srenath WebPage></title>
</head>
<body>
<h1>{{6*5}}</h1>

</body>
</html>
```

In [ ]:

```
#server.py

from flask import Flask,render_template

app=Flask(__name__)

@app.route('/')
def hello():
    name="Hello World"
    return render_template(f"index.html")


if __name__ =="__main__":
    app.run(debug=True)
```

Example: Sending a random number from server.py python code to html file

In [ ]:

```
# index.html

<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>Srenath WebPage></title>
</head>
<body>
<h1>{{6*5}}</h1>
<h1>Random Number:{{num}}</h1>

</body>
</html>
```

In [ ]:

```
#server.py

from flask import Flask,render_template
import random
app=Flask(__name__)

@app.route('/')
def hello():
    random_num=random.randint(1,10)
    return render_template(f"index.html",num=random_num)


if __name__ =="__main__":
    app.run(debug=True)
```

Challange: To change the year in the footer dynamically

In [ ]:

```
# index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Srenath WebPage></title>
</head>
<body>
<h1>{{6*5}}</h1>
<h1>Random Number:{{num}}</h1>

</body>
<footer> © Copyright {{cp}} Pallets.</footer>
</html>
```

In [ ]:

```
#server.py

from flask import Flask,render_template
import random
import datetime


app=Flask(__name__)

@app.route('/')
def hello():
    random_num=random.randint(1,10)
    year = datetime.datetime.now().year
    return render_template(f"index.html",num=random_num,cp=year)


if __name__ =="__main__":
    app.run(debug=True)
```

Challenge: Combining Jinja Templating with APIs

In [ ]:

```
#Adding some api to predict age and gender with given name
# index.html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Srenath WebPage></title>
</head>
<body>
<h1>Hey {{name.capitalize()}}</h1>
<h3>I think you are {{gender}}</h3>
<h3>And maybe {{age}} years old</h3>

</body>
<footer> © Copyright {{cp}} by Srenath-Industries.</footer>
</html>
```

In [ ]:

```
#server.py

from flask import Flask,render_template
import random
import datetime
import requests

app=Flask(__name__)

@app.route("/")
def hello():
    return "This is a Prediction"

@app.route('/<name>')
def predict(name):
    response = requests.get(f"https://api.agify.io/?name={name}")
    response1 = requests.get(f"https://api.genderize.io?name={name}")
    data = response.json()
    gender=response1.json()["gender"]
    name=data["name"]
    age=data["age"]
    return render_template(f"index.html",name=name,gender=gender,age=age)


if __name__ =="__main__":
    app.run(debug=True)
```

Multiline Statements with Jinja

In [ ]:

```
# Adding a python loops into the html through jinja
# we need to add {%   condition  %} as a syntax
# Getting the data from api and displaying the title
# kk.py

from flask import Flask,render_template
import random
import datetime
import requests

app=Flask(__name__)

@app.route("/")
def hello():
    return "This is a Prediction"

@app.route('/<name>')
def predict(name):
    response = requests.get(f"https://api.agify.io/?name={name}")
    response1 = requests.get(f"https://api.genderize.io?name={name}")
    data = response.json()
    gender=response1.json()["gender"]
    name=data["name"]
    age=data["age"]
```

```
    return render_template("index.html",name=name,gender=gender,age=age)


@app.route("/blogs")
def blogs():
    response=requests.get("https://api.npoint.io/82975389c85afb34e389")
    data2=response.json()
    return render_template("blogs.html",datas=data2)
if __name__ =="__main__":
    app.run(debug=True)
```

In [ ]:

```
#index.html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Srenath WebPage></title>
</head>
<body>
<h1>Hey {{name.capitalize()}}</h1>
<h3>I think you are {{gender}}</h3>
<h3>And maybe {{age}} years old</h3>

</body>
<footer> © Copyright {{cp}} by Srenath-Industries.</footer>
</html>
```

In [ ]:

```
# blogs.html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Blogs</title>
</head>
<body>
{% for blog in datas: %}
<h2>{{blog['title']}}</h2>

{% endfor %}
</body>
</html>
```

In [ ]:

```
# If condition in the html page with jinja

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Blogs</title>
</head>
<body>
{% for blog in datas: %}
{% if blog['id'] == 2: %}
<h2>{{blog['title']}}</h2>
{% endif %}


{% endfor %}
</body>
</html>
```

In [ ]:

```
from flask import Flask,render_template
import random
import datetime
```

```
import requests

app=Flask(__name__)

@app.route("/")
def hello():
    return "This is a Prediction"

@app.route('/<name>')
def predict(name):
    response = requests.get(f"https://api.agify.io/?name={name}")
    response1 = requests.get(f"https://api.genderize.io?name={name}")
    data = response.json()
    gender=response1.json()["gender"]
    name=data["name"]
    age=data["age"]
    return render_template("index.html",name=name,gender=gender,age=age)


@app.route("/blogs")
def blogs():
    response=requests.get("https://api.npoint.io/82975389c85afb34e389")
    data2=response.json()
    return render_template("blogs.html",datas=data2)
if __name__ =="__main__":
    app.run(debug=True)
```

URL Building with Flask

In [ ]:

```
# Adding a link for next website in html with syntax: {{url_for("blogs")}} where blogs is a function

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Srenath WebPage></title>
</head>
<body>
<h1>Hey {{name.capitalize()}}</h1>
<h3>I think you are {{gender}}</h3>
<h3>And maybe {{age}} years old</h3>
<a href="{{url_for('blogs')}}">Go to blog site</a>
</body>
<footer> © Copyright {{cp}} by Srenath-Industries.</footer>
</html>
```

In [ ]:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Blogs</title>
</head>
<body>
{% for blog in datas: %}
{% if blog['id'] == 2: %}
<h2>{{blog['title']}}</h2>
{% endif %}


{% endfor %}
</body>
</html>
```

In [ ]:

```
from flask import Flask,render_template
import random
import datetime
import requests
```

```
app=Flask(__name__)

@app.route("/")
def hello():
    return "This is a Prediction"

@app.route('/<name>')
def predict(name):
    response = requests.get(f"https://api.agify.io/?name={name}")
    response1 = requests.get(f"https://api.genderize.io?name={name}")
    data = response.json()
    gender=response1.json()["gender"]
    name=data["name"]
    age=data["age"]
    return render_template("index.html",name=name,gender=gender,age=age)


@app.route("/blogs")
def blogs():
    response=requests.get("https://api.npoint.io/82975389c85afb34e389")
    data2=response.json()
    return render_template("blogs.html",datas=data2)
if __name__ =="__main__":
    app.run(debug=True)
```

Blog Capstone Project Part 1 - Templating

In [ ]:
```
# Challange-1:
```

In [ ]:
```
#main.py
from flask import Flask, render_template
import requests

data=requests.get("https://api.npoint.io/bf7de003a5b591d61079").json()

app = Flask(__name__)

@app.route('/')
def home():
    return render_template("index.html",data=data)

@app.route('/post/<int:num>')
def reque(num):
    num=num-1
    list1=[]
    list1=[data[num]["title"],data[num]["subtitle"],data[num]["body"]]
    return render_template("post.html",data1=list1)


if __name__ == "__main__":
    app.run(debug=True)
```

In [ ]:
```
#post.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link href="https://fonts.googleapis.com/css2?family=Raleway" rel="stylesheet">
    <link rel="stylesheet" href="../static/css/styles.css">
</head>
<body>
<div class="wrapper">
    <div class="top">
        <div class="title"><h1>My Blog</h1></div>
           </div>
        <div class="content">
            <div class="card">
```

```
                <h1>{{data1[0]}}</h1>
                <h3>{{data1[1]}}</h3>
                <p>{{data1[2]}}</p>
            </div>

    </div>
</div>
</body>
<footer>
    <p>Made with ♥ in London.</p>
</footer>
</html>
```

In [ ]:

```
# index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link href="https://fonts.googleapis.com/css2?family=Raleway" rel="stylesheet">
    <link rel="stylesheet" href="../static/css/styles.css">
</head>
<body>
<div class="wrapper">
    <div class="top">
        <div class="title"><h1>My Blog</h1></div>
    </div>
    {% for i in data: %}
    <div class="content">
        <div class="card">
            <h2>{{i["title"]}}</h2>
            <p class="text">{{i["subtitle"]}}</p>
            <a href="{{ url_for('reque', num=i['id']) }}">Read</a>  # Sending the commands to the app.rout
        </div>
    </div>

    {% endfor %}


</div>
</body>
<footer>
    <p>Made with ♥ in London.</p>
</footer>
</html>
```

DAY-58 --> BootStrap

DAY-59

Advanced - Blog Capstone Project Part 2 - Adding Styling

In [ ]:

```
#main.py
from flask import Flask,render_template
import requests
app=Flask(__name__)

data=requests.get("https://api.npoint.io/bf7de003a5b591d61079").json()

@app.route("/")
def home():
    return render_template("index.html",data=data)

@app.route("/about")
def about():
    return render_template("about.html")
```

```
@app.route("/contact")
def contact():
    return render_template("contact.html")

@app.route("/post1/<int:num>")
def post1(num):
    num=num-1
    list2=[data[num]["title"],data[num]["subtitle"],data[num]["body"]]
    return render_template("post.html",data1=list2)

if __name__ =="__main__":
    app.run(debug=True)
```

