
52 Amazing
Python Projects
For
Developers
EDCORNER LEARNING

52 Amazing Python Projects For Developers

52 Amazing Python Projects For Developers

Edcorner Learning

Table of Contents

Introduction

Python is a general-purpose interpreted, interactive, object- oriented, and a powerful programming language with dynamic semantics. It is an easy language to learn and become expert. Python is one among those rare languages that would claim to be both easy and powerful. Python's elegant syntax and dynamic typing alongside its interpreted nature makes it an ideal language for scripting and robust application development in many areas on giant platforms.

Python helps with the modules and packages, which inspires program modularity and code reuse. The Python interpreter and thus the extensive standard library are all available in source or binary form for free of charge for all critical platforms and can be freely distributed. Learning Python doesn't require any pre- requisites. However, one should have the elemental understanding of programming languages.

This Book consist of 52 Python Projects for All Developers/Students to practice different projects and scenarios. Use these learnings in professional tasks or daily learning projects.

At the end of this book, you can download all this projects by using our link.

All 52 projects are divided into different modules, every project is special in its own way of performing daily task by a developer. Every project has its source codes which learners can copy and practice/use on their own systems. If there is special requirement for any projects, its already mentioned in the book.

Happy learning!!

Module 1 Project 1 -10

1. LinkedIn Email Scraper

Prerequisites:

1. Do `pip install -r requirements.txt` to make sure you have the necessary libraries.

2. Make sure you have a **chromedriver** installed and added to PATH.

3. Have the **URL** to your desired LinkedIn post ready (*make sure the post has some emails in the comments section*)

4. Have your **LinkedIn** account credentials ready

Executing Application

1. Replace the values of the URL, email and password variables in the code with your own data

2. Either hit **run** if your IDE has the option or just type in `python main.py` in the terminal.

3. The names and corresponding email address scraped from the post should appear in the **emails.csv** file.

Requirements:

selenium

email-validator

Source Code:

```
from selenium import webdriver
from email_validator import validate_email, EmailNotValidError
import csv


def LinkedInEmailScraper(userEmail, userPassword):
    emailList = {}
    browser = webdriver.Chrome()
    # example => 'https://www.linkedin.com/posts/faangpath_hiring-womxn-ghc2020-activity-6721287139721650176-QFCV/'
    url = '[INSERT URL TO LINKEDIN POST]'
    browser.get(url)  # visits page of the desired post

    browser.implicitly_wait(5)

    commentDiv = browser.find_element_by_xpath(
        '/html/body/main/section[1]/section[1]/div/div[3]/a[2]'
    )  # finds comment button
    loginLink = commentDiv.get_attribute('href')
    browser.get(loginLink)
```

```
email = browser.find_element_by_xpath('//*[@id="username"]')
password = browser.find_element_by_xpath('//*[@id="password"]')
email.send_keys(userEmail)  # inputs email in email field
password.send_keys(userPassword)  # inputs password in password field
submit = browser.find_element_by_xpath(
        '//*[@id="app__container"]/main/div[3]/form/div[3]/button')
submit.submit()  # submits form

browser.implicitly_wait(5)

commentSection = browser.find_element_by_css_selector(
        '.comments-comments-list')  # finds the comments section

for _ in range(
                3
        ):  # this can also be set to any number or "while True" if you want it to search through the whole comment section of the post
                try:
                        moreCommentsButton = commentSection.find_element_by_class_name(
                                'comments-comments-list__show-previous-container'
                        ).find_element_by_tag_name('button')
                        moreCommentsButton.click()
                        browser.implicitly_wait(5)
                except:
                        print('End of checking comments')
                        break

        browser.implicitly_wait(20)

        comments = commentSection.find_elements_by_tag_name(
                'article')  # finds all individual comments

        for comment in comments:
                try:
                        commenterName = comment.find_element_by_class_name(
                                'hoverable-link-text')  # finds name of commenter
                        commentText = comment.find_element_by_tag_name('p')
                        commenterEmail = commentText.find_element_by_tag_name(
                                'a').get_attribute('innerHTML')  # finds email of commenter
                        # validates email address
                        validEmail = validate_email(commenterEmail)
                        commenterEmail = validEmail.email
                except:
                        continue

                emailList[commenterName.get_attribute('innerHTML')] = commenterEmail
```

```
        browser.quit()
        return emailList


def DictToCSV(input_dict):
        '''
        Converts dictionary into csv
        '''
        with open('./LinkedIn Email Scraper/emails.csv', 'w') as f:
                f.write('name,email\n')
                for key in input_dict:
                        f.write('%s,%s\n' % (key, input_dict[key]))
                f.close()


if __name__ == '__main__':
        userEmail = '[INSERT YOUR EMAIL ADDRESS FOR LINKEDIN ACCOUNT]'
        userPassword = '[INSERT YOUR PASSWORD FOR LINKEDIN ACCOUNT'

        emailList = LinkedInEmailScraper(userEmail, userPassword)
        DictToCSV(emailList)
```

2. Cricbuzz scrapper

This python script will scrap cricbuzz.com to get live scores of the matches.

Setup

* Install the dependencies

    `pip install -r requirements.txt`

* Run the file

    `python live_score.py`

Requirement:

beautifulsoup4==4.9.3

bs4==0.0.1

pypiwin32==223

pywin32==228

soupsieve==2.0.1

urllib3==1.26.5

win10toast==0.9

Source Code:

```
from urllib.request import urlopen, Request
from bs4 import BeautifulSoup
from win10toast import ToastNotifier
import time

URL = 'http://www.cricbuzz.com/cricket-match/live-scores'


def notify(title, score):
    # Function for Windows toast desktop notification
    toaster = ToastNotifier()
    # toaster.show_toast(score, "Get! Set! GO!", duration=5,icon_path='cricket.ico')
    toaster.show_toast("CRICKET LIVE SCORE",
                       score,
                       duration=30,
                       icon_path='ipl.ico')


while True:
    request = Request(URL, headers={'User-Agent': 'XYZ/3.0'})
    response = urlopen(request, timeout=20).read()
```

```
data_content = response
# print(data_content)

# page = urlopen(URL)
soup = BeautifulSoup(data_content, 'html.parser')

update = []
# print(soup)
# print(soup.find_all('div',attrs={'class':'cb-col cb-col-100 cb-plyr-tbody cb-rank-hdr cb-lv-main'}))
for score in soup.find_all(
                'div',
                attrs={
                        'class':
                        'cb-col cb-col-100 cb-plyr-tbody cb-rank-hdr cb-lv-main'
                }):
# print(score)
header = score.find('div',
                                attrs={'class': 'cb-col-100 cb-col cb-schdl'})
header = header.text.strip()

status = score.find('div',
                                attrs={'class': 'cb-scr-wll-chvrn cb-lv-scrs-col'})
s = status.text.strip()

notify(header, s)
time.sleep(10)
```

3. Lyrics Download

This script can be used to download lyrics of any number of songs, by any number of Artists, until the API Limit is met.

The script uses [Genius API] (https://docs.genius.com/). It is a dedicated platform meant for music only.

Setup Instruction

- You need an API client, (it's free) follow the steps [here](https://docs.genius.com/).
- `pip install lyricsgenius` to install dedicated package.
- Good to go, follow guidelines mentioned as comments in code.
- The script is pretty much interactive, ensure you follow the guidelines.

Source Code:

```
import lyricsgenius as lg

# File for writing the Lyrics
filename = input('Enter a filename: ') or 'Lyrics.txt'
file = open(filename, "w+")

# Acquire a Access Token to connect with Genius API
genius = lg.Genius(
    'Client_Access_Token_Goes_Here',
    # Skip song listing
    skip_non_songs=True,
    # Terms that are redundant song names with same lyrics, e.g. Old Town Raod and Old Town Road Remix
    # have same lyrics
    excluded_terms=["(Remix)", "(Live)"],
    # In order to keep headers like [Chorus], [Bridge] etc.
    remove_section_headers=True)

# List of Artist and Maximum Songs
input_string = input("Enter name of Artists separated by spaces: ")
artists = input_string.split(" ")
```

```
def get_lyrics(arr, max_song):
    """
    Returns: Number of songs grabbed by Function
    Saves : Text File with Lyrics
    Parameters :
        arr : Artist
        max_song : Number of maximum songs to be grabbed
    """
    # Write lyrics of k songs by each artist in arr
    c = 0
    # A counter
    for name in arr:
        try:
            songs = (genius.search_artist(name,
                                          max_songs=max_song,
                                          sort='popularity')).songs
            s = [song.lyrics for song in songs]
            # A custom delimiter
            file.write("\n \n   <|endoftext|>   \n \n".join(s))
            c += 1
            print(f"Songs grabbed:{len(s)}")
        except:
            print(f"some exception at {name}: {c}")


# Function Call
get_lyrics(artists, 3)
```

4. Merge CSV files

With the help of the following simple python script, one would be able to merge CSV files present in the directory.

Dependencies
Requires Python 3 and `pandas`

Install requirements: `pip install -r "requirements.txt"`
OR
Install pandas: `pip install pandas`

How to use
Running
Put all the CSVs which are to be merged in a directory containing the script.
either run it from your code editor or IDE or type `python merge_csv_files.py` in your command line.

The final output would be a `combined_csv.csv` file in the same directory.

Requirements:

pandas==1.1.0

Source Code:

```
import glob
import pandas as pd

extension = 'csv'
all_filenames = [i for i in glob.glob('*.{}'.format(extension))]
combined_csv = pd.concat([pd.read_csv(f) for f in all_filenames ])
combined_csv.to_csv( "combined_csv.csv", index=False, encoding='utf-8-sig')
```

5. Merge pdfs

A simple python script which when executed merges two pdfs

Prerequisites

Run - "pip install PyPDF2"

How to run the script

It can be executed by running "python merge_pdfs.py"

Requirements:

PyPDF2==1.26.0

Source Code:

```
#!/usr/bin/env python

from PyPDF2 import PdfFileMerger


# By appending in the end
def by_appending():
    merger = PdfFileMerger()
    # Either provide file stream
    f1 = open("samplePdf1.pdf", "rb")
    merger.append(f1)
    # Or direct file path
    merger.append("samplePdf2.pdf")

    merger.write("mergedPdf.pdf")


# By inserting at after an specified page no.
def by_inserting():
    merger = PdfFileMerger()
    merger.append("samplePdf1.pdf")
    merger.merge(0, "samplePdf2.pdf")
    merger.write("mergedPdf1.pdf")
```

```
if __name__ == "__main__":
    by_appending()
    by_inserting()
```

6. Message Spam Detection

Short description of package/script

- Libraries Used:
    - pandas
    - string
    - re
    - nltk
    - sklearn
    - pickle
- The python code contains script for message spam detection based on Kaggle Dataset (dataset link inside the code)

Setup instructions

- Download the code
- Download the dataset
- Run the cells in the notebook

Detailed explanation of script, if needed

NA

Output

- Hello, I am James Bond: Not Spam
- Winner! Winner! Winner! Congrats! Call at xyz or email us at to claim your prize! Limited Time Offer!: Spam

Message Spam Detection Source Code:

```
# importing required libraries
import pandas as pd
import string
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.ensemble import RandomForestClassifier
import pickle
import warnings
import re
warnings.filterwarnings("ignore")

# reading the dataset
msg = pd.read_csv(
    "./Message_Spam_Detection/Cleaned_Dataset.csv", encoding='latin-1')
```

```
msg.drop(['Unnamed: 0'], axis=1, inplace=True)

        # seperating target and features
        y = pd.DataFrame(msg.label)
        x = msg.drop(['label'], axis=1)

        # countvectorization
        cv = CountVectorizer(max_features=5000)
        temp1 = cv.fit_transform(x['final_text'].values.astype('U')).toarray()
        tf = TfidfTransformer()
        temp1 = tf.fit_transform(temp1)
        temp1 = pd.DataFrame(temp1.toarray(), index=x.index)
        x = pd.concat([x, temp1], axis=1, sort=False)

        # drop final_text col
        x.drop(['final_text'], axis=1, inplace=True)

        # converting to int datatype
        y = y.astype(int)

        # randomforstclassifier model
        model = RandomForestClassifier(n_estimators=100, random_state=0)
        model.fit(x, y)

# User input
text = input("Enter text: ")

# data cleaning/preprocessing - removing punctuation and digits
updated_text = ''
for i in range(len(text)):
    if text[i] not in string.punctuation:
        if text[i].isdigit() == False:
                updated_text = updated_text+text[i]
text = updated_text

# data clearning/preprocessing - tokenization and convert to lower case
text = re.split("\W+", text.lower())

# data cleaning/preprocessing - stopwords
updated_list = []
stopwords = nltk.corpus.stopwords.words('english')
for i in range(len(text)):
    if text[i] not in stopwords:
        updated_list.append(text[i])
text = updated_list

            # data cleaning/preprocessing - lemmentizing
```

```
updated_list = []
wordlem = nltk.WordNetLemmatizer()
for i in range(len(text)):
        updated_list.append(wordlem.lemmatize(text[i]))
text = updated_list

# data cleaning/preprocessing - mergining token
text = " ".join(text)

text = cv.transform([text])
text = tf.transform(text)
pred = model.predict(text)
if pred == 0:
        print("Not Spam")
else:
        print("Spam")
```

Data Cleaning Source Code:

```
# importing required libraries
import pandas as pd
import string
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.ensemble import RandomForestClassifier
import pickle
import warnings
import re
warnings.filterwarnings("ignore")
nltk.download('stopwords')
nltk.download('wordnet')

# reading the dataset
# dataset: https://www.kaggle.com/uciml/sms-spam-collection-dataset
msg = pd.read_csv("./Message_Spam_Detection/dataset.csv", encoding='latin-1')
msg.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
msg.rename(columns={"v1": "label", "v2": "text"}, inplace=True)

        # mapping ham=0 and spam=1
        for i in msg.index:
        if msg['label'][i] == 'ham':
```

```
msg['label'][i] = 0
else:
msg['label'][i] = 1

# dropping duplicate columns
msg = msg.drop_duplicates()

# data cleaning/preprocessing - removing punctuation and digits
msg['cleaned_text'] = ""
for i in msg.index:
updated_list = []
for j in range(len(msg['text'][i])):
if msg['text'][i][j] not in string.punctuation:
if msg['text'][i][j].isdigit() == False:
            updated_list.append(msg['text'][i][j])
            updated_string = "".join(updated_list)
            msg['cleaned_text'][i] = updated_string
            msg.drop(['text'], axis=1, inplace=True)

# data clearning/preprocessing - tokenization and convert to lower case
msg['token'] = ""
for i in msg.index:
    msg['token'][i] = re.split("\W+", msg['cleaned_text'][i].lower())

# data cleaning/preprocessing - stopwords
msg['updated_token'] = ""
stopwords = nltk.corpus.stopwords.words('english')
for i in msg.index:
    updated_list = []
    for j in range(len(msg['token'][i])):
        if msg['token'][i][j] not in stopwords:
                    updated_list.append(msg['token'][i][j])
    msg['updated_token'][i] = updated_list
msg.drop(['token'], axis=1, inplace=True)

# data cleaning/preprocessing - lemmentizing
msg['lem_text'] = ""
wordlem = nltk.WordNetLemmatizer()
for i in msg.index:
    updated_list = []
    for j in range(len(msg['updated_token'][i])):
                    updated_list.append(wordlem.lemmatize(msg['updated_token'][i][j]))
                msg['lem_text'][i] = updated_list
            msg.drop(['updated_token'], axis=1, inplace=True)
```

```
# data cleaning/preprocessing - mergining token
msg['final_text'] = ""
for i in msg.index:
        updated_string = " ".join(msg['lem_text'][i])
        msg['final_text'][i] = updated_string
msg.drop(['cleaned_text', 'lem_text'], axis=1, inplace=True)

# cleaned dataset
msg.to_csv('Cleaned_Dataset.csv')
```

7. Movie Information Scraper

This script obtains movie details by scraping IMDB website.

Prerequisites

* beautifulsoup4

* requests

* Run `pip install -r requirements.txt` to install required external modules.

How to run the script

Execute `python3 movieInfoScraper.py` and type in the movie name when prompted.

