53 Must Do Python Projects For All

By Edcorner Learning

53 Must Do Python Projects For All

53 Must Do Python Projects For All

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 53 Must Do 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 53 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. Snake Game

Snake game is an Arcade Maze Game which has been developed by Gremlin Industries. The player's objective in the game is to achieve maximum points as possible by collecting food or fruits. The player loses once the snake hits the wall or hits itself.

Setup instructions

In order to run this script, You just need the following 3 modules -

- **Pygame:** It is a set of Python modules designed for writing video games.
- **Time:** This function is used to count the number of seconds elapsed since the epoch.
- **Random:** This function is used to generate random numbers in Python by using random module. **Pygame, Time and Random**

Source Code:

```
import pygame
import time
import random

pygame.init()

white = (255, 255, 255)
yellow = (255, 255, 102)
black = (0, 0, 0)
red = (213, 50, 80)
green = (0, 255, 0)
blue = (50, 153, 213)

dis_width = 600
dis_height = 400

dis = pygame.display.set_mode((dis_width, dis_height))
pygame.display.set_caption('Snake Game In Python')

clock = pygame.time.Clock()

snake_block = 10
snake_speed = 15
```

```
font_style = pygame.font.SysFont("bahnschrift", 25)
score_font = pygame.font.SysFont("comicsansms", 35)


def Your_score(score):
    value = score_font.render("Your Score: " + str(score), True, yellow)
    dis.blit(value, [0, 0])


def our_snake(snake_block, snake_list):
    for x in snake_list:
        pygame.draw.rect(dis, black, [x[0], x[1], snake_block,
snake_block])


def message(msg, color):
    mesg = font_style.render(msg, True, color)
    dis.blit(mesg, [dis_width / 6, dis_height / 3])


def gameLoop():
    game_over = False
    game_close = False

    x1 = dis_width / 2
    y1 = dis_height / 2
```

```
    x1_change = 0
    y1_change = 0

    snake_List = []
    Length_of_snake = 1

    foodx = round(random.randrange(0, dis_width - snake_block) / 10.0)
* 10.0
    foody = round(random.randrange(0, dis_height - snake_block) /
10.0) * 10.0

    while not game_over:

        while game_close == True:
            dis.fill(blue)
            message(
                "You Lost! Press 'C' to Play Again or 'Q' To Quit The Game",
                red)
            Your_score(Length_of_snake - 1)
            pygame.display.update()

            for event in pygame.event.get():
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_q:
                        game_over = True
```

```
                game_close = False
            if event.key == pygame.K_c:
                gameLoop()

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        game_over = True
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT:
            x1_change = -snake_block
            y1_change = 0
        elif event.key == pygame.K_RIGHT:
            x1_change = snake_block
            y1_change = 0
        elif event.key == pygame.K_UP:
            y1_change = -snake_block
            x1_change = 0
        elif event.key == pygame.K_DOWN:
            y1_change = snake_block
            x1_change = 0

if x1 >= dis_width or x1 < 0 or y1 >= dis_height or y1 < 0:
    game_close = True
x1 += x1_change
y1 += y1_change
```

```
        dis.fill(blue)
        pygame.draw.rect(dis, green, [foodx, foody, snake_block,
snake_block])
        snake_Head = []
        snake_Head.append(x1)
        snake_Head.append(y1)
        snake_List.append(snake_Head)
        if len(snake_List) > Length_of_snake:
            del snake_List[0]

        for x in snake_List[:-1]:
            if x == snake_Head:
                game_close = True

        our_snake(snake_block, snake_List)
        Your_score(Length_of_snake - 1)

        pygame.display.update()

        if x1 == foodx and y1 == foody:
            foodx = round(
                random.randrange(0, dis_width - snake_block) / 10.0) * 10.0
            foody = round(
                random.randrange(0, dis_height - snake_block) / 10.0) * 10.0
            Length_of_snake += 1
```

```
        clock.tick(snake_speed)



    pygame.quit()
    quit()
gameLoop()
```

3. Spaceship Game

- The python script makes use of Pygame, a popular GUI module, to develop an interactive multiplayer Spaceship Game.

- The 2 players compete to aim bullets at each other and the first player to lose their health, loses.

Requirements:

All the packages essential for running the script can be installed as follows:

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

Requirements

pygame==2.0.1

Source Code files:

main.py

utility.py

2. 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.

Requirement:

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)
```

4. Speech-to-Text Converter

This Python script converts the Speech input into Text using NLP (Natural Langauge Processing).

Requirements

Installation Required :

* Python Speech Recognition module:

`pip install speechrecognition`

* PyAudio:

* Use the following command for linux users

`sudo apt-get install python3-pyaudio`

* Windows users can install pyaudio by executing the following command in a terminal

`pip install pyaudio`

* Python pyttsx3 module:

`pip install pyttsx3`

How to run the script

- Enter the audio input by speaking into the microphone.
- Run converter_terminal.py script
- Output Text will be displayed

Requirements (Py modules used)

PyAudio==0.2.11

SpeechRecognition==3.8.1

Source Code:

```
import speech_recognition

def record_voice():
        microphone = speech_recognition.Recognizer()
```

```
        with speech_recognition.Microphone() as live_phone:
        microphone.adjust_for_ambient_noise(live_phone)

        print("I'm trying to hear you: ")
        audio = microphone.listen(live_phone)
        try:
        phrase = microphone.recognize_google(audio, language='en')
        return phrase
        except speech_recognition.UnkownValueError:
        return "I didn't understand what you said"

if __name__ == '__main__':
        phrase = record_voice()

        with open('you_said_this.txt','w') as file:
        file.write(phrase)

        print('the last sentence you spoke was saved in you_said_this.txt')
```

5. Speech-To-Text

A program that can convert Speech into Text using python

Dependencies:

pyttsx3
```python

pip install pyttsx3
```
pyaudio
```python
pip install pyaudio
```
SpeechRecognition
```
pip install SpeechRecognition
```
Run:

The text Will be saved in output.txt file

```
python speech-to-text.py
```

Source Code:

```
import pyttsx3
import speech_recognition as sr
import os
```

```
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()



def get():
        r = sr.Recognizer()



with sr.Microphone() as source:
print('say something!')
audio = r.listen(source)
print("done")
try:
text = r.recognize_google(audio)
print('google think you said:\n' +text)
except Exception as e:
print(e)
```

```
        remember = open('output.txt','w')
        remember.write(text)
        remember.close()

get()
```

6. Speed Test

Speed Test using python

Dependencies:

youtube_dl
```
pip3 install speedtest-cli
```

Source Code:

```
import subprocess

returned_text = subprocess.check_output("speedtest-cli", shell=True, universal_newlines=True)
print("The Result of Speed Test")
print(returned_text)
```

7. Spelling Checker

Here, you can input any word and check if it is having a correct spelling or not.

Prerequisites

First thing which you need to install is textblob library

<!--Install library-->

>pip install textblob

<!--For jupyter nb-->

You need to run this command in your terminal or your ide terminal.

<!--for jp nb-->

If you are using Jupyter Notebook you need to use the below command

<!--for jp nb-->

>import sys

<!--command-->

>!{sys.executable} -m pip install textblob

How to run the script

You can first install the textblob library and then you can run the python script.

Source Code:

```
from textblob import TextBlob    # importing textblob library

t = 1
while t:
    a = input("Enter the word to be checked:- ")  # incorrect spelling
    print("original text: "+str(a))     #printing original text

    b = TextBlob(a)  #correcting the text

    # prints the corrected spelling
    print("corrected text: "+str(b.correct()))
    t = int(input("Try Again? 1 : 0 "))
```

8. Split a video file by given time period

This script will split the video into two files when valid time periods are given.

```

pip install ffmpeg-python
```

usage

```python
python videosplitter.py test.mp4 0 50 out1.mp4 out2.mp4
```

OR

```python
python videosplitter.py -h
```

Requirements - ffmpeg==1.4

Source Code:

```
import ffmpeg
```

```
        import argparse


        parser = argparse.ArgumentParser(description='"Split A media file
                                        into two chunks"')
        parser.add_argument('inputfile', help="Input filename")
        parser.add_argument('starttime', type=float, help="Start time in
        seconds")
        parser.add_argument('endtime', type=float, help="End time in
        seconds")
        parser.add_argument('outputfile1', help="Output filename")
        parser.add_argument('outputfile2', help="Output filename")

        args = parser.parse_args()

        in1 = ffmpeg.input(args.inputfile)

        v1 = in1.filter('trim', start=float(args.starttime), end=(args.endtime))
        v2 = in1.filter('trim', start=float(args.endtime))


        out1 = ffmpeg.output(v1, args.outputfile1)
out2 = ffmpeg.output(v2, args.outputfile2)

out1.run()
out2.run()
```

9. Split Files

This accepts split index and file name than spilts it according to the index provided.

Prerequisites

To execute this script python must be installed the host system.

How to run the script

just type this in the terminal:-

`python split_files.py <csv/text_file> <split/line_number>`

Requirements:

pandas==1.1.0

Source Code:

```
import sys
import os
import shutil
import pandas as pd

class Split_Files:
    '''
        Class file for split file program
    '''
    def __init__(self, filename, split_number):
        '''
            Getting the file name and the split index
            Initializing the output directory, if present then truncate it.
            Getting the file extension
        '''
        self.file_name = filename
        self.directory = "file_split"
        self.split = int(split_number)
        if os.path.exists(self.directory):
            shutil.rmtree(self.directory)
        os.mkdir(self.directory)
        if self.file_name.endswith('.txt'):
            self.file_extension = '.txt'
        else:
            self.file_extension = '.csv'
```

```
        self.file_number = 1


    def split_data(self):
        '''
            spliting the input csv/txt file according to the index provided
        '''
        data = pd.read_csv(self.file_name, header=None)
        data.index += 1


        split_frame = pd.DataFrame()
        output_file = f"{self.directory}/split_file{self.file_number}
{self.file_extension}"


        for i in range(1, len(data)+1):
            split_frame = split_frame.append(data.iloc[i-1])
            if i % self.split == 0:
                output_file = f"{self.directory}/split_file{self.file_number}
{self.file_extension}"
                if self.file_extension == '.txt':
                    split_frame.to_csv(output_file, header=False, index=False,
sep=' ')
                else:
                    split_frame.to_csv(output_file, header=False, index=False)
                split_frame.drop(split_frame.index, inplace=True)
                self.file_number += 1
        if not split_frame.empty:
```

```
            output_file = f"{self.directory}/split_file{self.file_number}
{self.file_extension}"
            split_frame.to_csv(output_file, header=False, index=False)


if __name__ == '__main__':
    file, split_number = sys.argv[1], sys.argv[2]
    sp = Split_Files(file, split_number)
    sp.split_data()
```

10. Split folder into subfolders

Execute

python <input_folder_name> <files_count>

Source Code:

```
import glob
import os
from shutil import copy2
import sys


def get_files(path):
    '''
    return a list of files avialable in given folder
    '''
    files = glob.glob(f'{path}/*')
    return files


def getfullpath(path):
    '''
    Return absolute path of given file
```

```
    '''
    return os.path.abspath(path)


def copyfiles(src, dst):
    '''
    This function copy file from src to dst
    if dst dir is not there it will create new
    '''
    if not os.path.isdir(dst):
        os.makedirs(dst)
    copy2(src, dst)


def split(data, count):
    '''
    Split Given list of files and return generator
    '''
    for i in range(1, len(data), count):
        if i + count-1 > len(data):
            start, end = (i-1, len(data))
        else:
            start, end = (i-1, i+count-1)
        yield data[start:end]
```

```
def start_process(path, count):
    files = get_files(path)
    splited_data = split(files, count)

    for idx, folder in enumerate(splited_data):
        name = f'data_{idx}'
        for file in folder:
            copyfiles(getfullpath(file), getfullpath(name))



if __name__ == "__main__":
    '''
    driver code
    To run this script
    python split_and_copy.py <input folder path> <20>
    '''

    if len(sys.argv) != 3:
        print("Please provide correct parameters \
        \npython split_and_copy.py <input folder path> <count>")
        sys.exit(0)

    if len(sys.argv) == 3:
        path = sys.argv[1]
```

```
    if os.path.isdir(path):
        count = sys.argv[2]
        start_process(path, int(count))
    else:
        print('Given directory name is not an valid directory')
else:
    print('Wrong paramter are provided')
```

Module 2 Project 11-20

11. Spreadsheet Automation

Spreadsheet Automation Functionalities:

- First upload two datasets
- The script will we compare the two datasets
- The output will be a pie chart

Spreadsheet Automation Instructions:

Step 1:

Open Termnial

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

- **pandas**
- **plotly**

Source Code:

```
# importing libraries

import pandas as pd
import plotly.express as px
```

```
        # storing the dataset
        data1 = input("Enter first dataset")
        data2 = input("Enter second dataset")

        # reading the data
        data_read_1 = pd.read_excel(data1)
        data_read_2 = pd.read_excel(data2)

        # print(df_prices, df_home_1)

        reference = input("What is the basis of merging? ")
        data_total = data_read_2.merge(data_read_1, on=reference)