In [ ]:

```
# index.html

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
        <meta name="description" content="" />
        <meta name="author" content="" />
        <title>Clean Blog - Start Bootstrap Theme</title>
         <link rel="icon" type="image/x-icon" href="static/assets/favicon.ico" />
        {% include "header.html" %}
    </head>
    <body>

        <!-- Navigation-->

        <!-- Page Header-->
        <header class="masthead" style="background-image: url('static/assets/img/home-bg.jpg')">
            <div class="container position-relative px-4 px-lg-5">
                <div class="row gx-4 gx-lg-5 justify-content-center">
                    <div class="col-md-10 col-lg-8 col-xl-7">
                        <div class="site-heading">
                            <h1>Clean Blog</h1>
                            <span class="subheading">A Blog Theme by Start Bootstrap</span>
                        </div>
                    </div>
                </div>
            </div>
        </header>
        <!-- Main Content-->
        <div class="container px-4 px-lg-5">
            <div class="row gx-4 gx-lg-5 justify-content-center">
                <div class="col-md-10 col-lg-8 col-xl-7">
                    <br><br><br><br>
                    <!-- Post preview-->
                    {% for i in data: %}
                    <div class="post-preview">
                        <a href="{{ url_for('post1', num=i['id']) }}">
                            <h2 class="post-title">{{i["title"]}}</h2>
                            <h3 class="post-subtitle">{{i["subtitle"]}}</h3>
                        </a>
                        <p class="post-meta">
                            Posted by
                            <a href="{{ url_for('post1', num=i['id']) }}">Start Bootstrap</a>
                            on September 24, 2021
                        </p>
                    </div>
                    <!-- Divider-->
                    <hr class="my-4" />
                    {% endfor %}

                    <!-- Pager-->
                    <div class="d-flex justify-content-end mb-4"><a class="btn btn-primary text-uppercase"
                </div>
            </div>
        </div>
```

```
        <!-- Footer-->
        <footer class="border-top">
            <div class="container px-4 px-lg-5">
                <div class="row gx-4 gx-lg-5 justify-content-center">
                    <div class="col-md-10 col-lg-8 col-xl-7">
                        <ul class="list-inline text-center">
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-twitter fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-facebook-f fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-github fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                        </ul>
                        <div class="small text-center text-muted fst-italic">Copyright &copy; Your Website
                    </div>
                </div>
            </div>
        </footer>
        <!-- Bootstrap core JS-->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"></scrip
        <!-- Core theme JS-->
        <script src="static/js/scripts.js"></script>
    </body>
</html>
```

In [ ]:

```
# about.html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
        <meta name="description" content="" />
        <meta name="author" content="" />
        <title>Clean Blog - Start Bootstrap Theme</title>

        <link rel="icon" type="image/x-icon" href="static/assets/favicon.ico" />
        <!-- Font Awesome icons (free version)-->
        <script src="https://use.fontawesome.com/releases/v5.15.3/js/all.js" crossorigin="anonymous"></scr
        <!-- Google fonts-->
        <link href="https://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic" rel="stylesh
        <link href="https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700ita
        <!-- Core theme CSS (includes Bootstrap)-->
        <link href="static/css/styles.css" rel="stylesheet" />

    </head>
    <body>
{% include "header.html" %}
        <!-- Navigation-->

        <!-- Page Header-->
<br><br><br><br>
        <header class="masthead" style="background-image: url('static/assets/img/about-bg.jpg')">
            <div class="container position-relative px-4 px-lg-5">
```

```
                <div class="row gx-4 gx-lg-5 justify-content-center">
                    <div class="col-md-10 col-lg-8 col-xl-7">
                        <div class="page-heading">
                            <h1>About Me</h1>
                            <span class="subheading">This is what I do.</span>
                            <br>
                            <br>
                        </div>
                    </div>
                </div>
            </div>
        </header>
        <!-- Main Content-->
        <main class="mb-4">
            <div class="container px-4 px-lg-5">
                <div class="row gx-4 gx-lg-5 justify-content-center">
                    <div class="col-md-10 col-lg-8 col-xl-7">
                        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Saepe nostrum ullam e
                        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eius praesentium recu
                        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aut consequuntur magn
                    </div>
                </div>
            </div>
        </main>
        <!-- Footer-->
        <footer class="border-top">
            <div class="container px-4 px-lg-5">
                <div class="row gx-4 gx-lg-5 justify-content-center">
                    <div class="col-md-10 col-lg-8 col-xl-7">
                        <ul class="list-inline text-center">
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-twitter fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-facebook-f fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-github fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                        </ul>
                        <div class="small text-center text-muted fst-italic">Copyright &copy; Your Website
                    </div>
                </div>
            </div>
        </footer>
        <!-- Bootstrap core JS-->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"></scrip
        <!-- Core theme JS-->
        <script src="static/js/scripts.js"></script>
    </body>
</html>
```

In [ ]:

```
# contact.html
<!DOCTYPE html>
<html lang="en">
    <head>
```

```
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
        <meta name="description" content="" />
        <meta name="author" content="" />
        <title>Clean Blog - Start Bootstrap Theme</title>
        <link rel="icon" type="image/x-icon" href="static/assets/favicon.ico" />
        <!-- Font Awesome icons (free version)-->
        <script src="https://use.fontawesome.com/releases/v5.15.3/js/all.js" crossorigin="anonymous"></scr
        <!-- Google fonts-->
        <link href="https://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic" rel="stylesh
        <link href="https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700ita
        <!-- Core theme CSS (includes Bootstrap)-->
        <link href="static/css/styles.css" rel="stylesheet" />
    </head>
    <body>
        <!-- Navigation-->
        {% include "header.html" %}
<br><br><br><br><br>
        <!-- Page Header-->
        <header class="masthead" style="background-image: url('static/assets/img/contact-bg.jpg')">
            <div class="container position-relative px-4 px-lg-5">
                <div class="row gx-4 gx-lg-5 justify-content-center">
                    <div class="col-md-10 col-lg-8 col-xl-7">
                        <div class="page-heading">
                            <h1>Contact Me</h1>
                            <span class="subheading">Have questions? I have answers.</span>
                        </div>
                    </div>
                </div>
            </div>
        </header>
        <!-- Main Content-->
        <main class="mb-4">
            <div class="container px-4 px-lg-5">
                <div class="row gx-4 gx-lg-5 justify-content-center">
                    <div class="col-md-10 col-lg-8 col-xl-7">
                        <p>Want to get in touch? Fill out the form below to send me a message and I will g
                        <div class="my-5">
                            <!-- * * * * * * * * * * * * * * *-->
                            <!-- * * SB Forms Contact Form * *-->
                            <!-- * * * * * * * * * * * * * * *-->
                            <!-- This form is pre-integrated with SB Forms.-->
                            <!-- To make this form functional, sign up at-->
                            <!-- https://startbootstrap.com/solution/contact-forms-->
                            <!-- to get an API token!-->
                            <form id="contactForm" data-sb-form-api-token="API_TOKEN">
                                <div class="form-floating">
                                    <input class="form-control" id="name" type="text" placeholder="Enter y
                                    <label for="name">Name</label>
                                    <div class="invalid-feedback" data-sb-feedback="name:required">A name
                                </div>
                                <div class="form-floating">
                                    <input class="form-control" id="email" type="email" placeholder="Enter
                                    <label for="email">Email address</label>
                                    <div class="invalid-feedback" data-sb-feedback="email:required">An ema
                                    <div class="invalid-feedback" data-sb-feedback="email:email">Email is
                                </div>
                                <div class="form-floating">
                                    <input class="form-control" id="phone" type="tel" placeholder="Enter y
                                    <label for="phone">Phone Number</label>
                                    <div class="invalid-feedback" data-sb-feedback="phone:required">A phon
                                </div>
                                <div class="form-floating">
                                    <textarea class="form-control" id="message" placeholder="Enter your me
                                    <label for="message">Message</label>
                                    <div class="invalid-feedback" data-sb-feedback="message:required">A me
                                </div>
                                <br />
                                <!-- Submit success message-->
                                <!----->
                                <!-- This is what your users will see when the form-->
                                <!-- has successfully submitted-->
```

```
                                <div class="d-none" id="submitSuccessMessage">
                                    <div class="text-center mb-3">
                                        <div class="fw-bolder">Form submission successful!</div>
                                        To activate this form, sign up at
                                        <br />
                                        <a href="https://startbootstrap.com/solution/contact-forms">https:
                                    </div>
                                </div>
                                <!-- Submit error message-->
                                <!---->
                                <!-- This is what your users will see when there is-->
                                <!-- an error submitting the form-->
                                <div class="d-none" id="submitErrorMessage"><div class="text-center text-d
                                <!-- Submit Button-->
                                <button class="btn btn-primary text-uppercase disabled" id="submitButton"
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </main>
        <!-- Footer-->
        <footer class="border-top">
            <div class="container px-4 px-lg-5">
                <div class="row gx-4 gx-lg-5 justify-content-center">
                    <div class="col-md-10 col-lg-8 col-xl-7">
                        <ul class="list-inline text-center">
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-twitter fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-facebook-f fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-github fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                        </ul>
                        <div class="small text-center text-muted fst-italic">Copyright &copy; Your Website
                    </div>
                </div>
            </div>
        </footer>
        <!-- Bootstrap core JS-->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"></scrip
        <!-- Core theme JS-->
        <script src="js/scripts.js"></script>
        <!-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *-->
        <!-- * *                               SB Forms JS                                   * *-->
        <!-- * * Activate your form at https://startbootstrap.com/solution/contact-forms * *-->
        <!-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *-->
        <script src="https://cdn.startbootstrap.com/sb-forms-latest.js"></script>
    </body>
</html>
```