Requirements:

beautifulsoup4

requests==2.23.0

Source Code:

```
import os
import zipfile
import sys
import argparse

# Code to add the cli
parser = argparse.ArgumentParser()
parser.add_argument("-l", "--zippedfile", required=True, help="Zipped file")
args = vars(parser.parse_args())

#Catching the user defined zip file
zip_file = args['zippedfile']

file_name = zip_file

#To check if the entered zip file is present in the directory
if os.path.exists(zip_file) == False:
    sys.exit("No such file present in the directory")

        #Function to extract the zip file
        def extract(zip_file):
            file_name = zip_file.split(".zip")[0]
```

```
    if zip_file.endswith(".zip"):

        #Will use this to save the unzipped file in the current directory
        current_working_directory = os.getcwd()
        new_directory = current_working_directory + "/" + file_name
        #Logic to unzip the file
        with zipfile.ZipFile(zip_file, 'r') as zip_object:
            zip_object.extractall(new_directory)
        print("Extracted successfully!!!")
    else:
        print("Not a zip file")

extract(zip_file)
```

8. Movie Info Telegram Bot

Description

A telegram Bot made using python which scrapes IMDb website and has the following functionalities

1. Replies to a movie name with genre and rating of the movie
2. Replies to a genre with a list of top movies and tv shows belonging to that genre

Setup Instructions

1. Install required packages:

    pip install -r requirements.txt

2. Create a bot in telegram:

    1. Go to @BotFather and click /start and type /newbot and give it a name.
    2. Choose a username and get the token

3. Paste the token in a .env file (Take .env.example as an example)

4. Run the python script to start the bot

5. Type /start command to start conversation with the chatbot.

6. Type /name <movie_name> to get the genre and Rating of the movie. The bot replies with atmost three results.
7. Type /genre \<genre> to get a list of movies and TV shows belonging to that genres

Requirements:

APScheduler==3.6.3

beautifulsoup4==4.9.3

certifi==2020.12.5

python-dateutil==2.8.1

python-decouple==3.4

python-telegram-bot==13.1

pytz==2020.4

requests==2.25.1

six==1.15.0

soupsieve==2.1

tornado==6.1

urllib3==1.26.2

Source Code:

```
import logging
```

```
import requests
import re
import urllib.request
import urllib.parse
import urllib.error
from bs4 import BeautifulSoup
import ssl
import itertools
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters
import decouple

# Enable logging
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
                                level=logging.INFO)

logger = logging.getLogger(__name__)

TOKEN = decouple.config("API_KEY")

# Define a few command handlers. These usually take the two arguments update and
# context. Error handlers also receive the raised TelegramError object in error.


def start(update, context):
    """Send a message when the command /start is issued."""
    update.message.reply_text(
        'What can this bot do?\n\nThis bot gives brief information about any movie from IMDb website'
        + '\nSend /name movie_name to know the genre and rating of the movie.\nSend /genre genre_name to'
        + 'get the list of movies belonging to that genre'
    )


def help(update, context):
    """Send a message when the command /help is issued."""
    update.message.reply_text('Help!')


def genre(update, context):
    """Send a list of movies when the command /genre is issued."""
    url = 'https://www.imdb.com/search/title/'
    genre = str(update.message.text)[7:]
    print(genre)
    r = requests.get(url+'?genres='+genre)
    soup = BeautifulSoup(r.text, "html.parser")
    title = soup.find('title')
    if title.string == 'IMDb: Advanced Title Search - IMDb':
```

```
        update.message.reply_text("Sorry,No such genre.Try again")
    else:
        res = []
        res.append(title.string+'\n')
        tags = soup('a')
        for tag in tags:
                movie = re.search('<a href=\"/title/.*>(.*?)</a>', str(tag))
                try:
                        if "&amp;" in movie.group(1):
                                movie.group(1).replace("&amp;", "&")
                        res.append(movie.group(1))
                except:
                        pass
                stri = ""
                for i in res:
                        stri += i+'\n'
                update.message.reply_text(stri)


def name(update, context):
    """Send the first 3 search results of the movie name in IMDb site when the command /name is issued."""
    movie = str(update.message.text)[6:]
    print(movie)
    res = get_info(movie)
    stri = ""
    for i in res:
        for a in i:
                stri += a+'\n'
        stri += '\n'
    update.message.reply_text(stri)


def error(update, context):
    """Log Errors caused by Updates."""
    logger.warning('Update "%s" caused error "%s"', update, context.error)


def get_info(movie):
    "To scrape IMDb and get genre and rating of the movie "
    url = 'https://www.imdb.com/find?q='
    r = requests.get(url+movie+'&ref_=nv_sr_sm')
    soup = BeautifulSoup(r.text, "html.parser")
    title = soup.find('title')
    tags = soup('a')
```

```
pre_url = ""
count = 0
lis = []
res = []
for tag in tags:
    if(count > 2):
            break
    m = re.search('<a href=.*>(.*?)</a>', str(tag))
    try:
            lis = []
            link = re.search('/title/(.*?)/', str(m))
            new_url = 'https://www.imdb.com'+str(link.group(0))
            if new_url != pre_url:
html = requests.get(new_url)
soup2 = BeautifulSoup(html.text, "html.parser")
movietitle = soup2.find('title').string.replace('- IMDb', ' ')
a = soup2('a')
span = soup2('director')
for item in span:
    print(item)
genrestring = "Genre : "
for j in a:
    genre = re.search(
        '<a href=\"/search/title\?genres=.*> (.*?)</a>', str(j))
    try:
        genrestring += genre.group(1)+' '
    except:
        pass
atag = soup2('strong')
for i in atag:
    rating = re.search('<strong title=\"(.*?) based', str(i))
    try:
        rstring = "IMDb Rating : "+rating.group(1)
    except:
        pass
details = "For more details : "+new_url
lis.append(movietitle)
lis.append(genrestring)
lis.append(rstring)
lis.append(details)
pre_url = new_url
count += 1
```

```
                        res.append(lis)
            except:
                    pass
    return res


def main():
    """Start the bot."""
    # Create the Updater and pass it your bot's token.
    # Make sure to set use_context=True to use the new context based callbacks
    updater = Updater(TOKEN, use_context=True)

    # Get the dispatcher to register handlers
    dp = updater.dispatcher
    # on different commands - reply in Telegram
    dp.add_handler(CommandHandler("start", start))
    dp.add_handler(CommandHandler("help", help))
    dp.add_handler(CommandHandler("name", name))
    dp.add_handler(CommandHandler("genre", genre))

    # log all errors
    dp.add_error_handler(error)

    # Start the Bot
    updater.start_polling()

    # Run the bot until you press Ctrl-C or the process receives SIGINT,
    # SIGTERM or SIGABRT. This should be used most of the time, since
    # start_polling() is non-blocking and will stop the bot gracefully.
    updater.idle()


if __name__ == '__main__':
    main()
```

9. snapshot of given website

```
# snapshot of given website

## Set up
`pip install selenium`
`pip install chromedriver-binary==XX.X.XXXX.XX.X`

- 'XX.X.XXXX.XX.X' is chrome driver version.
- The version of 'chrome driver' need to match the version of your google chrome.
```

How to find your google chrome version

1. Click on the Menu icon in the upper right corner of the screen.
2. Click on Help, and then About Google Chrome.
3. Your Chrome browser version number can be found here.

Execute

`python snapshot_of_given_website.py <url>`

Snapshot is in current directory after this script runs.

Requirements:

selenium==3.141.0

chromedriver-binary==85.0.4183.38.0

Source Code:

```
# -*- cofing: utf-8 -*-
import sys
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import chromedriver_binary


script_name = sys.argv[0]

options = Options()
options.add_argument('--headless')
driver = webdriver.Chrome(options=options)

try:
    url = sys.argv[1]

    driver.get(url)
    page_width = driver.execute_script('return document.body.scrollWidth')
    page_height = driver.execute_script('return document.body.scrollHeight')
    driver.set_window_size(page_width, page_height)
    driver.save_screenshot('screenshot.png')
    driver.quit()
    print("SUCCESS")

except IndexError:
    print('Usage: %s URL' % script_name)
```

10. Music Player with Python

```
import pygame
import tkinter as tkr
from tkinter.filedialog import askdirectory
import os

music_player = tkr.Tk()
music_player.title("My Music Player")
music_player.geometry("450x350")
directory = askdirectory()
os.chdir(directory)
song_list = os.listdir()
play_list = tkr.Listbox(music_player, font="Helvetica 12 bold", bg='yellow', selectmode=tkr.SINGLE)
for item in song_list:
    pos = 0
    play_list.insert(pos, item)
    pos += 1
pygame.init()
pygame.mixer.init()

def play():
    pygame.mixer.music.load(play_list.get(tkr.ACTIVE))
    var.set(play_list.get(tkr.ACTIVE))
    pygame.mixer.music.play()
def stop():
    pygame.mixer.music.stop()
def pause():
    pygame.mixer.music.pause()
def unpause():
    pygame.mixer.music.unpause()
Button1 = tkr.Button(music_player, width=5, height=3, font="Helvetica 12 bold", text="PLAY", command=play,
bg="blue", fg="white")
Button2 = tkr.Button(music_player, width=5, height=3, font="Helvetica 12 bold", text="STOP", command=stop,
bg="red", fg="white")
Button3 = tkr.Button(music_player, width=5, height=3, font="Helvetica 12 bold", text="PAUSE",
command=pause, bg="purple", fg="white")
Button4 = tkr.Button(music_player, width=5, height=3, font="Helvetica 12 bold", text="UNPAUSE",
command=unpause, bg="orange", fg="white")

var = tkr.StringVar()
song_title = tkr.Label(music_player, font="Helvetica 12 bold", textvariable=var)

song_title.pack()
```

```
Button1.pack(fill="x")
Button2.pack(fill="x")
Button3.pack(fill="x")
Button4.pack(fill="x")
play_list.pack(fill="both", expand="yes")
music_player.mainloop()
```

Module 2 Project 11-20

11. News Updater with Voice

Get News API key From HERE:
[NewsAPI](https://newsapi.org/)

Add Your API Key:

```
newsapi = NewsApiClient(api_key='Add it Here')
```

Dependencies:

Newsapi
```python
pip install newsapi
```
pyttsx3
```python
pip install pyttsx3
```
pyaudio
```python
pip install pyaudio
```

Source Code:

```
from newsapi import NewsApiClient
import pyttsx3
import speech_recognition as sr
```

```
from time import sleep

engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice',voices[1].id)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()


def new():
    newsapi = NewsApiClient(api_key='')# Add your api key
    data = newsapi.get_top_headlines(q='corona',country='in',
                                                    language='en',
                                                    page_size=5)

at = data['articles']

for x,y in enumerate(at):
    print(f'{x} {y["description"]}')
    speak(f'{x} {y["description"]}')

speak("that's it for now i'll updating you in some time ")



    if __name__ == "__main__":
            while True:
            new()
            sleep(600)
```

12. News Scraper

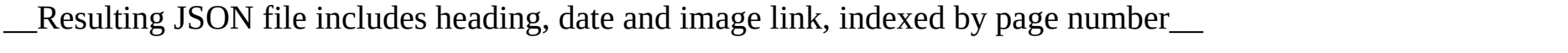
Financial-news-scraper

A scraper made using beautiful soup 4 in python. Tailor made for extracting news from moneycontrol.com. Issue pull request for different scrapers.

__The main page to start scraping from: https://www.moneycontrol.com/news/technical-call-221.html__

__The program scrapes news from next pages too by extracting website link in these buttons__

__Resulting JSON file includes heading, date and image link, indexed by page number__



Source code:

```
import re
import json
import requests
import datetime
from tqdm import tqdm
from bs4 import BeautifulSoup
from collections import defaultdict

submission = defaultdict(list)
#main url
src_url = 'https://www.moneycontrol.com/news/technical-call-221.html'

#get next page links and call scrap() on each link
def setup(url):
    nextlinks = []
    src_page = requests.get(url).text
    src = BeautifulSoup(src_page, 'lxml')

    #ignore <a> with void js as href
    anchors = src.find("div", attrs={"class": "pagenation"}).findAll(
        'a', {'href': re.compile('^((?!void).)*$')})
    nextlinks = [i.attrs['href'] for i in anchors]
    for idx, link in enumerate(tqdm(nextlinks)):
```

```
            scrap('https://www.moneycontrol.com'+link, idx)

    #scraps passed page url
    def scrap(url, idx):
        src_page = requests.get(url).text
        src = BeautifulSoup(src_page, 'lxml')

        span = src.find("ul", {"id": "cagetory"}).findAll('span')
        img = src.find("ul", {"id": "cagetory"}).findAll('img')

        #<img> has alt text attr set as heading of news, therefore get img link and heading from same tag
        imgs = [i.attrs['src'] for i in img]
        titles = [i.attrs['alt'] for i in img]
        date = [i.get_text() for i in span]

    #list of dicts as values and indexed by page number
    submission[str(idx)].append({'title': titles})
    submission[str(idx)].append({'date': date})
    submission[str(idx)].append({'img_src': imgs})

            #save data as json named by current date
            def json_dump(data):
                date = datetime.date.today().strftime("%B %d, %Y")
                with open('moneycontrol_'+str(date)+'.json', 'w') as outfile:
                    json.dump(submission, outfile)

            setup(src_url)
            json_dump(submission)
```

13. Noise Reduction

Noise Reduction Script

Implementing a feature that helps to filter an audio file by reducing the background noise similar to "Audacity".

Libraries used

Firstly import the following python libraries

* NumPy
* scipy.io.wavfile
* Matplotlib
* Os

Save the audio files and your code in the same folder

Run the python code

Detailed explanation of method used for "Noise Reduction Script"

* Imported the required libraries (NumPy, scipy.io.wavfile, and Matplotlib)
* Read the input audio file using scipy.io.wavfile library
* Converting the audio file into an array containg all the information of the given audio file and intiallizing the frame value.
* Calculating the first fourier transform of each window of the noisy audio file
* Subtracting the noise spectral mean from input spectral, and istft (Inverse Short-Time Fourier Transform)
* Finally getting an audio file with reduction in the background noise at a much higher extent

Requirements:

matplotlib==3.2.2

numpy==1.18.5

scipy==1.5.0

Source Code:

```
# Spectral Subtraction: Method used for noise reduction
import scipy.io.wavfile as wav
import numpy as np
import matplotlib.pyplot as plt

file = input("Enter the file path: ")
sr, data = wav.read(file)
fl = 400 #frame_length
frames = []  #empty list
for i in range(0,int(len(data)/(int(fl/2))-1)):
                arr = data[int(i*int(fl/2)):int(i*int(fl/2)+fl)]
                frames.append(arr)    #appending each array data into the frames list
```

```
frames = np.array(frames)   #converting the frames list into an array
ham_window = np.hamming(fl)    #using np.hamming
windowed_frames = frames*ham_window    #multiplying frames array with ham_window
dft = []   #empty list containing fft of windowed_frames
for i in windowed_frames:
        dft.append(np.fft.fft(i))   #now taking the first fourier transform of each window
dft = np.array(dft)  #converting dft into array

dft_mag_spec = np.abs(dft)      #converting dft into absolute values
dft_phase_spec = np.angle(dft)   #finding dft angle
noise_estimate = np.mean(dft_mag_spec,axis=0)    #mean
noise_estimate_mag = np.abs(noise_estimate)   #absolute value

estimate_mag = (dft_mag_spec-2*noise_estimate_mag)   #subtraction method
            estimate_mag[estimate_mag<0]=0
            estimate = estimate_mag*np.exp(1j*dft_phase_spec)  #calculating the final estimate
            ift = []   #taking ift as input list containing inverse fourier transform of estimate
            for i in estimate:
                  ift.append(np.fft.ifft(i))    #appending in ift list

clean_data = []
clean_data.extend(ift[0][:int(fl/2)])     #extending clean_data containg ift list
for i in range(len(ift)-1):
     clean_data.extend(ift[i][int(fl/2):]+ift[i+1][:int(fl/2)])
clean_data.extend(ift[-1][int(fl/2):])   #extending clean_data containing ift list
clean_data = np.array(clean_data)  #converting it into array

#finally plotting the graph showing the diffrence in the noise
fig = plt.figure(figsize=(8,5))
ax = plt.subplot(1,1,1)
ax.plot(np.linspace(0,64000,64000),data,label='Original',color="orange")
ax.plot(np.linspace(0,64000,64000),clean_data,label='Filtered',color="purple")
ax.legend(fontsize=12)
ax.set_title('Spectral Subtraction Method', fontsize=15)
filename = os.path.basename(file)
cleaned_file = "(Filtered_Audio)"+filename   #final filtered audio
wav.write(cleaned_file,rate=sr, data = clean_data.astype(np.int16))
plt.savefig(filename+"(Spectral Subtraction graph).jpg")  #saved file name as audio.wav(Spectral Subtraction
graph).jpg
```

14. NSE Stock Data

Running this Script would allow the user to go through NSE Stock data scraped from [NSE Website] (https://www.nseindia.com), based on their choice of available categories.