        # print(df_total)
        criteria_1 = input("Enter criteria 1")
criteria_2 = input("Enter criteria 2")
fig = px.pie(data_total[[criteria_1, criteria_2]],
             values=criteria_2, names=criteria_1)
fig.show()
```

12. Store emails in CSV

This project contains a simple script to extract email messages
from an IMAP server.

The messages are written to a simple four-column CSV file.

Dependencies

This depends on the BeautifulSoup library and `lxml`
for extracting text from HTML messages.

Running the script

You will need to have a file `credentials.txt`
with your IMAP server account name and password on separate lines.

Gmail - and many other IMAP providers -
requires you to create a separate "application password"
to allow this code to run, so probably do that first.
Then put that password in `credentials.txt`.

Then simply run

```
python store_emails.py
```

This generates `mails.csv` in the current directory.

The generated CSV file contains the following fields for each message:

* Date
* From (Sender)
* Subject
* Message text

Requirements:
beautifulsoup4
lxml

Source code:

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

import csv
```

```
        import email
        from email import policy
        import imaplib
        import logging
        import os
        import ssl

        from bs4 import BeautifulSoup



        credential_path = "credentials.txt"
        csv_path = "mails.csv"

        logger = logging.getLogger('imap_poller')

        host = "imap.gmail.com"
port = 993
ssl_context = ssl.create_default_context()



def connect_to_mailbox():
    # get mail connection
    mail = imaplib.IMAP4_SSL(host, port, ssl_context=ssl_context)

    with open(credential_path, "rt") as fr:
```

```
        user = fr.readline().strip()
        pw = fr.readline().strip()
        mail.login(user, pw)

    # get mail box response and select a mail box
    status, messages = mail.select("INBOX")
    return mail, messages



# get plain text out of html mails
def get_text(email_body):
    soup = BeautifulSoup(email_body, "lxml")
    return soup.get_text(separator="\n", strip=True)



def write_to_csv(mail, writer, N, total_no_of_mails):

    for i in range(total_no_of_mails, total_no_of_mails - N, -1):
        res, data = mail.fetch(str(i), "(RFC822)")

        response = data[0]
        if isinstance(response, tuple):
            msg = email.message_from_bytes(response[1], policy=policy.default)

            # get header data
```

```
email_subject = msg["subject"]
email_from = msg["from"]
email_date = msg["date"]
email_text = ""

# if the email message is multipart
if msg.is_multipart():
    # iterate over email parts
    for part in msg.walk():
        # extract content type of email
        content_type = part.get_content_type()
        content_disposition = str(part.get("Content-Disposition"))
        try:
            # get the email email_body
            email_body = part.get_payload(decode=True)
            if email_body:
                email_text = get_text(email_body.decode('utf-8'))
        except Exception as exc:
            logger.warning('Caught exception: %r', exc)
        if (
            content_type == "text/plain"
            and "attachment" not in content_disposition
        ):
            # print text/plain emails and skip attachments
            # print(email_text)
```

```
                pass
            elif "attachment" in content_disposition:
                pass

        else:
            # extract content type of email
            content_type = msg.get_content_type()
            # get the email email_body
            email_body = msg.get_payload(decode=True)
            if email_body:
                email_text = get_text(email_body.decode('utf-8'))

        if email_text is not None:
            # Write data in the csv file
            row = [email_date, email_from, email_subject, email_text]
            writer.writerow(row)
        else:
            logger.warning('%s:%i: No message extracted', "INBOX", i)

def main():
    mail, messages = connect_to_mailbox()

    logging.basicConfig(level=logging.WARNING)

    total_no_of_mails = int(messages[0])
```

```
    # no. of latest mails to fetch
    # set it equal to total_no_of_emails to fetch all mail in the inbox
    N = 2

    with open(csv_path, "wt", encoding="utf-8", newline="") as fw:
        writer = csv.writer(fw)
        writer.writerow(["Date", "From", "Subject", "Text mail"])
        try:
            write_to_csv(mail, writer, N, total_no_of_mails)
        except Exception as exc:
            logger.warning('Caught exception: %r', exc)


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

13. String search from multiple files

Finds a file with the inputted string in the specified folder of your choice.

Prerequisites

Python3 is the only prerequisites! No external modules are needed to run.

How to run the script

In order to run this script you must have Python3 installed, not Python2. The command to run this is simply `python3 findstring.py`, and you'll be prompted with two questions, the string to search, and where to look.

Source Code:

```
import os
```

```
text = input("input text : ")

path = input("path : ")

# os.chdir(path)


def getfiles(path):
    f = 0
    os.chdir(path)
    files = os.listdir()
    # print(files)
    for file_name in files:
        abs_path = os.path.abspath(file_name)
        if os.path.isdir(abs_path):
            getfiles(abs_path)
        if os.path.isfile(abs_path):
            f = open(file_name, "r")
            if text in f.read():
                f = 1
                print(text + " found in ")
                final_path = os.path.abspath(file_name)
                print(final_path)
                return True
```

```
    if f == 1:
        print(text + " not found! ")
        return False


getfiles(path)
```

14. Take A Break

1. Get or set some favorite URLs for the user
2. measure 2 hours of time that has passed
3. promt the browser to open at one of the set URLs
4. have a loop to do this

Source Code files:

15. Terminal-based hangman game

This project contains a simple python script to play terminal-based hangman game.

Prerequisites

None

How to run the script

- Run the hangman.py script.
- Start to guess the word.

Source Code:

```
import random
```

```
from json import load



# function to randomly get one word from words.py and convert the word to uppercase
def get_word():
    with open('words.json') as json_file:
        data = load(json_file)
    wordArray = data["word_list"]
    word = random.choice(wordArray)
    word = word.upper()
    return word



# function to play the game
def play(word):


    # intialise variable
    word_completion = "_" * len(word)  # generate a line to show the number of word
    guessed = False  # indicate the status of guess
    guessed_letters = []  # store guessed letters
    guessed_words = []  # store guessed words
    tries = 6  # user have 6 times of wrong
    # display message and the format of the hangman
    print("Let's play Hangman!")
```

```
    print(display_hangman(tries))
    print(word_completion)
    print("\n")
    print("Length of the word: ", len(word))
    print("\n")

    # user can keep guessing when the tries is more than 0 and the answer is not found yet.
    while not guessed and tries > 0:

        # Display message and ask for user input and convert it into uppercase
        guess = input("Please guess a letter or the word: ").upper()

        # check the length of the user input and is it alpha or not
        if len(guess) == 1 and guess.isalpha():

            # display message when user guess the same letter twice
            if guess in guessed_letters:
                print("You already guessed the letter", guess)

            # display message and deduct the tries when user guess the wrong letter
            elif guess not in word:
                print(guess, "is not in the word.")
                tries -= 1
                guessed_letters.append(guess)
```

```
            # dispay message and store the letter when the user guess the correct
letter
            else:
                print("Good job,", guess, "is in the word!")
                guessed_letters.append(guess)
                word_as_list = list(word_completion)

                indices = [i for i, letter in enumerate(word) if letter == guess]
                for index in indices:
                    word_as_list[index] = guess

                # join the guess word in the word_completion
                word_completion = "".join(word_as_list)

                # if there is not blank space in word_completion change the status
of guess to true
                if "_" not in word_completion:
                    guessed = True

        # check the length of the user input and is it alpha or not
        elif len(guess) == len(word) and guess.isalpha():
            # display message when user guess the same letter twice
            if guess in guessed_words:
                print("You already guessed the word", guess)
```

```
            # display message and deduct the tries when user guess the wrong
letter
            elif guess != word:
                print(guess, "is not the word.")
                tries -= 1
                guessed_words.append(guess)

            # change the status of guess
            else:
                guessed = True
                word_completion = word

        # display error message for user
        else:
            print("Not a valid guess.")

        # display the format of hangman each time of guess
        print(display_hangman(tries))
        print(word_completion)
        print("\n")
        print("Length of the word: ", len(word))
        print("\n")

    # if the variable of guess is true means user win the game
    if guessed:
        print("Congrats, you guessed the word! You win!")
```

```
    # else means user lose the game.
    else:
        print("Sorry, you ran out of tries. The word was " + word + ". Maybe next time!")


# function to display the format of hangman
def display_hangman(tries):
    stages = ["""
                --------
                |      |
                |      0
                |     \\|/
                |      |
                |     / \\
                -
            """,
            """
                --------
                |      |
                |      0
                |     \\|/
                |      |
                |     /
                -
            """,
```

```
"""
    --------
    |      |
    |      0
    |     \\|/
    |      |
    |
    -
""",
"""
    --------
    |      |
    |      0
    |     \\|
    |      |
    |
    -
""",
"""
    --------
    |      |
    |      0
    |      |
    |      |
    |
```

```
            -
        """,
        """
            --------
            |      |
            |      0
            |
            |
            |
            -
        """,
        """
            --------
            |      |
            |
            |
            |
            |
            -
        """
        ]
    return stages[tries]


# main function to start the game
```

```
def main():
    word = get_word()
    play(word)
    while input("Play Again? (Y/N): ").upper() == "Y":
        word = get_word()
        play(word)


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

16. Terminal Progress bar with image Resizing

Terminal Progress bar with image Resizing

Here I just take example of image resizing for displaying progress bar.

when we convert lots of images at time we can use progress bar to show how many images are resized.

For this purpose I am using tqdm librabry

` pip install tqdm `

This Library is for showing progress bar

For Resizing images

` pip install Pillow `

Requirements:

tqdm==4.48.2

PIL==1.1.6

Source Code:

```
from tqdm import tqdm
```

```
from PIL import Image
import os
from time import sleep


def Resize_image(size, image):
    if os.path.isfile(image):
        try:
            im = Image.open(image)
            im.thumbnail(size, Image.ANTIALIAS)
            im.save("resize/" + str(image) + ".jpg")
        except Exception as ex:
            print(f"Error: {str(ex)} to {image}")


path = input("Enter Path to images : ")
size = input("Size Height , Width : ")
size = tuple(map(int, size.split(",")))

os.chdir(path)

list_images = os.listdir(path)
if "resize" not in list_images:
    os.mkdir("resize")
```

```
for image in tqdm(list_images, desc="Resizing Images"):
    Resize_image(size, image)
    sleep(0.1)
print("Resizing Completed!")
```

17. Text to Speech

When executed the text from abc.txt will be turned into an mp3, saved and then played on your device.

Prerequisites

- abc.txt with your text

- the gTTS==2.1.1 module (pip install gTTS to download)

- the os module (pip install os)

How to run the script

Write your desired text into the abc.txt file

then execute the txtToSpeech.py file. This can be

done by typing 'python txtToSpeech.py' into your Terminal.

Requirements - gTTS==2.1.1

Source Code:

```
from gtts import gTTS
import os
```

```
file = open("abc.txt", "r").read()

speech = gTTS(text=file, lang='en', slow=False)
speech.save("voice.mp3")
os.system("voice.mp3")

#print(file)
```

18. Text Editor

Source Code:

```
from tkinter import *
import tkinter.filedialog


class TextEditor:

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

    def open_file(self, event=None):

    txt_file = tkinter.filedialog.askopenfilename(parent=root,
initialdir="./examples")

    if txt_file:

    self.text_area.delete(1.0, END)
```

```
with open(txt_file) as _file:

self.text_area.insert(1.0, _file.read())

root.update_idletasks()

def save_file(self, event=None):
file = tkinter.filedialog.asksaveasfile(mode='w')

if file != None:
data = self.text_area.get('1.0', END + '-1c')

file.write(data)
file.close()

def __init__(self, root):
self.text_to_write = ""

root.title("TextEditor")

root.geometry("600x550")

frame = Frame(root, width=600, height=550)

scrollbar = Scrollbar(frame)
```

```
        self.text_area = Text(frame , width=600, height=550,
yscrollcommand=scrollbar.set, padx = 10, pady=10)

        scrollbar.config(command=self.text_area.yview)

        scrollbar.pack(side="right", fill="y")

        self.text_area.pack(side="left", fill="both", expand=True)

        frame.pack()

        the_menu = Menu(root)

        file_menu = Menu(the_menu, tearoff=0)
        file_menu.add_command(label="Open", command=self.open_file)
        file_menu.add_command(label="Save", command=self.save_file)

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

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



        root = Tk()
```

```
text_editor = TextEditor(root)
root.mainloop()
```

19. Textfile Analysis

```
# -*- cofing: utf-8 -*-
import os
import sys
import collections
import string

script_name = sys.argv[0]

res = {
    "total_lines":"",
    "total_characters":"",
    "total_words":"",
    "unique_words":"",
    "special_characters":""
}

try:
    textfile = sys.argv[1]
    with open(textfile, "r", encoding = "utf_8") as f:

        data = f.read()
        res["total_lines"] = data.count(os.linesep)
```

```
        res["total_characters"] = len(data.replace(" ","")) - res["total_lines"]
        counter = collections.Counter(data.split())
        d = counter.most_common()
        res["total_words"] = sum([i[1] for i in d])
        res["unique_words"] = len([i[0] for i in d])
        special_chars = string.punctuation
        res["special_characters"] = sum(v for k, v in
collections.Counter(data).items() if k in special_chars)


         except IndexError:
            print('Usage: %s TEXTFILE' % script_name)
         except IOError:
            print('"%s" cannot be opened.' % textfile)


         print(res)
```

20. Tic Tac Toe

Description

A python based 2-player Tic Tac Toe game.