In [ ]:

```
#header.html
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

        <!-- Font Awesome icons (free version)-->
        <script src="https://use.fontawesome.com/releases/v5.15.3/js/all.js" crossorigin="anonymous"></scr
        <!-- Google fonts-->
        <link href="https://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic" rel="stylesh
        <link href="https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700ita
        <!-- Core theme CSS (includes Bootstrap)-->
        <link href="static/css/styles.css" rel="stylesheet" />
        <!--Java Script -->
        <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha3
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sh

</head>
<body>
 <!-- Navigation -->
  <nav class="navbar navbar-expand-lg navbar-light fixed-top" id="mainNav">
    <div class="container">
      <a class="navbar-brand" href="index.html">Start Bootstrap</a>
      <button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse" data-target
        Menu
        <i class="fas fa-bars"></i>
      </button>
      <div class="collapse navbar-collapse" id="navbarResponsive">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item">
            <a class="nav-link" href="{{url_for('home')}}">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{{url_for('about')}}">About</a>
          </li>

          <li class="nav-item">
            <a class="nav-link" href="{{url_for('contact')}}">Contact</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>

</body>
</html>
```

In [ ]:

```
#post.html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
        <meta name="description" content="" />
        <meta name="author" content="" />
        <title>Clean Blog - Start Bootstrap Theme</title>
        <link rel="icon" type="image/x-icon" href="static/assets/favicon.ico" />
        <!-- Font Awesome icons (free version)-->
        <script src="https://use.fontawesome.com/releases/v5.15.3/js/all.js" crossorigin="anonymous"></scr
        <!-- Google fonts-->
        <link href="https://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic" rel="stylesh
        <link href="https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700ita
        <!-- Core theme CSS (includes Bootstrap)-->
        <link href="css/styles.css" rel="stylesheet" />
    </head>
    <body>
        <!-- Navigation-->
     {% include "header.html" %}
        <!-- Page Header-->
        <header class="masthead" style="background-image: url('static/assets/img/post-bg.jpg')">
```

```
    <div class="container position-relative px-4 px-lg-5">
        <div class="row gx-4 gx-lg-5 justify-content-center">
            <div class="col-md-10 col-lg-8 col-xl-7">
                <div class="post-heading">



                </div>
            </div>
        </div>

    </div>

</header>
<div style="text-align: center">
    <br>
        <br>
        <br><br>
        <br><br>
        <br><br>

         <h1>{{data1[0]}}</h1>
                    <h2 class="subheading">{{data1[1]}}</h2>
                   <p class="subheading">{{data1[2]}}</p>
                    <span class="meta">
                        Posted by
                        <a href="#!">Start Bootstrap</a>
                        on August 24, 2021
                    </span>

</div>

<!-- Post Content-->
<article class="mb-4">
    <div class="container px-4 px-lg-5">
        <div class="row gx-4 gx-lg-5 justify-content-center">
            <div class="col-md-10 col-lg-8 col-xl-7">
            </div></div></div>

</article>
<!-- Footer-->
<footer class="border-top">
    <div class="container px-4 px-lg-5">
        <div class="row gx-4 gx-lg-5 justify-content-center">
            <div class="col-md-10 col-lg-8 col-xl-7">
                <ul class="list-inline text-center">
                    <li class="list-inline-item">
                        <a href="#!">
                            <span class="fa-stack fa-lg">
                                <i class="fas fa-circle fa-stack-2x"></i>
                                <i class="fab fa-twitter fa-stack-1x fa-inverse"></i>
                            </span>
                        </a>
                    </li>
                    <li class="list-inline-item">
                        <a href="#!">
                            <span class="fa-stack fa-lg">
                                <i class="fas fa-circle fa-stack-2x"></i>
                                <i class="fab fa-facebook-f fa-stack-1x fa-inverse"></i>
                            </span>
                        </a>
                    </li>
                    <li class="list-inline-item">
                        <a href="#!">
                            <span class="fa-stack fa-lg">
                                <i class="fas fa-circle fa-stack-2x"></i>
                                <i class="fab fa-github fa-stack-1x fa-inverse"></i>
                            </span>
                        </a>
                    </li>
                </ul>
                <div class="small text-center text-muted fst-italic">Copyright &copy; Your Website
```

```
                    </div>
                </div>
            </div>
        </footer>
        <!-- Bootstrap core JS-->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"></scrip
        <!-- Core theme JS-->
        <script src="static/js/scripts.js"></script>
    </body>
</html>
```

DAY-60

Make POST Requests with Flask and HTML Forms

In [ ]:

```
# Creating a forms with html and getting the inputted data from the html tags

#index.html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>


<form action="{{ url_for('login') }}" method="post">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" ><br>
  <label for="password">Password:</label><br>
  <input type="text" id="password" name="password" ><br><br>
  <input type="submit" value="Submit">
</form>


</body>
</html>
```

In [ ]:

```
#main.py

from flask import Flask , render_template,request

app=Flask(__name__)

@app.route("/")
def hello():
    return render_template("index.html")


@app.route("/login",methods=["GET","POST"])
def login():
    if request.method == "POST":
        fname=request.form["fname"]
        password=request.form["password"]
        return f"<h1>Name:{fname},Password:{password}</h1>"


if __name__ =="__main__":
    app.run(debug=True)
```

In [2]:

```
# CHALLENGE: Update the code in contact.html and main.py so that you print the information the
# user has entered into the form and return a <h1> that says "Successfully sent your message".
```

In [ ]:
```python
# main.py

from flask import Flask,render_template,request
import requests
app=Flask(__name__)

data=requests.get("https://api.npoint.io/bf7de003a5b591d61079").json()

@app.route("/")
def home():
    return render_template("index.html",data=data)

@app.route("/about")
def about():
    return render_template("about.html")

@app.route("/contact")
def contact():
    return render_template("contact.html")

@app.route("/post1/<int:num>")
def post1(num):
    num=num-1
    list2=[data[num]["title"],data[num]["subtitle"],data[num]["body"]]
    return render_template("post.html",data1=list2)

@app.route("/form-entry",methods=["GET","POST"])
def recieve_data():
    data=request.form
    print(data["name"])
    print(data["email"])
    print(data["phone"])
    print(data["message"])
    return "<h1>Sucessfully added </h1>"

if __name__ =="__main__":
    app.run(debug=True)
```

In [ ]:
```python
# contact.html

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
        <meta name="description" content="" />
        <meta name="author" content="" />
        <title>Clean Blog - Start Bootstrap Theme</title>
        <link rel="icon" type="image/x-icon" href="static/assets/favicon.ico" />
        <!-- Font Awesome icons (free version)-->
        <script src="https://use.fontawesome.com/releases/v5.15.3/js/all.js" crossorigin="anonymous"></scr
        <!-- Google fonts-->
        <link href="https://fonts.googleapis.com/css?family=Lora:400,700,400italic,700italic" rel="stylesh
        <link href="https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700ita
        <!-- Core theme CSS (includes Bootstrap)-->
        <link href="static/css/styles.css" rel="stylesheet" />
    </head>
    <body>
        <!-- Navigation-->
        {% include "header.html" %}
 <br><br><br><br><br>
        <!-- Page Header-->
        <header class="masthead" style="background-image: url('static/assets/img/contact-bg.jpg')">
            <div class="container position-relative px-4 px-lg-5">
                <div class="row gx-4 gx-lg-5 justify-content-center">
                    <div class="col-md-10 col-lg-8 col-xl-7">
                        <div class="page-heading">
                            <h1>Contact Me</h1>
                            <span class="subheading">Have questions? I have answers.</span>
                        </div>
                    </div>
```

```
            </div>
        </div>
    </header>
    <!-- Main Content-->
    <main class="mb-4">
        <div class="container px-4 px-lg-5">
            <div class="row gx-4 gx-lg-5 justify-content-center">
                <div class="col-md-10 col-lg-8 col-xl-7">
                    <p>Want to get in touch? Fill out the form below to send me a message and I will g
                    <div class="my-5">
                        <!-- * * * * * * * * * * * * * * *-->
                        <!-- * * SB Forms Contact Form * *-->
                        <!-- * * * * * * * * * * * * * * *-->
                        <!-- This form is pre-integrated with SB Forms.-->
                        <!-- This form is pre-integrated with SB Forms.-->
                        <!-- To make this form functional, sign up at-->
                        <!-- https://startbootstrap.com/solution/contact-forms-->
                        <!-- to get an API token!-->


                      <form name="sentMessage" id="contactForm" action="{{ url_for('recieve_data') }}" met
      <div class="control-group">
        <div class="form-group floating-label-form-group controls">
          <label>Name</label>
          <input type="text" name="name" class="form-control" placeholder="Name" id="name" required da
          <p class="help-block text-danger"></p>
        </div>
      </div>
      <div class="control-group">
        <div class="form-group floating-label-form-group controls">
          <label>Email Address</label>
          <input type="email" name="email" class="form-control" placeholder="Email Address" id="email"
          <p class="help-block text-danger"></p>
        </div>
      </div>
      <div class="control-group">
        <div class="form-group col-xs-12 floating-label-form-group controls">
          <label>Phone Number</label>
          <input type="tel" name="phone" class="form-control" placeholder="Phone Number" id="phone" re
          <p class="help-block text-danger"></p>
        </div>
      </div>
      <div class="control-group">
        <div class="form-group floating-label-form-group controls">
          <label>Message</label>
          <textarea rows="5" name="message" class="form-control" placeholder="Message" id="message" re
          <p class="help-block text-danger"></p>
        </div>
      </div>
      <br>
      <div id="success"></div>
      <button type="submit" class="btn btn-primary" id="sendMessageButton">Send</button>
    </form>
                    </div>
                </div>
            </div>
        </div>
    </main>
    <!-- Footer-->
    <footer class="border-top">
        <div class="container px-4 px-lg-5">
            <div class="row gx-4 gx-lg-5 justify-content-center">
                <div class="col-md-10 col-lg-8 col-xl-7">
                    <ul class="list-inline text-center">
                        <li class="list-inline-item">
                            <a href="#!">
                                <span class="fa-stack fa-lg">
                                    <i class="fas fa-circle fa-stack-2x"></i>
                                    <i class="fab fa-twitter fa-stack-1x fa-inverse"></i>
                                </span>
                            </a>
                        </li>
```

```
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-facebook-f fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                            <li class="list-inline-item">
                                <a href="#!">
                                    <span class="fa-stack fa-lg">
                                        <i class="fas fa-circle fa-stack-2x"></i>
                                        <i class="fab fa-github fa-stack-1x fa-inverse"></i>
                                    </span>
                                </a>
                            </li>
                        </ul>
                        <div class="small text-center text-muted fst-italic">Copyright &copy; Your Website
                    </div>
                </div>
            </div>
        </footer>
        <!-- Bootstrap core JS-->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"></scrip
        <!-- Core theme JS-->
        <script src="js/scripts.js"></script>
        <!-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *-->
        <!-- * *                               SB Forms JS                               * *-->
        <!-- * * Activate your form at https://startbootstrap.com/solution/contact-forms * *-->
        <!-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *-->
        <script src="https://cdn.startbootstrap.com/sb-forms-latest.js"></script>
    </body>
</html>
```