Setup instructions

In order to run this script, you need to have Python and pip installed on your system. After you're done installing Python and pip, run the following command from your terminal to install the requirements from the same folder (directory) of the project.

```

pip install -r requirements.txt
```

As this script uses selenium, you will need to install the chrome webdriver from [this link] (https://sites.google.com/a/chromium.org/chromedriver/downloads)

After satisfying all the requirements for the project, Open the terminal in the project folder and run

```
    python stocks.py
    ```

    or

    ```
    python3 stocks.py
    ```

    depending upon the python version. Make sure that you are running the command from the same virtual environment in which the required modules are installed.

    Requirements:

    requests

    beautifulsoup4

    selenium

Source Code:

```
import requests
from bs4 import BeautifulSoup
import tkinter as tk
```

```
from tkinter import ttk
from tkinter import font as tkFont
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time

driver_path = input('Enter path for chromedriver: ')

# Categories and their URL slugs
most_active = {'Most Active equities - Main Board':'mae_mainboard_tableC','Most Active equities - SME':'mae_sme_tableC','Most Active equities - ETFs':'mae_etf_tableC',
                        'Most Active equities - Price Spurts':'mae_pricespurts_tableC', 'Most Active equities - Volume Spurts':'mae_volumespurts_tableC'}
top_20 = {'NIFTY 50 Top 20 Gainers':'topgainer-Table','NIFTY 50 Top 20 Losers':'toplosers-Table'}

# Function to generate request url based on user choice
def generate_url():
    category_choice = category.get()
    if(category_choice in most_active):
        page = 'most-active-equities'
    else:
        page = 'top-gainers-loosers'
    url = 'https://www.nseindia.com/market-data/{}'.format(page)
    return url

# Function to scrape stock data from generated URL
def scraper():
    url = generate_url()
    driver = webdriver.Chrome(driver_path)
    driver.get(url)

    # Wait for results to load
    time.sleep(5)
    html = driver.page_source

    # Start scraping resultant html data
    soup = BeautifulSoup(html, 'html.parser')

    # Based on choice scrape div
    category_choice = category.get()
    if category_choice in most_active :
        category_div = most_active[category_choice]
    else :
        category_div = top_20[category_choice]

    # Find the table to scrape
    results = soup.find("table", {"id": category_div})
```

```
rows = results.findChildren('tr')

table_data = []
row_values = []
# Append stock data into a list
for row in rows:
    cells = row.findChildren(['th', 'td'])
    for cell in cells:
            value = cell.text.strip()
            value = " ".join(value.split())
            row_values.append(value)
    table_data.append(row_values)
    row_values = []

# Formatting the stock data stored in the list
stocks_data = ""
for stock in table_data:
    single_record = ""
    for cell in stock:
            format_cell = "{:<20}"
            single_record += format_cell.format(cell[:20])
    single_record += "\n"
    stocks_data += single_record

 # Adding the formatted data into tkinter GUI
query_label.config(state=tk.NORMAL)
query_label.delete(1.0,"end")
query_label.insert(1.0,stocks_data)
query_label.config(state=tk.DISABLED)
driver.close()

        # Creating tkinter window
        window = tk.Tk()
        window.title('NSE Stock data')
        window.geometry('1200x1000')
window.configure(bg='white')

style = ttk.Style()
style.configure('my.TButton', font=('Helvetica', 16))
style.configure('my.TFrame', background='white')

# label text for title
ttk.Label(window, text="NSE Stock market data",
        background='white', foreground="SpringGreen2",
        font=("Helvetica", 30, 'bold')).grid(row=0, column=1)
```

```
# label
ttk.Label(window, text="Select Market data to get:", background = 'white',
          font=("Helvetica", 15)).grid(column=0,
                                       row=5, padx=10, pady=25)

# Combobox creation
category = ttk.Combobox(
    window, width=60, state='readonly',font="Helvetica 15")

submit_btn = ttk.Button(window, text="Get Stock Data!", style='my.TButton', command = scraper)

# Adding combobox drop down list
category['values'] = ('Most Active equities - Main Board','Most Active equities - SME','Most Active equities - ETFs','Most Active equities - Price Spurts',
                      'Most Active equities - Volume Spurts','NIFTY 50 Top 20 Gainers','NIFTY 50 Top 20 Losers')

category.grid(column=1, row=5, padx=10)
category.current(0)

submit_btn.grid(row=5, column=3, pady=5, padx=15, ipadx=5)

frame = ttk.Frame(window, style='my.TFrame')
frame.place(relx=0.50, rely=0.12, relwidth=0.98, relheight=0.90, anchor="n")

# To display stock data
query_label = tk.Text(frame ,height="52" ,width="500", bg="alice blue")
query_label.grid(row=7,  columnspan=2)

window.mainloop()
```

15. Number Guessing Game

This game allows you to check your luck and intuition :)

You should find the number computer guessed

Usage

Just run "python main.py" in cmd command line after setting the project directory

Here you can see sample

Source Code:

```
  import random

  print("Number guessing game")

  # randint function to generate the
  # random number b/w 1 to 9
  number = random.randint(1, 9)

  # number of chances to be given
  # to the user to guess the number
      # or it is the inputs given by user
      # into input box here number of
      # chances are 5
      chances = 0

      print("Guess a number (between 1 and 9):")

      # While loop to count the number
      # of chances
      while True:

# Enter a number between 1 to 9
guess = int(input())

# Compare the user entered number
```

```
# with the number to be guessed
if guess == number:

    # if number entered by user
    # is same as the generated
    # number by randint function then
    # break from loop using loop
    # control statement "break"
    print(
            f'CONGRATULATIONS! YOU HAVE GUESSED THE \
            NUMBER {number} IN {chances} ATTEMPTS!')
    # Printing final statement using the f-strings method;
    break

# Check if the user entered
# number is smaller than
# the generated number
elif guess < number:
    print("Your guess was too low: Guess a number higher than", guess)

# The user entered number is
# greater than the generated
# number
else:
    print("Your guess was too high: Guess a number lower than", guess)

# Increase the value of chance by 1
chances += 1
```

16. Files-Sorter

This is python script that sort the files in Download directory to other directories depending on extension.

Source Code:

```
import os
import shutil
os.chdir("E:\downloads")
#print(os.getcwd())
#check number of files in  directory
files = os.listdir()

#list of extension (You can add more if you want)
extentions = {
    "images": [".jpg", ".png", ".jpeg", ".gif"],
    "videos": [".mp4", ".mkv"],
    "musics": [".mp3", ".wav"],
    "zip": [".zip", ".tgz", ".rar", ".tar"],
    "documents": [".pdf", ".docx", ".csv", ".xlsx", ".pptx", ".doc", ".ppt", ".xls"],
    "setup": [".msi", ".exe"],
    "programs": [".py", ".c", ".cpp", ".php", ".C", ".CPP"],
    "design": [".xd", ".psd"]


}

#sort to specific folder depend on extenstions
def sorting(file):
    keys = list(extentions.keys())
    for key in keys:
        for ext in extentions[key]:
                # print(ext)
                if file.endswith(ext):
                    return key


#iterat through each file
for file in files:
    dist = sorting(file)
```

```
if dist:
    try:
        shutil.move(file, "../download-sorting/" + dist)
    except:
        print(file + " is already exist")
else:
    try:
        shutil.move(file, "../download-sorting/others")
    except:
        print(file + " is already exist")
```

17. PageSpeed

This script generates the PageSpeed API results for a website.

Packages required:

- requests
- json

Instructions

To use the package, just check the test.py file.

Output

Depending upon the use of the script, it can generate a json file for the pagespeed results, and it also returns the regular response object.

.

PageSpeed.py Source Code:

```
import requests
import json
from responses import PageSpeedResponse


class PageSpeed(object):
    """
    Google PageSpeed analysis client

    Attributes:
        api_key (str): Optional API key for client account.
        endpoint (str): Endpoint for HTTP request
    """

    def __init__(self, api_key=None):
        self.api_key = api_key
        self.endpoint = 'https://www.googleapis.com/pagespeedonline/v5/runPagespeed'

    def analyse(self, url, strategy='desktop', category='performance'):
        """
        Run PageSpeed test

        Args:
            url (str): The URL to fetch and analyse.
```

```
            strategy (str, optional): The analysis strategy to use. Acceptable values: 'desktop', 'mobile'
            category (str, optional): A Lighthouse category to run; if none are given, only Performance category
will be run

        Returns:
            response: PageSpeed API results
        """
        strategy = strategy.lower()

        params = {
            'strategy': strategy,
            'url': url,
            'category': category,
        }

        if self.api_key:
            params['key'] = self.api_key

        # Sanity Check
        if strategy not in ('mobile', 'desktop'):
            raise ValueError('invalid strategy: {0}'.format(strategy))

        # Returns raw data
        raw = requests.get(self.endpoint, params=params)

        response = PageSpeedResponse(raw)

        return response

    def save(self, response, path='./'):
        json_data = response._json
        with open(path + "json_data.json", 'w+') as f:
            json.dump(json_data, f, indent=2)


if __name__ == "__main__":
    ps = PageSpeed()
    response = ps.analyse('https://www.example.com', strategy='mobile')
    ls = [
        response.url, response.loadingExperience,
        response.originLoadingExperience,
        response.originLoadingExperienceDetailed,
        response.loadingExperienceDetailed, response.finalUrl,
        response.requestedUrl, response.version, response.userAgent
    ]  # , response.lighthouseResults]
    ps.save(response)
    print(ls)
```

Responses.py Source Code:

```
import json


class Response(object):
    """
    Base Response Object

    Attributes:
        self.json (dict): JSON representation of response
        self._request (str): URL of
        self._response (`requests.models.Response` object): Response object from requests module
    """

    def __init__(self, response):
        response.raise_for_status()

        self._response = response
        self._request = response.url
        self._json = json.loads(response.content)


class PageSpeedResponse(Response):
    """
    PageSpeed Response Object

    Attributes:
        self.url (str):
        self.speed (int):
        self.statistics (`Statistics` object):
    """
    @property
    def url(self):
        return self._json['id']

    @property
    def loadingExperience(self):
        return self._json['loadingExperience']['overall_category']

    @property
    def originLoadingExperience(self):
        return self._json['originLoadingExperience']['overall_category']

    @property
    def originLoadingExperienceDetailed(self):
        metrics = self._json['originLoadingExperience']['metrics']
        keys_ = list(metrics.keys())
```

```
        originLoadingExperienceDetailed_ = {}

        for each in keys_:
                originLoadingExperienceDetailed_[each] = metrics[each]['category']

        return originLoadingExperienceDetailed_

    @property
    def loadingExperienceDetailed(self):
        metrics = self._json['loadingExperience']['metrics']
        keys_ = list(metrics.keys())

        loadingExperienceDetailed_ = {}

        for each in keys_:
                loadingExperienceDetailed_[each] = metrics[each]['category']

        return loadingExperienceDetailed_

    # In case of re-directs
    @property
    def requestedUrl(self):
        return self._json['lighthouseResult']['requestedUrl']

    @property
    def finalUrl(self):
        return self._json['lighthouseResult']['finalUrl']

    @property
    def version(self):
        return self._json['lighthouseResult']['lighthouseVersion']

    @property
    def userAgent(self):
        return self._json['lighthouseResult']['userAgent']

    @property
    def lighthouseResults(self):
        return self._json['lighthouseResult']
```

Test.py Source Code:

```
import pagespeed
from pagespeed import PageSpeed

ps = PageSpeed()

response = ps.analyse('https://www.example.com', strategy='mobile')
ls = [
```

```
    response.url, response.loadingExperience, response.originLoadingExperience,
    response.originLoadingExperienceDetailed,
    response.loadingExperienceDetailed, response.finalUrl,
    response.requestedUrl, response.version, response.userAgent
]  # , response.lighthouseResults]
ps.save(response)
print(ls)
```

18. Paint App

```
from tkinter import *
import tkinter.font

class PaintApp:
        drawing_tool = "pencil"
        left_button = "up"

        x_position, y_position = None, None

        x1_line_pt, y1_line_pt, x2_line_pt, y2_line_pt = None, None, None, None

        @staticmethod
        def quit_app(event=None):
        root.quit()

        def __init__(self, root):
        drawing_area = Canvas(root)
        drawing_area.pack()

        drawing_area.bind("<Motion>", self.motion)
        drawing_area.bind("<ButtonPress-1>", self.left_button_down)
        drawing_area.bind("<ButtonRelease-1>", self.left_button_up)

        the_menu = Menu(root)

        file_menu = Menu(the_menu, tearoff=0)
        file_menu.add_command(label="Line", command=self.set_line_drawing_tool)
        file_menu.add_command(label="Pencil", command=self.set_pencil_drawing_tool)
        file_menu.add_command(label="ARC", command=self.set_arc_drawing_tool)
        file_menu.add_command(label="Rectangle", command=self.set_rectangle_drawing_tool)
        file_menu.add_command(label="Oval", command=self.set_oval_drawing_tool)
        file_menu.add_command(label="Text", command=self.set_text_drawing_tool)

        file_menu.add_separator()
        file_menu.add_command(label="Quit", command=self.quit_app)

        the_menu.add_cascade(label="Options", menu=file_menu)
        root.config(menu=the_menu)

        def set_line_drawing_tool(self):
        self.drawing_tool = "line"

        def set_pencil_drawing_tool(self):
        self.drawing_tool = "pencil"
```

```
def set_arc_drawing_tool(self):
self.drawing_tool = "arc"

def set_rectangle_drawing_tool(self):
self.drawing_tool = "rectangle"

def set_oval_drawing_tool(self):
self.drawing_tool = "oval"

def set_text_drawing_tool(self):
self.drawing_tool = "text"

def left_button_down(self, event=None):
self.left_button = "down"
self.x1_line_pt = event.x
self.y1_line_pt = event.y


def left_button_up(self, event=None):
self.left_button = "up"
self.x_position = None
self.y_position = None

self.x2_line_pt = event.x
self.y2_line_pt = event.y

if self.drawing_tool=="line":
self.line_draw(event)
if self.drawing_tool=="pencil":
self.pencil_draw(event)
if self.drawing_tool=="arc":
self.arc_draw(event)
if self.drawing_tool=="oval":
self.oval_draw(event)
if self.drawing_tool=="rectangle":
self.rect_draw(event)
if self.drawing_tool=="text":
self.text_draw(event)

def motion(self, event=None):
if self.drawing_tool=="pencil":
self.pencil_draw(event)

self.x_position = event.x
self.y_position = event.y

def pencil_draw(self, event=None):
```

```
        if self.left_button =="down":
        if self.x_position is not None and self.y_position is not None:
        event.widget.create_line(self.x_position, self.y_position, event.x, event.y, smooth=True)


        def line_draw(self, event=None):
        if  None not in (self.x1_line_pt, self.x2_line_pt, self.y1_line_pt, self.y2_line_pt):
        event.widget.create_line(self.x1_line_pt, self.x2_line_pt, self.y1_line_pt, self.y2_line_pt, smooth=True,
fill="green")

        def arc_draw(self, event=None):
        if  None not in (self.x1_line_pt, self.x2_line_pt, self.y1_line_pt, self.y2_line_pt):
        coords = self.x1_line_pt, self.x2_line_pt, self.y1_line_pt, self.y2_line_pt

        event.widget.create_arc(coords, start=0, extent=150, style=ARC, fill="blue")
        def oval_draw(self, event=None):
        if  None not in (self.x1_line_pt, self.x2_line_pt, self.y1_line_pt, self.y2_line_pt):
        event.widget.create_oval(self.x1_line_pt, self.x2_line_pt, self.y1_line_pt, self.y2_line_pt, fill="midnight
blue", outline="yellow", width=2)

        def rect_draw(self, event=None):
        if  None not in (self.x1_line_pt, self.x2_line_pt, self.y1_line_pt, self.y2_line_pt):
        event.widget.create_rectangle(self.x1_line_pt, self.x2_line_pt, self.y1_line_pt, self.y2_line_pt, fill="red",
outline="pink", width=2)

        def text_draw(self, event=None):
        if  None not in (self.x1_line_pt, self.y1_line_pt):
        text_font = tkinter.font.Font(family="Helvetica", size=20, weight="bold", slant="italic")
        event.widget.create_text(self.x1_line_pt, self.y1_line_pt, fill="lightblue", font=text_font,
text="helloooo!")
root = Tk()
paint_app = PaintApp(root)
root.mainloop()
```

19. Password Manager

```
import os.path
# Python program to generate random
# password using Tkinter module
import random
import pyperclip
from tkinter import *
from tkinter.ttk import *