It takes input for the respective x and y coordinates of the two players.

The two players are named as X and O

and will enter their desired coordinates alternatively to win the game.

Prerequisites

Use any Python online compiler of download python IDE from https://www.python.org/

How to run

Just run

```sh
python tic_tac_toe.py
```

Source Code:

```
def start():
    global board
    board = [
        [",",","],
        [",",","],
        [",",","]
    ]

def print_board():
    print(' ---------')
    for row in board:
        print(' ',row[0],'|',row[1],'|',row[2])
        print(' ---------')

def have_empty_room():
    for row in board:
        for room in row:
            if not room:
                return True
    return False

def set_room_state(roomxy,state):
    x = int(roomxy[0])-1
```

```
    y = int(roomxy[1])-1
    row = board[x]
    room = row[y]
    if not room:
        board[x][y] = state
        return True
    return False

def check_xy(xy):
    xy = str(xy)
    if len(xy) != 2:
        return False
    if int(xy[0]) > 3 or int(xy[0]) < 1 or int(xy[1]) > 3 or int(xy[1]) < 1:
        return False
    return True

def check_for_win():
    if board[0][0] == board[0][1] == board[0][2] != '':
        winner = board[0][0]
        print(f'{winner} won!')

    elif board[1][0] == board[1][1] == board[1][2] != '':
        winner = board[1][0]
        print(f'{winner} won!')
```

```
elif board[2][0] == board[2][1] == board[2][2] != '':
    winner = board[2][0]
    print(f'{winner} won!')

elif board[0][0] == board[1][0] == board[2][0] != '':
    winner = board[0][0]
    print(f'{winner} won!')

elif board[0][1] == board[1][1] == board[2][1] != '':
    winner = board[0][1]
    print(f'{winner} won!')

elif board[0][2] == board[1][2] == board[2][2] != '':
    winner = board[0][0]
    print(f'{winner} won!')

elif board[0][0] == board[1][1] == board[2][2] != '':
    winner = board[0][0]
    print(f'{winner} won!')

elif board[0][2] == board[1][1] == board[2][0] != '':
    winner = board[0][2]
    print(f'{winner} won!')

else:
```

```
        return False

    return True


turn = 'o'
start()

while have_empty_room():
    print_board()
    print('\n')
    if turn == 'o':
        turn = 'x'
    else:
        turn = 'o'
    print(f'{turn}\'s Turn!')
    while True:
        xy = int(input('enter x and y: '))
        if check_xy(xy):
            if set_room_state(str(xy),turn):
                break
            print('This room is full!')
            continue
        print('Error!')
        continue
```

```
    if check_for_win():
        break
print_board()
print('Game Over')
input()
```

Module 3 Projects 21-30

21. Tic-Tac-Toe-AI

Adding a simple AI to the Tic-Tac-Toe Game:

3 modes:

- Player vs. Player (2 - player mode)

- Player vs. AI (1 - player mode)

- AI vs. AI (*for fun*)

References

Logic

- Optimal Tic Tac Toe Moves

DEMO:

The board will be printed out every time a player makes a move.

The board will look like this!

The positions of this 3 x 3 board is same as the **keypad on the right side of your key board**.

Source Code:

```
####        TIC TAC TOE    ####

#START;
```

```
#FUNCTIONS;


def default():
        #To be printed as Default;
    print("\nWelcome! Let's play TIC TAC TOE!\n")



def rules():
    print("The board will look like this!")
    print("The positions of this 3 x 3 board is same as the right side of your key board.\n")
    print(" 7 | 8 | 9 ")
    print("-----------")
    print(" 4 | 5 | 6 ")
    print("-----------")
    print(" 1 | 2 | 3 ")
    print("\nYou just have to input the position(1-9).")



def play():
        #Asking if the player is ready;
    return input("\nAre you ready to play the game? Enter [Y]es or [N]o.\t").upper().startswith('Y')
```

```
def names():
    #Player names input;

    p1_name=input("\nEnter NAME of PLAYER 1:\t").capitalize()
    p2_name=input("Enter NAME of PLAYER 2:\t").capitalize()
    return (p1_name, p2_name)



def choice():
    #Player choice input;
    p1_choice = ' '
    p2_choice = ' '
    while p1_choice != 'X' or p1_choice != 'O':          #while loop; if the entered value isn't X or O;

        #WHILE LOOP STARTS


        p1_choice = input(f"\n{p1_name}, Do you want to be X or O?\t")[0].upper()
        #The input above has [0].upper() in the end;
        #So the user can enter x, X, xxxx or XXX; the input will always be taken as X;
        #Thereby, increasing the user input window;


        if p1_choice == 'X' or p1_choice == 'O':
```

```
            #if entered value is X or O; get out of the loop;
            break
        print("INVALID INPUT! Please Try Again!")
        #if the entered value isn't X or O, re-run the while loop;

        #WHILE LOOP ENDS
    #Assigning the value to p2 and then diplaying the values;
    if p1_choice == 'X':
        p2_choice = 'O'
    elif p1_choice == 'O':
        p2_choice = 'X'

    return (p1_choice, p2_choice)



def first_player():
    #This function will randomly decide who will go first;
    import random
    return random.choice((0, 1))



def display_board(board, avail):
    print("    " + " {} | {} | {} ".format(board[7],board[8],board[9]) + "            "
+ " {} | {} | {} ".format(avail[7],avail[8],avail[9]))
    print("    " + "-----------" + "            " + "-----------")
```

```
    print("    " + " {} | {} | {} ".format(board[4],board[5],board[6]) + "          "
+ " {} | {} | {} ".format(avail[4],avail[5],avail[6]))

    print("    " + "-----------" + "          " + "-----------")

    print("    " + " {} | {} | {} ".format(board[1],board[2],board[3]) + "          "
+ " {} | {} | {} ".format(avail[1],avail[2],avail[3]))



def player_choice(board, name, choice):

    position = 0

    #Initialising position as 0^; so it passes through the while loop;

    while position not in [1,2,3,4,5,6,7,8,9] or not space_check(board,
position):

        position = int(input(f'\n{name} ({choice}), Choose your next position:
(1-9) \t'))


        if position not in [1,2,3,4,5,6,7,8,9] or not space_check(board, position)
or position == "":

            #To check whether the given position is in the set [1-9] or whether it
is empty or occupied;

            print(f"INVALID INPUT. Please Try Again!\n")

    print("\n")

    return position



# THIS IS THE FUNCTION WHERE AI IS ADDED:

def CompAI(board, name, choice):

    position = 0
```

```
    possibilities = [x for x, letter in enumerate(board) if letter == ' ' and x != 0]


    # including both X and O, since if computer will win, he will place a choice there, but if the component will win --> we have to block that move
    for let in ['O', 'X']:
        for i in possibilities:
            # Creating a copy of the board everytime, placing the move and checking if it wins;
            # Creating a copy like this  and not this boardCopy = board, since changes to boardCopy changes the original board;
            boardCopy = board[:]
            boardCopy[i] = let
            if(win_check(boardCopy, let)):
                position = i
                return position


    openCorners = [x for x in possibilities if x in [1, 3, 7, 9]]


    if len(openCorners) > 0:
        position = selectRandom(openCorners)
        return position


    if 5 in possibilities:
        position = 5
        return position
```

```
    openEdges = [x for x in possibilities if x in [2, 4, 6, 8]]

    if len(openEdges) > 0:
        position = selectRandom(openEdges)
        return position



def selectRandom(board):
    import random
    ln = len(board)
    r = random.randrange(0,ln)
    return board[r]


def place_marker(board, avail, choice, position):
    #To mark/replace the position on the board list;
    board[position] = choice
    avail[position] = ' '


def space_check(board, position):
    #To check whether the given position is empty or occupied;
    return board[position] == ' '
```

```
def full_board_check(board):
    #To check if the board is full, then the game is a draw;
    for i in range(1,10):
        if space_check(board, i):
            return False
    return True



def win_check(board, choice):
    #To check if one of the following patterns are true; then the respective player has won!;


    #HORIZONTAL CHECK;
    return (
      ( board[1] == choice and board[2] == choice and board[3] == choice )
    or ( board[4] == choice and board[5] == choice and board[6] == choice )
    or ( board[7] == choice and board[8] == choice and board[9] == choice )
    #VERTICAL CHECK;
    or ( board[1] == choice and board[4] == choice and board[7] == choice )
    or ( board[2] == choice and board[5] == choice and board[8] == choice )
    or ( board[3] == choice and board[6] == choice and board[9] == choice )
    #DIAGONAL CHECK;
    or ( board[1] == choice and board[5] == choice and board[9] == choice )
    or ( board[3] == choice and board[5] == choice and board[7] == choice )  )
```

```
def delay(mode):
    if mode == 2:
        import time
        time.sleep(2)

def replay():
    #If the users want to play the game again?
    return input('\nDo you want to play again? Enter [Y]es or [N]o:
').lower().startswith('y')



#MAIN PROGRAM STARTS;

print("\n\t\t NAMASTE! \n")
input("Press ENTER to start!")

default()
rules()



while True:

##################################################################

    #Creating the board as a list; to be kept replacing it with user input;
    theBoard = [' ']*10
```

```
    #Creating the available options on the board:
    available = [str(num) for num in range(0,10)] # a List Comprehension
    #available = '0123456789'



    print("\n[0]. Player vs. Computer")
    print("[1]. Player vs. Player")
    print("[2]. Computer vs. Computer")
    mode = int(input("\nSelect an option [0]-[2]: "))
    if mode == 1:
        #Asking Names;
        p1_name, p2_name = names()
        # Asking Choices; Printing choices; X or O;
        p1_choice, p2_choice = choice()
        print(f"\n{p1_name}:", p1_choice)
        print(f"{p2_name}:", p2_choice)


    elif mode == 0:
        p1_name = input("\nEnter NAME of PLAYER who will go against the
Computer:\t").capitalize()
        p2_name = "Computer"
        # Asking Choices; Printing choices; X or O;
        p1_choice, p2_choice = choice()
        print(f"\n{p1_name}:", p1_choice)
```

```
        print(f"{p2_name}:", p2_choice)

    else:
        p1_name = "Computer1"
        p2_name = "Computer2"
        p1_choice, p2_choice = "X", "O"
        print(f"\n{p1_name}:", p1_choice)
        print(f"\n{p2_name}:", p2_choice)



    #Printing randomly who will go first;
    if first_player():
        turn = p2_name
    else:
        turn = p1_name

    print(f"\n{turn} will go first!")

    #Asking the user, if ready to play the game; Output will be True or False;
    if(mode == 2):
        ent = input("\nThis is going to be fast! Press Enter for the battle to
begin!\n")
        play_game = 1
    else:
        play_game = play()
```

```
    while play_game:

        ############################
        #PLAYER1
        if turn == p1_name:

            #Displaying the board;
            display_board(theBoard, available)

            #Position of the input;
            if mode != 2:
                position = player_choice(theBoard, p1_name, p1_choice)
            else:
                position = CompAI(theBoard, p1_name, p1_choice)
                print(f'\n{p1_name} ({p1_choice}) has placed on {position}\n')

            #Replacing the ' ' at *position* to *p1_choice* in *theBoard* list;
            place_marker(theBoard, available, p1_choice, position)

            #To check if Player 1 has won after the current input;
            if win_check(theBoard, p1_choice):
                display_board(theBoard, available)

print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
                if(mode):
```

```
                print(f'\n\nCONGRATULATIONS {p1_name}! YOU HAVE
WON THE GAME!\n\n')
            else:
                print('\n\nTHE Computer HAS WON THE GAME!\n\n')

print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
            play_game = False


        else:
            #To check if the board is full; if yes, the game is a draw;
            if full_board_check(theBoard):
                display_board(theBoard, available)
                print("~~~~~~~~~~~~~~~~~~~")
                print('\nThe game is a DRAW!\n')
                print("~~~~~~~~~~~~~~~~~~~")
                break
            #If none of the above is possible, next turn of Player 2;
            else:
                turn = p2_name



    ##############################
    #PLAYER2
    elif turn == p2_name:

        #Displaying the board;
```

```
        display_board(theBoard, available)

        #Position of the input;
        if(mode == 1):
            position = player_choice(theBoard, p2_name, p2_choice)
        else:
            position = CompAI(theBoard, p2_name, p2_choice)
            print(f'\n{p2_name} ({p2_choice}) has placed on {position}\n')

        #Replacing the ' ' at *position* to *p2_choice* in *theBoard* list;
        place_marker(theBoard, available, p2_choice, position)

        #To check if Player 2 has won after the current input;
        if win_check(theBoard, p2_choice):
            display_board(theBoard, available)

print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
            if(mode):
                print(f'\n\nCONGRATULATIONS {p2_name}! YOU HAVE
WON THE GAME!\n\n')
            else:
                print('\n\nTHE Computer HAS WON THE GAME!\n\n')

print("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
            play_game = False
```

```
        else:
            #To check if the board is full; if yes, the game is a draw;
            if full_board_check(theBoard):
                display_board(theBoard, available)
                print("~~~~~~~~~~~~~~~~~~")
                print('\nThe game is a DRAW!\n')
                print("~~~~~~~~~~~~~~~~~~")
                break
            #If none of the above is possible, next turn of Player 2;
            else:
                turn = p1_name


    #If the users want to play the game again?
    if replay():
        #if Yes;
        continue
    else:
        #if No;
        break


#################################################################

print("\n\n\t\t\tTHE END!")
```

22. Time to load website

This script takes a url from the user and returns the time taken to load that website.