DAY-61

Creating Forms with Flask-WTF

In [ ]:

```
#main.py

from flask import Flask, render_template
from flask_wtf import FlaskForm                                 # two reqired packages for creating html form
from wtforms import StringField

app = Flask(__name__)
app.secret_key = "any-string-you-want-just-keep-it-secret"

# WE are creating a class LoginForm and adding FlaskForm features to the LoginForm i,e like making a copy
# Crating a variable or a attribute called email,password and creating a Email and Password Input Box with
class LoginForm(FlaskForm):
    #StringField is to create input box and Email is the name assigned in it i,e it displays it before inp
    email = StringField(label='Email')       #It automatically creates name and a input box and also we ca
    password = StringField(label='Password')

@app.route("/")
def home():
    return render_template('index.html')

@app.route("/login")
def login():
    login=LoginForm()
    return render_template("login.html",login=login)


if __name__ == '__main__':
    app.run(debug=True)
```

In [ ]:

```
#index.html
<!DOCTYPE HTML>
<html>
<head>
<title>Secrets</title>
</head>
<body>
<div class="jumbotron">
<div class="container">
<h1>Welcome</h1>
<p>Are you ready to discover my secret?</p>
<button class="btn btn-primary btn-lg" href=" {{ url_for('login') }} ">Login</button>
</div>
</div>
</body>
</html>
```

You might have already done this, but in the Quickstart, they set the form action to "/", which is a static path. It's always a good idea to use dynamically built urls like this

In [ ]:
```
#login.html

<!DOCTYPE HTML>

<html>
<head>
<title>Login</title>
</head>
<body>
       <div class="container">
<h1>Login</h1>
#Instead of having static way("/login") we can use dynamic way of url_for('login') it will be good
#Because the dynamic way it runs continously and gets updated
<form method="POST" action="{{ url_for('login') }}">
{{login.csrf_token}}
{{login.email.label}} {{login.email(size=30)}}
{{login.password.label}}{{login.password(size=30)}}
<input type="submit" value="Log In">
</form>
        <!-- This is where our form will go. -->
        </div>
</body>
</html>
```

Code Improvements for Our WTForms

In [1]:
```
# https://wtforms.readthedocs.io/en/2.3.x/fields/#basic-fields
```

In [ ]:
```
#Changing the String field to Password Fiels
```

The arguments given when creating a StringField or PasswordField is for the label property of the form field. Even though the Quickstart doesn't add it, I prefer adding the keyword argument when it's not clear what the argument is for.

In [ ]:
```
This is the (label) property in use
in login.html when our form object is passed over.
```

In [ ]:
```
from flask import Flask, render_template
from flask_wtf import FlaskForm
from wtforms import StringField,PasswordField

app = Flask(__name__)
app.secret_key = "any-string-you-want-just-keep-it-secret"


class LoginForm(FlaskForm):
    email = StringField(label='Email')
```

```
    #In PasswordField we get the password as .........
    password = PasswordField(label='Password')

@app.route("/")
def home():
    return render_template('index.html')

@app.route("/login")
def login():
    login=LoginForm()
    return render_template("login.html",login=login)


if __name__ == '__main__':
    app.run(debug=True)
```

In [ ]:

```
#Index.html and login.html files are same
```

Adding Submitfield in the Code and reordering the code

In [ ]:

```
#main.py
from flask import Flask, render_template
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField


class LoginForm(FlaskForm):
    email = StringField(label='Email')
    password = PasswordField(label='Password')
    submit = SubmitField(label="Log In")


app = Flask(__name__)
app.secret_key = "any-string-you-want-just-keep-it-secret"


@app.route("/")
def home():
    return render_template("index.html")


@app.route("/login")
def login():
    login_form = LoginForm()
    return render_template('login.html', form=login_form)


if __name__ == '__main__':
    app.run(debug=True)
```

In [ ]:

```
#login.html
<html>
<head>
<title>Login</title>
</head>
<body>
    <div class="container">
        <h1>Login</h1>
        <form method="POST" action="{{ url_for('login') }}">
            {{ form.csrf_token }}
            <p>
            {{ form.email.label }} <br> {{ form.email(size=30) }}
            </p>
            <p>
            {{ form.password.label }} <br> {{ form.password(size=30) }}
            </p>
            {{ form.submit }}
```

```
        </form>
    </div>
</body>
</html>
```

In [ ]:

```
#index.html
<!DOCTYPE HTML>
<html>
<head>
<title>Secrets</title>
</head>
<body>
<div class="jumbotron">
<div class="container">
    <h1>Welcome</h1>
    <p>Are you ready to discover my secret?</p>
    <button class="btn btn-primary btn-lg" href=" {{ url_for('login') }} ">Login</button>
</div>
</div>
</body>
</html>
```

Adding Validation to Forms with Flask-WTF

In [3]:

```
# https://wtforms.readthedocs.io/en/2.3.x/crash_course/#validators
```

In [4]:

```
# It will not accept the person to click the button unless all input box are filled
# i,e like adding mandatory fields
#Same index.html and login.html

#main.py
from flask import Flask, render_template
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
#We need to import DataRequired method from validators class
from wtforms.validators import DataRequired


class LoginForm(FlaskForm):
    email = StringField(label='Email',validators=[DataRequired()]) #For the mandatory inputs we need to ad
    password = PasswordField(label='Password',validators=[DataRequired()])
    submit = SubmitField(label="Log In")


app = Flask(__name__)
app.secret_key = "any-string-you-want-just-keep-it-secret"


@app.route("/")
def home():
    return render_template("index.html")


@app.route("/login")
def login():
    login_form = LoginForm()
    return render_template('login.html', form=login_form)


if __name__ == '__main__':
    app.run(debug=True)
```

Displaying Error in the console

In [ ]:

```
#https://wtforms.readthedocs.io/en/2.3.x/crash_course/#displaying-errors
```

In [ ]:
```
# It will display error if there is a error

#login.html
<html>
<head>
    <title>Login</title>
</head>
<body>
    <div class="container">
        <h1>Login</h1>
        <form method="POST" action="{{ url_for('login') }}">
            {{ form.csrf_token }}
            <p>
                {{ form.email.label }} <br> {{ form.email(size=30) }}
                {% for error in form.email.error: %}           #Creating a for loop to display all the err
                <span style="color:red">{{error}}</span>
                {% endfor %}
            </p>
            <p>
            {{ form.password.label }} <br> {{ form.password(size=30) }}
                {% for error in form.password.error: %}      #Creating a for loop to display all the error
                <span style="color:red">{{error}}</span>
                {% endfor %}
            </p>
            {{ form.submit }}
        </form>
    </div>
</body>
</html>
```

Validate the user's entry when they hit submit

In [ ]:
```
#main.py
from flask import Flask, render_template
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import DataRequired


class LoginForm(FlaskForm):
    email = StringField(label='Email',validators=[DataRequired()])
    password = PasswordField(label='Password',validators=[DataRequired()])
    submit = SubmitField(label="Log In")


app = Flask(__name__)
app.secret_key = "any-string-you-want-just-keep-it-secret"


@app.route("/")
def home():
    return render_template("index.html")

#Making th elogin url to get and post method
@app.route("/login",methods=["GET","POST"])
def login():
    login_form = LoginForm()
    #Making the submit button responsive
    login_form.validate_on_submit()
    return render_template('login.html', form=login_form)


if __name__ == '__main__':
    app.run(debug=True)


#index.html and login.html are same
```

In [ ]:
```
# In order to remove all validation we use novalidate syntak next to url place
```

```
<html>
<head>
    <title>Login</title>
</head>
<body>
    <div class="container">
        <h1>Login</h1>
        <form method="POST" action="{{ url_for('login') }}" novalidate >  #Here we put novalidate
            {{ form.csrf_token }}
            <p>
                {{ form.email.label }} <br> {{ form.email(size=30) }}
                {% for error in form.email.error: %}
                <span style="color:red">{{error}}</span>
                {% endfor %}
            </p>
            <p>
            {{ form.password.label }} <br> {{ form.password(size=30) }}
                {% for error in form.password.error: %}
                <span style="color:red">{{error}}</span>
                {% endfor %}
            </p>
            {{ form.submit }}
        </form>
    </div>
</body>
</html>
```

Add validators in email and add password length of 8 characters

In [5]:
```
#https://wtforms.readthedocs.io/en/2.3.x/validators/#module-wtforms.validators
```

Adding password autharisation to the code

In [ ]:
```
#main.py

from flask import Flask, render_template
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import DataRequired,Length,Email


class LoginForm(FlaskForm):
    email = StringField(label='Email',validators= [DataRequired(),Email()])
    password = PasswordField(label='Password',validators=[Length(max=8,message="Field must be at least 8 c
    submit = SubmitField(label="Log In")


app = Flask(__name__)
app.secret_key = "any-string-you-want-just-keep-it-secret"


@app.route("/")
def home():
    return render_template("index.html")


@app.route("/login",methods=["GET","POST"])
def login():
    login_form = LoginForm()
    if login_form.validate_on_submit():  #This condition will check is there a submit button available
        if login_form.email.data == "admin@email.com" and login_form.password.data == "12345678":  #Condit
            return render_template("success.html") #It renders the html page
        else:
            return render_template("denied.html")  #It renders denied html page
    return render_template('login.html', form=login_form)
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

In [ ]:

```
# index.html

<!DOCTYPE HTML>
<html>
<head>
<title>Secrets</title>
</head>
<body>
<div class="jumbotron">
<div class="container">
    <h1>Welcome</h1>
    <p>Are you ready to discover my secret?</p>
    <button class="btn btn-primary btn-lg" href=" {{ url_for('login') }} ">Login</button>
</div>
</div>
</body>
</html>
```

In [ ]:

```
# login.html

<html>
<head>
    <title>Login</title>
</head>
<body>
    <div class="container">
        <h1>Login</h1>
        <form method="POST" action="{{ url_for('login') }}"  >
            {{ form.csrf_token }}
            <p>
                {{ form.email.label }} <br> {{ form.email(size=30) }}
                {% for error in form.email.error: %}
                <span style="color:red">{{error}}</span>
                {% endfor %}
            </p>
            <p>
            {{ form.password.label }} <br> {{ form.password(size=30) }}
                {% for error in form.password.error: %}
                <span style="color:red">{{error}}</span>
                {% endfor %}
            </p>
            {{ form.submit }}
        </form>
    </div>
</body>
</html>
```

In [ ]:

```
#sucess.html

<!DOCTYPE HTML>
<html>
<head>
    <title>Success</title>
</head>
<body>
    <div class="container">
        <h1>Top Secret </h1>
        <iframe src="https://giphy.com/embed/Ju7l5y9osyymQ" width="480" height="360" frameBorder="0" class
        <p><a href="https://giphy.com/gifs/rick-astley-Ju7l5y9osyymQ">via GIPHY</a></p>
    </div>
</body>
</html>
```

In [ ]:

```
# denied.html

<!DOCTYPE HTML>
<html>
<head>
<title>Access Denied</title>
</head>
<body>
<div class="container">
    <h1>Access Denied </h1>
    <iframe src="https://giphy.com/embed/1xeVd1vr43nHO" width="480" height="271" frameBorder="0" class="gi
    <p><a href="https://giphy.com/gifs/cheezburger-funny-dog-fails-1xeVd1vr43nHO">via GIPHY</a></p>
</div>
</body>
</html>
```

Inheriting Templates Using Jinja2

In [6]:
```
# Template inheritance is similar to Class inheritance, you can take a parent template and extend its styl
```

In [ ]:
```
Previously, we saw that we can inject a header.html and footer.html using Jinja and the code might look so

{% include "header.html" %}
Web page content
{% include "footer.html" %}


This is a really flexible way of using Jinja to Template a website. It means that if your header and foote
can just insert them into all your webpages.
```

In [ ]:
```
Template Inheritance
However, often you'll find that you actually want to use the same design template for your entire website,
change some code in your header or footer. In these cases, it's better to use Template Inheritance instead

Template inheritance is similar to Class inheritance, you can take a parent template and extend its stylin



For example, if we create a base.html file that has the following code:

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    {% block content %}{% endblock %}
</body>
</html>
It has predefined areas (or blocks) where new content can be inserted by a child webpage inheriting from t

1. We could re-write the success.html page to inherit from this base.html template:

#1.
{% extends "base.html" %}
#2.
{% block title %}Success{% endblock %}
#3.
{% block content %}
   <div class="container">
      <h1>Top Secret </h1>
      <iframe src="https://giphy.com/embed/Ju7l5y9osyymQ" width="480" height="360" frameBorder="0" class="
      <p><a href="https://giphy.com/gifs/rick-astley-Ju7l5y9osyymQ">via GIPHY</a></p>
   </div>
{% endblock %}

#1. This line of code tells the templating engine (Jinja) to use "base.html" as the template for this page
```

```
#2. This block inserts a custom title into the header of the template.

#3. This block provides the content of the website. The part that is going to vary between webpages.
```

In [ ]:

```
# Changing the sucess.html code which extends base.html

#base.html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    {% block content %}{% endblock %}
</body>
</html>
```

In [ ]:

```
#sucess.html

<!DOCTYPE HTML>
<html>
<head>
<title>Success</title>
</head>
<body>
{% extends "base.html" %}
{% block title %}Success{% endblock %}
{% block content %}
<div class="container">
    <h1>Top Secret </h1>
    <iframe src="https://giphy.com/embed/Ju7l5y9osyymQ" width="480" height="360" frameBorder="0" class="gi
    <p><a href="https://giphy.com/gifs/rick-astley-Ju7l5y9osyymQ">via GIPHY</a></p>
</div>
{% endblock %}
</body>
</html>
```

In [ ]:

```
#denied.html

<!DOCTYPE HTML>
<html>
<head>
<title>Access Denied</title>
</head>
<body>
{% extends "base.html" %}
{% block title %}Success{% endblock %}
{% block content %}
<div class="container">
    <h1>Access Denied </h1>
    <iframe src="https://giphy.com/embed/1xeVd1vr43nHO" width="480" height="271" frameBorder="0" class="gi
    <p><a href="https://giphy.com/gifs/cheezburger-funny-dog-fails-1xeVd1vr43nHO">via GIPHY</a></p>
</div>
{% endblock %}
</body>
</html>
```

In [ ]:

```
# Adding some background color feature wi=hich inherit both sucess and denied page
#base.html

<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
    <style>
{% block styling %}              #If you are adding a style in python format we need to add block styling
body{
    background: purple;         #Adding purple background to base.html and to inherited html files like su
}
{% endblock %}
</style>
</head>
<body>
    {% block content %}{% endblock %}
</body>
</html>
```

So now you can see how easy it is to modify all web pages in your website if you use the same template. But what if on the denied page we also wanted to make the

red? We would need to modify the internal styling in the

	Undergraduate Major	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
0	Accounting	46000.0	77100.0	42200.0	152000.0	Business
1	Aerospace Engineering	57700.0	101000.0	64300.0	161000.0	STEM
2	Agriculture	42600.0	71900.0	36300.0	150000.0	Business
3	Anthropology	36800.0	61500.0	33800.0	138000.0	HASS
4	Architecture	41600.0	76800.0	50600.0	136000.0	Business

In [9]:
```
#Viewing the no of rows and columns
df.shape
```

Out[9]:
```
(51, 6)
```

In [10]:
```
#Viewing the list of columns
df.columns
```

Out[10]:
```
Index(['Undergraduate Major', 'Starting Median Salary',
       'Mid-Career Median Salary', 'Mid-Career 10th Percentile Salary',
       'Mid-Career 90th Percentile Salary', 'Group'],
      dtype='object')
```

In [11]:
```
#Checking is there any missing values
#If there are details available it returns False
#If there is nan then it returns true
df.isna()
```

Out[11]:

	Undergraduate Major	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False
6	False	False	False	False	False	False
7	False	False	False	False	False	False
8	False	False	False	False	False	False
9	False	False	False	False	False	False
10	False	False	False	False	False	False
11	False	False	False	False	False	False
12	False	False	False	False	False	False
13	False	False	False	False	False	False
14	False	False	False	False	False	False
15	False	False	False	False	False	False
16	False	False	False	False	False	False
17	False	False	False	False	False	False
18	False	False	False	False	False	False
19	False	False	False	False	False	False
20	False	False	False	False	False	False
21	False	False	False	False	False	False
22	False	False	False	False	False	False
23	False	False	False	False	False	False
24	False	False	False	False	False	False
25	False	False	False	False	False	False
26	False	False	False	False	False	False
27	False	False	False	False	False	False
28	False	False	False	False	False	False
29	False	False	False	False	False	False
30	False	False	False	False	False	False
31	False	False	False	False	False	False
32	False	False	False	False	False	False
33	False	False	False	False	False	False
34	False	False	False	False	False	False
35	False	False	False	False	False	False
36	False	False	False	False	False	False
37	False	False	False	False	False	False
38	False	False	False	False	False	False

	Undergraduate Major	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
39	False	False	False	False	False	False
40	False	False	False	False	False	False
41	False	False	False	False	False	False
42	False	False	False	False	False	False
43	False	False	False	False	False	False
44	False	False	False	False	False	False
45	False	False	False	False	False	False
46	False	False	False	False	False	False
47	False	False	False	False	False	False
48	False	False	False	False	False	False
49	False	False	False	False	False	False
50	False	True	True	True	True	True

In [16]:
```
df.tail()
```

Out[16]:

	Undergraduate Major	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
46	Psychology	35900.0	60400.0	31600.0	127000.0	HASS
47	Religion	34100.0	52000.0	29700.0	96400.0	HASS
48	Sociology	36500.0	58200.0	30700.0	118000.0	HASS
49	Spanish	34000.0	53100.0	31000.0	96400.0	HASS
50	Source: PayScale Inc.	NaN	NaN	NaN	NaN	NaN

In [18]:
```
# Droping the all the nan values rows
df.dropna(inplace=True)
```

In [20]:
```
df.tail()
```

Out[20]:

	Undergraduate Major	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
45	Political Science	40800.0	78200.0	41200.0	168000.0	HASS
46	Psychology	35900.0	60400.0	31600.0	127000.0	HASS
47	Religion	34100.0	52000.0	29700.0	96400.0	HASS
48	Sociology	36500.0	58200.0	30700.0	118000.0	HASS
49	Spanish	34000.0	53100.0	31000.0	96400.0	HASS

In [21]:
```
# For accessing the particular column  we used squared brackets []
df["Group"]
```

Out[21]:
```
0     Business
1         STEM
2     Business
3         HASS
4     Business
```