# Function for calculation of password
```

```
def low():
        entry.delete(0, END)

        # Get the length of passowrd
        length = var1.get()

        lower = "abcdefghijklmnopqrstuvwxyz"
        upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
        digits = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
!@#$%^&*()"
        password = ""

        # if strength selected is low
        if var.get() == 1:
        for i in range(0, length):
        password = password + random.choice(lower)
        return password

        # if strength selected is medium
        elif var.get() == 0:
        for i in range(0, length):
        password = password + random.choice(upper)
        return password

        # if strength selected is strong
        elif var.get() == 3:
        for i in range(0, length):
        password = password + random.choice(digits)
        return password
        else:
        print("Please choose an option")

# Function for generation of password
def generate():
        password1 = low()
        entry.insert(10, password1)


# Function for copying password to clipboard
def copy1():
        random_password = entry.get()
        pyperclip.copy(random_password)


def checkExistence():
```

```
    if os.path.exists("info.txt"):
        pass
    else:
        file = open("info.txt", 'w')
        file.close()

def appendNew():
            file = open("info.txt", 'a')
            userName = entry1.get()
            website= entry2.get()
            Random_password=entry.get()
            usrnm = "UserName: " + userName + "\n"
            pwd = "Password: " + Random_password + "\n"
            web = "Website: " + website + "\n"
            file.write("---------------------------------\n")
            file.write(usrnm)
            file.write(pwd)
            file.write(web)
            file.write("---------------------------------\n")
            file.write("\n")
            file.close
    # This function will append new password in the txt file
            file = open("info.txt", 'a')


def readPasswords():
    file = open('info.txt', 'r')
    content = file.read()
    file.close()
    print(content)

# Main Function
checkExistence()
# create GUI window
root = Tk()
var = IntVar()
var1 = IntVar()


# Title of your GUI window
root.title("Python Password Manager")


# create label for length of password
```

```
c_label = Label(root, text="Length")
c_label.grid(row=1)

# create Buttons Copy which will copy
# password to clipboard and Generate
# which will generate the password
copy_button = Button(root, text="Copy", command=copy1)
copy_button.grid(row=0, column=2)
generate_button = Button(root, text="Generate", command=generate)
generate_button.grid(row=0, column=3)

# Radio Buttons for deciding the
# strength of password
# Default strength is Medium
radio_low = Radiobutton(root, text="Low", variable=var, value=1)
radio_low.grid(row=1, column=2, sticky='E')
radio_middle = Radiobutton(root, text="Medium", variable=var, value=0)
radio_middle.grid(row=1, column=3, sticky='E')
radio_strong = Radiobutton(root, text="Strong", variable=var, value=3)
radio_strong.grid(row=1, column=4, sticky='E')
combo = Combobox(root, textvariable=var1)

# Combo Box for length of your password
combo['values'] = (8, 9, 10, 11, 12, 13, 14, 15, 16,
                   17, 18, 19, 20, 21, 22, 23, 24, 25,
                   26, 27, 28, 29, 30, 31, 32, "Length")
combo.current(0)
combo.bind('<<ComboboxSelected>>')
combo.grid(column=1, row=1)

# create label and entry to show
# password generated
userName = Label(root, text="Enter username here")
userName.grid(row=2)
entry1 = Entry(root)
entry1.grid(row=2, column=1)


# create label and entry to show
# password generated
website = Label(root, text="Enter website address here")
website.grid(row=3)
entry2 = Entry(root)
entry2.grid(row=3, column=1)
```

```
Random_password = Label(root, text="Generated password")
Random_password.grid(row=4)
entry = Entry(root)
entry.grid(row=4, column=1)


save_button = Button(root, text="Save", command=appendNew)
        save_button.grid(row=2, column=2)
        show_button = Button(root, text="Show all passwords", command=readPasswords)
        show_button.grid(row=2, column=3)

        # start the GUI
        root.mainloop()
```

20. Password Manager GUI

This script opens up a Password Manager GUI on execution. It allows the user to store all their passwords in a local SQL database

Setup instructions

In order to run this script, you need to have Python and pip installed on your system.

After you're done installing Python and pip, Open the terminal in the project folder and run

```
python passwords.py <Master password>
```

or

```
python3 passwords.py <Master password>
```

depending upon the python version. Make sure that you are running the command from the same virtual environment in which the required modules are installed.

Source Code:

```
from tkinter import *
from tkinter import messagebox, simpledialog
import sqlite3
from sqlite3 import Error
import sys

# Store Master password
master_password = sys.argv[1]

# Function to connect to the SQL Database


def sql_connection():
    try:
        con = sqlite3.connect('passwordManager.db')
        return con
    except Error:
        print(Error)

# Function to create table
```

```
def sql_table(con):
        cursorObj = con.cursor()
        cursorObj.execute(
                "CREATE TABLE IF NOT EXISTS passwords(website text, username text, pass text)")
        con.commit()


# Call functions to connect to database and create table
con = sql_connection()
sql_table(con)

# Create submit function for database


def submit(con):
    cursor = con.cursor()
    # Insert Into Table
    if website.get() != "" and username.get() != "" and password.get() != "":
        cursor.execute("INSERT INTO passwords VALUES (:website, :username, :password)",
                       {
                           'website': website.get(),
                           'username': username.get(),
                           'password': password.get()
                       }
                       )
        con.commit()
        # Message box
        messagebox.showinfo("Info", "Record Added in Database!")

        # After data entry clear the text boxes
        website.delete(0, END)
        username.delete(0, END)
        password.delete(0, END)

    else:
        messagebox.showinfo("Alert", "Please fill all details!")

# Create Query Function


def query(con):

    password = simpledialog.askstring("Password", "Enter Master Password")
    if(password == master_password):
        # set button text
        query_btn.configure(text="Hide Records", command=hide)
```

```
cursor = con.cursor()
# Query the database
cursor.execute("SELECT *, oid FROM passwords")
records = cursor.fetchall()

p_records = 'ID'.ljust(10) + 'Website'.ljust(40) + \
        'Username'.ljust(70)+'Password'.ljust(100)+'\n'

for record in records:
        single_record = ""
        single_record += (str(record[3]).ljust(10) +
                                str(record[0]).ljust(40)+str(record[1]).ljust(70)+str(record[2]).ljust(10
        single_record += '\n'
        # print(single_record)
                p_records += single_record
            query_label['text'] = p_records
            # Commit changes
            con.commit()
            p_records = ""

        else:
            messagebox.showinfo("Failed!", "Authentication failed!")

# Create Function to Hide Records


def hide():
        query_label['text'] = ""
        query_btn.configure(text="Show Records", command=lambda: query(con))


root = Tk()
root.title("Password Manager")
root.geometry("500x400")
root.minsize(600, 400)
root.maxsize(600, 400)

frame = Frame(root, bg="#774A9F", bd=5)
frame.place(relx=0.50, rely=0.50, relwidth=0.98, relheight=0.45, anchor="n")

# Create Text Boxes
website = Entry(root, width=30)
website.grid(row=1, column=1, padx=20, pady=5)
username = Entry(root, width=30)
username.grid(row=2, column=1, padx=20, pady=5)
password = Entry(root, width=30)
```

```
password.grid(row=3, column=1, padx=20, pady=5)

# Create Text Box Labels
website_label = Label(root, text="Website:")
website_label.grid(row=1, column=0)
username_label = Label(root, text=" Username:")
username_label.grid(row=2, column=0)
password_label = Label(root, text="Password:")
password_label.grid(row=3, column=0)


# Create Buttons
submit_btn = Button(root, text="Add Password", command=lambda: submit(con))
submit_btn.grid(row=5, column=1, pady=5, padx=15, ipadx=35)
query_btn = Button(root, text="Show All", command=lambda: query(con))
query_btn.grid(row=6, column=1, pady=5, padx=5, ipadx=35)

# Create a Label to show stored passwords
global query_label
query_label = Label(frame, anchor="nw", justify="left")
query_label.place(relwidth=1, relheight=1)


def main():
    root.mainloop()


if __name__ == '__main__':
    main()
```

Module 3 Projects 21-30

21. PDF & Image Text Reader That Can Speak

PDF & Image Text Reader That can Speak using python , pyttsx3 & Tesserct

Installation:
- Install tesserct-ocr using this command:
    - On Ubuntu
      ```
      sudo apt-get install tesseract-ocr
      ```
    - On Mac
      ```
      brew install tesseract
      ```
    - On Windows, download installer from [here](https://github.com/UB-Mannheim/tesseract/wiki)

- Install python binding for tesseract, pytesseract, using this pip command:
  ```
  pip install pytesseract
  ```

- Install image processing library in python, pillow using this pip command:
  ```
  pip install pillow
  ```

For working with pdf files:
- Install imagemagick using this command:
    - On Ubuntu
      ```
      sudo apt-get install imagemagick
      ```
    - For other platforms, download installer from [here](https://imagemagick.org/script/download.php)

- Install python binding for imagemagick, wand, using this pip command:
  ```
  pip install wand
  ```
- Install Pyttsx3:
```
```

pip install pyttsx3
```

**Pdf Reader  Source Code:**

```
import io
from PIL import Image
import pytesseract
from wand.image import Image as wi
import pyttsx3
import speech_recognition as sr

engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
#print(voices[1].id)
engine.setProperty('voice',voices[0].id)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

pytesseract.pytesseract.tesseract_cmd = r"C:\\Program Files\\Tesseract-OCR\tesseract.exe" #Path to the tesseract

pdf = wi(filename = "sample.pdf", resolution = 300)
pdfImage = pdf.convert('jpeg')

imageBlobs = []

for img in pdfImage.sequence:
        imgPage = wi(image = img)
        imageBlobs.append(imgPage.make_blob('jpeg'))

recognized_text = []

for imgBlob in imageBlobs:
        im = Image.open(io.BytesIO(imgBlob))
        text = pytesseract.image_to_string(im, lang = 'eng')
        recognized_text.append(text)

imageBlobs = str(text)
recognized_text = text
print(recognized_text)
speak(recognized_text)
remember = open('remember.txt','w')
remember.write(text)
```
```

remember.close()

Image reader Source Code:

```
import pytesseract #pip install tesseract
import os
from PIL import Image
import pyttsx3

engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
#print(voices[1].id)
engine.setProperty('voice',voices[0].id)

def speak(audio):
engine.say(audio)
engine.runAndWait()


    pytesseract.pytesseract.tesseract_cmd = r"C:\\Program Files\\Tesseract-OCR\tesseract.exe" #Path to the
    tesseract

    img = Image.open('img2.jpg')# add Image name here with file extention
    text = pytesseract.image_to_string(img)
    print(text)
    remember = open('remember.txt','w')
    remember.write(text)
    remember.close()
    speak(text)
```

22. PDF-To-CSV Converter

Requirements - tabula-py

Source Code:

```
import tabula  # simple wrapper for tabula-java, read tables from PDF into csv
import os
print("[-+-] starting pdf_csv.py...")
print("[-+-] import a pdf and convert it to a csv")
# --------------------------------------------------------------------------
print("[-+-] importing required packages for pdf_csv.py...")
# from modules.defaults import df # local module
print("[-+-] pdf_csv.py packages imported! \n")
# --------------------------------------------------------------------------

# --------------------------------------------------------------------------


def pdf_csv():  # convert pdf to csv
    print("[-+-] default filenames:")
    filename = "sample1"
    pdf = filename + ".pdf"
    csv = filename + ".csv"
    print(pdf)
    print(csv + "\n")

    print("[-+-] default directory:")
    print("[-+-] (based on current working directory of python file)")
    defaultdir = os.getcwd()
    print(defaultdir + "\n")

    print("[-+-] default file paths:")
    pdf_path = os.path.join(defaultdir, pdf)
    csv_path = os.path.join(defaultdir, csv)
    print(pdf_path)
    print(csv_path + "\n")

    print("[-+-] looking for default pdf...")
    if os.path.exists(pdf_path) == True:  # check if the default pdf exists
        print("[-+-] pdf found: " + pdf + "\n")
        pdf_flag = True
    else:
        print("[-+-] looking for another pdf...")
```

```
arr_pdf = [
        defaultdir for defaultdir in os.listdir()
        if defaultdir.endswith(".pdf")
]
if len(arr_pdf) == 1:  # there has to be only 1 pdf in the directory
        print("[-+-] pdf found: " + arr_pdf[0] + "\n")
        pdf_path = os.path.join(defaultdir, arr_pdf[0])
        pdf_flag = True
elif len(arr_pdf) > 1:  # there are more than 1 pdf in the directory
        print("[-+-] more than 1 pdf found, exiting script!")
        pdf_flag = False
        # TODO add option to select from available pdfs
else:
        print("[-+-] pdf cannot be found, exiting script!")
        pdf_flag = False

if pdf_flag == True:
    # check if csv exists at the default file path
    # if csv does not exist create a blank file at the default path
    try:
        print("[-+-] looking for default csv...")
        open(csv_path, "r")
        print("[-+-] csv found: " + csv + "\n")
    except IOError:
        print("[-+-] did not find csv at default file path!")
        print("[-+-] creating a blank csv file: " + csv + "... \n")
        open(csv_path, "w")

    print("[-+-] converting pdf to csv...")
    #    print("[-+-] pdf to csv conversion suppressed! \n")
    try:
        tabula.convert_into(pdf_path,
                            csv_path,
                            output_format="csv",
                            pages="all")
        print("[-+-] pdf to csv conversion complete!\n")
    except IOError:
        print("[-+-] pdf to csv conversion failed!")

    print("[-+-] converted csv file can be found here: " + csv_path + "\n")

    print("[-+-] finished pdf_csv.py successfully!")


# ----------------------------------------------------------------------
```

```
# ------------------------------------------------------------------------
pdf_csv()  # run the program
# ------------------------------------------------------------------------
```

23. Plagiarism Checker

Plagiarism Checker

How it works

- In order to compute the similarity between two text documents, the textual raw data is transformed into vectors.

- Then it is transformed into arrays of numbers and then from that by using a basic knowledge vector to compute the the similarity between them.

Dependencies

- Install scikit-learn by:

$ pip install scikit-learn

Running the app

- There are four text documents in the repository.

- Basically the code will compare all the .txt files and check for any similarity.

$ python plagiarism.py

Source code:

```
# OS Module for loading paths of textfiles. TfidfVectorizer to perform word embedding on the textual data and cosine similarity to compute the plagiarism.
import os
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
student_files = [doc for doc in os.listdir() if doc.endswith('.txt')]
student_notes = [open(File).read() for File in student_files]
# Two lambda functions, one to convert the text to arrays of numbers and the other one to compute the similarity between them.

def vectorize(Text): return TfidfVectorizer().fit_transform(Text).toarray()

def similarity(doc1, doc2): return cosine_similarity([doc1, doc2])

# Vectorize the Textual Data
vectors = vectorize(student_notes)
s_vectors = list(zip(student_files, vectors))
# computing the similarity among students
def check_plagiarism():
```

```
    plagiarism_results = set()
    global s_vectors
    for student_a, text_vector_a in s_vectors:
        new_vectors = s_vectors.copy()
        current_index = new_vectors.index((student_a, text_vector_a))
        del new_vectors[current_index]
        for student_b, text_vector_b in new_vectors:
                sim_score = similarity(text_vector_a, text_vector_b)[0][1]
                student_pair = sorted((student_a, student_b))
                score = (student_pair[0], student_pair[1], sim_score)
                plagiarism_results.add(score)
    return plagiarism_results
for data in check_plagiarism():
print(data)
```

24. Pomodoro Clock with GUI

- The given python script creates your very own pomodoro timer/tomato clock with a user friendly GUI.

- A pomodoro clock is a scientifically proven productivity timer dividing your work in time intervals of 25 minutes of high-focus period and a 5 minutes break interval.

Intalling requirements:

```
$ pip install -r requirements.txt
```

Running the script:

```
$ python Pomodoro_gui.py
```

Requirements:

pygame==2.0.1

tkinter==8.6

Source Code:

```
import time
import tkinter as tk
from tkinter import messagebox
import pygame
from datetime import timedelta

pygame.mixer.init()
pomo_count = 0
break_count = 0
enable = 0

# path of host file in windows
host_path = r"C:\Windows\System32\drivers\etc\hosts"

# URL of websites to block
block_list = []

# redirecting above URLs to this localhost to ensure blocking
redirect = "127.0.0.1"

def block_websites():
    """
    The function will open the host file and add the block-list websites to
```

```
the file if it is not already present and redirect it to the localhost
for blocking
"""
global web_var
global enable
global block_list
global host_path
url = web_var.get()
block_list.append(url)
try:
    # Opening the host file in reading and writing mode
    with open(host_path, 'r+') as h_file:
            content = h_file.read()
            for website in block_list:

                # Website is already blocked
                if website in content:
                    pass

                # To redirect the website to be blocked
                else:
                    h_file.write(redirect + "\t" + website + "\n")

    tk.messagebox.showinfo("Blocked", f"{url} successfully blocked!")
    enable = 1
    web_var.set("")

except PermissionError:
    tk.messagebox.showinfo("Error", "Run cmd in the admin mode and then try again!")
    web_var.set("")
    except (FileNotFoundError, NameError):
        tk.messagebox.showinfo("Error", "Functionality not supported in your OS!")
        web_var.set("")


def remove_websites():
    """
    The function will unblock the block_list websites by opening the file
    and removing the changes we made before
    """
    global block_list
    global host_path
    try:
        if enable:
```

```
# Opening the host file in reading and writing mode
with open(host_path, "r+") as file:

        # making each line of file into a list
        content = file.readlines()

        # sets the file pointer at the beginning of the file
        file.seek(0)

        # Traversing through each line of the host file and
        # checking for the websites to be blocked
        for lines in content:
            if not any(website in lines for website in block_list):
                file.write(lines)
                # Truncating the file to its original size
                file.truncate()

            block_list.clear()
            enable = 0
    except:
        pass
    finally:
        pass


def blocker():
    """
    The function asks input from user to block websites for high focus mode.
    """
    global enable
    global popup_4
    popup_4 = tk.Toplevel(root)
    popup_4.title("Website Blocker!")
    popup_4.geometry("360x220")
    popup_4.config( bg = 'DodgerBlue4')

    global block_list
    global web_var
    web_var=tk.StringVar()

    pass_label = tk.Label(popup_4, text = 'Enter URL to block:', font = ('Arial',12, 'bold'), bg = 'DodgerBlue4', fg =
'white')
    pass_entry = tk.Entry(popup_4, textvariable = web_var, font = ('Arial',12, 'bold'))

    sub_btn = tk.Button(popup_4, text = 'Block', font = ('Arial',12, 'bold'), command = block_websites, bg='gold',
activebackground='yellow')
```

```
    text_to_put = '*Supported for windows ONLY\n*You can add multiple urls\n*Don\'t forget to unblock after'

    instructions = tk.Label(popup_4, text = text_to_put, font = ('Arial',12, 'bold'), justify='left', bg = 'sky blue')

    unblock_btn = tk.Button(popup_4, text = 'Unblock all', font = ('Arial',12, 'bold'), command = remove_websites, state='disabled', width = 23, height = 2, bg='gold', activebackground='yellow')

    if enable:
        unblock_btn.config(state='normal')

    pass_label.place(x=25, y=10)
    pass_entry.place(x=25, y=34)
    sub_btn.place(x=255, y=30)
    instructions.place(x=25, y=80)
    unblock_btn.place(x=50, y=150)

def break_timer():
    """
    5 min timer popup window acting as a callback function to the break timer button
    """
    global enable
    global popup_2
    popup_2 = tk.Toplevel(root)
    popup_2.title("Break Timer!")
    popup_2.geometry("370x120")
    round = 0

    try:
        # Creating a continous loop of text of time on the screen for 25 mins
        t = 5*60
        while t>-1:
                minute_count = t // 60
                second_count = t % 60
                timer = '{:02d}:{:02d}'.format(minute_count, second_count)
                time_display = tk.Label(popup_2, text = timer, bg = 'DodgerBlue4', fg = 'white', font = ('STIX', 90, 'bold'))
                time_display.place(x=0,y=0)
                popup_2.update()
                time.sleep(1)
                t -= 1
    except:
        pass

    # Setting up an alarm sound and popup window to let user know when the time is up
    if t == -1:
```

```
            tk.messagebox.showinfo("Time's up!", "Break is over!\nTime to get to work!")
            popup_2.destroy()
            global break_count
            pygame.mixer.music.load("./Pomodoro_GUI/beep.wav")
            pygame.mixer.music.play(loops=1)
            break_count += 1


def show_report():
    """
    The function acts as a callback for show report button and shows the report the hours
    of work they have put in.
    """
    global popup_3
    popup_3 = tk.Toplevel(root)
    popup_3.title("Report")
    popup_3.geometry("370x170")
    popup_3.config( bg = 'DodgerBlue4')

    pomo_time = str(timedelta(minutes=pomo_count*25))[:-3]
    break_time = str(timedelta(minutes=pomo_count*5))[:-3]
    tk.Label(popup_3, text=f"Number of Pomodoros completed: {pomo_count}", justify=tk.LEFT, bg =
'DodgerBlue4', fg = 'white', font=('Arial',12,'bold')).place(x = 10, y = 10)
    tk.Label(popup_3, text=f"Number of breaks completed: {break_count}", justify=tk.LEFT, bg = 'DodgerBlue4',
fg = 'white', font=('Arial',12,'bold')).place(x = 10, y = 50)
    tk.Label(popup_3, text=f"Hours of work done: {pomo_time} hrs", justify=tk.LEFT,  bg = 'DodgerBlue4', fg =
'white', font=('Arial',12,'bold')).place(x = 10, y = 90)
    tk.Label(popup_3, text=f"Hours of break taken: {break_time} hrs", justify=tk.LEFT,  bg = 'DodgerBlue4', fg =
'white', font=('Arial',12,'bold')).place(x = 10, y = 130)


def pomodoro_timer():
"""
25 min timer popup window acting as a callback function to the work timer button
"""
global popup_1
popup_1 = tk.Toplevel(root)
popup_1.title("Work Timer!")
popup_1.geometry("370x120")
round = 0

try:
    # Creating a continous loop of text of time on the screen for 25 mins
    t = 25*60
    while t>-1:
            minute_count = t // 60
```

```
                second_count = t % 60
                timer = '{:02d}:{:02d}'.format(minute_count, second_count)
                time_display = tk.Label(popup_1, text = timer, bg = 'DodgerBlue4', fg = 'white', font = ('STIX', 90,
'bold'))
                time_display.place(x=0,y=0)
                popup_1.update()
                time.sleep(1)
                t -= 1
    except:
        pass

    # Setting up an alarm sound and popup window to let user know when the time is up
    if t == -1:
        tk.messagebox.showinfo("Time's up!", "Pomodoro completed successfully!\nYou deserve a break!")
        popup_1.destroy()
        global pomo_count
        pomo_count += 1
        pygame.mixer.music.load("./Pomodoro_GUI/beep.wav")
        pygame.mixer.music.play(loops=0)


def main():
    """
    This function produces the main screen of the Pomodoro timer with options to
    select the 25mins work timer, 5mins break timer, block websites for extra focus and
    another option to see the statistics of the time you've put in the work
    """
    # Creating the root window (main screen)
    global root
    root = tk.Tk()
    root.title('Timer')
    root.geometry('470x608')

    # Setting the screen background
    bg = tk.PhotoImage(file = "./Pomodoro_GUI/bg.png")
    label1 = tk.Label( root, image = bg)
    label1.place(x = 0, y = 0)

    intro1 = tk.Label(root, text = 'POMODORO TIMER', bg = 'snow', fg = 'maroon', font = ('Arial', 25, 'bold'))
    intro1.place(x=100, y=100)

    blocker_btn = tk.Button(root, text = 'WEBSITE BLOCKER', command = blocker, font = ('Arial', 12, 'bold'),
bg='gold', activebackground='yellow', height = 3, width = 25)
    blocker_btn.place(x=100, y=150)

    start_btn = tk.Button(root, text = 'START WORK TIMER', command = pomodoro_timer, font = ('Arial', 12,
```

```
'bold'), bg='gold', activebackground='yellow', height = 3, width = 25)
    start_btn.place(x=100, y=250)

    break_btn = tk.Button(root, text = 'START BREAK TIMER', command = break_timer, font = ('Arial', 12,
'bold'), bg='gold', activebackground='yellow', height = 3, width = 25)
    break_btn.place(x=100, y=350)

    report_btn = tk.Button(root, text = 'SHOW REPORT', command = show_report, font = ('Arial', 12, 'bold'),
bg='gold', activebackground='yellow', height = 3, width = 25)
    report_btn.place(x=100, y=450)

    root.mainloop()


if __name__ == '__main__':
            main()
```

25. Pyduku

Solve Sudoku , or let Python solve it for you!

- [X] Play sudoku yourself
- [X] Let Python play and solve it for you!
- [X] Generate random sudoku puzzle

Built with
- [Python3](https://www.python.org/)

Source code:

```
import tkinter as tk
from tkinter import font
# from time import sleep
import random

count = 0

class Sudoku:
    #Canvas background
    canvas_bg = "#fafafa" #impure white
    #Grid lines
    line_normal = "#4f4f4f" #dark grey
    line_thick = "#000000" #pure black
    #cell highlight box
    hbox_green = "#15fa00" #light green
    hbox_red = "#d61111" #red

    def __init__(self, master):
        #A record of all cells and their attributes
self.grid = {}
#A small edit window which will be initilized and displayed on click
self.e = None
```

```
self.canvas_width = 300
self.canvas_height = 300
#The sudoku grid
self.canvas = tk.Canvas(master,bg=self.canvas_bg, width=self.canvas_width, height=self.canvas_height)
self.t = tk.Entry(self.canvas)
self.t.bind("<KeyRelease>",self.keyPressed)
self.canvas.grid(columnspan=3)
self.canvas.bind("<Button 1>",self.click)
#Solve button
self.btn_solve = tk.Button(master,text='Solve', command=self.wrapper, width=8)
self.btn_solve.grid(row=1, padx=5, pady=5)
#Generate button
self.btn_gen = tk.Button(master,text='Generate', command=self.Generate, width=8)
self.btn_gen.grid(row=1, column=1, padx=5, pady=5, sticky=tk.E)
#Difficulty selector
self.set_difficulty = tk.IntVar(master,1)
self.difficulty_selector = tk.OptionMenu(master,self.set_difficulty,1,2,3,4,5)
self.difficulty_selector.grid(row=1, column=2, pady=5, sticky=tk.W)
#Individual cell width and height
self.cell_width = self.canvas_width/9
self.cell_height = self.canvas_height/9
#Draw vertical lines
for x in range(1,9):
        width=1
        fill=self.line_normal
        if(x%3==0):
                #Draw thicker black lines for seperating 3x3 boxes
                width=2
                fill=self.line_thick
        else:
                #Draw normal thin dark-grey lines
                width=1
                fill=self.line_normal
        self.canvas.create_line(self.cell_width*x, 0, self.cell_width*x, self.canvas_height, width=width,
fill=fill)
    #Draw horizontal lines in the same way
    for y in range(1,9):
        width=1
        fill=self.line_normal
        if(y%3==0):
                width=2
                fill=self.line_thick
```

```
                else:
                    width=1
                    fill=self.line_normal
                self.canvas.create_line(0, self.cell_height*y, self.canvas_width, self.cell_height*y, width=width,
fill=fill)

    def click(self, eventorigin):
        x = eventorigin.x
        y = eventorigin.y
        #Calcilate top-left x,y coords of cell clicked by mouse
        rect_x = int(x/self.cell_width)*self.cell_width
        rect_y = int(y/self.cell_height)*self.cell_height
        #Coords for drawing a square to highlight clicked cell
        coords =
[rect_x,rect_y,rect_x+self.cell_width,rect_y,rect_x+self.cell_width,rect_y+self.cell_height,rect_x,rect_y+self.cell_he
        # For some stupid reason, this line below didn't work as expected. So I had to choose the hard way.
        # h_box = self.canvas.create_rectangle(rect_x, rect_y, self.cell_width, self.cell_height, outline="#15fa00",
width=3)
        #Get cell info
        editable = self.getCell(x/self.cell_width,y/self.cell_height)[1]
        if editable:
            #It's a cell you can edit
            #Show a green box highlight and edit
            h_box = self.canvas.create_polygon(coords, outline=self.hbox_green, fill='', width=3)
            self.edit(rect_x, rect_y)
        else:
            #It's a cell containing a clue number, cannot edit
            #Show a red box highlight
            h_box = self.canvas.create_polygon(coords, outline=self.hbox_red, fill='', width=3)
        self.canvas.after(200,lambda : self.canvas.delete(h_box))

    def edit(self,cordx:int,cordy:int):
        #Create a entry inside a small canvas window
        #make sure it's actuall initilized before deleting it
        if self.e is None:
            #Not initilized, else block skipped
            pass
        else:
            #Canvas window initilized, delete and reset it to current position
            self.canvas.delete(self.e)
        #Create a mini edit window that just fits the cell
        self.e = self.canvas.create_window(cordx+1,cordy+1,window=self.t,width=self.cell_width-
1,height=self.cell_height-2,anchor=tk.NW)
        #Clean up
```

```
        self.t.delete(0,tk.END)
        self.t.focus_set()

    def keyPressed(self, event):
        val = self.t.get().strip()
        try:
                #If input is a number between 1-9, this won't raise any errors
                val = int(val)
                if(val>9 or val<0):
                        raise ValueError
        except ValueError:
                print("Invalid input!")
                self.t.delete(0,tk.END)
        else:
                #Get x,y coords of edit window and calculate cell row,column values
                x,y = (self.t.winfo_x())/self.cell_width,(self.t.winfo_y())/self.cell_height
                #Update cell with new value
                self.updateCell(val,x,y)
                self.canvas.delete(self.e)

    def updateCell(self,value,x,y,editable=True):
        #Get cell information stored in dict self.grid
        t = self.getCell(x,y)
        #Update values
        t[0] = value
        t[1] = editable
        text=value
        if value==0:
                text=' '
        #Update display value by using item id
        self.canvas.itemconfigure(t[2],text=text)
        self.canvas.update()
        #Update the dict
        self.grid[(x,y)] = t

    def getCell(self, x:int, y:int):
        #Returns info of cell at 'x' row 'y' column
        x=int(x)
        y=int(y)
        val = self.grid[(x,y)]
        return val

    def populate(self, X:[[]]):
        #Populates the sudoku grid with given 9x9 matrix and also store it in a dict
```

```
c = self.canvas
#The bookeeping is managed as shown below
'''Dict->(X,Y) : [value,True/Flase,id]
                    ^       ^       ^    ^
              X,Y coords  value  editable  object id'''
for i in range(9):
        for j in range(9):
            #Calculate x,y position of center of cell
            text_x = j*self.cell_width+self.cell_width/2
            text_y = i*self.cell_height+self.cell_height/2
            val = X[i][j]
            if val == 0:
                t = c.create_text(text_x,text_y,text=' ',font=('Times', 14))
                self.grid[(j,i)] = [ val, True, t]
            else:
                t = c.create_text(text_x,text_y,text=val,font=('Times', 15, 'bold'))
                self.grid[(j,i)] = [ val, False, t]

def clearGrid(self):
    #Utility function to clear the grid, this will also wipe out the puzzle from memory
    for i in range(9):
            for j in range(9):
                self.updateCell(0,i,j)

def getValue(self, row:int, col:int):
    #Return value at row, column
    return self.grid[(row,col)][0]

def printGrid(self):
    #Utility function to print the grid
    for i in range(9):
            x=[]
            for j in range(9):
                x.append(self.getValue(j,i))
            print(x)

def wrapper(self):
    #A small wrapper funtion that performs some small tasks before solving
    global count
    #Reset the count
    count = 0
    #Delete edit boxes if any
    self.canvas.delete(self.e)
    #Lock the buttons and start solving
```

```
        self.btn_gen.configure(state='disabled')
        self.btn_solve.configure(state='disabled')
        self.solve()
        #After solving, set the buttons back to normal
        self.btn_gen.configure(state='normal')
        self.btn_solve.configure(state='normal')

    def solve(self):
        global count
        #Start by finding an empty cell
        x,y = self.findEmpty()
        #If no cells are empty, our job is done
        if (x,y)==(None,None):
                #Print the no. of times solve() was called
                print("Recursed", count, "times.")
                return True

    #Keep a track of the number of function calls
    count+=1

    #Try putting in numbers from 1-9
    for i in range(1,10):
            #Check if number will satisfy sub-grid rule and row-column rule
            if self.is_SubGrid_Safe(i,x,y) and self.is_Cell_Safe(i,x,y):
                    #Yes, then update the cell
                    self.updateCell(i,x,y,False)
                    # self.canvas.after(10,self.updateCell(i,x,y))
                    #Now repeat for remaining cells
                    nxt = self.solve()
                    if nxt:
                            #All went well, so return true
                            return True
                    else:
                            #The value chose earlier is wrong, so backtrack
                            self.updateCell(0,x,y,True)

        #Cannot find any number, so return false (backtrack)
        return False

    def Generate(self, level=1):
        #Disable the buttons
        self.btn_solve.configure(state='disabled')
        self.btn_gen.configure(state='disabled')
        #Generate random puzzle with difficulty level 'level'
```

```
#Start by generate diagonal sub-grids with randomly shuffled nos from 1-9
nos = list(range(1,10))
rand_grid = []
for i in range(9):
        if i%3==0:
            random.shuffle(nos)
        t=[0]*9
        for j in range(3):
            t_pos = int(i/3)*3+j
            n_pos = (i%3)*3
            t[t_pos] = nos[n_pos+j]
        rand_grid.append(t)
#Clean up
        self.clearGrid()
        #Cover up the window with a label
        cover_label = tk.Label(text="GENERATING",font=('Arial',16))
        cover =
self.canvas.create_window(0,0,window=cover_label,width=self.canvas_width,height=self.canvas_height,anchor=tk.
        #Populate with the diagonal sub-grids
        self.populate(rand_grid)
        #Solve to get the completed puzzle
        self.solve()
        global count
        #Reset count
        count = 0
        #Remove random numbers based on set difficulty level, this needs work. Any Math wizards around?
        level = self.set_difficulty.get()
        if level<=2: level+=2
        for i in range(9):
            for j in range(9):
                remove = level>random.randint(1,5)
                if remove:
                    self.updateCell(0,i,j)

#Set the fonts right
g=self.grid
for i in g.keys():
        cell = g[i]
        if cell[1]:
            #Editable cells have Times-14-regular font
            self.canvas.itemconfigure(cell[2],font=('Times',14))
        else:
            #Non-editable cells have Times-15-bold font
```

```
                    self.canvas.itemconfigure(cell[2],font=('Times',15,'bold'))
        #Finally, lift the cover for the user to see the puzzle
        self.canvas.delete(cover)
        #and set the buttons back to normal
        self.btn_solve.configure(state='normal')
        self.btn_gen.configure(state='normal')

    def findEmpty(self):
        #Utility function to find an empty cell
        for i in range(9):
                for j in range(9):
                    cell_val = self.getCell(j,i)[1]
                    if cell_val:
                        return (j,i)
        return (None,None)

    def is_SubGrid_Safe(self,val,x,y)->bool:
        #Checks if the sub-grid rule is satisfied for the number 'val' at given row 'x',column 'y'
        #Figure out the sub grid x,y
        sgrid_x = int(x/3)*3
        sgrid_y = int(y/3)*3
        #Search the sub-grid
        for i in range(sgrid_x,sgrid_x+3):
                for j in range(sgrid_y,sgrid_y+3):
                    #Check only non-editable cells, ignore cells edited by user
                    if val==self.getValue(i,j) and not self.getCell(i,j)[1]:
                        #This number already exists, rule violated
                        return False
        #No duplicate number found in sub-grid, rule intact
        return True

    def is_Cell_Safe(self,val,x,y)->bool:
        #Check if the number 'val' already exists in the row 'x' or column 'y'
        for i in range(9):
                #Check only non-editable cells, ignore cells edited by user
                if val==self.getValue(x,i) and not self.getCell(x,i)[1]:
                    return False
                if val==self.getValue(i,y) and not self.getCell(i,y)[1]:
                    return False
        #Row-column rule intact
        return True


#######################################
```

```
master = tk.Tk()
master.title("PyDoku")
master.resizable(False, False)
game=Sudoku(master)
ex1= [
    [3, 0, 6, 5, 0, 8, 4, 0, 0],
    [5, 2, 0, 0, 0, 0, 0, 0, 0],
    [0, 8, 7, 0, 0, 0, 0, 3, 1],
    [0, 0, 3, 0, 1, 0, 0, 8, 0],
    [9, 0, 0, 8, 6, 3, 0, 0, 5],
    [0, 5, 0, 0, 9, 0, 6, 0, 0],
    [1, 3, 0, 0, 0, 0, 2, 5, 0],
    [0, 0, 0, 0, 0, 0, 0, 7, 4],
    [0, 0, 5, 2, 0, 6, 3, 0, 0]
    ]
#Here's an extreme puzzel, ref : https://www.sudokuwiki.org/Daily_Sudoku
ex2=[
    [0, 5, 0, 0, 0, 0, 0, 0, 0],
    [3, 0, 8, 0, 7, 0, 2, 0, 0],
    [0, 0, 9, 3, 0, 6, 8, 0, 0],
    [0, 8, 0, 0, 0, 9, 5, 0, 0],
    [9, 0, 0, 0, 0, 0, 0, 0, 1],
    [0, 0, 3, 8, 0, 0, 0, 9, 0],
    [0, 0, 6, 5, 0, 7, 3, 0, 0],
    [0, 0, 1, 0, 4, 0, 6, 0, 7],
    [0, 0, 0, 0, 0, 0, 0, 4, 0]
    ]

        game.populate(ex1)
        tk.mainloop()
```

26. PYQT5 Password Generator

Main.py Source Code:

```
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QPushButton, QLineEdit, QMessageBox
from PyQt5 import QtGui
import sys
import random_pass as rp
import logging

logging.basicConfig(filename="passwords.txt", format="%(message)s", level=logging.INFO)
with open("showMessage", "r") as f:
    showM = f.read()
    if showM == "1":
        showM = True
    else:
        showM = False


class window(QMainWindow):
    def __init__(self):
        super().__init__()

        self.x = 500
        self.y = 500
        self.title = "password-gen"


    def start(self):
        self.setGeometry(100, 100, self.x, self.y)
        self.setWindowTitle(self.title)
        self.setFixedSize(self.x, self.y)
        self.setWindowIcon(QtGui.QIcon("lock.png"))

        self.label1 = QLabel(self)
        self.label1.setText("characters:")
        self.label1.move(190, 50)

        self.charsInput = QLineEdit(self)
        self.charsInput.setText("a,b,c,d")
        self.charsInput.setGeometry(190, 100, 100, 30)

        self.label2 = QLabel(self)
```

```
self.label2.setText("length:")
self.label2.move(190, 150)

self.passLength = QLineEdit(self)
self.passLength.setText("5")
self.passLength.move(190, 200)

self.button1 = QPushButton(self)
self.button1.setText("generate password")
self.button1.clicked.connect(self.generatePassword)
self.button1.setGeometry(170, 240, 150, 30)


self.deletePassButton = QPushButton(self)
self.deletePassButton.setText("Delete")
self.deletePassButton.setGeometry(10, 460, 120, 30)
self.deletePassButton.clicked.connect(self.deletePopUp)
self.show()


def generatePassword(self):
    global showM
    self.chars = self.charsInput.text().split(",")
    self.passLen = int(self.passLength.text())

    self.password = rp.randomPass(self.chars, self.passLen)

    logging.info(f"password: {self.password} ")

    if showM:
            self.messageBox()


def messageBox(self):
    message = QMessageBox()
    message.setText("The password was written to password.txt, show this again?")
    message.setWindowTitle("password")
    message.setIcon(QMessageBox.Question)
    message.setStandardButtons(QMessageBox.No|QMessageBox.Yes)
    message.buttonClicked.connect(self.YesNo)


    x = message.exec_()

def YesNo(self, button):
    if button.text() == "&Yes":
            pass
    elif button.text() == "&No":
```

```
                with open("showMessage", "w") as f:
                        f.write("0")


def deletePasswords(self, button):
    if button.text() == "&Yes":
                try:
                        with open("passwords.txt", "w") as f:
                                f.write("")
                except:
                        raise FileNotFoundError("password file not found please press generate password")


def deletePopUp(self):
        message = QMessageBox()
        message.setText("Are you shure you want to delete all the passwords")
        message.setIcon(QMessageBox.Warning)
        message.setStandardButtons(QMessageBox.Yes|QMessageBox.Cancel)
        message.setDefaultButton(QMessageBox.Cancel)
        message.buttonClicked.connect(self.deletePasswords)

        x = message.exec_()

if __name__ == "__main__":

    app = QApplication(sys.argv)

    win = window()
    win.start()


    sys.exit(app.exec_())
```

Random Pass.py Source Code:

```
import random


def randCahr(chars):
                ranChar = random.choice(chars)

                return ranChar


def randomPass(chars, passLen):
                password = ""
```

```
for i in range(passLen):
char = randCahr(chars)
password += char

return password
```

27. Python Auto Draw

DEMO:

To run it on your PC:

* Make sure you have Python 3.7.x or Python 3.8.x installed, if not, click [here] (https://www.python.org/downloads/) to install!

* Install PyAutoGUI: `pip install pyautogui`

* Clone this into your Desktop: `git clone "https://github.com/tusharnankani/PythonAutoDraw"`

* Open Command Line or Terminal

* Change directory to a respective game: `cd "Desktop\PythonAutoDraw"`

* Run: `python python-auto-draw.py`

BASICS:

<code>
>>> import pyautogui
</code>

`>>> screenWidth, screenHeight = pyautogui.size()` # Get the size of the primary monitor.

`>>> currentMouseX, currentMouseY = pyautogui.position()` # Get the XY position of the mouse.

`>>> pyautogui.moveTo(100, 150)` # Move the mouse to XY coordinates.

`>>> pyautogui.click()` # Click the mouse.
`>>> pyautogui.click(100, 200)` # Move the mouse to XY coordinates and click it.
`>>> pyautogui.click('button.png')` # Find where button.png appears on the screen and click it.

`>>> pyautogui.move(0, 10)` # Move mouse 10 pixels down from its current position.
`>>> pyautogui.doubleClick()` # Double click the mouse.
`>>> pyautogui.moveTo(500, 500, duration=2, tween=pyautogui.easeInOutQuad)` # Use tweening/easing function to move mouse over 2 seconds.

`>>> pyautogui.write('Hello world!', interval=0.25)` # type with quarter-second pause in between each key
`>>> pyautogui.press('esc')` # Press the Esc key. All key names are in pyautogui.KEY_NAMES

`>>> pyautogui.keyDown('shift')` # Press the Shift key down and hold it.
`>>> pyautogui.press(['left', 'left', 'left', 'left'])` # Press the left arrow key 4 times.
`>>> pyautogui.keyUp('shift')` # Let go of the Shift key.

`>>> pyautogui.hotkey('ctrl', 'c')` # Press the Ctrl-C hotkey combination.

`>>> pyautogui.alert('This is the message to display.')` # Make an alert box appear and pause the program until OK is clicked.

Source Code:

```
import pyautogui
import time

# time to change tabs from editor to paint;
time.sleep(10)

# it will remain clicked till program ends;
pyautogui.click()

# can be varied according to convininence
distance = 250

while distance > 0:
        # right
pyautogui.dragRel(distance, 0, duration = 0.1)
distance -= 5

# down
pyautogui.dragRel(0, distance, duration = 0.1)

# left
pyautogui.dragRel(-distance, 0, duration = 0.1)

distance -= 5

#up
pyautogui.dragRel(0, -distance, duration = 0.1)
```

28. Pyweather

Python Script that forecasts the weather of any given city

Source Code:

```
#Importing required modules
import requests, json

#enter your API key from openweathermap.org here
api_key = 'Your API key goes here'

#base url to store url from api
base_url = "http://api.openweathermap.org/data/2.5/weather?"

#input city name here
city_name = input('Enter city name: ')

complete_url = base_url + 'appid=' + api_key + '&q=' + city_name
response = requests.get(complete_url)
x = response.json()

#checking validity of city name
if x['cod'] != '404':
    y = x['main']
    current_temperature = y['temp']
    current_pressure = y['pressure']
    current_humidity = y['humidity']
    z = x['weather']
    weather_description = z[0]['description']
    q = x['wind']
    wind_speed = q['speed']
    wind_direction = q['deg']
    k = x['clouds']
    cloudliness = k['all']

    print('Temperature (in Kelvin) = ' + str(current_temperature) + '\n Atmospheric Pressure (in hPa) = ' +
str(current_pressure) + '\n Humidity (in percentage) = ' + str(current_humidity) + '\n Wind Speed (in m/s) = ' +
str(wind_speed) + '\n Wind Direction (in degrees) = ' + str(wind_direction) + '\n Cloudliness (in percentage) = ' +
str(cloudliness) +  '\n Weather Description = ' + str(weather_description) )
else:
```

```
print('City Not Found')
```

29. QR code generator using Python

This script take a link of any URL and generate a QR code corresponding to it.

Library Used

* qrcode

To install required external modules

* Run `pip install qrcode`

How to run the script

- Provide your desired URL in the script
- Execute `python3 generate_qrcode.py`

Source Code:

```
import qrcode

input_URL = "https://www.google.com/"

qr = qrcode.QRCode(
    version=1,
    error_correction=qrcode.constants.ERROR_CORRECT_L,
    box_size=15,
    border=4,
)
qr.add_data(input_URL)
qr.make(fit=True)

img = qr.make_image(fill_color="red", back_color="white")
img.save("url_qrcode.png")

print(qr.data_list)
```

30. Racing Bar Chart Animation

Packages Needed

Make sure you are using a python virtual environment

`pip install jupyterlab`

`pip install pandas`

`pip install requests`

OR

`pip install -r requirements.txt`

Requirements

jupyterlab==2.2.2

matplotlib==3.3.0

notebook==6.1.1

numpy==1.19.1

pandas==1.1.0

requests==2.24.0

Source Code File:

animated_barchart.ipynb

Module 4 Projects 31-40

31. Random Password Generator

THIS SIMPLE PROJECT WAS MADE USING PYTHON LIBRARY FUNCTIONS LIKE `string` & `random`.

* `string.ascii_letters`

    - The concatenation of the ascii_lowercase and ascii_uppercase constants described below. This value is not locale-dependent.

* `string.ascii_lowercase`

    - The lowercase letters <kbd>abcdefghijklmnopqrstuvwxyz</kbd>. This value is not locale-dependent and will not change.

* `string.ascii_uppercase`

    - The uppercase letters <kbd>ABCDEFGHIJKLMNOPQRSTUVWXYZ</kbd>. This value is not locale-dependent and will not change.

* `string.digits`

    - The string <kbd>0123456789</kbd>.

* `string.hexdigits`

    - The string <kbd>0123456789abcdefABCDEF</kbd>.

* `string.octdigits`

The string <kbd>01234567</kbd>.

* `string.punctuation`

    - String of ASCII characters which are considered punctuation characters in the C locale: `!"#$%&'()*+,-./:;<=>?@[\]^_{|}~`

Python-Password Generator Source Code:

```
import random
import string

total = string.ascii_letters + string.digits + string.punctuation
```

```
length = 16

password = "".join(random.sample(total, length))

print(password)
```

Random_password_gen Source Code:

```
import random
import math

alpha = "abcdefghijklmnopqrstuvwxyz"
num = "0123456789"
special = "@#$%&*"

# pass_len=random.randint(8,13)  #without User INput
pass_len = int(input("Enter Password Length"))

# length of password by 50-30-20 formula
alpha_len = pass_len//2
num_len = math.ceil(pass_len*30/100)
special_len = pass_len-(alpha_len+num_len)
```

```
password = []


def generate_pass(length, array, is_alpha=False):
    for i in range(length):
        index = random.randint(0, len(array) - 1)
        character = array[index]
        if is_alpha:
                case = random.randint(0, 1)
                if case == 1:
                        character = character.upper()
        password.append(character)


# alpha password
```

```
generate_pass(alpha_len, alpha, True)
# numeric password
generate_pass(num_len, num)
# special Character password
generate_pass(special_len, special)
# suffle the generated password list
random.shuffle(password)
# convert List To string
gen_password = ""
for i in password:
        gen_password = gen_password + str(i)
print(gen_password)
```

32. Random Wikipedia Article

An application to save any random article from Wikipedia to a text file.

Use:

``` pip install htmlparser``` and ``` pip install beautifulsoup4```

Requirements:

HTMLParser==0.0.2

Source Code:

```
from bs4 import BeautifulSoup
import requests

# Trying to open a random wikipedia article
# Special:Random opens random articles
res = requests.get("https://en.wikipedia.org/wiki/Special:Random")
res.raise_for_status()

# pip install htmlparser
wiki = BeautifulSoup(res.text, "html.parser")

r = open("random_wiki.txt", "w+", encoding='utf-8')

# Adding the heading to the text file
heading = wiki.find("h1").text

r.write(heading + "\n")
for i in wiki.select("p"):
    # Optional Printing of text
    # print(i.getText())
    r.write(i.getText())

r.close()
print("File Saved as random_wiki.txt")
```
```

33. Random word from list

This is a useful program that chooses a random word from a given list.

How to run script

``` bash
python Random_word_from_list.py
```

Make sure you have a file in the same directory you wish to choose a random word from.

```

**Source Code:**

```
import sys
import random

# check if filename is supplied as a command line argument
if sys.argv[1:]:
    filename = sys.argv[1]
else:
    filename = input("What is the name of the file? (extension included): ")
try:
    file = open(filename)
except (FileNotFoundError, IOError):
    print("File doesn't exist!")
    exit()
# handle exception

# get number of lines
num_lines = sum(1 for line in file if line.rstrip())

# generate a random number between possible interval
random_line = random.randint(0, num_lines)

re-iterate from first line
```
```

```
file.seek(0)

for i, line in enumerate(file):
    if i == random_line:
        print(line.rstrip())  # rstrip removes any trailing newlines :)
        break
```

34. High Quality YouTube Video Downloader

This is a Python Script which generates random email addresses

Requirements

For this script to run you need to have progressbar package installed

Run the command in terminal to install package

```
$ pip install progressbar
```

Run the program using command

```
$ python random_email_generator.py
```

Source Code:

```
import random
import string
import csv
import progressbar
''' Ask user for total number of emails required'''
def getcount():
    rownums = input("How many email addresses?: ")
    try:
        rowint = int(rownums)
        return rowint
    except ValueError:
        print("Please enter an integer value")
        return getcount()


'''Below function creates a random length of email between 1-20 characters length and adds domain and extension to give the resulting email'''


def makeEmail():
    extensions = ['com', 'net', 'org', 'gov']
    domains = [
        'gmail', 'yahoo', 'comcast', 'verizon', 'charter', 'hotmail',
```

```
        'outlook', 'frontier'
    ]

    finalext = extensions[random.randint(0, len(extensions) - 1)]
    finaldom = domains[random.randint(0, len(domains) - 1)]

    accountlen = random.randint(1, 20)

    finalacc = ''.join(
        random.choice(string.ascii_lowercase + string.digits)
        for _ in range(accountlen))

    finale = finalacc + "@" + finaldom + "." + finalext
    return finale
# Take the total count of emails and pass them to getcount()
howmany = getcount()

# counter for While loop
counter = 0

# empty array to add emails
emailarray = []

print("Creating email addresses...")
print("Progress: ")

prebar = progressbar.ProgressBar(maxval=int(howmany))

for i in prebar(range(howmany)):
    while counter < howmany:
        emailarray.append(str(makeEmail()))
        counter += 1
        prebar.update(i)

print("Creation completed.")

for i in emailarray:
    print(i)
```

35. Raspberry-Pi-Sonoff

It is an Sonoff using Rapberry Pi

Hardware Requirements:

1.Raspberry Pi (Any Version Will Work)

<img src="img/rpi.png">

2.Relay Board

<img src="img/relay.png">

Run

Run Main.py File on RPI

```
python main.py
```

A Flask Server Will Run on http://0.0.0.0:8000/ Connect Relay With GPIO 2

Source Code:

```
from flask import Flask, render_template, request, redirect
from gpiozero import LED
from time import sleep
led = LED(2)

app = Flask(__name__)

@app.route("/")
def home():
    if led.value == 1:
        status = 'ON'
    else:
        status = 'OFF'
    return render_template('home.html', status=status)
@app.route("/on")

def on():
    led.on()
    return "LED on"

@app.route("/off")
```

```
def off():
    led.off()
    return "LED off"

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8000)
```

36. Recursive Password Generator

<!--Remove the below lines and add yours -->

Generates a random password with the length specified using recursivity

Prerequisites

<!--Remove the below lines and add yours -->

None

How to run the script

<!--Remove the below lines and add yours -->

Execute `python3 generator.py`

Source Code:

```
import random
import string

def stretch(text,maxlength):
    if len(text) < maxlength:
        randomChar = get_random_char()
        return stretch(text+randomChar,maxlength)
    else:
        return text

def get_random_char():
    chars = string.printable
    randomChar = chars[random.randint(0,len(chars)-1)]
    return randomChar

while 1:
    maxlen = input(' [?] Enter a length for your password (e for exit): ')
    try:
        maxlength = int(maxlen)
        print("",stretch('',maxlength),"'\n")
    except:
```

```
if maxlen == 'e':
        break
print('Please Enter an integer')
```

37. Reddit_Scraper_without_API

- Using BeautifulSoup, a python library useful for web scraping, this script helps to scrape a desired subreddit to obtain all relevant data regarding its posts.

- In the `fetch_reddit.py` , we take user input for the subreddit name, tags and the maximum count of posts to be scraped, we fetch and store all this information in a database file.

- In the `display_reddit.py` , we display the desired results from the database to the user.

Setup instructions

- The requirements can be installed as follows:

```shell
    $ pip install -r requirements.txt
```

Requirements:

beautifulsoup4==4.9.3

certifi==2020.12.5

chardet==4.0.0

idna==2.10

requests==2.25.1

soupsieve==2.2.1

urllib3==1.26.4

Source Code files:

fetch_reddit.py

display_reddit.py

38. Reduce Image Size

Script to reduce the size of image file using the openCV library of python.

Prerequisites

openCV library

`pip install opencv-python`

How to run the script

- Add the image in jpg format with name as 'input.jpg' in this folder.
- Run reduce_image_size.py script.
- resized output image will be generated in this folder.

Source Code:

```
# import openCV library for image handling
import cv2

# read image to be resized by imread() function of openCV library
img = cv2.imread('input.jpg')
print(img.shape)

# set the ratio of resized image
k = 5
width = int((img.shape[1])/k)
height = int((img.shape[0])/k)

# resize the image by resize() function of openCV library
scaled = cv2.resize(img, (width, height), interpolation=cv2.INTER_AREA)
print(scaled.shape)



# show the resized image using imshow() function of openCV library
cv2.imshow("Output", scaled)
```

```
        cv2.waitKey(500)
        cv2.destroyAllWindows()

# get the resized image output by imwrite() function of openCV library
cv2.imwrite('resized_output_image.jpg', scaled)
```

39. Rock Paper Scissors Game

To be Played with a Computer.

* You can enter the number of games you want to play.

* There is also a score window which is displayed after every turn.

Source Code:

```
#START;

import random
#DEFAULT;
my_dict={'R':"Rock",'P':"Paper",'S':"Scissors"}
user_count=0
comp_count=0

#INPUT;
games=int(input("\nEnter the number of games you want to play: "))

while(comp_count+user_count<games):
  #WHILE LOOP STARTS;

  flag=0

  user_input=input("\nUser's Input: ")[0]
  user_input=user_input.upper()
  #The [0] after the input() will assign the first charcter of input to the variable;
  #Hence, the user can enter anything, anyway;
  #Example: The user can enter Rock or rock or r or R or ro or any such thing which represents Rock;
  #It will always take input as a R
  #Thereby, increasing the user input window;

  for i in my_dict.keys():
  if(user_input==i):                    #If the entered input is confined to Rock, Paper or Scissors;
  flag=1
  break
  if(flag!=1):              #If not, run the loop again;
  print("INVALID INPUT")
  continue

  comp_input=random.choice(list(my_dict.keys()))       #Random Key from the dictionary my_dict i.e. R,P
or S;

  print("Computer's Input: ", my_dict[comp_input])
```

```
        if ( user_input=='R' and comp_input=='P' ) or ( user_input=='P' and comp_input=='S' ) or ( user_input=='S' and comp_input=='R' ):
            comp_count+=1
        elif ( user_input=='P' and comp_input=='R' ) or ( user_input=='S' and comp_input=='P' ) or ( user_input=='R' and comp_input=='S' ):
            user_count+=1
        else:
            print("TIE")

        print("\nSCORE:")
        print("User Score:",user_count,"\tComputer Score:",comp_count,"\n")

        #LOOP ENDS;

print("\n\t\tFINAL SCORE:")
print("User Score:",user_count,"\t\t\tComputer Score:",comp_count,"\n")
if user_count>comp_count:
        print("\n\tCONGRATULATIONS! YOU WON!")
elif user_count<comp_count:
        print("\n\t\tSORRY! YOU LOST!")
else:
        print("\n\t\tOOPS! IT'S A TIE!")



#END;
```

40. Room Security Using Laptop Webcam

Dependencies:

Open CV
```python

pip install opencv-python
```

Flask
```python
pip install flask
```

Run:

Run this Script to run a local server
```python

python main.py
```

Live Stream
```

http://0.0.0.0:5000
```

main.py Source Code:

```
# main.py
# import the necessary packages
from flask import Flask, render_template, Response
from camera import VideoCamera
app = Flask(__name__)
@app.route('/')
def index():
    # rendering webpage
    return render_template('index.html')
def gen(camera):
    while True:
        #get camera frame
        frame = camera.get_frame()
        yield (b'--frame\r\n'
                    b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
```

```
@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')
if __name__ == '__main__':
    # defining server ip address and port
    app.run(host='0.0.0.0',port='5000', debug=False)
```

Camera.py Source Code:

```
import cv2

face_cascade=cv2.CascadeClassifier("faces.xml")
ds_factor=0.6
class VideoCamera(object):
  def __init__(self):
    #capturing video
    self.video = cv2.VideoCapture(0)

  def __del__(self):
     #releasing camera
     self.video.release()
  def get_frame(self):
    #extracting frames
     ret, frame = self.video.read()
     frame=cv2.resize(frame,None,fx=ds_factor,fy=ds_factor,
     interpolation=cv2.INTER_AREA)
     gray=cv2.cvtColor(frame ,cv2.COLOR_BGR2GRAY)
     face_rects=face_cascade.detectMultiScale(gray,1.3,5)
     for (x,y,w,h) in face_rects:
      cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
      break
# encode OpenCV raw frame to jpg and displaying it
ret, jpeg = cv2.imencode('.jpg', frame)
return jpeg.tobytes()
```

Module 5 Projects 41-50

41. Scrape News From HackerNews website

Scrape News From HackerNews website

A script that scrapes a number of pages from HackerNews

How to run the script

In command go to the file directory, and "run python main.py" in commandline

Source Code:

```
import requests
import os
from bs4 import BeautifulSoup, SoupStrainer
# Makes Output Directory if it does not exist
if not os.path.exists(os.path.join(os.getcwd(), 'HackerNews')):
    os.makedirs(os.path.join(os.getcwd(), 'HackerNews'))
'''
@params page_no: The page number of HackerNews to fetch.
Adding only page number in order to add multiprocess support in future.
@params verbose: Adds verbose output to screen instead
of running the program silently.
'''


def fetch(page_no, verbose=False):
    # Should be unreachable, but just in case
    if page_no <= 0:
        raise ValueError('Number of Pages must be greater than zero')
    page_no = min(page_no, 20)
    i = page_no
    if verbose:
        print('Fetching Page {}...'.format(i))
    try:
```

```
res = requests.get('https://news.ycombinator.com/?p=' + str(i))
only_td = SoupStrainer('td')
soup = BeautifulSoup(res.content, 'html.parser', parse_only=only_td)
tdtitle = soup.find_all('td', attrs={'class': 'title'})
tdmetrics = soup.find_all('td', attrs={'class': 'subtext'})
with open(os.path.join('HackerNews', 'NewsPage{}.txt'.format(i)), 'w+') as f:
        f.write('-' * 80)
        f.write('\n')
        f.write('Page {}'.format(i))
        tdtitle = soup.find_all('td', attrs={'class': 'title'})
        tdrank = soup.find_all(
            'td',
            attrs={
                'class': 'title',
                'align': 'right'})
    tdtitleonly = [t for t in tdtitle if t not in tdrank]
    tdmetrics = soup.find_all('td', attrs={'class': 'subtext'})
    tdt = tdtitleonly
    tdr = tdrank
    tdm = tdmetrics
    num_iter = min(len(tdr), len(tdt))
    for idx in range(num_iter):
        f.write('\n' + '-' * 80 + '\n')
        rank = tdr[idx].find('span', attrs={'class': 'rank'})
        titl = tdt[idx].find('a', attrs={'class': 'storylink'})
        url = titl['href'] if titl and titl['href'].startswith(
            'https') else 'https://news.ycombinator.com/' + titl['href']
        site = tdt[idx].find('span', attrs={'class': 'sitestr'})
        score = tdm[idx].find('span', attrs={'class': 'score'})
    time = tdm[idx].find('span', attrs={'class': 'age'})
    author = tdm[idx].find('a', attrs={'class': 'hnuser'})
    f.write(
        '\nArticle Number: ' +
        rank.text.replace(
            '.',
            '') if rank else '\nArticle Number: Could not get article number')
    f.write(
        '\nArticle Title: ' +
        titl.text if titl else '\nArticle Title: Could not get article title')
    f.write(
        '\nSource Website: ' +
        site.text if site else '\nSource Website: https://news.ycombinator.com')
```

```
            f.write(
                '\nSource URL: ' +
                url if url else '\nSource URL: No URL found for this article')
            f.write(
                '\nArticle Author: ' +
                author.text if author else '\nArticle Author: Could not get article author')
            f.write(
                '\nArticle Score: ' +
                score.text if score else '\nArticle Score: Not Scored')
            f.write(
                '\nPosted: ' +
                time.text if time else '\nPosted: Could not find when the article was posted')
            f.write('\n' + '-' * 80 + '\n')
        except (requests.ConnectionError, requests.packages.urllib3.exceptions.ConnectionError) as e:
            print('Connection Failed for page {}'.format(i))
        except requests.RequestException as e:
            print("Some ambiguous Request Exception occurred. The exception is " + str(e))


    while(True):
        try:
            pages = int(
                        input('Enter number of pages that you want the HackerNews for (max 20): '))
            v = input('Want verbose output y/[n] ?')
            verbose = v.lower().startswith('y')
            if pages > 20:
                        print('A maximum of only 20 pages can be fetched')
            pages = min(pages, 20)
            for page_no in range(1, pages + 1):
                        fetch(page_no, verbose)
            break
        except ValueError:
            print('\nInvalid input, probably not a positive integer\n')
            continue
```

42. Quote Scraper

Prerequisites

* beautifulsoup4

* requests

Run `pip install -r requirements.txt` to install required external modules.

How to run the script

Execute `python3 quote_scraper.py`

Requirements:

beautifulsoup4

requests==2.23.0

Source Code:

```
from bs4 import BeautifulSoup
import requests
import csv

# URL to the website
url='http://quotes.toscrape.com'

# Getting the html file and parsing with html.parser
html=requests.get(url)
bs=BeautifulSoup(html.text,'html.parser')

# Tries to open the file
try:
    csv_file=open('quote_list.csv','w')
    fieldnames=['quote','author','tags']
    dictwriter=csv.DictWriter(csv_file,fieldnames=fieldnames)

    # Writes the headers
    dictwriter.writeheader()

    #While next button is found in the page the loop runs
    while True:
```

```
# Loops through quote in the page
for quote in bs.findAll('div',{'class':'quote'}):
        #Extract the text part of quote, author and tags
        text=quote.find('span',{'class':'text'}).text
        author=quote.find('small',{'class':'author'}).text
        tags=[]
        for tag in quote.findAll('a',{'class':'tag'}):
            tags.append(tag.text)
        #Writes the current quote,author and tags to a csv file
        dictwriter.writerow({'quote':text,'author':author,'tags':tags})

#Finds the link to next page
next=bs.find('li',{'class':'next'})
if not next:
            break

        #Gets and parses the html file of next page
        html=requests.get(url+next.a.attrs['href'])
        bs=BeautifulSoup(html.text,'html.parser')
except:
    print('Unknown Error!!!')
finally:
    csv_file.close()
```

43. Scraping Medium Articles

Scraping Medium Articles

Well [Medium](https://medium.com/) is a website containing great articles and used by many programmers.

This script asks the user for the url of a medium article, scrapes it's text and saves it to a text file into a folder named scraped_articles in the same directory.

There are 3 text files in the folder scraped_articles as an example of how the article is scraped.

Prerequisites

`pip` install the modules given in requirements.txt

Have a working network connection on the device

How to run the script

Run it like any other python file

Requirement:

beautifulsoup4==4.9.1

requests==2.23.0

Source Code:

```
import os
import sys
import requests
import re
from bs4 import BeautifulSoup

# switching to current running python files directory
os.chdir('\\'.join(__file__.split('/')[:-1]))
# function to get the html of the page
def get_page():
    global url
    url = input('Enter url of a medium article: ')
    # handling possible error
    if not re.match(r'https?://medium.com/',url):
    print('Please enter a valid website, or make sure it is a medium article')
    sys.exit(1)
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'html.parser')
    return soup
```

```
# function to remove all the html tags and replace some with specific strings
def purify(text):
    rep = {"<br>": "\n", "<br/>": "\n", "<li>":  "\n"}
    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))
    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], text)
    text = re.sub('\<(.*?)\>', '', text)
    return text

# function to compile all of the scraped text in one string
def collect_text(soup):
        fin = f'url: {url}\n\n'
        main = (soup.head.title.text).split('|')
        global title
        title = main[0].strip()
        fin += f'Title: {title.upper()}\n{main[1].strip()}'

        header = soup.find_all('h1')
        j = 1

        try:
        fin += '\n\nINTRODUCTION\n'
        for elem in list(header[j].previous_siblings)[::-1]:
        fin += f'\n{purify(str(elem))}'
        except:
        pass

        fin += f'\n\n{header[j].text.upper()}'
        for elem in header[j].next_siblings:
        if elem.name == 'h1':
        j+=1
                fin += f'\n\n{header[j].text.upper()}'
                continue
                fin += f'\n{purify(str(elem))}'
                return fin

# function to save file in the current directory
def save_file(fin):
        if not os.path.exists('./scraped_articles'):
        os.mkdir('./scraped_articles')
        fname = './scraped_articles/' + '_'.join(title.split()) + '.txt'
        with open(fname, 'w', encoding='utf8') as outfile:
        outfile.write(fin)
        print(f'File saved in directory {fname}')
```

```
# driver code
if __name__ == '__main__':
        fin = collect_text(get_page())
        save_file(fin)
```

44. Screen Recorder

It records the computer screen.

Modules Used

- time
- PIL
- numpy
- cv2

How it works

- While Running the script it captures the screen frames.

- Then it returns the screen record with realtime changes.

Source Code:

```
import cv2
import numpy as np
from PIL import ImageGrab
import time


def screenrecorder():
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    name = int(round(time.time() * 1000))
    name = '{}.avi'.format(name)
    out = cv2.VideoWriter(name, fourcc, 5.0, (1920, 1080))

    while True:
        img = ImageGrab.grab()
        img_np = np.array(img)
        frame = cv2.cvtColor(img_np, cv2.COLOR_BGR2RGB)
        cv2.imshow("Screen Recorder", frame)
        out.write(frame)

        if cv2.waitKey(1) == 27:
      break

out.release()
cv2.destroyAllWindows()
```

screenrecorder()

45. Send Email With Python

```
import smtplib
import csv
from string import Template
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
def read_template(filename):
with open(filename, ‘r’, encoding=’utf-8') as template_file:
  template_file_content = template_file.read()
return Template(template_file_content)
def main():
  message_template = read_template(‘template.txt’)
            MY_ADDRESS = ‘**********@gmail.com’
            PASSWORD = ‘**********’
            # set up the SMTP server
            s = smtplib.SMTP(host=’smtp.gmail.com’, port=587)
              s.starttls()
              s.login(MY_ADDRESS, PASSWORD)

with open(“details.csv”, “r”) as csv_file:
   csv_reader = csv.reader(csv_file, delimiter=’,’)
   # the below statement will skip the first row
   next(csv_reader)
   for lines in csv_reader:
    msg = MIMEMultipart() # create a message
# add in the actual person name to the message template
     message =    message_template.substitute(PERSON_NAME=row[0],MATH=row[2],
     ENG=row[3],SCI=row[4])
     print(message)
# setup the parameters of the message
     msg[‘From’]=MY_ADDRESS
     msg[‘To’]=lines[1]
     msg[‘Subject’]=”Mid term grades”
# add in the message body
     msg.attach(MIMEText(message, ‘plain’))
# send the message via the server set up earlier.
     s.send_message(msg)
     del msg
# Terminate the SMTP session and close the connection
              s.quit()
            if __name__ == ‘__main__’:
```

```
main()
```

46. Send Emails from CSV File

```
# Send Emails from CSV File

This project contains a simple bulk email script
which sends the same message to a list of recipients.

## Dependencies

This project only requires the Python standard library
(more specifically, the `csv`, `email`, and `smtplib` modules).

## Running the script

The script requires two configuration files:

* `emails.csv` should contain the email addresses to send the message to.
* `credentials.txt` should contain your SMTP server login credentials,
  with your user name and your password on sepate lines,
  with no additional whitespace or other decorations.

The project's directory contains two example files which you'll
probably both want and need to edit.

Once you have these files set up, simply

```
python Send_emails.py
```

Development ideas

A proper email sender would use `Cc:` or `Bcc:` and send the same
message just once.

Don't play frivolously with this; your email provider,
and/or the recipient's,
may have automatic filters which quickly block anyone who sends
multiple identical messages.

The script simply hardcodes the conventions for Gmail.com.
Other providers may use a different port number and authentication regime.
```

Source Code:

```
import csv
from email.message import EmailMessage
import smtplib


def get_credentials(filepath):
    with open("credentials.txt", "r") as f:
        email_address = f.readline()
        email_pass = f.readline()
    return (email_address, email_pass)


def login(email_address, email_pass, s):
    s.ehlo()
    # start TLS for security
    s.starttls()
    s.ehlo()
    # Authentication
    s.login(email_address, email_pass)
    print("login")


def send_mail():
    s = smtplib.SMTP("smtp.gmail.com", 587)
    email_address, email_pass = get_credentials("./credentials.txt")
    login(email_address, email_pass, s)

    # message to be sent
    subject = "Welcome to Python"
    body = """Python is an interpreted, high-level,
    general-purpose programming language.\n
    Created by Guido van Rossum and first released in 1991,
    Python's design philosophy emphasizes code readability\n
    with its notable use of significant whitespace"""

    message = EmailMessage()
    message.set_content(body)
    message['Subject'] = subject

    with open("emails.csv", newline="") as csvfile:
        spamreader = csv.reader(csvfile, delimiter=" ", quotechar="|")
        for email in spamreader:
                s.send_message(email_address, email[0], message)
                print("Send To " + email[0])

    # terminating the session
```

```
s.quit()
print("sent")


    if __name__ == "__main__":
        send_mail()
```

47. Send Text

If you do import sys, you'll get to access the functions and variables in the module sys via sys.foo or sys.bar(). This can get a lot of typing, especially if using something from submodules (e.g. I often have to access django.contrib.auth.models.User). To avoid such this redundancy, you can bring one, many or all of the variables and functions into the global scope. from os.path import exists allows you to use the function exists() without having to prepend it with os.path. all the time.

If you'd like to import more than one variable or function from os.path, you could do from os.path import foo, bar.

Source Code:

```
from twilio.rest import Client

# Your Account SID from twilio.com/console
account_sid = "0000"
# Your Auth Token from twilio.com/console
auth_token  = "0000"

client = Client(account_sid, auth_token)

message = client.messages.create(
    to="0000",
    from_="0000",
    body="Hello from Python!")

print(message.sid)
```

48. Set Alarm

Set Alarm

This script lets you set an alarm and plays your selected music after the selected time.
THIS SCRIPT ONLY WORKS ON WINDOWS

Usage
```
$ python alarm.py
```

Sample output
```
$ python3 alarm.py

############################
###### Alarm Program ######
############################

Set the alarm time (e.g. 01:10): 00:01

        Select any alarm music:

        1. The Four Seasons
        2. Carnival
        3. Renaissance
        4. Variations
        5. Dreamy Nights
        6. Lakhau Hajarau
        7. New Horizon
        8. Crusade
        9. Mozart Wakes
        10. Morning Calm

Enter the index of the listed musics (e.g. 1): 1
>> Alarm music has been set --> The Four Seasons

>>> Alarm has been set successfully for 00:01! Please dont close the program! <<<
```

Source Code:

```
import datetime
import os
```

```
import re
import subprocess


def rename_files_with_whitespaces(cd, files, extra_path=""):
    for file in files:
        if " " in file:
                renamed_file = file.replace(" ", "_")
                os.rename(os.path.join(cd, extra_path, file), os.path.join(cd, extra_path, renamed_file))


def clean_filename(file):
    return ' '.join(map(str.capitalize, file[:-4].split('_')))


def set_alarm():
    stop = False
    error = True
    while error:
        user_set_time = ":".join(map(lambda x: str(x).zfill(2), input("\nSet the alarm time (e.g. 01:10): ").split(":")))

        if re.match(r"^[0-9]{2}:[0-9]{2}$", user_set_time):
                playback_time = f"{user_set_time}:00.000000"
                error = False
        else:
                print(">>> Error: Time format invalid! Please try again!\n")

    cd = os.path.dirname(os.path.realpath(__file__))
    musics_path = os.path.join(cd, "musics")

    rename_files_with_whitespaces(cd, os.listdir(musics_path), "musics")

    musics = os.listdir(musics_path)
    if len(musics) < 1:
        print(">>> Error: No music in the musics folder! Please add music first!\n")
        exit()

    elif len(musics) == 1:
        print(">> Alarm music has been set default --> " + clean_filename(musics[0]))
        selected_music = musics[0]

    else:
        error = True
        while error:
                try:
                        print("\nSelect any alarm music:\n")
                        for i in range(1, len(musics) + 1):
```

```
                print(f"{i}. {clean_filename(musics[i - 1])}")

            user_input = int(input("\nEnter the index of the listed musics (e.g. 1): "))
            selected_music = musics[user_input - 1]
            print(">> Alarm music has been set --> "+ clean_filename(selected_music))
            error = False

        except:
            print(">>> Error: Invalid Index! Please try again!\n")

print(f"\n>>> Alarm has been set successfully for {user_set_time}! Please dont close the program! <<<")
while stop == False:
    current_time = str(datetime.datetime.now().time())
    if current_time >= playback_time:
            stop = True
            subprocess.run(('cmd', '/C', 'start', f"{cd}\\musics\\{selected_music}"))
            print(">>> Alarm ringing! Closing the program!! Bye Bye!!! <<<")

def display_header(header):
    print("")
    print("############################".center(os.get_terminal_size().columns))
    print(f"###### {header} ######".center(os.get_terminal_size().columns))
    print("############################".center(os.get_terminal_size().columns))

if __name__ == "__main__":
    display_header("Alarm Program")
    set_alarm()
```

49. Shutdown or restart your device

Power Options

<!--Remove the below lines and add yours -->

This script shuts down or restarts your computer

Prerequisites

<!--Remove the below lines and add yours -->

None

How to run the script

<!--Remove the below lines and add yours -->

Steps on how to run the script along with suitable examples.

1. Type the following on the command line:

python PowerOptions.py

2. Press enter and wait for prompt. Type “r” to restart or “s” to shut down

Example:

python PowerOptions.py

Use 'r' for restart and 's' for shutdown: r

Source Code:

```
import os
import platform

def shutdown():
    if platform.system() == "Windows":
        os.system('shutdown -s')
    elif platform.system() == "Linux" or platform.system() == "Darwin":
        os.system("shutdown -h now")
    else:
        print("Os not supported!")

def restart():
    if platform.system() == "Windows":
        os.system("shutdown -t 0 -r -f")
    elif platform.system() == "Linux" or platform.system() == "Darwin":
        os.system('reboot now')
    else:
        print("Os not supported!")


command = input("Use \'r\' for restart and \'s\' for shutdown: ").lower()
```

```
if command == "r":
    restart()
elif command == "s":
    shutdown()
else:
    print("Wrong letter")
```

50. SINE vs COSINE

```
import numpy as np
import matplotlib.pyplot as plot
# Get x values of the sine wave
time = np.linspace(-2*np.pi, 2*np.pi, 256, endpoint=True)
# Amplitude of the sine wave is sine of a variable like time
amplitude_sin = np.sin(time)
amplitude_cos = np.cos(time)
# Plot a sine wave using time and amplitude obtained for the sine wave
plot.plot(time, amplitude_sin)
plot.plot(time, amplitude_cos)
# Give a title for the sine wave plot
plot.title('Sine & Cos wave')
# Give x axis label for the sine wave plot
plot.xlabel('Time')
# Give y axis label for the sine wave plot
plot.ylabel('Amplitude')
plot.grid(True, which='both')
plot.axhline(y=0, color='k')
plot.show()
```

51. Website Blocker

Website Blocker

Description

This is a script that aims to implement a website blocking utility for Windows-based systems. It makes use of the computer's hosts files and runs it as a background process, preventing access to the sites entered by the user in array format.

Third-party libraries required:

The project requires Python's datetime library only

Importing the Libraries:

Open Command Prompt on your computer and type the following:

On the script's console, type:

`import time

`from datetime import datetime as dt`

Running the Script:

After opening the script in your Python IDE, execute the code so that you get the console output window. Open your browser and try to visit the websites you blocked. When the script runs successfully, you will see `This site can't be reached` error on the browser.

Note:

> In some systems, access to the computers's hosts files maybe denied by default to prevent malware attacks. So the script while executing may show an error while modifying the hosts files.

`Please visit [here](https://www.technipages.com/windows-access-denied-when-modifying-hosts-or-lmhosts-file) for a brief readup on how to solve the issue.`

Output:

This is how the browser acts when you try to visit the website that you blocked:

<img src="https://xtendedview.com/wp-content/uploads/2020/03/This-Site-Cant-be-Reached.jpg">

The acess will be denied to all the mentioned sites as per you changes the list

Source Code:

```
import time
from datetime import datetime as dt

# Windows host file path
hostsPath = r"C:\Windows\System32\drivers\etc\hosts"
redirect = "127.0.0.1"

# Add the website you want to block, in this list
websites = [
    "www.youtube.com", "youtube.com", "www.facebook.com",
    "facebook.com"
]
while True:
    # Duration during which, website blocker will work
    if dt(dt.now().year,
          dt.now().month,
          dt.now().day, 9) < dt.now() < dt(dt.now().year,
                                           dt.now().month,
                                           dt.now().day, 18):
        print("Access denied to Website")
    with open(hostsPath, 'r+') as file:
        content = file.read()
        for site in websites:
            if site in content:
                pass
            else:
                file.write(redirect + " " + site + "\n")
```

```
        else:
                with open(hostsPath, 'r+') as file:
                    content = file.readlines()
                file.seek(0)
        for line in content:
                if not any(site in line for site in websites):
                    file.write(line)
                file.truncate()
    print("Allowed access!")
time.sleep(5)
```

52. SMS Automation

SMS Automation Functionalities:

- First register in Twilio and add the phone numbers to whom messages are to be sent.
- On running the script and entering the api key and phone numbers the message will be delivered

Setup:

- First register on Twilio.
- Then verify your phone number from which you want to send the message.
- Now add the phone number of the receiver and verify it.
- Also allow the geo-location [permission](https://www.twilio.com/console/sms/settings/geo-permissions).

Automation Instructions:

Step 1:

Open Terminal

Step 2:

Locate to the directory where python file is located

Step 3:

Run the command: python script.py/python3 script.py

Step 4:

Sit back and Relax. Let the Script do the Job.

Requirements

- twilio

Source Code:

```
from twilio.rest import Client

api = input("Enter your ACCOUNT SID: ")
auth = input("Enter your AUTH TOKEN: ")
from_number = input("Enter number from which you want to send the SMS: ")
message = input("Enter the massage: ")
```

```
to_number = input(
    "Enter comma separated numbers to which you want to send the SMS: ")
lists = to_number.split(",")
groupnum = []
for i in lists:
    groupnum.append(i)

account_sid = api
auth_token = auth
client = Client(account_sid, auth_token)

for i in range(len(groupnum)):
    client.messages.create(from_=from_number, body=message, to=groupnum[i])
```

How to download this project:

As you are our special readers you deserve special privileges. Please download all this projects for further practise using following steps.

1. Goto - https://www.edcredibly.com/s/store/courses/description/52-Python-Projects which is our own website.
2. Apply coupon code – **SPECIAL** to make this course free available for you.
3. Checkout without paying anything and enrol.
4. Download the file and enjoy.

Cheers, Happy learning!!

ABOUT THE AUTHOR

“Edcorner Learning” and have a significant number of students on **Udemy** with more than **90000+ Student and Rating of 4.1 or above.**

Edcorner Learning is Part of Edcredibly.

Edcredibly is an online eLearning platform provides Courses on all trending technologies that maximizes learning outcomes and career opportunity for professionals and as well as students. Edcredibly have a significant number of 100000+ students on their own platform and have a **Rating of 4.9 on Google Play Store – Edcredibly App**.

Feel Free to check or join our courses on:

Edcredibly Website - https://www.edcredibly.com/

Edcredibly App –

https://play.google.com/store/apps/details?id=com.edcredibly.courses

Edcorner Learning Udemy - https://www.udemy.com/user/edcorner/

Do check our other eBooks available on Kindle Store.