How to use this ?

1. Just type the following on the command prompt:

python time_to_load_website.py

2. It will reuest you to provide a url. Provide the url and hit enter to see the script in action.

Sample use:

    <img src="sample.PNG" alt="sample">

Source Code:

```
from urllib.request import urlopen
```

```
import time


def get_load_time(url):
    """This function takes a user defined url as input
    and returns the time taken to load that url in seconds.

    Args:
        url (string): The user defined url.

    Returns:
        time_to_load (float): The time taken to load the website in seconds.
    """

    if ("https" or "http") in url:  # Checking for presence of protocols
        open_this_url = urlopen(url)  # Open the url as entered by the user
    else:
        open_this_url = urlopen("https://" + url)  # Adding https to the url
    start_time = time.time()  # Time stamp before the reading of url starts
    open_this_url.read()  # Reading the user defined url
    end_time = time.time()  # Time stamp after the reading of the url
    open_this_url.close()  # Closing the instance of the urlopen object
    time_to_load = end_time - start_time

    return time_to_load
```

```
if __name__ == '__main__':
    url = input("Enter the url whose loading time you want to check: ")
    print(f"\nThe time taken to load {url} is {get_load_time(url):.2} seconds.")
```

23. Todo App using flask

Perform Operation like

1. Add Task

2. Delete Task

3. Update Task

To run app

- Create virtual Environment

- Install requirements

`pip install requirements.txt`

- run app

`py app.py`

Requirements

Flask==1.1.2

Flask-SQLAlchemy==2.4.4

Source code:

```
from flask import Flask, render_template, url_for, request, redirect
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
```

```
app = Flask(__name__)
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///test.db"
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
db = SQLAlchemy(app)



class Todo(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.String(200), nullable=False)
    completed = db.Column(db.Integer, default=0)
    pub_date = db.Column(db.DateTime, nullable=False,
default=datetime.utcnow)

    def __repr__(self):
        return "<Task %r>" % self.id



@app.route("/", methods=["POST", "GET"])
def index():
    if request.method == "POST":
        task_content = request.form["task"]
        new_task = Todo(content=task_content)
        try:
            db.session.add(new_task)
            db.session.commit()
            return redirect("/")
```

```
        except:
            return "There is an issue"
    else:
        tasks = Todo.query.order_by(Todo.pub_date).all()
        return render_template("index.html", tasks=tasks)


@app.route("/delete/<int:id>")
def delete(id):
    task = Todo.query.get_or_404(id)
    try:
        db.session.delete(task)
        db.session.commit()
        return redirect("/")
    except:
        return "This is an Problem while deleting"


@app.route("/update/<int:id>", methods=["POST", "GET"])
def update(id):
    task = Todo.query.get_or_404(id)
    if request.method == "POST":
        task.content = request.form["task"]

        try:
```

```
            db.session.commit()
            return redirect("/")
        except:
            return "There is an issue"
    else:
        tasks = Todo.query.order_by(Todo.pub_date).all()

        return render_template("index.html", update_task=task, tasks=tasks)


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

1. Twitter Scrapper Without API

Tweet hashtag based scraper without Twitter API

- Here, we make use of snscrape to scrape tweets associated with a particular hashtag. Snscrape is a python library that scrapes twitter without the use of API keys.

- We have 2 scripts associated with this project one to fetch tweets with snscrape and store it in the database (we use SQLite3), and the other script displays the tweets from the database.

- Using snscrape, we are storing the hashtag, the tweet content, user id, as well as the URL of the tweets in the database.

Requirements

Packages associated can be installed as:

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

Running the script

For running the script which fetches tweets and other info associated with the hashtag and storing in the database:

```sh
  $ python fetch_hashtags.py
```

For running the script to display the tweet info stored in the database:

```sh
  $ python display_hashtags.py
```

Requirements:

beautifulsoup4==4.9.3

certifi==2020.12.5

chardet==4.0.0

idna==2.10

lxml==4.6.2

PySocks==1.7.1

requests==2.25.1

snscrape==0.3.4

soupsieve==2.2

urllib3==1.26.4

Source Code:

25. Typing Speed Test

```
import time
string = "Python is an interpreted, high-level programming language"
word_count = len(string.split())
border = '-+-'*10

def createbox():
    print(border)
    print()
    print('Enter the phrase as fast as possible and with accuracy')
    print()

while  1:
    t0 = time.time()
    createbox()
    print(string,'\n')
    inputText = str(input())
    t1 = time.time()
    lengthOfInput = len(inputText.split())
    accuracy = len(set(inputText.split()) & set(string.split()))
    accuracy = (accuracy/word_count)
    timeTaken = (t1 - t0)
```

```
wordsperminute = (lengthOfInput/timeTaken)*60
#Showing results now
print('Total words \t :' ,lengthOfInput)
print('Time used \t :',round(timeTaken,2),'seconds')
print('Your accuracy \t :',round(accuracy,3)*100,'%')
print('Speed is \t :' , round(wordsperminute,2),'words per minute')
print("Do you want to retry",end='')
if input():
    continue
else:
    print('Thank you , bye bye .')
    time.sleep(1.5)
    break
```

26. Instagram Unfollower Bot

bb8 - Your Personal bot

`bb8` is a cute name for a great bot to check for the people that you follow who don't follow you back on Instagram.

How to run

* Install the latest chrome driver and place it in 'C:\Program Files (x86)\chromedriver.exe'. You can download it

  from [here](https://chromedriver.chromium.org/)

* Run the script, enter your username and password for the instagram account.

* That's it. The terminal will soon return you a list of all the accounts that you follow, which don't follow you back.

Side Note

Do remember to download the dependencies in the requirements.txt file!

Modules used

* selenium

Development status

This bot is currently working. However, changes on the Instagram frontend may require

this script to be edited.

``

Source Code:

```
from selenium import webdriver
from getpass import getpass
import time


# Class for the bot



class InstaBot:


    # Initializes bot
    def __init__(self):
        self.username = input('Enter your username:')
        self.pw = getpass('Enter your password(will NOT appear as you type):')
        self.PATH = r"C:\Program Files (x86)\chromedriver.exe"
        self.driver = webdriver.Chrome(self.PATH)


    # Starts Instagram
    def start(self):
        self.driver.get('https://www.instagram.com/')
        time.sleep(2)
```

```
    return

# Logs into your account, also closes various dialogue boxes that open on
# the way
def login(self):

    user_field = self.driver.find_element_by_xpath(
        '//*[@id="loginForm"]/div/div[1]/div/label/input')
    pw_field = self.driver.find_element_by_xpath(
        '//*[@id="loginForm"]/div/div[2]/div/label/input')
    login_button = self.driver.find_element_by_xpath(
        '//*[@id="loginForm"]/div/div[3]/button/div')
    user_field.send_keys(self.username)
    pw_field.send_keys(self.pw)
    login_button.click()
    time.sleep(2.5)
    not_now1 = self.driver.find_element_by_xpath(
        '//*[@id="react-root"]/section/main/div/div/div/div/button')
    not_now1.click()
    time.sleep(2)
    not_now2 = self.driver.find_element_by_xpath(
        '/html/body/div[4]/div/div/div/div[3]/button[2]')
    not_now2.click()
    time.sleep(1)
    return
```

```
# Opens your profile
def open_profile(self):
    profile_link = self.driver.find_element_by_xpath(
        '//*[@id="react-root"]/section/main/section/div[3]'
        '/div[1]/div/div[2]/div[1]/a')
    profile_link.click()
    time.sleep(2)
    return

# Opens the list of the people you follow
def open_following(self):
    following_link = self.driver.find_element_by_xpath(
        '/html/body/div[1]/section/main/div/header/section/ul/li[3]/a')
    following_link.click()
    return

# Gets the list of the people you follow
def get_following(self):
    xpath = '/html/body/div[4]/div/div/div[2]'
    self.following = self.scroll_list(xpath)
    return

# Opens the link to 'Followers'
def open_followers(self):
```

```
    followers_link = self.driver.find_element_by_xpath(
        '//*[@id="react-root"]/section/main/div/header/section/ul/li[2]/a')
    followers_link.click()
    return

# Gets the list of followers
def get_followers(self):
    xpath = '/html/body/div[4]/div/div/div[2]'
    self.followers = self.scroll_list(xpath)
    return

# Scrolls a scroll box and retrieves their names
def scroll_list(self, xpath):

    time.sleep(2)
    scroll_box = self.driver.find_element_by_xpath(xpath)
    last_ht, ht = 0, 1

    # Keep scrolling till you can't go down any further
    while last_ht != ht:
        last_ht = ht
        time.sleep(1)
        ht = self.driver.execute_script(
            """
            arguments[0].scrollTo(0, arguments[0].scrollHeight);
```

```
            return arguments[0].scrollHeight;
            """, scroll_box)

    # Gets the list of accounts
    links = scroll_box.find_elements_by_tag_name('a')
    names = [name.text for name in links if name.text != '']

    # Closes the box
    close_btn = self.driver.find_element_by_xpath(
        '/html/body/div[4]/div/div/div[1]/div/div[2]/button/div')
    close_btn.click()

    return names

# Prints the list of people you follow who don't follow you back in
# terminal
def get_unfollowers(self):

    self.unfollowers = [
        x for x in self.following if x not in self.followers
    ]
    for name in self.unfollowers:
        print(name)
    return
```

```
    # Closes the driver
    def close(self):
        self.driver.quit()
        return


def main():

    # Bot method calls
    bb8 = InstaBot()

    bb8.start()

    bb8.login()

    bb8.open_profile()
    bb8.open_following()

    bb8.get_following()
    bb8.open_followers()

    bb8.get_followers()
    bb8.get_unfollowers()
```

```
    bb8.close()


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

27. Unique words in text file

Script to display unique words in a given text file.

Source Code:

```
import re

# script to fetch unique sorted words from a text file.
list_of_words = []

# Alternate Method to insert file
# filename = input("Enter file name: ")
filename = "text_file.txt"

with open(filename, "r") as f:
    for line in f:
        # if case is ignored then Great and great are same words
        list_of_words.extend(re.findall(r"[\w]+", line.lower()))
        # else use this alternate method:
        # list_of_words.extend(re.findall(r"[\w]+", line))


# Creating a dictionary to store the number of occurence of a word
unique = {}
```

```
for each in list_of_words:
    if each not in unique:
        unique[each] = 0
    unique[each] += 1

# Creating a list to sort the final unique words
s = []

# If occurence of a word(val) is 1 then it is unique
for key, val in unique.items():
    if val == 1:
        s.append(key)

print(sorted(s))
```

28. Unstructured Supplementary Service Data

Unstructured Supplementary Service Data (USSD), sometimes referred to as "Quick Codes" or "Feature codes", is a communications protocol used by GSM cellular telephones to communicate with the mobile network operator's computers. USSD can be used for WAP browsing, prepaid callback service, mobile-money services, location-based content services, menu-based information services, and as part of configuring the phone on the network

MODULES REQUIRED

1. random

2.time

3. sys

EXECUTION PROCESS

1. fork code

2. git clone SSH

3. open on device using a python IDE

4. run the script

Source Code:

```
import time
import sys

print('Welcome To fastrack USSD Banking Project...')
time.sleep(8)

bank_list="""
1. Access Bank
2. Fidelity Bank
3. Guarantee Trust Bank
4. Heritage Bank
5. Polaris Bank
6. Stanbic IBTC
7. Unity Bank
8. Wema Bank
"""

gen_bvn = " "

def BVN_checker( ):
    global gen_bvn
    bvn = [str(i) for i in range (5)]
    gen_bvn= "".join(bvn)
```

```
def open_acct( ):
        global gen_bvn
        print("Welcome to our online Account opening services.")
        print("loading...")
# creating an empty list to serve as a temporary place holder.
        temp_storage= [ ]
        f_name= input("Enter your first name:")
        s_name= input ("Enter your second name:")
        sex = input("Enter sex [M/F]:")
        BVN_checker( )
        temp_storage.append(f_name)
        temp_storage.append(s_name)
        temp_storage.append(sex)
        temp_storage.append(gen_bvn)
        details= " ".join(temp_storage)
        split_details = details.split(" ")
        #print(split_details)
        print(split_details[0]+" "+split_details[1])
        print(split_details[2])
        print("Your bvn is :"+split_details[3])
        print("1. Press # to go back to options menu\n2. Press * to exit")
        bck=input(":")
        if bck=='#':
            options_menu( )
        else:
```

```
            sys.exit( )
        exit( )
def upgrade_migrate( ):
        print("Welcome to our online Upgrade/Migration services.\n 1. Ugrade\n 2. Migrate")
        print("press # is go back to the Main Menu.")
        prompt = input("Enter preferred Choice:")
        if prompt=="1":
        time.sleep(5)
        print("Upgrading...")
        exit( )
        elif prompt == "2":
        time.sleep(5)
        print("Migrating...")
        exit( )
        elif prompt == "#":
        options_menu( )
        else:
        sys.exit( )


def balance ( ):
        print("ACCOUNT\tBALANCE\n CHECKER")
        print("press # is go back to the Main Menu.")
        pin=input("Enter your 4 digit pin:")
# isdigit( ) is used to  check for digits within a str while the nested if is used to make sure the user inputs 4 digits.
```