```
5         HASS
6         STEM
7     Business
8         STEM
9         STEM
10        STEM
11        HASS
12        STEM
13        STEM
14    Business
15        HASS
16        HASS
17    Business
18        HASS
19        STEM
20        HASS
21        HASS
22    Business
23    Business
24        HASS
25        STEM
26        HASS
27    Business
28        HASS
29    Business
30        STEM
31        STEM
32        HASS
33        HASS
34        HASS
35        STEM
36    Business
37        STEM
38        STEM
39        HASS
40    Business
41        HASS
42        HASS
43        STEM
44        STEM
45        HASS
46        HASS
47        HASS
48        HASS
49        HASS
Name: Group, dtype: object
```

In [24]:
```
#Finding the maximum salary
df["Starting Median Salary"].max()
```

Out[24]:
```
74300.0
```

In [28]:
```
# idxmax() helps to return the index value whose salary is in max value
df["Starting Median Salary"].idxmax()
```

Out[28]:
```
43
```

In [30]:
```
df["Starting Median Salary"].loc[43]
```

Out[30]:
```
74300.0
```

In [31]:
```
# With this loc we can track all the details in the row
df.loc[43]
```

Out[31]:
```
Undergraduate Major                 Physician Assistant
Starting Median Salary                          74300.0
Mid-Career Median Salary                        91700.0
Mid-Career 10th Percentile Salary               66400.0
Mid-Career 90th Percentile Salary              124000.0
```

```
Group                                 STEM
Name: 43, dtype: object
```

CHALLANGE

In [33]:
```
df.head()
```

Out[33]:

	Undergraduate Major	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
0	Accounting	46000.0	77100.0	42200.0	152000.0	Business
1	Aerospace Engineering	57700.0	101000.0	64300.0	161000.0	STEM
2	Agriculture	42600.0	71900.0	36300.0	150000.0	Business
3	Anthropology	36800.0	61500.0	33800.0	138000.0	HASS
4	Architecture	41600.0	76800.0	50600.0	136000.0	Business

In [45]:
```
df.loc[df["Mid-Career Median Salary"].idxmax()]
```

Out[45]:
```
Undergraduate Major                 Chemical Engineering
Starting Median Salary                           63200.0
Mid-Career Median Salary                        107000.0
Mid-Career 10th Percentile Salary                71900.0
Mid-Career 90th Percentile Salary               194000.0
Group                                               STEM
Name: 8, dtype: object
```

In [44]:
```
df.loc[df["Starting Median Salary"].idxmin()]
```

Out[44]:
```
Undergraduate Major                 Spanish
Starting Median Salary              34000.0
Mid-Career Median Salary            53100.0
Mid-Career 10th Percentile Salary   31000.0
Mid-Career 90th Percentile Salary   96400.0
Group                                  HASS
Name: 49, dtype: object
```

In [43]:
```
df.loc[df["Mid-Career Median Salary"].idxmin()]
```

Out[43]:
```
Undergraduate Major                 Education
Starting Median Salary                34900.0
Mid-Career Median Salary              52000.0
Mid-Career 10th Percentile Salary     29300.0
Mid-Career 90th Percentile Salary    102000.0
Group                                   HASS
Name: 18, dtype: object
```

In [47]:
```
# Subtract function which helps to subtract all the data
df['Mid-Career 90th Percentile Salary'].subtract(df['Mid-Career 10th Percentile Salary'])
```

Out[47]:
```
0     109800.0
1      96700.0
2     113700.0
3     104200.0
4      85400.0
5      96200.0
6      98100.0
7     108200.0
8     122100.0
9     102700.0
10     84600.0
11    105500.0
12     95900.0
13     98000.0
14    114700.0
```

```
15     74800.0
16    116300.0
17    159400.0
18     72700.0
19     98700.0
20     99600.0
21    102100.0
22    147800.0
23     70000.0
24     92000.0
25    111000.0
26     76000.0
27     66400.0
28    112000.0
29     88500.0
30    115900.0
31     84500.0
32     71300.0
33    118800.0
34    106600.0
35    100700.0
36    132900.0
37    137800.0
38     99300.0
39    107300.0
40     50700.0
41     65300.0
42    132500.0
43     57600.0
44    122000.0
45    126800.0
46     95400.0
47     66700.0
48     87300.0
49     65400.0
dtype: float64
```

In [48]:
```
# Adding a Column to the dataframe
#Adding the column in the first column
df.insert(1,"After Subtract",df['Mid-Career 90th Percentile Salary'].subtract(df['Mid-Career 10th Percenti
```

In [49]:
```
df
```

Out[49]:

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
0	Accounting	109800.0	46000.0	77100.0	42200.0	152000.0	Business
1	Aerospace Engineering	96700.0	57700.0	101000.0	64300.0	161000.0	STEM
2	Agriculture	113700.0	42600.0	71900.0	36300.0	150000.0	Business
3	Anthropology	104200.0	36800.0	61500.0	33800.0	138000.0	HASS
4	Architecture	85400.0	41600.0	76800.0	50600.0	136000.0	Business
5	Art History	96200.0	35800.0	64900.0	28800.0	125000.0	HASS
6	Biology	98100.0	38800.0	64800.0	36900.0	135000.0	STEM
7	Business Management	108200.0	43000.0	72100.0	38800.0	147000.0	Business
8	Chemical Engineering	122100.0	63200.0	107000.0	71900.0	194000.0	STEM
9	Chemistry	102700.0	42600.0	79900.0	45300.0	148000.0	STEM
10	Civil Engineering	84600.0	53900.0	90500.0	63400.0	148000.0	STEM
11	Communications	105500.0	38100.0	70000.0	37500.0	143000.0	HASS
12	Computer Engineering	95900.0	61400.0	105000.0	66100.0	162000.0	STEM
13	Computer Science	98000.0	55900.0	95500.0	56000.0	154000.0	STEM

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
14	Construction	114700.0	53700.0	88900.0	56300.0	171000.0	Business
15	Criminal Justice	74800.0	35000.0	56300.0	32200.0	107000.0	HASS
16	Drama	116300.0	35900.0	56900.0	36700.0	153000.0	HASS
17	Economics	159400.0	50100.0	98600.0	50600.0	210000.0	Business
18	Education	72700.0	34900.0	52000.0	29300.0	102000.0	HASS
19	Electrical Engineering	98700.0	60900.0	103000.0	69300.0	168000.0	STEM
20	English	99600.0	38000.0	64700.0	33400.0	133000.0	HASS
21	Film	102100.0	37900.0	68500.0	33900.0	136000.0	HASS
22	Finance	147800.0	47900.0	88300.0	47200.0	195000.0	Business
23	Forestry	70000.0	39100.0	62600.0	41000.0	111000.0	Business
24	Geography	92000.0	41200.0	65500.0	40000.0	132000.0	HASS
25	Geology	111000.0	43500.0	79500.0	45000.0	156000.0	STEM
26	Graphic Design	76000.0	35700.0	59800.0	36000.0	112000.0	HASS
27	Health Care Administration	66400.0	38800.0	60600.0	34600.0	101000.0	Business
28	History	112000.0	39200.0	71000.0	37000.0	149000.0	HASS
29	Hospitality & Tourism	88500.0	37800.0	57500.0	35500.0	124000.0	Business
30	Industrial Engineering	115900.0	57700.0	94700.0	57100.0	173000.0	STEM
31	Information Technology (IT)	84500.0	49100.0	74800.0	44500.0	129000.0	STEM
32	Interior Design	71300.0	36100.0	53200.0	35700.0	107000.0	HASS
33	International Relations	118800.0	40900.0	80900.0	38200.0	157000.0	HASS
34	Journalism	106600.0	35600.0	66700.0	38400.0	145000.0	HASS
35	Management Information Systems (MIS)	100700.0	49200.0	82300.0	45300.0	146000.0	STEM
36	Marketing	132900.0	40800.0	79600.0	42100.0	175000.0	Business
37	Math	137800.0	45400.0	92400.0	45200.0	183000.0	STEM
38	Mechanical Engineering	99300.0	57900.0	93600.0	63700.0	163000.0	STEM
39	Music	107300.0	35900.0	55000.0	26700.0	134000.0	HASS
40	Nursing	50700.0	54200.0	67000.0	47600.0	98300.0	Business
41	Nutrition	65300.0	39900.0	55300.0	33900.0	99200.0	HASS
42	Philosophy	132500.0	39900.0	81200.0	35500.0	168000.0	HASS
43	Physician Assistant	57600.0	74300.0	91700.0	66400.0	124000.0	STEM
44	Physics	122000.0	50300.0	97300.0	56000.0	178000.0	STEM
45	Political Science	126800.0	40800.0	78200.0	41200.0	168000.0	HASS
46	Psychology	95400.0	35900.0	60400.0	31600.0	127000.0	HASS
47	Religion	66700.0	34100.0	52000.0	29700.0	96400.0	HASS
48	Sociology	87300.0	36500.0	58200.0	30700.0	118000.0	HASS
49	Spanish	65400.0	34000.0	53100.0	31000.0	96400.0	HASS

In [51]:
```
#Sorting the values by after Subtract

df.sort_values("After Subtract",ascending=False)
```

Out[51]:

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
17	Economics	159400.0	50100.0	98600.0	50600.0	210000.0	Business
22	Finance	147800.0	47900.0	88300.0	47200.0	195000.0	Business
37	Math	137800.0	45400.0	92400.0	45200.0	183000.0	STEM
36	Marketing	132900.0	40800.0	79600.0	42100.0	175000.0	Business
42	Philosophy	132500.0	39900.0	81200.0	35500.0	168000.0	HASS
45	Political Science	126800.0	40800.0	78200.0	41200.0	168000.0	HASS
8	Chemical Engineering	122100.0	63200.0	107000.0	71900.0	194000.0	STEM
44	Physics	122000.0	50300.0	97300.0	56000.0	178000.0	STEM
33	International Relations	118800.0	40900.0	80900.0	38200.0	157000.0	HASS
16	Drama	116300.0	35900.0	56900.0	36700.0	153000.0	HASS
30	Industrial Engineering	115900.0	57700.0	94700.0	57100.0	173000.0	STEM
14	Construction	114700.0	53700.0	88900.0	56300.0	171000.0	Business
2	Agriculture	113700.0	42600.0	71900.0	36300.0	150000.0	Business
28	History	112000.0	39200.0	71000.0	37000.0	149000.0	HASS
25	Geology	111000.0	43500.0	79500.0	45000.0	156000.0	STEM
0	Accounting	109800.0	46000.0	77100.0	42200.0	152000.0	Business
7	Business Management	108200.0	43000.0	72100.0	38800.0	147000.0	Business
39	Music	107300.0	35900.0	55000.0	26700.0	134000.0	HASS
34	Journalism	106600.0	35600.0	66700.0	38400.0	145000.0	HASS
11	Communications	105500.0	38100.0	70000.0	37500.0	143000.0	HASS
3	Anthropology	104200.0	36800.0	61500.0	33800.0	138000.0	HASS
9	Chemistry	102700.0	42600.0	79900.0	45300.0	148000.0	STEM
21	Film	102100.0	37900.0	68500.0	33900.0	136000.0	HASS
35	Management Information Systems (MIS)	100700.0	49200.0	82300.0	45300.0	146000.0	STEM
20	English	99600.0	38000.0	64700.0	33400.0	133000.0	HASS
38	Mechanical Engineering	99300.0	57900.0	93600.0	63700.0	163000.0	STEM
19	Electrical Engineering	98700.0	60900.0	103000.0	69300.0	168000.0	STEM
6	Biology	98100.0	38800.0	64800.0	36900.0	135000.0	STEM
13	Computer Science	98000.0	55900.0	95500.0	56000.0	154000.0	STEM
1	Aerospace Engineering	96700.0	57700.0	101000.0	64300.0	161000.0	STEM
5	Art History	96200.0	35800.0	64900.0	28800.0	125000.0	HASS
12	Computer Engineering	95900.0	61400.0	105000.0	66100.0	162000.0	STEM
46	Psychology	95400.0	35900.0	60400.0	31600.0	127000.0	HASS
24	Geography	92000.0	41200.0	65500.0	40000.0	132000.0	HASS
29	Hospitality & Tourism	88500.0	37800.0	57500.0	35500.0	124000.0	Business

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
48	Sociology	87300.0	36500.0	58200.0	30700.0	118000.0	HASS
4	Architecture	85400.0	41600.0	76800.0	50600.0	136000.0	Business
10	Civil Engineering	84600.0	53900.0	90500.0	63400.0	148000.0	STEM
31	Information Technology (IT)	84500.0	49100.0	74800.0	44500.0	129000.0	STEM
26	Graphic Design	76000.0	35700.0	59800.0	36000.0	112000.0	HASS
15	Criminal Justice	74800.0	35000.0	56300.0	32200.0	107000.0	HASS
18	Education	72700.0	34900.0	52000.0	29300.0	102000.0	HASS
32	Interior Design	71300.0	36100.0	53200.0	35700.0	107000.0	HASS
23	Forestry	70000.0	39100.0	62600.0	41000.0	111000.0	Business
47	Religion	66700.0	34100.0	52000.0	29700.0	96400.0	HASS
27	Health Care Administration	66400.0	38800.0	60600.0	34600.0	101000.0	Business
49	Spanish	65400.0	34000.0	53100.0	31000.0	96400.0	HASS
41	Nutrition	65300.0	39900.0	55300.0	33900.0	99200.0	HASS
43	Physician Assistant	57600.0	74300.0	91700.0	66400.0	124000.0	STEM
40	Nursing	50700.0	54200.0	67000.0	47600.0	98300.0	Business

In [53]:
```
df.head()
```

Out[53]:

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
0	Accounting	109800.0	46000.0	77100.0	42200.0	152000.0	Business
1	Aerospace Engineering	96700.0	57700.0	101000.0	64300.0	161000.0	STEM
2	Agriculture	113700.0	42600.0	71900.0	36300.0	150000.0	Business
3	Anthropology	104200.0	36800.0	61500.0	33800.0	138000.0	HASS
4	Architecture	85400.0	41600.0	76800.0	50600.0	136000.0	Business

In [55]:
```
df.sort_values("Mid-Career 90th Percentile Salary",ascending=False)
```

Out[55]:

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
17	Economics	159400.0	50100.0	98600.0	50600.0	210000.0	Business
22	Finance	147800.0	47900.0	88300.0	47200.0	195000.0	Business
8	Chemical Engineering	122100.0	63200.0	107000.0	71900.0	194000.0	STEM
37	Math	137800.0	45400.0	92400.0	45200.0	183000.0	STEM
44	Physics	122000.0	50300.0	97300.0	56000.0	178000.0	STEM
36	Marketing	132900.0	40800.0	79600.0	42100.0	175000.0	Business
30	Industrial Engineering	115900.0	57700.0	94700.0	57100.0	173000.0	STEM
14	Construction	114700.0	53700.0	88900.0	56300.0	171000.0	Business

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
45	Political Science	126800.0	40800.0	78200.0	41200.0	168000.0	HASS
42	Philosophy	132500.0	39900.0	81200.0	35500.0	168000.0	HASS
19	Electrical Engineering	98700.0	60900.0	103000.0	69300.0	168000.0	STEM
38	Mechanical Engineering	99300.0	57900.0	93600.0	63700.0	163000.0	STEM
12	Computer Engineering	95900.0	61400.0	105000.0	66100.0	162000.0	STEM
1	Aerospace Engineering	96700.0	57700.0	101000.0	64300.0	161000.0	STEM
33	International Relations	118800.0	40900.0	80900.0	38200.0	157000.0	HASS
25	Geology	111000.0	43500.0	79500.0	45000.0	156000.0	STEM
13	Computer Science	98000.0	55900.0	95500.0	56000.0	154000.0	STEM
16	Drama	116300.0	35900.0	56900.0	36700.0	153000.0	HASS
0	Accounting	109800.0	46000.0	77100.0	42200.0	152000.0	Business
2	Agriculture	113700.0	42600.0	71900.0	36300.0	150000.0	Business
28	History	112000.0	39200.0	71000.0	37000.0	149000.0	HASS
10	Civil Engineering	84600.0	53900.0	90500.0	63400.0	148000.0	STEM
9	Chemistry	102700.0	42600.0	79900.0	45300.0	148000.0	STEM
7	Business Management	108200.0	43000.0	72100.0	38800.0	147000.0	Business
35	Management Information Systems (MIS)	100700.0	49200.0	82300.0	45300.0	146000.0	STEM
34	Journalism	106600.0	35600.0	66700.0	38400.0	145000.0	HASS
11	Communications	105500.0	38100.0	70000.0	37500.0	143000.0	HASS
3	Anthropology	104200.0	36800.0	61500.0	33800.0	138000.0	HASS
21	Film	102100.0	37900.0	68500.0	33900.0	136000.0	HASS
4	Architecture	85400.0	41600.0	76800.0	50600.0	136000.0	Business
6	Biology	98100.0	38800.0	64800.0	36900.0	135000.0	STEM
39	Music	107300.0	35900.0	55000.0	26700.0	134000.0	HASS
20	English	99600.0	38000.0	64700.0	33400.0	133000.0	HASS
24	Geography	92000.0	41200.0	65500.0	40000.0	132000.0	HASS
31	Information Technology (IT)	84500.0	49100.0	74800.0	44500.0	129000.0	STEM
46	Psychology	95400.0	35900.0	60400.0	31600.0	127000.0	HASS
5	Art History	96200.0	35800.0	64900.0	28800.0	125000.0	HASS
29	Hospitality & Tourism	88500.0	37800.0	57500.0	35500.0	124000.0	Business
43	Physician Assistant	57600.0	74300.0	91700.0	66400.0	124000.0	STEM
48	Sociology	87300.0	36500.0	58200.0	30700.0	118000.0	HASS
26	Graphic Design	76000.0	35700.0	59800.0	36000.0	112000.0	HASS
23	Forestry	70000.0	39100.0	62600.0	41000.0	111000.0	Business
32	Interior Design	71300.0	36100.0	53200.0	35700.0	107000.0	HASS
15	Criminal Justice	74800.0	35000.0	56300.0	32200.0	107000.0	HASS
18	Education	72700.0	34900.0	52000.0	29300.0	102000.0	HASS

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
27	Health Care Administration	66400.0	38800.0	60600.0	34600.0	101000.0	Business
41	Nutrition	65300.0	39900.0	55300.0	33900.0	99200.0	HASS
40	Nursing	50700.0	54200.0	67000.0	47600.0	98300.0	Business
47	Religion	66700.0	34100.0	52000.0	29700.0	96400.0	HASS
49	Spanish	65400.0	34000.0	53100.0	31000.0	96400.0	HASS

In [56]:
```
df.sort_values("After Subtract",ascending=False)
```

Out[56]:

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
17	Economics	159400.0	50100.0	98600.0	50600.0	210000.0	Business
22	Finance	147800.0	47900.0	88300.0	47200.0	195000.0	Business