###```i am to put the pin trial in a while loop```###REMINDER!!!

```
        if len(pin)!=4:
                print("Make sure its a 4digit pin.")
                time.sleep(5)
                balance( )
        else:
                        if pin.isdigit( ):
                           time.sleep(5)
                           print("Loading...")
                           exit( )
                        elif pin== "#":
                           options_menu( )
                        else:
                           time.sleep(15)
                           print("wrong pin")
                           sys.exit( )
def transf( ):
        print("1. Transfer self\n2. Transfer others")
        print("press # is go back to the Main Menu.")
        trnsf=input(":")
        if trnsf == "#" :
           options_menu( )
        elif trnsf == "1":
```

```
            time.sleep(5)
            print("Sending...")
            exit( )
        elif trnsf=="2":
            time.sleep(5)
            num=int(input("Enter receivers mobile number:"))
            print("Transferring to",num)
            exit( )
        else:
            if trnsf.isdigit( )!= True:
                time.sleep(5)
                print("Not an option")
                sys.exit( )
            elif trnsf.isdigit( ) and len(trnsf)>2:
                time.sleep( 5)
                print("wrong password.")
                sys.exit( )
            else:
                time.sleep(10)
                print("An error has occurred")
                sys.exit( )

def funds( ):
        time.sleep(3)
        print(bank_list)
```

```
        bnk = input("Select receipients Bank:")
        acc_num= input("Entet account number:")
        print("Sending to",acc_num)
        hash= input("1.Press # to go back to options menu\n2. Press * to go
exit.")
        if hash == "#":
           options_menu( )
        elif hash == "*":
            exit( )
        else:
                                sys.exit( )



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

###i'm yet to catch an error for non -digit and more than one
digit###REMINDER!!!
#-#-------------------------------------------------

# This is the function for options.
def options_menu( ) :
        print("1. Open Account\n2. Upgrade/Migrate\n3. Balance\n4.
Transfer\n5. Funds")
        select_options ={
          '1':open_acct,
          '2':upgrade_migrate,
          '3': balance,
```

```
        '4':transf,
        '5':funds}
    choice=input("Enter an option:")
    if select_options.get(choice):
      select_options[choice]()
    else:
    sys.exit()
# This is the function which prompts the user as to whether the user wishes to continue or stop transaction.
def exit( ):
    exit= input("Do you wish to make another transaction [Y/N] :")
    if exit== "N":
    sys.exit( )
    elif exit == "#":
       options_menu( )
    else:
    log_in( )
# This is the function for logging using the fast code
*919#
def log_in( ):
  try:
    a=0
    while a<3:
      a+=1
      USSD=input("ENTER USSD:")
      if(USSD !="*919#"):
```

```
            print("please re-enter USSD ...")
        else:
            print("Welcome to our online services how may we help you")
            options_menu( )
            exit( )
    else:
     time.sleep(10)
     print("checking discrepancies...")
     time.sleep(5)
     print("An error has occured.")

  except:
            sys.exit( )

log_in( )
```

29. Unzip File

Unzip File Functionalities:

- Upload the zip file which is to be unzipped
- Then the script will return all the unzipped files into the Unzip files folder

Unzip File Instructions:

Step 1:

Open Termnial

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

- zipfile

Source Code:

```
import zipfile

target = input(r"Enter file to be unzipped: ")
handle = zipfile.ZipFile(target)
handle.extractall("./Unzip file/Unzip files")
handle.close()
```

30. URL Shortner

```
from __future__ import with_statement
import contextlib
from urllib.parse import urlencode
from urllib import urlencode
from urllib.request import urlopen
from urllib2 import urlopen
import sys


def short_url(url):
    request_url = ('http://tinyurl.com/api-create.php?' +
urlencode({'url':url}))
    with contextlib.closing(urlopen(request_url)) as response:
        return response.read().decode('utf-8 ')
def main():
    for url in map(short_url, sys.argv[1:]):
        print(url)


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

Module 4 Projects 31-40

31. Video To Audio Converter in python

```
from pytube import YouTube
import pytube
import os

def main():
    video_url = input('Enter YouTube video URL: ')

    if os.name == 'nt':
        path = os.getcwd() + '\\'
    else:
        path = os.getcwd() + '/'

    name = pytube.extract.video_id(video_url)

YouTube(video_url).streams.filter(only_audio=True).first().download(filenam
    location = path + name + '.mp4'
    renametomp3 = path + name + '.mp3'

    if os.name == 'nt':
        os.system('ren {0} {1}'. format(location, renametomp3))
    else:
```

```
        os.system('mv {0} {1}'. format(location, renametomp3))

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

32. Voice Translators

Dependencies:

Google Translate
```python

pip install googletrans
```
pyttsx3
```python

pip install pyttsx3
```
pyaudio
```python
pip install pyaudio
```
speech recongnition
```python
pip install SpeechRecognition
```

Source Code:

```
from googletrans import Translator
import pyttsx3
import speech_recognition as sr

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

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

def takeCommand():

    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
  audio = r.listen(source)

try:
  print("Recognizing...")
  query = r.recognize_google(audio, language='en-in')
  print(f"AK47 Said:{query}\n")
```

```
    except Exception as e:
        print(e)
        print("Say that again Please...")
        speak("Say that again Please...")
        return "None"
    return query

def Translate():
        speak("what I should Translate??")
        sentence = takeCommand()
        trans = Translator()




        trans_sen = trans.translate(sentence,src='en',dest='ca')
        print(trans_sen.text)
        speak(trans_sen.text)

Translate()
```

33. Hashing Passwords

Wallpaper-Changer-using-Python

Dependencies:

Get Your API HERE :- [Unsplash](https://unsplash.com/developers)

Wget

```python

pip install wget
```

Add API KEY in Wallpapers.py file:

```

access_key = '' # add your unspash api key here
```

Run:

```
python wallpapers.py
```

Source Code:

```
# Get the wallpaper from the internet
# Save it to a temp directory
# Set the wallpaper
# Automate the calls to this script

import os
import requests
import wget
import subprocess
import time
import ctypes
SPI_SETDESKWALLPAPER = 20


def get_wallpaper():
        access_key = '' # add your unspash api key here
        url = 'https://api.unsplash.com/photos/random?client_id=' + access_key
        params = {
        'query': 'HD wallpapers',
```

```
        'orientation': 'landscape'
        }

        response = requests.get(url, params=params).json()
        image_source = response['urls']['full']

        image = wget.download(image_source,
'C:/Users/projects/wallpaper.jpg') # add the path here
        return image

def change_wallpaper():
        wallpaper = get_wallpaper()
        ctypes.windll.user32.SystemParametersInfoW(SPI_SETDESKWALLPA
0, "C:\\Users\\projects\\wallpaper.jpg" , 0) # add the path  here as well

def main():
        try:
        while True:
        change_wallpaper()
        time.sleep(10)

        except KeyboardInterrupt:
        print("\nHope you like this one! Quitting.")
        except Exception as e:
        pass
if __name__ == "__main__":
```

main()

34. Weather App

```
# import all functions from the tkinter
from tkinter import *
from tkinter import messagebox
def tell_weather() :
        import requests, json
        api_key = "api_key"
        base_url = "http://api.openweathermap.org/data/2.5/weather?"
        city_name = city_field.get()
        complete_url = base_url + "appid =" + api_key + "&q =" + city_name
        response = requests.get(complete_url)
        x = response.json()
        if x["cod"] != "404" :
        y = x["main"]
        current_temperature = y["temp"]
        current_pressure = y["pressure"]
        current_humidiy = y["humidity"]
        z = x["weather"]
        weather_description = z[0]["description"]
        temp_field.insert(15, str(current_temperature) + " Kelvin")
        atm_field.insert(10, str(current_pressure) + " hPa")
        humid_field.insert(15, str(current_humidiy) + " %")
        desc_field.insert(10, str(weather_description) )
```

```
        else :
        messagebox.showerror("Error", "City Not Found \n"
        "Please enter valid city name")
        city_field.delete(0, END)



def clear_all() :
        city_field.delete(0, END)
        temp_field.delete(0, END)
        atm_field.delete(0, END)
        humid_field.delete(0, END)
        desc_field.delete(0, END)
        city_field.focus_set()

if __name__ == "__main__" :
        root = Tk()
        root.title("Weather Application")

        # Set the background colour of GUI window
        root.configure(background = "light blue")

        # Set the configuration of GUI window
        root.geometry("425x175")
```

```
        # Create a Weather Gui Application label
        headlabel = Label(root, text = "Weather Gui Application", fg = 'white',
bg = 'Black')
        # Create a City name : label
        label1 = Label(root, text = "City name : ", fg = 'white', bg = 'dark gray')
        # Create a City name : label
        label2 = Label(root, text = "Temperature :", fg = 'white', bg = 'dark
gray')

        # Create a atm pressure : label
        label3 = Label(root, text = "atm pressure :", fg = 'white', bg = 'dark
gray')

        # Create a humidity : label
        label4 = Label(root, text = "humidity :", fg = 'white', bg = 'dark gray')

        # Create a description :label
        label5 = Label(root, text = "description :", fg = 'white', bg = 'dark gray')
        headlabel.grid(row = 0, column = 1)
        label1.grid(row = 1, column = 0, sticky ="E")
        label2.grid(row = 3, column = 0, sticky ="E")
        label3.grid(row = 4, column = 0, sticky ="E")
        label4.grid(row = 5, column = 0, sticky ="E")
        label5.grid(row = 6, column = 0, sticky ="E")
```

```
    city_field = Entry(root)
    temp_field = Entry(root)
    atm_field = Entry(root)
    humid_field = Entry(root)
    desc_field = Entry(root)

    city_field.grid(row = 1, column = 1, ipadx ="100")
    temp_field.grid(row = 3, column = 1, ipadx ="100")
    atm_field.grid(row = 4, column = 1, ipadx ="100")
    humid_field.grid(row = 5, column = 1, ipadx ="100")
    desc_field.grid(row = 6, column = 1, ipadx ="100")

    button1 = Button(root, text = "Submit", bg = "pink", fg = "black",
command = tell_weather)

    button2 = Button(root, text = "Clear", bg = "pink" , fg = "black",
command = clear_all)
    button1.grid(row = 2, column = 1)
    button2.grid(row = 7, column = 1)
    # Start the GUI
    root.mainloop()
```

35. Website Summarization API

This project is carried out for the purpose of building a machine learning model for summarising a website from urls;

Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

Prerequisites

Python distribution

```
Anaconda
```

Installing

Install Anaconda python distribution on your system

Create a virtual environment called env.

```
python -m venv app
```

Activate the virtual environment

```
LINUX/Mac: source app/bin/activate

Windows: app\Scripts\activate
```

Upgrade to the latest pip

```
pip install --upgrade pip
```

Install dependencies using requirements file

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

Note: Your virtual environment must always be activated before running any command

Deployment

Start app (Make sure to enter a valid website to an existing website)

Example of valid commands

```
python app.py simple --url https://facebook.com --sentence 1 --language english
python app.py simple --url https://facebook.com
python app.py simple --url https://korapay.com
python app.py bulk --path ./csv/valid_websites.csv
```

APIs

This are command options in full:

```
A command line utility for website Summarization.
---------------------------------------------
These are common commands for this app.

positional arguments:
```

action This has to be 'summarize'

optional arguments:

-h, --help show this help message and exit

--website PATH website of the url to be summarised

Requirements:

utils==1.0.1

sumeval==0.2.2

tensorflow==2.3.0

wget==3.2

sumy==0.8.1

model==0.6.0

numpy==1.19.1

newspaper==0.1.0.7

nltk==3.5

gensim==3.8.3

Source Code:

```
#!/usr/bin/python
from utils.summarize import summarize
import csv
import shutil
import os
```

```
    import textwrap
    import logging
    import argparse
    import sys


    def parse_args(argv):
        parser = argparse.ArgumentParser(
            formatter_class=argparse.RawDescriptionHelpFormatter,
    description=textwrap.dedent('''\
        A command line utility for website summarization.
        ---------------------------------------------
        These are common commands for this app.'''))
parser.add_argument(
    'action',
    help='This action should be summarize')
parser.add_argument(
    '--url',
    help='A link to the website url'
)
parser.add_argument(
    '--sentence',
    help='Argument to define number of sentence for the summary',
    type=int,
    default=2)
```

```
    parser.add_argument(
        '--language',
        help='Argument to define language of the summary',
        default='English')
    parser.add_argument(
        '--path',
        help='path to csv file')

    return parser.parse_args(argv[1:])


def readCsv(path):
    print('\n\n Processing Csv file \n\n')
    sys.stdout.flush()
    data = []
    try:
        with open(path, 'r') as userFile:
            userFileReader = csv.reader(userFile)
            for row in userFileReader:
                data.append(row)
    except:
        with open(path, 'r', encoding="mbcs") as userFile:
            userFileReader = csv.reader(userFile)
            for row in userFileReader:
                data.append(row)
```

```
    return data


def writeCsv(data, LANGUAGE, SENTENCES_COUNT):
    print('\n\n Updating Csv file \n\n')
    sys.stdout.flush()
    with open('beneficiary.csv', 'w') as newFile:
        newFileWriter = csv.writer(newFile)
        length = len(data)
        position = data[0].index('website')
        for i in range(1, length):
            if i == 1:
                _data = data[0]
                _data.append("summary")
                newFileWriter.writerow(_data)
            try:
                __data = data[i]
                summary = summarize(
                    (data[i][position]), LANGUAGE, SENTENCES_COUNT)
                __data.append(summary)
                newFileWriter.writerow(__data)
            except:
                print('\n\n Error Skipping line \n\n')
                sys.stdout.flush()
```

```
def processCsv(path, LANGUAGE, SENTENCES_COUNT):
    try:
        print('\n\n Proessing Started \n\n')
        sys.stdout.flush()
        data = readCsv(path)
        writeCsv(data, LANGUAGE, SENTENCES_COUNT)
    except:
        print('\n\n Invalid file in file path \n\n')
        sys.stdout.flush()



def main(argv=sys.argv):
        # Configure logging
    logging.basicConfig(filename='applog.log',
                        filemode='w',
                        level=logging.INFO,
                        format='%(levelname)s:%(message)s')
    args = parse_args(argv)
    action = args.action
    url = args.url
    path = args.path
    LANGUAGE = "english" if args.language is None else args.language
    SENTENCES_COUNT = 2 if args.sentence is None else args.sentence
    if action == 'bulk':
```

```
    if path is None:
        print(
            '\n\n Invalid Entry!, please Ensure you enter a valid file path \n\n')
        sys.stdout.flush()
        return
    # guide against errors
    try:
        processCsv(path, LANGUAGE, SENTENCES_COUNT)
    except:
        print(
            '\n\n Invalid Entry!, please Ensure you enter a valid file path \n\n')
        sys.stdout.flush()
    print('Completed')
    sys.stdout.flush()
    if os.path.isfile('beneficiary.csv'):
        return shutil.move('beneficiary.csv', path)
    return
if action == 'simple':
    # guide against errors
    try:
        summarize(url, LANGUAGE, SENTENCES_COUNT)
    except:
        print(
            '\n\n Invalid Entry!, please Ensure you enter a valid web link \n\n')
        sys.stdout.flush()
```

```
        print('Completed')
        sys.stdout.flush()
    else:
        print(
            '\nAction command is not supported\n for help: run python3 app.py -
h'
        )
        sys.stdout.flush()
        return


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

36. Web Scrapping Comment

- This script will take a url of youtube video and it will give csv file for users and comments .

Prerequisites

- You only need to have installed selenium which is used for automation.
- Run the below script to install selenium
- $ pip install selenium

How to run the script

- Simply replace your own youtube video url in the webscrapindcomment.py
- And run command in the same directory
- python webscrapindcomment.py

Requirements- selenium==3.141.0

Source Code:

```
# -*- coding: utf-8 -*-
```

```
from selenium import webdriver
import csv
import time

items=[]
driver=webdriver.Chrome(r"C:/Users/hp/Anaconda3/chromedriver.exe"

driver.get('https://www.youtube.com/watch?v=iFPMz36std4')

driver.execute_script('window.scrollTo(1, 500);')

#now wait let load the comments
time.sleep(5)

driver.execute_script('window.scrollTo(1, 3000);')



username_elems = driver.find_elements_by_xpath('//*[@id="author-text"]')
comment_elems = driver.find_elements_by_xpath('//*[@id="content-text"]')
for username, comment in zip(username_elems, comment_elems):
    item = {}
    item['Author'] = username.text
```

```
    item['Comment'] = comment.text
    items.append(item)
filename = 'C:/Users/hp/Desktop/commentlist.csv'
with open(filename, 'w', newline='', encoding='utf-8') as f:
    w = csv.DictWriter(f,['Author','Comment'])
    w.writeheader()
    for item in items:
        w.writerow(item)
```

37. Website Blocker

This script lets you block websites on your computer by editing your hosts file.

Usage

First add your Blocked Websites to the array in both scripts.

On Linux: `sudo python website_blocker.py`

On Windows, run the script as Administrator

To unblock the websites, run the `website_unblocker.py` script.

Website block Source Code:

import platform

```
if platform.system() == "Windows":
        pathToHosts=r"C:\Windows\System32\drivers\etc\hosts"
elif platform.system() == "Linux":
        pathToHosts=r"/etc/hosts"

redirect="127.0.0.1"
websites=["https://www.websitename.com"]

with open(pathToHosts,'r+') as file:
    content=file.read()
    for site in websites:
        if site in content:
            pass
        else:
            file.write(redirect+" "+site+"\n")
```

Website Unblock Source Code:

```
        import platform

        if platform.system() == "Windows":
```

```
        pathToHosts=r"C:\Windows\System32\drivers\etc\hosts"
elif platform.system() == "Linux":
        pathToHosts=r"/etc/hosts"

websites=["https://www.websitename.com"]

with open(pathToHosts,'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()
```

38. Whatsapp Bot

Perform Operation like

1. Put your details

2. connect with internet

3. Pass your message

To run app

- Create virtual Environment

- Install requirements

`pip install requirements.txt`

- run app

`python main.py`

Source Code:

```
import pywhatkit
from datetime import datetime
```

```
now = datetime.now()

chour = now.strftime("%H")
mobile = input('Enter Mobile No of Receiver : ')
message = input('Enter Message you wanna send : ')
hour = int(chour) + int(input('Enter hour : '))
minute = int(input('Enter minute : '))

pywhatkit.sendwhatmsg(mobile,message,hour,minute)
```

39. Whatsapp Automation

How to run this Python Script?

1. Install [chromedriver] (https://chromedriver.storage.googleapis.com/index.html?path=2.25/) (choose your specific version)

2. `pip install selenium`

3. Make sure you have added the **correct path** to your chrome driver

4. Enter the name of the person you want to send the message to **exactly the way it is saved.**

5. Type in the message you want to send.

6. You will have **15s** to scan for whatsapp web.

7. Message has been sent.

Source Code:

```
# Selenium is required for automation
# sleep is required to have some time for scanning
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from time import sleep


def whatsapp(to, message):
    person = [to]
    string = message
    chrome_driver_binary = "C:\\Program
Files\\Google\\Chrome\\Application\\chromedriver.exe"
    # Selenium chromedriver path
    driver = webdriver.Chrome(chrome_driver_binary)
    driver.get("https://web.whatsapp.com/")
    sleep(15)
    # This will find the person we want to send the message to in the list
    for name in person:
        user =
driver.find_element_by_xpath("//span[@title='{}']".format(name))
        user.click()
        text_box = driver.find_element_by_xpath(
```

```
            '//*[@id="main"]/footer/div[1]/div[2]/div/div[2]')
        try:
            text_box.send_keys(string)
            sendbutton = driver.find_elements_by_xpath(
                '//*[@id="main"]/footer/div[1]/div[3]/button')[0]
            sendbutton.click()
            sleep(10)
            print('Message Sent!!')
        except:
            print('Error occured....')


if __name__ == "__main__":
    to = input('Who do you want to send a message to? Enter the name: ')
    content = input("What message to you want to send? Enter the message: ")
    whatsapp(to, content)
```

40. Instagram Follow- NotFollow

Send and schedule a message in WhatsApp by only seven lines of Python Script.

Modules Used

- pywhatkit

pip install [requirements.txt]

How it works

- First login your WhatsApp web version by scanning QR Code.

- By just providing the string format (receiver(recipient) Phone number with country code, message you want to send to receiver, schedule time in 24hrs format).

- Then on scheduled time it opens on WhatsApp web on your default browser and sends your message to the receiver phone number.

Requirements - pywhatkit

Source Code:

"""

WhatsApp Auto Messenger

- Send message to your friend or group by using just 7 lines of Python Script

"""

```
import pywhatkit
phoneno = input("Enter Receiver(recipient) Phone Number :")
message = input("Enter Message You want to send :")
print("Enter Schedule Time to send WhatsApp message to recipient :")
Time_hrs = int(input("- At What Hour :"))
Time_min = int(input("- At What Minutes :"))
pywhatkit.sendwhatmsg(phoneno, message, Time_hrs, Time_min)
```

Tip : Do you want to send and schedule a messages to any WhatsApp group then use below code and provide inside attributes value.

pywhatkit.sendwhatmsg_to_group(GroupID, message, time_hour, time_min, wait_time)

Note : Group ID is something that is in its invite link,

Module 5 Projects 41-50

41. Wikipedia infobox scraper

- The given python script uses beautifulSoup to scrape Wikipedia pages according to the given user query and obtain data from its wikipedia infobox.

Requirements:

```
$ 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:

```
from bs4 import BeautifulSoup
import requests
from tkinter import *

info_dict = {}
```

```
def error_box():
    """
    A function to create a pop-up, in case the code errors out
    """
    global mini_pop

    mini_pop = Toplevel()
    mini_pop.title('Error screen')

    mini_l = Label(mini_pop, text=" !!!\nERROR FETCHING DATA",
fg='red', font=('Arial',10,'bold'))
    mini_l.grid(row=1, column=1, sticky='nsew')
    entry_str.set("")



def wikiScraper():
    """
    Function scrapes the infobox lying under the right tags and displays
    the data obtained from it in a new window
    """
    global info_dict

    # Modifying the user input to make it suitable for the URL
    entry = entry_str.get()
    entry = entry.split()
```

```
query = '_'.join([i.capitalize() for i in entry])
req = requests.get('https://en.wikipedia.org/wiki/'+query)

# to check for valid URL
if req.status_code == 200:
    # for parsing through the html text
    soup = BeautifulSoup(req.text, 'html.parser')

    # Finding text within infobox and storing it in a dictionary
    info_table = soup.find('table', {'class': 'infobox'})

    try:
        for tr in info_table.find_all('tr'):
            try:
                if tr.find('th'):
                    info_dict[tr.find('th').text] = tr.find('td').text
            except:
                pass

    except:
        error_box()

    # Creating a pop up window to show the results
    global popup
    popup = Toplevel()
```

```
        popup.title(query)

        r = 1

        for k, v in info_dict.items():
            e1 = Label(popup, text=k+" : ", bg='cyan4', font=('Arial',10,'bold'))
            e1.grid(row=r, column=1, sticky='nsew')

            e2 = Label(popup, text=info_dict[k], bg="cyan2", font=('Arial',10,
'bold'))
            e2.grid(row=r, column=2, sticky='nsew')

            r += 1
            e3 = Label(popup, text="", font=('Arial',10,'bold'))
            e3.grid(row=r, sticky='s')
            r += 1

        entry_str.set("")
        info_dict = {}

    else:
        print('Invalid URL')
        error_box()


# Creating a window to take user search queries
```

```
root = Tk()
root.title('Wikipedia Infobox')

global entry_str
entry_str = StringVar()

search_label = LabelFrame(root, text="Search: ", font = ('Century Schoolbook L',17))
search_label.pack(pady=10, padx=10)

user_entry = Entry(search_label, textvariable = entry_str, font = ('Century Schoolbook L',17))
user_entry.pack(pady=10, padx=10)

button_frame = Frame(root)
button_frame.pack(pady=10)

submit_bt = Button(button_frame, text = 'Submit', command = wikiScraper, font = ('Century Schoolbook L',17))
submit_bt.grid(row=0, column=0)

root.mainloop()
```

42. Wikipedia Scrapper in Python

```
import wikipedia as wiki

print(wiki.search("Python"))
print(wiki.suggest("Pyth"))
print(wiki.summary("Python"))

wiki.set_lang("fr")
print(wiki.summary("Python"))

wiki.set_lang("en")
p = wiki.page("Python")

#To get the Title
print(p.title)

#To get the url of the article
print(p.url)

#To scrape the full article
print(p.content)

#To get all the images in the article
print(p.images)
```

```
#And to get all the referals used by Wikipedia in the article
print(p.links)
```

43. Instagram Image download

Wordcloud Images for Wikipedia Article

Python script that prompts the user for an input, searches for the corresponding article on wikipedia and generates a wordcloud based on the searched article.

Prerequisites

`pip install` the models in `requirements.txt` from your command prompt.

How to run the script

Run like any other python file. Upon executing, the wordcloud image will be saved to the current directory. The script will also prompt a y/n if the user wants to see the generated image during execution.

![script execution](script_execution.jpg)

Requirement:

beautifulsoup4==4.9.1

certifi==2020.6.20

chardet==3.0.4

cycler==0.10.0

idna==2.10

kiwisolver==1.2.0

matplotlib==3.3.1

numpy==1.19.1

Pillow==7.2.0

pyparsing==2.4.7

python-dateutil==2.8.1

requests==2.24.0

six==1.15.0

soupsieve==2.0.1

urllib3==1.25.10

wikipedia==1.4.0

wordcloud==1.8.0

Source Code:

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
import wikipedia
import sys
```

```
import warnings
# supressing unnecessary warnings
warnings.filterwarnings("ignore")


# function to search the wikipedia article and generate the wordcloud
def gen_cloud(topic):
    try:
        content = str(wikipedia.page(topic).content)
    except:
        print("Error, try searching something else...")
        sys.exit()
    STOPWORDS.add('==')
    stopwords = set(STOPWORDS)
    wordcloud = WordCloud(stopwords=stopwords, max_words=200,
background_color="black", width=600, height=350).generate(content)
    return wordcloud


# function to save the wordcloud to current directory
def save_cloud(wordcloud):
    wordcloud.to_file("./wordcloud.png")


# function to display the wordcloud with matplotlib
def show_cloud(wordcloud):
```

```
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()


# driver code
if __name__ == '__main__':
    topic = input("What do you want to search: ").strip()
    wordcloud = gen_cloud(topic)
    save_cloud(wordcloud)
    print("Wordcloud saved to current directory as wordcloud.png")
    desc = input("Do you wish to see the output(y/n): ")
    if desc == 'y':
        show_cloud(wordcloud)
    sys.exit()
```

44. Wikipedia summary script with GUI

Running this Script would open up a wikipedia summary generator GUI which can be used to get summary about any topic of the user's choice from wikipedia

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
```

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

```
python summary.py
```

or

```
python3 summary.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:

Pymediawiki

Source Code:

```
from tkinter import Tk, Frame, Toplevel, Entry, Button, Text, Scrollbar, END, INSERT
from tkinter.messagebox import showerror
from mediawiki import MediaWiki
wikipedia = MediaWiki()

# Function to get summary using wikipedia module and display it


def get_summary():
    try:
        # clear text area
        answer.delete(1.0, END)
        # show summary in text area
        topic = keyword_entry.get()
        p = wikipedia.page(topic)
        answer.insert(INSERT, p.summary)
    except Exception as error:
        showerror("Error", error)
```

```
# create a GUI window and configure it
root = Tk()
root.title("Wikipedia Summary")
root.geometry("770x650")
root.resizable(False, False)
root.configure(bg="dark grey")

# create a frame for entry and button
top_frame = Frame(root, bg="dark grey")
top_frame.pack(side="top", fill="x", padx=50, pady=10)

# create a frame for text area where summary will be displayed
bottom_frame = Frame(root, bg="dark grey")
bottom_frame.pack(side="top", fill="x", padx=10, pady=10)

# create a entry box where user can enter a keyword
keyword_entry = Entry(top_frame, font=("Arial", 20, "bold"), width=25,
bd=4)
keyword_entry.pack(side="left", ipady=6)

# create a search button
search_button = Button(top_frame, text="Get Summary", font=(
    "Arial", 16, "bold"), width=15, bd=4, command=get_summary)
search_button.pack(side="right")
```

```
# create a scroll bar for text area
scroll = Scrollbar(bottom_frame)

# create a text area where summary will be displayed
answer = Text(bottom_frame, font=("Arial", 18), fg="black",
              width=55, height=20, bd=5, yscrollcommand=scroll.set)
answer.pack(side="left", fill="y")
scroll.pack(side="left", fill="y")

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

45. Word Games

```
import sys
list1=['a','b','c','d','e','f','g','h','i','j','k','l','m']
list2=['n','o','p','q','r','s','t','u','v','w','x','y','z']
w=input("Enter a word ")
prepartner=prepartner1=[]
postpartner1=postpartner=[]
for i in w:
    if(i in list1):
        prepartner.append(i)
    if(i in list2):
        postpartner.append(i)
for j in prepartner:
    list1index=list1.index(j)
    if(list2[list1index] in postpartner):#testing if all prepartners has postpartners
        pass
    else:
        print("YOU LOST")
        sys.exit()
         prepartner1=prepartner
         postpartner1=postpartner
         for k in prepartner:
```

```
    x=prepartner.index(k)
    y=postpartner.index(list2[list1.index(k)])
    if(w.index(prepartner[x])<w.index(postpartner[y])):
        if(w.index(postpartner[y])-w.index(prepartner[x])==1):#testing3a
            prepartner1.pop(x)
            postpartner1.pop(y)
    else:
        print("YOU LOST")
        sys.exit()
postpartner1.reverse()
count=0
for l in prepartner1:

if(prepartner1.index(l)==postpartner1.index(list2[list1.index(l)])):#testir
        count+=1
if(count==len(prepartner1)):
    print("GAME WON")
else:
    print("GAME LOST")
```

46. Worksetup Automation

Dependencies:

pyinstaller
```
pip install pyinstaller
```

ADD PATH FOR TEXT EDITOR OR IDE HERE:

```
codePath = "C:\\Program Files\\Sublime Text 3\\sublime_text.exe"#ADD THE PATH OF TXET EDITOR OR IDE HERE
```

Run:

Convert the python file into .exe
```
pyinstaller -F workstation.py
```

Start-Up Setup

1.PRESS WINDOWS + R to open RUN

2.Type

```
shell:startup
```

3.Copy the exe file to start-up folder

4.Restart the system it should run the exe as soon as system starts

Source Code:

```
import os
import webbrowser as wb

def workstation():
    codePath = "C:\\Program Files\\Sublime Text 3\\sublime_text.exe"#ADD THE PATH OF TXET EDITOR OR IDE HERE
```

```
    os.startfile(codePath)
    chrome_path = 'C:/Program Files (x86)/Google/Chrome/Application/chrome.exe %s'#ADD THE PATH OF CHROME HERE
    URLS = (
        "stackoverflow.com",
        "github.com/ ",
        "gmail.com",
        "google.com",
        "youtube.com"
        )#ADD THE WEBSITES YOU USE WHIE WORKING
    for url in URLS:
    wb.get(chrome_path).open(url)
workstation()
```

47. Set a Random desktop background

This script will download a random image from [unsplash] (https://source.unsplash.com/random) and set it as the desktop background.

The image will be saved as "random.jpg" makesure that there are no files saved as "random.jpg" in the current directory

Requirements

Linux

Install [Nitrogen](https://wiki.archlinux.org/index.php/Nitrogen)

```
pip install requests
```

Usage

```python
python background_linux.py
```

OR

```python
python background_windows.py
```

Source Code:

```
from requests import get
import os
import ctypes
import sys

url = "https://source.unsplash.com/random"
file_name = "random.jpg"

def is_64bit():
    return sys.maxsize > 2 ** 32


def download(url, file_name):
    '''
    downloading the file and saving it
    '''
    with open(file_name, "wb") as file:
        response = get(url)
```

```
        file.write(response.content)



def setup(pathtofile,version):
    name_of_file = pathtofile
    path_to_file = os.path.join(os.getcwd(), name_of_file)
    SPI_SETDESKWALLPAPER = 20
    if is_64bit():

ctypes.windll.user32.SystemParametersInfoW(SPI_SETDESKWALLPAPER
0, path_to_file, 0)
    else:

ctypes.windll.user32.SystemParametersInfoA(SPI_SETDESKWALLPAPER,
0, path_to_file, 0)



if __name__ == "__main__":


   download(url, file_name)
        setup(file_name)
    except Exception as e:
        print(f"Error {e}")
        raise NotImplementedError
```

Source Code file for linux:

48. Compress folder and files

usage

python zipfiles.py file_name(or folder name)

example:

python zipfiles.py test.txt

python zipfiles.py ./test (folder)

A Compressed file("filename.zip") will be generated after the program is run

Source Code:

```
import zipfile
```

```
import sys
import os


# compress file function
def zip_file(file_path):
    compress_file = zipfile.ZipFile(file_path + '.zip', 'w')
    compress_file.write(path, compress_type=zipfile.ZIP_DEFLATED)
    compress_file.close()


# Declare the function to return all file paths of the particular directory
def retrieve_file_paths(dir_name):
    # setup file paths variable
    file_paths = []

    # Read all directory, subdirectories and file lists
    for root, directories, files in os.walk(dir_name):
        for filename in files:
            # Create the full file path by using os module.
            file_path = os.path.join(root, filename)
            file_paths.append(file_path)

    # return all paths
    return file_paths
```

```
def zip_dir(dir_path, file_paths):
    # write files and folders to a zipfile
    compress_dir = zipfile.ZipFile(dir_path + '.zip', 'w')
    with compress_dir:
        # write each file separately
        for file in file_paths:
            compress_dir.write(file)



if __name__ == "__main__":
    path = sys.argv[1]

    if os.path.isdir(path):
        files_path = retrieve_file_paths(path)
        # print the list of files to be zipped
        print('The following list of files will be zipped:')
        for file_name in files_path:
            print(file_name)
        zip_dir(path, files_path)
    elif os.path.isfile(path):
        print('The %s will be zipped:' % path)
        zip_file(path)
    else:
```

```
print('a special file(socket,FIFO,device file), please input file or dir')
```

49. Organize files in a directory

Script to organize files in a directory (alphabetical order)

usage

```python
python main.py
```

Folder will be generated and files will be moved accordingly.

Source Code:

'''

```
    This script will sort and move the files in the directory
    (the alphabetical order).

    'apple.txt' --> 'A'
    'ryan.txt' --> 'R'
    '01010.txt' --> 'Misc'

    '''
    import os
    import shutil

    filenames = []


    def getfoldername(filename):
        '''
        'Test.txt' --> 't'
        '010.txt' --> 'misc'
        'zebra.txt' --> 'z'
'Alpha@@.txt' --> 'a'
'!@#.txt' --> 'misc'
'''
if filename[0].isalpha():
   return filename[0].lower()
else:
```

```
        return 'misc'


def readdirectory():
    '''
    read the filename in the current directory and append them to a list
    '''
    global filenames
    for files in os.listdir(os.getcwd()):
        if os.path.isfile(os.path.join(os.getcwd(), files)):
            filenames.append(files)
    filenames.remove('main.py')  # removing script from the file list


# getting the first letters of the file & creating a file in the current_dir
def createfolder():
    '''
    creating a folders
    '''
    global filenames
    for f in filenames:
        if os.path.isdir(getfoldername(f)):
            print("folder already created")
        else:
            os.mkdir(getfoldername(f))
```

```
            print('creating folder...')


# moving the file into the proper folder
def movetofolder():
    '''
    movetofolder('zebra.py','z')

    'zebra.py'(moved to) 'z'
    '''
    global filenames
    for i in filenames:
        filename = i
        file = getfoldername(i)
        source = os.path.join(os.getcwd(), filename)
        destination = os.path.join(os.getcwd(), file)
        print(f"moving {source} to {destination}")
        shutil.move(source, destination)


if __name__ == '__main__':
    readdirectory()
    createfolder()
    movetofolder()
```

50. Youtube Trending Feed Scrapper

It's a 2 scripts that is used to scrap and read the first 10 trending news in YouTube from any its available categories. Let be What's happening right ``Now``, in ``Gaming``, in ``Music``, or in ``Movies`` You will get it on your local machine.

Installation

* Install the following Python libraries:

> ``pip3 install selenium pymongo mongoengine pandas``

* Place ChromeDriver in the same directory of the script. You can download it from [here] (https://sites.google.com/a/chromium.org/chromedriver/downloads).

(Note: Download the one with the same version of your Chrome browser.)

* Install MongoDB Community Server on your machine. You can refer to the installation from [here] (https://docs.mongodb.com/manual/administration/install-community/).

Usage

The scripts allows you to save the scrapped content using 2 methods:

1) A MongoDB called ``Youtube`` and saved in a collection called

``trending``.

2) A CSV file called ``Youtube.csv``.

You can save using either or both, It's up to your desires. The same goes with ``scrap_reader.py``, It can read from either MongoDB or the CSV file.

* For saving-to/reading-from a MongoDB, pass the ``-m`` argument.
* For saving-to/reading-from a CSV file, pass the ``-c`` argument.

Output

whatever the used argument to save the data is, it will be saved containing these video attributes:

1) Video Section
2) Video Title
3) Video Link
4) Video Channel
5) Video Views
6) Video Date

Source Code Files:

youtube_scrapper.py

scrap_reader.py

51. LinkedIn My Connections Scrapper

It's a script built with the help of Selenium and Pandas to scrap LinkedIn connections list along with the skills of each connection if you want to. Using just a one-line command you can sit back and have a CSV file prepared for your cause.

Installation

Make sure you have the following Python libraries:

> pip3 install selenium pandas

The rest should be present as core Python modules.

Next thing is to place ChromeDriver.exe in the same directory of the script. You can download it from [here] (https://sites.google.com/a/chromium.org/chromedriver/downloads)

(Note: Download the one with the same version of your Chrome browser.)

Usage

For basic use:

> python scrapper.py -e \<email\> -p \<password\>

For scrapping skills:

> python scrapper.py -e \<email\> -p \<password\> -s

Furthur Notes

- The time of script progress depends on the number of connections the account has. For basic use, the script can take a time complexity of O(n^2).
- For skills scraping, the time will rise even more depending on each profile and its contained details.
- The scripts print out a couple of messages to explain in which phase it is.
- efficiency is also affected by Internet speed.

Output

Basic use will output a \"scrap.csv\" file that will contain columns of Name, Headline, & Link. There will be a skills column but it will be empty.

Using the skills scrapper mode will add the skills of each profile to that column, each skill will be " -- " separated.

Source Code:

```
# Linkedin My_Connections Scrapper
from selenium.webdriver.common.action_chains import ActionChains
from optparse import OptionParser
```

```
from selenium import webdriver
import pandas as pd
import time
import sys
import re

pattern_name = "\\n(.+)\\n"  # Used to extract names
pattern_headline = 'occupation\\n(.+)\\n'  # Used to extract headlines

# Help menu
usage = """
<Script> [Options]

[Options]
   -h, --help        Show this help message and exit.
   -e, --email       Enter login email
   -p, --password    Enter login password
  -s, --skills      Flag to scrap each profile, and look at its skill set
```

Operation Modes:

> Basic mode

This will scrap all LinkedIn connections list with there corresponding Name, Headline, and Profile link.

> Skills scrapper mode (-s/--skills)

(Time Consuming mode)

This will do the same job of basic mode but along with visiting each

```
profile and extracting the skills of each.
"""

# Load args
parser = OptionParser()
parser.add_option("-e", "--email", dest="email", help="Enter login email")
parser.add_option("-p", "--password", dest="password",
                  help="Enter login password")
parser.add_option("-s", "--skills", action="store_true", dest="skills",
                  help="Flag to scrap each profile, and look at its skill set")


def login(email, password):
    """LinkedIn automated login function"""
            # Get LinkedIn login page
            driver = webdriver.Chrome("chromedriver.exe")
            driver.get("https://www.linkedin.com")
            # Locate Username field and fill it
            session_key = driver.find_element_by_name("session_key")
            session_key.send_keys(email)
            # Locate Password field and fill it
            session_password =
        driver.find_element_by_name("session_password")
            session_password.send_keys(password)
            # Locate Submit button and click it
            submit = driver.find_element_by_class_name("sign-in-
```

```
        form__submit-button")
            submit.click()
            # Check credentials output
            if driver.title != "LinkedIn":
                print("Provided E-mail/Password is wrong!")
                driver.quit()
                sys.exit()
            # Return session
            return driver


        def scrap_basic(driver):
    """Returns 3 lists of Names, Headlines, and Profile Links"""
    driver.get("https://www.linkedin.com/mynetwork/invite-connect/connections/")
    # Bypassing Ajax Call through scrolling the page up and down multiple times
    # Base case is when the height of the scroll bar is constant after 2 complete scrolls
    time_to_wait = 3  # Best interval for a 512KB/Sec download speed - Change it according to your internet speed
    last_height = driver.execute_script("return document.body.scrollHeight")
    while True:
        # Scroll down to bottom
        driver.execute_script(
            "window.scrollTo(0, document.body.scrollHeight);")
```

```
        # This loop is for bypassing a small bug upon scrolling that causes the
Ajax call to be cancelled
        for i in range(2):
            time.sleep(time_to_wait)
            driver.execute_script("window.scrollTo(0, 0);")  # Scroll up to top
            time.sleep(time_to_wait)
            # Scroll down to bottom
            driver.execute_script(
                "window.scrollTo(0, document.body.scrollHeight);")

        new_height = driver.execute_script(
            "return document.body.scrollHeight")  # Update scroll bar height
        if new_height == last_height:
            break
        last_height = new_height

    # Extract card without links
    extracted_scrap = driver.find_elements_by_class_name(
        "mn-connection-card__details")
    extracted_scrap = [_.text for _ in extracted_scrap]
    # Append data to a seperate list
    names = []
    headlines = []
    for card in extracted_scrap:
        # Try statements just in case of headline/name type errors
```

```
        try:
            names.append(re.search(pattern_name, card)[0])
        except:
            names.append(" ")

        try:
            headlines.append(re.search(pattern_headline, card)[0])
        except:
            headlines.append(" ")

    # Extract links
    extracted_scrap = driver.find_elements_by_tag_name('a')
    links = []
    for i in extracted_scrap:
        link = i.get_attribute("href")
        if "https://www.linkedin.com/in" in link and not link in links:
            links.append(link)
    # Return outputs
    return driver, names, headlines, links


def scrap_skills(driver, links):
    skill_set = []
    length = len(links)
    for i in range(length):
```

```
        link = links[i]  # Get profile link
        driver.get(link)


        # Bypassing Ajax Call through scrolling through profile multiple
sections
        time_to_wait = 3
        last_height = driver.execute_script(
            "return document.body.scrollHeight")
        while True:
            # Scroll down to bottom
            driver.execute_script(
                "window.scrollTo(0, document.body.scrollHeight);")


            # This loop is for bypassing a small bug upon scrolling that causes the
Ajax call to be cancelled
            for i in range(2):
                time.sleep(time_to_wait)
                driver.execute_script(
                    "window.scrollTo(0, document.body.scrollHeight/4);")
                driver.execute_script(
                    "window.scrollTo(0, document.body.scrollHeight/3);")
                driver.execute_script(
                    "window.scrollTo(0, document.body.scrollHeight/2);")
                driver.execute_script(
                    "window.scrollTo(0, document.body.scrollHeight*3/4);")
                time.sleep(time_to_wait)
```

```
        # Scroll down to bottom
        driver.execute_script(
            "window.scrollTo(0, document.body.scrollHeight);")

    new_height = driver.execute_script(
        "return document.body.scrollHeight")  # Update scroll bar height
    if new_height == last_height:
        break
    last_height = new_height

# Locate button
buttons = driver.find_elements_by_tag_name('button')
length = len(buttons)
for button_num in range(length):
    i = buttons[button_num].get_attribute("data-control-name")
    if i == "skill_details":
        button = buttons[button_num]
        break
# Scroll then click the button
actions = ActionChains(driver)
actions.move_to_element(button).click().perform()
# Finally extract the skills
skills = driver.find_elements_by_xpath(
    "//*[starts-with(@class,'pv-skill-category-entity__name-text')]")
skill_set_list = []
```

```
        for skill in skills:
            skill_set_list.append(skill.text)
        # Append each skill set to its corresponding name
        # Appending all to one string
        skill_set.append(" -- ".join(skill_set_list))
    # Return session & skills
    return driver, skill_set


def save_to_csv(names, headlines, links, skills):
    # If skills argument was false
    if skills is None:
        skills = [None]*len(names)
    # Make a dataframe and append data to it
    df = pd.DataFrame()
    for i in range(len(names)):
        df = df.append({"Name": names[i], "Headline": headlines[i],
                    "Link": links[i], "Skills": skills[i]}, ignore_index=True)
    # Save to CSV
    df.to_csv("scrap.csv", index=False, columns=[
            "Name", "Headline", "Link", "Skills"])


# Start checkpoint
if __name__ == "__main__":
```

```
(options, args) = parser.parse_args()

# Inputs
email = options.email
password = options.password
skills = options.skills

driver = login(email, password)  # Login Phase
print("Successfull Login!")
print("Commencing 'My-Connections' list scrap...")
driver, names, headlines, links = scrap_basic(driver)  # Basic Scrap Phase
print("Finished basic scrap, scrapped {}".format(len(names)))

if skills:
    print("Commencing 'Skills' scrap...")
    driver, skill_set = scrap_skills(driver, links)  # Skills Scrap Phase
    print("Finished Skills scrap.")
            print("Saving to CSV file...")
            save_to_csv(names, headlines, links, skill_set)  # Save to CSV
        else:
            save_to_csv(names, headlines, links, None)  # Save to CSV

        print("Scrapping session has ended.")
        # End Session
        driver.quit()
```

52. Download Audio – Youtube

How To Download Audio Of A YouTube Video?

1. Setup python and pip if you haven’t already

2. Install virtualenv

`pip install virtualenv`

3. Create Virtual environment

`virtualenv venv`

4. Activate virtual environment

- `source venv/bin/activate` (Linux)
- `venv\Scripts\activate` (Windows)

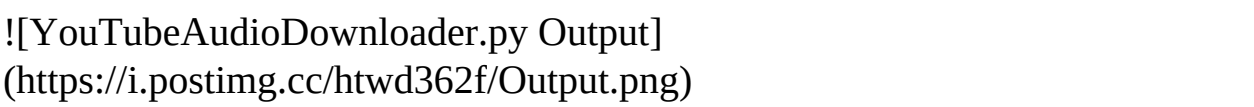
5. Install requirements

`pip install -r requirements.txt`

6. Specify url of the YouTube video whoose audio you want (in YouTubeAudioDownloader.py)

ex: `url = "https://www.youtube.com/watch?v=ZSXN_dpG5jk"`

7. Run YouTubeAudioDownloader.py



You will find file of .webm format downloaded in the same folder.

How To convert .webm to .mp3 format?

1. Install moviepy in the same environment

`pip install moviepy==1.0.3`

2. Specify path of .webm file (in WebmToMp3.py)

ex: `clip = mp.AudioFileClip("C:/Users/sejal/Desktop/YouTube Audio Downloader/Rainbow.webm").subclip()`

3. Specify path where .mp3 file should be saved (in WebmToMp3.py)

ex: `clip.write_audiofile("C:/Users/sejal/Desktop/YouTube Audio

Downloader/rainbow.mp3")`

4. Run WebmToMp3.py

Requirements:

pafy==0.5.5

youtube-dl==2020.12.31

Source Code files:

53. Youtube Video Downloader

YouTube Video Downloader

The objective of this project is to download any type of video in a fast and easy way from youtube in your device.

In this, user has to copy the youtube video URL that they want to download and simply paste that URL in the ‘paste link here’ section and click on the download button, it will start downloading the video. When video downloading finishes, it shows a message ‘downloaded’ popup on the window below the download button.

Prerequisites

To implement this, we use basic concept of python, tkinter and pytube library.

- **Tkinter** is a standard GUI library and it is one of the easiest ways to build a GUI application.
- **pytube** used for downloading videos from youtube

To install the required modules run pip installer command on the command line:

```
pip install tkinter
pip install pytube
```

These are the following steps to build:

Step 1: Import libraries

Start the project by importing the required modules.

In this script implementation, we import Tkinter and pytube modules.

Step 2: Create display window

- **Tk()** used to initialize tkinter to create display window
- **geometry()** used to set the window’s width and height
- **resizable(0,0)** set the fix size of window
- **title()** used to give the title of window
- **Label()** widget use to display text that users can’t able to modify.
- **root** is the name of the window
- **text** which we display the title of the label
- **font** in which our text is written
- **pack** organized widget in block

Step 3: Create field to enter link

- **link** is a string type variable that stores the youtube video link that the user enters.
- **Entry()** widget is used when we want to create an input text field.
- **width** sets the width of entry widget
- **textvariable** used to retrieve the value of current text variable to the entry widget
- **place()** use to place the widget at a specific position

Step 4: Create function to start downloading

`url` variable gets the youtube link from the link variable by **get()** function and then **str()** will convert the link in string datatype.

The video is downloaded in the first present stream of that video by **stream.first()** method.

Button() widget used to display button on the window.

- **text** which we display on the label
- **font** in which the text is written
- **bg** sets the background color
- **command** is used to call the function

root.mainloop() is a method that executes when we want to run the program.

Output

After running this script, you will be able to see this:

Source Code:

```
from tkinter import *
from pytube import YouTube

root = Tk()
root.geometry('700x300')
```

```
root.resizable(0, 0)
root.title("YouTube Video Downloader")

Label(root, text='Copy the link of the video you want to download from YouTube',
      font='arial 15 bold').pack()

# enter link
link = StringVar()

Label(root, text='Paste Link Here:', font='arial 15 bold').place(x=270, y=60)
Entry(root, width=80, textvariable=link).place(x=32, y=90)

# function to download video



def Downloader():

    url = YouTube(str(link.get()))
    video = url.streams.first()
    video.download()
    Label(root, text='DOWNLOADED', font='arial 15').place(x=270, y=210)
```

```
Button(root, text='DOWNLOAD', font='arial 15 bold', bg='white',
      padx=2, command=Downloader).place(x=280, y=150)

root.mainloop()
```

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/53-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.