37	Math	137800.0	45400.0	92400.0	45200.0	183000.0	STEM
36	Marketing	132900.0	40800.0	79600.0	42100.0	175000.0	Business
42	Philosophy	132500.0	39900.0	81200.0	35500.0	168000.0	HASS
45	Political Science	126800.0	40800.0	78200.0	41200.0	168000.0	HASS
8	Chemical Engineering	122100.0	63200.0	107000.0	71900.0	194000.0	STEM
44	Physics	122000.0	50300.0	97300.0	56000.0	178000.0	STEM
33	International Relations	118800.0	40900.0	80900.0	38200.0	157000.0	HASS
16	Drama	116300.0	35900.0	56900.0	36700.0	153000.0	HASS
30	Industrial Engineering	115900.0	57700.0	94700.0	57100.0	173000.0	STEM
14	Construction	114700.0	53700.0	88900.0	56300.0	171000.0	Business
2	Agriculture	113700.0	42600.0	71900.0	36300.0	150000.0	Business
28	History	112000.0	39200.0	71000.0	37000.0	149000.0	HASS
25	Geology	111000.0	43500.0	79500.0	45000.0	156000.0	STEM
0	Accounting	109800.0	46000.0	77100.0	42200.0	152000.0	Business
7	Business Management	108200.0	43000.0	72100.0	38800.0	147000.0	Business
39	Music	107300.0	35900.0	55000.0	26700.0	134000.0	HASS
34	Journalism	106600.0	35600.0	66700.0	38400.0	145000.0	HASS
11	Communications	105500.0	38100.0	70000.0	37500.0	143000.0	HASS
3	Anthropology	104200.0	36800.0	61500.0	33800.0	138000.0	HASS
9	Chemistry	102700.0	42600.0	79900.0	45300.0	148000.0	STEM
21	Film	102100.0	37900.0	68500.0	33900.0	136000.0	HASS
35	Management Information Systems (MIS)	100700.0	49200.0	82300.0	45300.0	146000.0	STEM
20	English	99600.0	38000.0	64700.0	33400.0	133000.0	HASS
38	Mechanical Engineering	99300.0	57900.0	93600.0	63700.0	163000.0	STEM
19	Electrical Engineering	98700.0	60900.0	103000.0	69300.0	168000.0	STEM

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
6	Biology	98100.0	38800.0	64800.0	36900.0	135000.0	STEM
13	Computer Science	98000.0	55900.0	95500.0	56000.0	154000.0	STEM
1	Aerospace Engineering	96700.0	57700.0	101000.0	64300.0	161000.0	STEM
5	Art History	96200.0	35800.0	64900.0	28800.0	125000.0	HASS
12	Computer Engineering	95900.0	61400.0	105000.0	66100.0	162000.0	STEM
46	Psychology	95400.0	35900.0	60400.0	31600.0	127000.0	HASS
24	Geography	92000.0	41200.0	65500.0	40000.0	132000.0	HASS
29	Hospitality & Tourism	88500.0	37800.0	57500.0	35500.0	124000.0	Business
48	Sociology	87300.0	36500.0	58200.0	30700.0	118000.0	HASS
4	Architecture	85400.0	41600.0	76800.0	50600.0	136000.0	Business
10	Civil Engineering	84600.0	53900.0	90500.0	63400.0	148000.0	STEM
31	Information Technology (IT)	84500.0	49100.0	74800.0	44500.0	129000.0	STEM
26	Graphic Design	76000.0	35700.0	59800.0	36000.0	112000.0	HASS
15	Criminal Justice	74800.0	35000.0	56300.0	32200.0	107000.0	HASS
18	Education	72700.0	34900.0	52000.0	29300.0	102000.0	HASS
32	Interior Design	71300.0	36100.0	53200.0	35700.0	107000.0	HASS
23	Forestry	70000.0	39100.0	62600.0	41000.0	111000.0	Business
47	Religion	66700.0	34100.0	52000.0	29700.0	96400.0	HASS
27	Health Care Administration	66400.0	38800.0	60600.0	34600.0	101000.0	Business
49	Spanish	65400.0	34000.0	53100.0	31000.0	96400.0	HASS
41	Nutrition	65300.0	39900.0	55300.0	33900.0	99200.0	HASS
43	Physician Assistant	57600.0	74300.0	91700.0	66400.0	124000.0	STEM
40	Nursing	50700.0	54200.0	67000.0	47600.0	98300.0	Business

In [60]:
```
df.sort_values("Mid-Career Median Salary",ascending=False)
```

Out[60]:

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
8	Chemical Engineering	122100.0	63200.0	107000.0	71900.0	194000.0	STEM
12	Computer Engineering	95900.0	61400.0	105000.0	66100.0	162000.0	STEM
19	Electrical Engineering	98700.0	60900.0	103000.0	69300.0	168000.0	STEM
1	Aerospace Engineering	96700.0	57700.0	101000.0	64300.0	161000.0	STEM
17	Economics	159400.0	50100.0	98600.0	50600.0	210000.0	Business
44	Physics	122000.0	50300.0	97300.0	56000.0	178000.0	STEM
13	Computer Science	98000.0	55900.0	95500.0	56000.0	154000.0	STEM
30	Industrial Engineering	115900.0	57700.0	94700.0	57100.0	173000.0	STEM
38	Mechanical Engineering	99300.0	57900.0	93600.0	63700.0	163000.0	STEM
37	Math	137800.0	45400.0	92400.0	45200.0	183000.0	STEM

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
43	Physician Assistant	57600.0	74300.0	91700.0	66400.0	124000.0	STEM
10	Civil Engineering	84600.0	53900.0	90500.0	63400.0	148000.0	STEM
14	Construction	114700.0	53700.0	88900.0	56300.0	171000.0	Business
22	Finance	147800.0	47900.0	88300.0	47200.0	195000.0	Business
35	Management Information Systems (MIS)	100700.0	49200.0	82300.0	45300.0	146000.0	STEM
42	Philosophy	132500.0	39900.0	81200.0	35500.0	168000.0	HASS
33	International Relations	118800.0	40900.0	80900.0	38200.0	157000.0	HASS
9	Chemistry	102700.0	42600.0	79900.0	45300.0	148000.0	STEM
36	Marketing	132900.0	40800.0	79600.0	42100.0	175000.0	Business
25	Geology	111000.0	43500.0	79500.0	45000.0	156000.0	STEM
45	Political Science	126800.0	40800.0	78200.0	41200.0	168000.0	HASS
0	Accounting	109800.0	46000.0	77100.0	42200.0	152000.0	Business
4	Architecture	85400.0	41600.0	76800.0	50600.0	136000.0	Business
31	Information Technology (IT)	84500.0	49100.0	74800.0	44500.0	129000.0	STEM
7	Business Management	108200.0	43000.0	72100.0	38800.0	147000.0	Business
2	Agriculture	113700.0	42600.0	71900.0	36300.0	150000.0	Business
28	History	112000.0	39200.0	71000.0	37000.0	149000.0	HASS
11	Communications	105500.0	38100.0	70000.0	37500.0	143000.0	HASS
21	Film	102100.0	37900.0	68500.0	33900.0	136000.0	HASS
40	Nursing	50700.0	54200.0	67000.0	47600.0	98300.0	Business
34	Journalism	106600.0	35600.0	66700.0	38400.0	145000.0	HASS
24	Geography	92000.0	41200.0	65500.0	40000.0	132000.0	HASS
5	Art History	96200.0	35800.0	64900.0	28800.0	125000.0	HASS
6	Biology	98100.0	38800.0	64800.0	36900.0	135000.0	STEM
20	English	99600.0	38000.0	64700.0	33400.0	133000.0	HASS
23	Forestry	70000.0	39100.0	62600.0	41000.0	111000.0	Business
3	Anthropology	104200.0	36800.0	61500.0	33800.0	138000.0	HASS
27	Health Care Administration	66400.0	38800.0	60600.0	34600.0	101000.0	Business
46	Psychology	95400.0	35900.0	60400.0	31600.0	127000.0	HASS
26	Graphic Design	76000.0	35700.0	59800.0	36000.0	112000.0	HASS
48	Sociology	87300.0	36500.0	58200.0	30700.0	118000.0	HASS
29	Hospitality & Tourism	88500.0	37800.0	57500.0	35500.0	124000.0	Business
16	Drama	116300.0	35900.0	56900.0	36700.0	153000.0	HASS
15	Criminal Justice	74800.0	35000.0	56300.0	32200.0	107000.0	HASS
41	Nutrition	65300.0	39900.0	55300.0	33900.0	99200.0	HASS
39	Music	107300.0	35900.0	55000.0	26700.0	134000.0	HASS

	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary	Group
32	Interior Design	71300.0	36100.0	53200.0	35700.0	107000.0	HASS
49	Spanish	65400.0	34000.0	53100.0	31000.0	96400.0	HASS
18	Education	72700.0	34900.0	52000.0	29300.0	102000.0	HASS
47	Religion	66700.0	34100.0	52000.0	29700.0	96400.0	HASS

In [63]:
```
# Groping the data according to the groups in the college

df.groupby("Group").count()
```

Out[63]:

Group	Undergraduate Major	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary
Business	12	12	12	12	12	12
HASS	22	22	22	22	22	22
STEM	16	16	16	16	16	16

In [64]:
```
# Finding the mean of the data grouped

df.groupby("Group").mean()
```

Out[64]:

Group	After Subtract	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 90th Percentile Salary
Business	103958.333333	44633.333333	75083.333333	43566.666667	147525.000000
HASS	95218.181818	37186.363636	62968.181818	34145.454545	129363.636364
STEM	101600.000000	53862.500000	90812.500000	56025.000000	157625.000000

DAY-72 and DAY-73 --> Basic Matplotlib Plots

DAY-74 -->Matplotlib Plots

DAY-75 --> Learning about Matplotlib Plots Dataset

DAY-76

In [2]:
```
# Learning about Numpy data
```

DAY-77 DAY-78 DAY-79 DAY-80 --> Practice

DAY-81 -- DAY-100 TASKS

In [ ]: