Follow me on twitter - @harendraverma2

Follow me on medium.com - @harendraverma21

1. Convert JSON to CSV

This script will convert your JSON data to a CSV file. It takes `.json` a file as input and provides `.csv` the file as output.

Installation

```
pip install json
```

```
import json
if __name__ == '__main__':
    try:
        with open('input.json', 'r') as f:
            data = json.loads(f.read())

        output = ','.join([*data[0]])
        for obj in data:
            output += f'\n{obj["Name"]},{obj["age"]},{obj["birthyear"]}'

        with open('output.csv', 'w') as f:
            f.write(output)
    except Exception as ex:
        print(f'Error: {str(ex)}')
```

2. Password Generator

This simple Python project is using `random` and `string` package to generate a random string of a given length.

```
import random
import string
total = string.ascii_letters + string.digits + string.punctuation
length = 16
password = "".join(random.sample(total, length))
print(password)
```

3. String search from multiple files

Finds a file with the supplied string in the folder of your choosing.

```
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)
```

4. Fetch all links from a given webpage

This script gets all links from a particular website and saves them as a text file.

Installation

```
pip install beautifulsoup4 requests
```

```
import requests as rq
from bs4 import BeautifulSoup

url = input("Enter Link: ")
if ("https" or "http") in url:
    data = rq.get(url)
else:
    data = rq.get("https://" + url)
soup = BeautifulSoup(data.text, "html.parser")
```

```
links = []
for link in soup.find_all("a"):
    links.append(link.get("href"))

# Writing the output to a file (myLinks.txt) instead of to stdout
# You can change 'a' to 'w' to overwrite the file each time
with open("myLinks.txt", 'a') as saved:
    print(links[:10], file=saved)
```

5. Image Watermarking

This project will take a photograph and put a watermark of your choice on it.

Installation

```
pip install Pillow
```

```
import os
from PIL import Image

def 
watermark_photo(input_image_path,watermark_image_path,output_image_path):
    base_image = Image.open(input_image_path)
    watermark = Image.open(watermark_image_path).convert("RGBA")
    # add watermark to your image
    position = base_image.size
    newsize = (int(position[0]*8/100),int(position[0]*8/100))
    # print(position)
    watermark = watermark.resize(newsize)
    # print(newsize)
    # return watermark

    new_position = position[0]-newsize[0]-20,position[1]-newsize[1]-20
    # create a new transparent image
    transparent = Image.new(mode='RGBA',size=position,color=(0,0,0,0))
    # paste the original image
    transparent.paste(base_image,(0,0))
    # paste the watermark image
    transparent.paste(watermark,new_position,watermark)
    image_mode = base_image.mode
    print(image_mode)
    if image_mode == 'RGB':
        transparent = transparent.convert(image_mode)
    else:
        transparent = transparent.convert('P')
    transparent.save(output_image_path,optimize=True,quality=100)
    print("Saving"+output_image_path+"...")

folder = input("Enter Folder Path:")
```

```
watermark = input("Enter Watermark Path:")
os.chdir(folder)
files = os.listdir(os.getcwd())
print(files)

if not os.path.isdir("output"):
    os.mkdir("output")

c = 1
for f in files:
    if os.path.isfile(os.path.abspath(f)):
        if f.endswith(".png") or f.endswith(".jpg"):
            watermark_photo(f,watermark,"output/"+f)
```

6. Scrap and Download all images from the WEB Page

This script will utilize the selenium and beautifulsoup4 packages to download all photos from a specified webpage.

Installation

1. We need to install selenium and beautifulsoup4 using pip python package manager `pip install selenium beautifulsoup4`.
2. Then download chrome driver as per your chrome browser version and OS from here — https://chromedriver.chromium.org/
3. You have to enter the chromedriver path asked by the program.

```
from selenium import webdriver
import requests as rq
import os
from bs4 import BeautifulSoup
import time

# path= E:\web scraping\chromedriver_win32\chromedriver.exe
path = input("Enter Path : ")

url = input("Enter URL : ")

output = "output"


def get_url(path, url):
    driver = webdriver.Chrome(executable_path=r"{}".format(path))
    driver.get(url)
    print("loading.....")
    res = driver.execute_script("return
document.documentElement.outerHTML")

    return res
```

```
def get_img_links(res):
    soup = BeautifulSoup(res, "lxml")
    imglinks = soup.find_all("img", src=True)
    return imglinks


def download_img(img_link, index):
    try:
        extensions = [".jpeg", ".jpg", ".png", ".gif"]
        extension = ".jpg"
        for exe in extensions:
            if img_link.find(exe) > 0:
                extension = exe
                break

        img_data = rq.get(img_link).content
        with open(output + "\\" + str(index + 1) + extension, "wb+") as f:
            f.write(img_data)

        f.close()
    except Exception:
        pass


result = get_url(path, url)
time.sleep(60)
img_links = get_img_links(result)
if not os.path.isdir(output):
    os.mkdir(output)

for index, img_link in enumerate(img_links):
    img_link = img_link["src"]
    print("Downloading...")
    if img_link:
        download_img(img_link, index)
print("Download Complete!!")
```

Executing the Script

To run the following script you have to open the terminal into the script's root directory and need to enter the following command

```
python3 scrap-img.py
```

It will ask for the chrome driver path which you have just downloaded and a URL from which you want to download images.

7. Low Battery Notification

This python script displays a notice regarding the device’s battery percentage.

Installation

To run this script we need to download the psutil, py-notifier, and win10tost by running the following commands.

```
1. psutil
    > pip install psutil

2. pynotifier
    > pip install py-notifier

3. win10toast
    > pip install win10toast
```

```
# pip install psutil
import psutil

battery = psutil.sensors_battery()
plugged = battery.power_plugged
percent = battery.percent

if percent <= 30 and plugged!=True:

    # pip install py-notifier
    # pip install win10toast
    from pynotifier import Notification

    Notification(
        title="Battery Low",
        description=str(percent) + "% Battery remain!!",
        duration=5,  # Duration in seconds

    ).send()
```

Executing the Script

Open your terminal into the root directory of your script file and run the following command

```
python3 battery.py
```

8. Calculate Your Age

This script prints your age in three different ways: **Years**, **Months**, **Days**

```
import time
from calendar import isleap

# judge the leap year
def judge_leap_year(year):
    if isleap(year):
        return True
    else:
        return False


# returns the number of days in each month
def month_days(month, leap_year):
    if month in [1, 3, 5, 7, 8, 10, 12]:
        return 31
    elif month in [4, 6, 9, 11]:
        return 30
    elif month == 2 and leap_year:
        return 29
    elif month == 2 and (not leap_year):
        return 28


name = input("input your name: ")
age = input("input your age: ")
localtime = time.localtime(time.time())

year = int(age)
month = year * 12 + localtime.tm_mon
day = 0

begin_year = int(localtime.tm_year) - year
end_year = begin_year + year

# calculate the days
for y in range(begin_year, end_year):
    if (judge_leap_year(y)):
        day = day + 366
    else:
        day = day + 365

leap_year = judge_leap_year(localtime.tm_year)
for m in range(1, localtime.tm_mon):
    day = day + month_days(m, leap_year)

day = day + localtime.tm_mday
print("%s's age is %d years or " % (name, year), end="")
print("%d months or %d days" % (month, day))
```

Executing the Script

It is quite simple to execute the script!

Simply open a terminal in the folder containing your script and enter the following command:

```
python3 calculate.py
```

Then you have to enter the name and age

```
input your name: XYZ
input your age: 33
Output - XYZ's age is 33 years or 406 months or 12328 days
```

9. Organized download folder with different categories

This is a Python script that sorts files in the Download directory into other folders based on their extension.

```
import os
import shutil
os.chdir("E:\downloads")
#print(os.getcwd())

#check number of files in  directory
files = os.listdir()

#list of extension (You can add more if you want)
extentions = {
    "images": [".jpg", ".png", ".jpeg", ".gif"],
    "videos": [".mp4", ".mkv"],
    "musics": [".mp3", ".wav"],
    "zip": [".zip", ".tgz", ".rar", ".tar"],
    "documents": [".pdf", ".docx", ".csv", ".xlsx", ".pptx", ".doc",
".ppt", ".xls"],
    "setup": [".msi", ".exe"],
    "programs": [".py", ".c", ".cpp", ".php", ".C", ".CPP"],
    "design": [".xd", ".psd"]
}
#sort to specific folder depend on extenstions
def sorting(file):
    keys = list(extentions.keys())
    for key in keys:
        for ext in extentions[key]:
            # print(ext)
            if file.endswith(ext):
                return key

#iterat through each file
for file in files:
    dist = sorting(file)
    if dist:
        try:
            shutil.move(file, "../download-sorting/" + dist)
        except:
            print(file + " is already exist")
    else:
        try:
            shutil.move(file, "../download-sorting/others")
        except:
            print(file + " is already exist")
```

10. Send Emails in Bulk From CSV File

This project includes a straightforward bulk email script that delivers the same message to a list of recipients.

Installation

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

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

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

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

def send_mail():
    s = smtplib.SMTP("smtp.gmail.com", 587)
    email_address, email_pass = get_credentials("./credentials.txt")
    login(email_address, email_pass, s)
    # message to be sent
    subject = "Welcome to Python"
    body = """Python is an interpreted, high-level,
    general-purpose programming language.\n
    Created by Guido van Rossum and first released in 1991,
    Python's design philosophy emphasizes code readability\n
    with its notable use of significant whitespace"""

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

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

    # terminating the session
    s.quit()
    print("sent")

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

Executing the Script

The script necessitates the use of two configuration files:

- *emails.csv* should include the email addresses to which the message should be sent.
- *credentials.txt* should include your SMTP server login credentials, with your user name and password on separate lines and no extra whitespace or decorations.

The project directory contains two sample files that you will almost certainly desire and need to alter.

Once you've got these files in place, all you have to do is

```
python Send_emails.py
```

11. Get the IP Address and Hostname of A Website

This script will be used to fetch the IP address and hostname of any website.

```
# Get Ipaddress and Hostname of Website
# importing socket library
import socket

def get_hostname_IP():
    hostname = input("Please enter website address(URL):")
    try:
        print (f'Hostname: {hostname}')
        print (f'IP: {socket.gethostbyname(hostname)}')
    except socket.gaierror as error:
        print (f'Invalid Hostname, error raised is {error}')

get_hostname_IP()
```

12. Terminal Progress Bar

Here I just take the example of image resizing for displaying the progress bar. when we convert lots of images at a time we can use the progress bar to show how many images are resized.

Installation

- Library to show the progress bar

```
pip install tqdm
```

- Library to resize the images

```
pip install Pillow
```

```
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!")
```

13. Wifi Password Ejector

Get saved wifi password from your windows operating system

```
import subprocess

data = (
    subprocess.check_output(["netsh", "wlan", "show", "profiles"])
    .decode("utf-8")
    .split("\n")
)
profiles = [i.split(":")[1][1:-1] for i in data if "All User Profile" in i]
for i in profiles:
    results = (
        subprocess
        .check_output(["netsh", "wlan", "show", "profile", i, "key=clear"])
        .decode("utf-8")
        .split("\n")
    )
    results = [b.split(":")[1][1:-1] for b in results if "Key Content" in
b]
    try:
        print("{:<30}|  {:<}".format(i, results[0]))
    except IndexError:
        print("{:<30}|  {:<}".format(i, ""))
```

14. Snapshot of The Given Website

This script will capture a screenshot of the website provided.

Installation

Read more about `selenium` here

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

```
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)
```

15. Split Files Into Chunks

This script accepts split index and file name then splits it according to the index provided.

```
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()
```

16. Encrypt and Decrypt Texts

A small python program that encodes and decodes text.

##$ Prerequisites

- pycryptodome 3.9.8
- Python 3

```
from Crypto.Cipher import AES
from Crypto import Random
from binascii import b2a_hex
import sys

# get the plaintext
plain_text = sys.argv[1]

# The key length must be 16 (AES-128), 24 (AES-192), or 32 (AES-256) Bytes.
key = b'this is a 16 key'

# Generate a non-repeatable key vector with a length
# equal to the size of the AES block
iv = Random.new().read(AES.block_size)
```

```
# Use key and iv to initialize AES object, use MODE_CFB mode
mycipher = AES.new(key, AES.MODE_CFB, iv)

# Add iv (key vector) to the beginning of the encrypted ciphertext
# and transmit it together
ciphertext = iv + mycipher.encrypt(plain_text.encode())


# To decrypt, use key and iv to generate a new AES object
mydecrypt = AES.new(key, AES.MODE_CFB, ciphertext[:16])

# Use the newly generated AES object to decrypt the encrypted ciphertext
decrypttext = mydecrypt.decrypt(ciphertext[16:])

# output
file_out = open("encrypted.bin", "wb")
file_out.write(ciphertext[16:])
file_out.close()

print("The key k is: ", key)
print("iv is: ", b2a_hex(ciphertext)[:16])
print("The encrypted data is: ", b2a_hex(ciphertext)[16:])
print("The decrypted data is: ", decrypttext.decode())
```

17. Capture Screenshots At Regular Intervals of Time

Python script to capture screenshots at regular intervals of time.

```
import os
import argparse
import pyautogui
import time

parser = argparse.ArgumentParser()
parser.add_argument("-p", "--path", help="absolute path to store 
screenshot.", default=r"./images")
parser.add_argument("-t", "--type", help="h (in hour) or m (in minutes) or 
s (in seconds)", default='h')
parser.add_argument("-f", "--frequency", help="frequency for taking 
screenshot per h/m/s.", default=1, type=int)

args = parser.parse_args()
sec = 0.

if args.type == 'h':
    sec = 60 * 60 / args.frequency
elif args.type == 'm':
    sec = 60 / args.frequency

if sec < 1.:
    sec = 1.

if os.path.isdir(args.path) != True:
    os.mkdir(args.path)

try:
    while True:
        t = time.localtime()
        current_time = time.strftime("%H_%M_%S", t)
        file = current_time + ".jpg"
        image = pyautogui.screenshot(os.path.join(args.path,file))
        print(f"{file} saved successfully.\n")
        time.sleep(sec)

except KeyboardInterrupt:
    print("End of script by user interrupt")
```

How to run?

```
python screenshot.py # takes screenshot at interval of 1 hour
python screenshot.py -t m -f 5 # takes 5 screenshots in 1 minute
python screenshot.py -p path_to_directory # screenshots will be saved to
path_to_directory
```

18. Decimal to Binary converter

A small python program that converts binary and decimal, and decimal to binary.

```
try:
    menu = int(input("Choose an option: \n 1. Decimal to binary \n 2.
Binary to decimal\n Option: "))
    if menu < 1 or menu > 2:
        raise ValueError
    if menu == 1:
        dec = int(input("Input your decimal number:\nDecimal: "))
        print("Binary: {}".format(bin(dec)[2:]))
    elif menu == 2:
        binary = input("Input your binary number:\n Binary: ")
        print("Decimal: {}".format(int(binary, 2)))
except ValueError:
    print ("please choose a valid option")
```

19. CLI Todo App

Simple Todo app with command-line interface. Supports adding, deleting, and viewing task entries.

Installation

```
pip install click
```

```
import click

@click.group()
@click.pass_context
def todo(ctx):
    '''Simple CLI Todo App'''
    ctx.ensure_object(dict)
    #Open todo.txt – first line contains latest ID, rest contain tasks and
IDs
    with open('./todo.txt') as f:
        content = f.readlines()
    #Transfer data from todo.txt to the context
    ctx.obj['LATEST'] = int(content[:1][0])
    ctx.obj['TASKS'] = {en.split('```')[0]:en.split('```')[1][:-1] for en
in content[1:]}

@todo.command()
@click.pass_context
def tasks(ctx):
    '''Display tasks'''
    if ctx.obj['TASKS']:
        click.echo('YOUR TASKS\n**********')
        #Iterate through all the tasks stored in the context
        for i, task in ctx.obj['TASKS'].items():
            click.echo('• ' + task + ' (ID: ' + i + ')')
        click.echo('')
    else:
        click.echo('No tasks yet! Use ADD to add one.\n')

@todo.command()
@click.pass_context
@click.option('-add', '--add_task', prompt='Enter task to add')
def add(ctx, add_task):
    '''Add a task'''
    if add_task:
        #Add task to list in context
        ctx.obj['TASKS'][ctx.obj['LATEST']] = add_task
        click.echo('Added task "' + add_task + '" with ID ' +
str(ctx.obj['LATEST']))
        #Open todo.txt and write current index and tasks with IDs
(separated by " ``` ")
        curr_ind = [str(ctx.obj['LATEST'] + 1)]
        tasks = [str(i) + '```' + t for (i, t) in ctx.obj['TASKS'].items()]
        with open('./todo.txt', 'w') as f:
            f.writelines(['%s\n' % en for en in curr_ind + tasks])

@todo.command()
@click.pass_context
```

~~~~
@click.option('-fin', '--fin_taskid', prompt='Enter ID of task to finish',
type=int)
def done(ctx, fin_taskid):
    '''Delete a task by ID'''
    #Find task with associated ID
    if str(fin_taskid) in ctx.obj['TASKS'].keys():
        task = ctx.obj['TASKS'][str(fin_taskid)]
        #Delete task from task list in context
        del ctx.obj['TASKS'][str(fin_taskid)]
        click.echo('Finished and removed task "' + task + '" with id ' +
str(fin_taskid))
        #Open todo.txt and write current index and tasks with IDs
(separated by " ``` ")
        if ctx.obj['TASKS']:
            curr_ind = [str(ctx.obj['LATEST'] + 1)]
            tasks = [str(i) + '```' + t for (i, t) in
ctx.obj['TASKS'].items()]
            with open('./todo.txt', 'w') as f:
                f.writelines(['%s\n' % en for en in curr_ind + tasks])
        else:
            #Resets ID tracker to 0 if list is empty
            with open('./todo.txt', 'w') as f:
                f.writelines([str(0) + '\n'])
    else:
        click.echo('Error: no task with id ' + str(fin_taskid))

if __name__ == '__main__':
    todo()
~~~~

How to use it?

Running

Either run it from your code editor or Ide or type `python todo.py [command]` in your command line. (instead of [command] add the desired command u want)

Commands

`add` Adds a task. Prompts the user for task text.

`done` Deletes a task. Prompts the user for task id.

`tasks` Displays all inputted tasks.

20. Currency Converter

A small python program that converts currency with live info

```
import requests
import json
import sys
from pprint import pprint
```

```
# The below 4 lines bring out the value of currency from the api at
fixer.io.  I had to register there, the key is unique to me.
url = "http://data.fixer.io/api/latest?
access_key=33ec7c73f8a4eb6b9b5b5f95118b2275"
data = requests.get(url).text
data2 = json.loads(data) #brings whether request was successful,timestamp
etc
fx = data2["rates"]

currencies = [
    "AED : Emirati Dirham,United Arab Emirates Dirham",
    "AFN : Afghan Afghani,Afghanistan Afghani",
    "ALL : Albanian Lek,Albania Lek",
    "AMD : Armenian Dram,Armenia Dram",
    "ANG : Dutch Guilder,Netherlands Antilles
Guilder,Bonaire,Cura&#231;ao,Saba,Sint Eustatius,Sint Maarten",
    "AOA : Angolan Kwanza,Angola Kwanza",
    "ARS : Argentine Peso,Argentina Peso,Islas Malvinas",
    "AUD : Australian Dollar,Australia Dollar,Christmas Island,Cocos
(Keeling) Islands,Norfolk Island,Ashmore and Cartier Islands,Australian
Antarctic Territory,Coral Sea Islands,Heard Island,McDonald
Islands,Kiribati,Nauru",
    "AWG : Aruban or Dutch Guilder,Aruba Guilder",
    "AZN : Azerbaijan Manat,Azerbaijan Manat",
    "BAM : Bosnian Convertible Mark,Bosnia and Herzegovina Convertible
Mark",
    "BBD : Barbadian or Bajan Dollar,Barbados Dollar",
    "BDT : Bangladeshi Taka,Bangladesh Taka",
    "BGN : Bulgarian Lev,Bulgaria Lev",
    "BHD : Bahraini Dinar,Bahrain Dinar",
    "BIF : Burundian Franc,Burundi Franc",
    "BMD : Bermudian Dollar,Bermuda Dollar",
    "BND : Bruneian Dollar,Brunei Darussalam Dollar",
    "BOB : Bolivian Bol&#237;viano,Bolivia Bol&#237;viano",
    "BRL : Brazilian Real,Brazil Real",
    "BSD : Bahamian Dollar,Bahamas Dollar",
    "BTC : Bitcoin,BTC, XBT",
    "BTN : Bhutanese Ngultrum,Bhutan Ngultrum",
    "BWP : Botswana Pula,Botswana Pula",
    "BYN : Belarusian Ruble,Belarus Ruble",
    "BYR : Belarusian Ruble,Belarus Ruble",
    "BZD : Belizean Dollar,Belize Dollar",
    "CAD : Canadian Dollar,Canada Dollar",
    "CDF : Congolese Franc,Congo/Kinshasa Franc",
    "CHF : Swiss Franc,Switzerland Franc,Liechtenstein,Campione
d&#039;Italia,B&#252;singen am Hochrhein",
    "CLF : Chilean Unit of Account",
    "CLP : Chilean Peso,Chile Peso",
    "CNY : Chinese Yuan Renminbi,China Yuan Renminbi",
    "COP : Colombian Peso,Colombia Peso",
    "CRC : Costa Rican Colon,Costa Rica Colon",
    "CUC : Cuban Convertible Peso,Cuba Convertible Peso",
    "CUP : Cuban Peso,Cuba Peso",
    "CVE : Cape Verdean Escudo,Cape Verde Escudo",
    "CZK : Czech Koruna,Czech Republic Koruna",
    "DJF : Djiboutian Franc,Djibouti Franc",
    "DKK : Danish Krone,Denmark Krone,Faroe Islands,Greenland",
    "DOP : Dominican Peso,Dominican Republic Peso",
    "DZD : Algerian Dinar,Algeria Dinar",
    "EGP : Egyptian Pound,Egypt Pound,Gaza Strip",
    "ERN : Eritrean Nakfa,Eritrea Nakfa",
    "ETB : Ethiopian Birr,Ethiopia Birr,Eritrea",
    "EUR : Euro,Euro Member Countries,Andorra,Austria,Azores,Baleares
(Balearic Islands),Belgium,Canary Islands,Cyprus,Finland,France,French
Guiana,French Southern Territories,Germany,Greece,Guadeloupe,Holland
(Netherlands),Holy See (Vatican City),Ireland
(Eire),Italy,Luxembourg,Madeira
Islands,Malta,Monaco,Montenegro,Netherlands",
    "FJD : Fijian Dollar,Fiji Dollar",
    "FKP : Falkland Island Pound,Falkland Islands (Malvinas) Pound",
    "GBP : British Pound,United Kingdom Pound,United Kingdom
(UK),England,Northern Ireland,Scotland,Wales,Falkland
Islands,Gibraltar,Guernsey,Isle of Man,Jersey,Saint Helena and
Ascension,South Georgia and the South Sandwich Islands,Tristan da Cunha",
    "GEL : Georgian Lari,Georgia Lari",
    "GGP : Guernsey Pound,Guernsey Pound",
    "GHS : Ghanaian Cedi,Ghana Cedi",
    "GIP : Gibraltar Pound,Gibraltar Pound",
    "GMD : Gambian Dalasi,Gambia Dalasi",
    "GNF : Guinean Franc,Guinea Franc",
    "GTQ : Guatemalan Quetzal,Guatemala Quetzal",
    "GYD : Guyanese Dollar,Guyana Dollar",
    "HKD : Hong Kong Dollar,Hong Kong Dollar",
    "HNL : Honduran Lempira,Honduras Lempira",
    "HRK : Croatian Kuna,Croatia Kuna",
    "HTG : Haitian Gourde,Haiti Gourde",
    "HUF : Hungarian Forint,Hungary Forint",
    "IDR : Indonesian Rupiah,Indonesia Rupiah,East Timor",
    "ILS : Israeli Shekel,Israel Shekel,Palestinian Territories",
    "IMP : Isle of Man Pound,Isle of Man Pound",
    "INR : Indian Rupee,India Rupee,Bhutan,Nepal",
    "IQD : Iraqi Dinar,Iraq Dinar",
    "IRR : Iranian Rial,Iran Rial",
    "ISK : Icelandic Krona,Iceland Krona",
    "JEP : Jersey Pound,Jersey Pound",
    "JMD : Jamaican Dollar,Jamaica Dollar",
    "JOD : Jordanian Dinar,Jordan Dinar",
    "JPY : Japanese Yen,Japan Yen",
    "KES : Kenyan Shilling,Kenya Shilling",
    "KGS : Kyrgyzstani Som,Kyrgyzstan Som",
    "KHR : Cambodian Riel,Cambodia Riel",
    "KMF : Comorian Franc,Comorian Franc",
    "KPW : North Korean Won,Korea (North) Won",
    "KRW : South Korean Won,Korea (South) Won",
    "KWD : Kuwaiti Dinar,Kuwait Dinar",
    "KYD : Caymanian Dollar,Cayman Islands Dollar",
    "KZT : Kazakhstani Tenge,Kazakhstan Tenge",
    "LAK : Lao Kip,Laos Kip",
    "LBP : Lebanese Pound,Lebanon Pound",
    "LKR : Sri Lankan Rupee,Sri Lanka Rupee",
    "LRD : Liberian Dollar,Liberia Dollar",
    "LSL : Basotho Loti,Lesotho Loti",
    "LTL : Lithuanian litas",
    "LVL : Latvia Lats",
    "LYD : Libyan Dinar,Libya Dinar",
    "MAD : Moroccan Dirham,Morocco Dirham,Western Sahara",
    "MDL : Moldovan Leu,Moldova Leu",
    "MGA : Malagasy Ariary,Madagascar Ariary",
    "MKD : Macedonian Denar,Macedonia Denar",
    "MMK : Burmese Kyat,Myanmar (Burma) Kyat",
    "MNT : Mongolian Tughrik,Mongolia Tughrik",
    "MOP : Macau Pataca,Macau Pataca",
    "MRU : Mauritanian Ouguiya,Mauritania Ouguiya",
    "MUR : Mauritian Rupee,Mauritius Rupee",
    "MVR : Maldivian Rufiyaa,Maldives (Maldive Islands) Rufiyaa",
    "MWK : Malawian Kwacha,Malawi Kwacha",
    "MXN : Mexican Peso,Mexico Peso",
    "MYR : Malaysian Ringgit,Malaysia Ringgit",
    "MZN : Mozambican Metical,Mozambique Metical",
    "NAD : Namibian Dollar,Namibia Dollar",
    "NGN : Nigerian Naira,Nigeria Naira",
    "NIO : Nicaraguan Cordoba,Nicaragua Cordoba",
    "NOK : Norwegian Krone,Norway Krone,Bouvet Island,Svalbard,Jan
Mayen,Queen Maud Land,Peter I Island",
    "NPR : Nepalese Rupee,Nepal Rupee,India (unofficially near India-Nepal
border)",
    "NZD : New Zealand Dollar,New Zealand Dollar,Cook Islands,Niue,Pitcairn
Islands,Tokelau",
    "OMR : Omani Rial,Oman Rial",
    "PAB : Panamanian Balboa,Panama Balboa",
    "PEN : Peruvian Sol,Peru Sol",
    "PGK : Papua New Guinean Kina,Papua New Guinea Kina",
    "PHP : Philippine Peso,Philippines Peso",
    "PKR : Pakistani Rupee,Pakistan Rupee",
    "PLN : Polish Zloty,Poland Zloty",
    "PYG : Paraguayan Guarani,Paraguay Guarani",
    "QAR : Qatari Riyal,Qatar Riyal",
    "RON : Romanian Leu,Romania Leu",
    "RSD : Serbian Dinar,Serbia Dinar",
    "RUB : Russian Ruble,Russia Ruble,Tajikistan,Abkhazia,South Ossetia",
    "RWF : Rwandan Franc,Rwanda Franc",
    "SAR : Saudi Arabian Riyal,Saudi Arabia Riyal",
    "SBD : Solomon Islander Dollar,Solomon Islands Dollar",
    "SCR : Seychellois Rupee,Seychelles Rupee",
    "SDG : Sudanese Pound,Sudan Pound",
    "SEK : Swedish Krona,Sweden Krona",
    "SGD : Singapore Dollar,Singapore Dollar",
    "SHP : Saint Helenian Pound,Saint Helena Pound",
    "SLL : Sierra Leonean Leone,Sierra Leone Leone",
    "SOS : Somali Shilling,Somalia Shilling",
    "SRD : Surinamese Dollar,Suriname Dollar",
    "STN : Sao Tomean Dobra,S&#227;o Tom&#233; and Pr&#237;ncipe Dobra",
    "SVC : Salvadoran Colon,El Salvador Colon",
    "SYP : Syrian Pound,Syria Pound",
    "SZL : Swazi Lilangeni,eSwatini Lilangeni",
    "THB : Thai Baht,Thailand Baht",
    "TJS : Tajikistani Somoni,Tajikistan Somoni",
    "TMT : Turkmenistani Manat,Turkmenistan Manat",
    "TND : Tunisian Dinar,Tunisia Dinar",
    "TOP : Tongan Pa&#039;anga,Tonga Pa&#039;anga",
    "TRY : Turkish Lira,Turkey Lira,North Cyprus",
    "TTD : Trinidadian Dollar,Trinidad and Tobago Dollar,Trinidad,Tobago",
    "TWD : Taiwan New Dollar,Taiwan New Dollar",
    "TZS : Tanzanian Shilling,Tanzania Shilling",
    "UAH : Ukrainian Hryvnia,Ukraine Hryvnia",
    "UGX : Ugandan Shilling,Uganda Shilling",
    "USD : US Dollar,United States Dollar,America,American Samoa,American
Virgin Islands,British Indian Ocean Territory,British Virgin
Islands,Ecuador,El Salvador,Guam,Haiti,Micronesia,Northern Mariana
Islands,Palau,Panama,Puerto Rico,Turks and Caicos Islands,United States
Minor Outlying Islands,Wake Island,East Timor",
    "UYU : Uruguayan Peso,Uruguay Peso",
    "UZS : Uzbekistani Som,Uzbekistan Som",
    "VEF : Venezuelan Bol&#237;var,Venezuela Bol&#237;var",
    "VND : Vietnamese Dong,Viet Nam Dong",
    "VUV : Ni-Vanuatu Vatu,Vanuatu Vatu",
    "WST : Samoan Tala,Samoa Tala",
    "XAF : Central African CFA Franc BEAC,Communaut&#233; Financi&#232;re
Africaine (BEAC) CFA Franc BEAC,Cameroon,Central African
Republic,Chad,Congo/Brazzaville,Equatorial Guinea,Gabon",
    "XAG : Silver Ounce,Silver",
    "XAU : Gold Ounce,Gold",
    "XCD : East Caribbean Dollar,East Caribbean Dollar,Anguilla,Antigua and
Barbuda,Dominica,Grenada,The Grenadines and Saint Vincent,Montserrat",
    "XDR : IMF Special Drawing Rights,International Monetary Fund (IMF)
Special Drawing Rights",
    "XOF : CFA Franc,Communaut&#233; Financi&#232;re Africaine (BCEAO)
Franc,Benin,Burkina Faso,Ivory Coast,Guinea-
Bissau,Mali,Niger,Senegal,Togo",
    "XPF : CFP Franc,Comptoirs Fran&#231;ais du Pacifique (CFP)
Franc,French Polynesia,New Caledonia,Wallis and Futuna Islands",
    "YER : Yemeni Rial,Yemen Rial",
    "ZAR : South African Rand,South Africa Rand,Lesotho,Namibia",
    "ZMK : Zambian Kwacha,Zambia Kwacha",
    "ZMW : Zambian Kwacha,Zambia Kwacha",
    "ZWL : Zimbabwean Dollar,Zimbabwe Dollar",
]


# The below function calculates the actual conversion
def function1():
    query = input(
        "Please specify the amount of currency to convert, from currency,
to currency (with space in between).\nPress SHOW to see list of currencies
available. \nPress Q to quit. \n"
```

```
    )
    if query == "Q":
        sys.exit()
    elif query == "SHOW":
        pprint(currencies)
        function1()
    else:
        qty, fromC, toC = query.split(" ")
        fromC = fromC.upper()
        toC = toC.upper()
        qty = float(round(int(qty), 2))
        amount = round(qty * fx[toC] / fx[fromC], 2)
        print(f"{qty} of currency {fromC} amounts to {amount} of currency
{toC} today")


try:
    function1()
except KeyError:
    print("You seem to have inputted wrongly, retry!")
    function1()
```

21. Create a simple stopwatch

```
import tkinter as Tkinter
from datetime import datetime
counter = 0
running = False


def counter_label(label):
    def count():
        if running:
            global counter
            # To manage the intial delay.
            if counter == 0:
                display = 'Ready!'
            else:
                tt = datetime.utcfromtimestamp(counter)
                string = tt.strftime('%H:%M:%S')
                display = string

            label['text'] = display

            # label.after(arg1, arg2) delays by
            # first argument given in milliseconds
            # and then calls the function given as second argument.
            # Generally like here we need to call the
            # function in which it is present repeatedly.
            # Delays by 1000ms=1 seconds and call count again.
            label.after(1000, count)
```

```
                counter += 1

        # Triggering the start of the counter.
        count()


    # start function of the stopwatch
    def Start(label):
        global running
        running = True
        counter_label(label)
        start['state'] = 'disabled'
        stop['state'] = 'normal'
        reset['state'] = 'normal'


    # Stop function of the stopwatch
    def Stop():
        global running
        start['state'] = 'normal'
        stop['state'] = 'disabled'
        reset['state'] = 'normal'
        running = False


    # Reset function of the stopwatch
    def Reset(label):
        global counter
        counter = 0
        # If reset is pressed after pressing stop.
        if not running:
            reset['state'] = 'disabled'
            label['text'] = '00:00:00'
        # If reset is pressed while the stopwatch is running.
        else:
            label['text'] = '00:00:00'


    root = Tkinter.Tk()
    root.title("Stopwatch")

    # Fixing the window size.
    root.minsize(width=250, height=70)
    label = Tkinter.Label(root, text='Ready!', fg='black', font='Verdana 30
    bold')
    label.pack()
    f = Tkinter.Frame(root)
    start = Tkinter.Button(f, text='Start', width=6, command=lambda:
    Start(label))
    stop = Tkinter.Button(f, text='Stop', width=6, state='disabled',
    command=Stop)
    reset = Tkinter.Button(f, text='Reset', width=6, state='disabled',
    command=lambda: Reset(label))
    f.pack(anchor='center', pady=5)
```

```
start.pack(side='left')
stop.pack(side='left')
reset.pack(side='left')
root.mainloop()
```

22. Python script to compress folders and files

```
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')
```

23. Find IMDB Ratings

This script is used to retrieve the ratings and genres of films in your films folder that matches those on IMDb; the data is scraped from IMDB's official website and saved in a CSV file.

The CSV file may then be utilized for analysis, sorted by rating, and so on.

```
from bs4 import BeautifulSoup
import requests
import pandas as pd
import os

# Setting up session
s = requests.session()

# List contaiting all the films for which data has to be scraped from IMDB
films = []

# Lists contaiting web scraped data
names = []
ratings = []
genres = []

# Define path where your films are present
# For eg: "/Users/utkarsh/Desktop/films"
path = input("Enter the path where your films are: ")

# Films with extensions
filmswe = os.listdir(path)

for film in filmswe:
    # Append into my films list (without extensions)
    films.append(os.path.splitext(film)[0])
    # print(os.path.splitext(film)[0])

for line in films:
    # x = line.split(", ")
    title = line.lower()
    # release = x[1]
    query = "+".join(title.split())
    URL = "https://www.imdb.com/search/title/?title=" + query
    print(URL)
    # print(release)
    try:
        response = s.get(URL)

        #getting contect from IMDB Website
        content = response.content

        # print(response.status_code)

        soup = BeautifulSoup(response.content, features="html.parser")
        #searching all films containers found
        containers = soup.find_all("div", class_="lister-item-content")
        for result in containers:
            name1 = result.h3.a.text
            name = result.h3.a.text.lower()

            # Uncomment below lines if you want year specific as well,
define year variable before this
            # year = result.h3.find(
            # "span", class_="lister-item-year text-muted unbold"
            # ).text.lower()

            #if film found (searching using name)
            if title in name:
                #scraping rating
                rating = result.find("div",class_="inline-block ratings-
imdb-rating")["data-value"]
                #scraping genre
                genre = result.p.find("span", class_="genre")
                genre = genre.contents[0]
                #appending name, rating and genre to individual lists
                names.append(name1)
                ratings.append(rating)
                genres.append(genre)

    except Exception:
        print("Try again with valid combination of tile and release year")

#storing in pandas dataframe
df = pd.DataFrame({'Film Name':names,'Rating':ratings,'Genre':genres})

#making csv using pandas
df.to_csv('film_ratings.csv', index=False, encoding='utf-8')
```

How to run the script?

- Install the requirements.
- Inside the find_IMDb_rating.py, update the directory path.
- Type the following command: python find_IMDb_rating.py
- A CSV file with a rating will be created in the same directory as the python file.

24. Web Scrapping Youtube Comment

This script will take a URL of a youtube video and it will give CSV file for users and comments.

```
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)
```

25. 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)

```
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")
```

26. Convert Image Format

This script will convert all JPG images to PNG and PNG images to JPG in the present directory tree recursively (i.e. will change the format in images inside sub-directories too.)

```
from PIL import Image
import sys
import os

try:
  im = None
  for root, dirs, files in os.walk("."):
    for filename in files:
        if filename.endswith('.jpg'):
          im = Image.open(filename).convert("RGB")
          im.save(filename.replace('jpg', 'png'), "png")
        elif filename.endswith('.png'):
          im = Image.open(filename).convert("RGB")
          im.save(filename.replace('png', 'jpg'), "jpeg")
        else:
          print('dont have image to convert')
except IOError:
  print('directory empty!')
  sys.exit()
```

27. Random Wikipedia Article

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

Installation

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

```
from bs4 import BeautifulSoup
import requests

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

# pip install htmlparser
```

```
wiki = BeautifulSoup(res.text, "html.parser")

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

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

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

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

28. Check Website Connectivity

This script includes a basic utility for checking website connections.

Website URLs should be listed one per line in the input file websites.txt.

A two-column report with the URL of each tested site and its state is included in the output file website status.csv.

The script just checks for a 200 status code from the webserver.

Each time you run the utility, the result file will be overwritten.

```
import csv
import requests

status_dict = {"Website": "Status"}
def main():
    with open("websites.txt", "r") as fr:
        for line in fr:
            website = line.strip()
            status = requests.get(website).status_code
            status_dict[website] = "working" if status == 200 \
                else "not working"

    # print(status_dict)
    with open("website_status.csv", "w", newline="") as fw:
        csv_writers = csv.writer(fw)
        for key in status_dict.keys():
            csv_writers.writerow([key, status_dict[key]])

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

29. # Current Weather

This Script will help you to find the current weather of any entered place. This script is using openweathermap.org to find the current weather.

Requirement

To run this script you need to have api key, to get an API key you first signup here

After getting the api key to add in the code:

```
# Python program to find current weather details of any city using
openweathermap api
import requests

# Enter your API key here
api_key = "Your_API_Key"

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

# Give city name
city_name = input("Enter city name : ")

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"]
    print(" Temperature (in kelvin unit) = " +
                    str(current_temperature) +
          "\n atmospheric pressure (in hPa unit) = " +
                    str(current_pressure) +
          "\n humidity (in percentage) = " +
                    str(current_humidiy) +
          "\n description = " +
                    str(weather_description))

else:
    print(" City Not Found ")
```

30. GUI Calculator App

```python
# -*- coding: utf-8 -*-
from tkinter import Tk, END, Entry, N, E, S, W, Button
from tkinter import font
from tkinter import Label
from functools import partial


def get_input(entry, argu):
    entry.insert(END, argu)


def backspace(entry):
    input_len = len(entry.get())
    entry.delete(input_len - 1)


def clear(entry):
    entry.delete(0, END)


def calc(entry):
    input_info = entry.get()
    try:
        output = str(eval(input_info.strip()))
    except ZeroDivisionError:
        popupmsg()
        output = ""
    clear(entry)
    entry.insert(END, output)


def popupmsg():
    popup = Tk()
    popup.resizable(0, 0)
    popup.geometry("120x100")
    popup.title("Alert")
    label = Label(popup, text="Cannot divide by 0 ! \n Enter valid values")
    label.pack(side="top", fill="x", pady=10)
    B1 = Button(popup, text="Okay", bg="#DDDDDD", command=popup.destroy)
    B1.pack()


def cal():
    root = Tk()
    root.title("Calc")
    root.resizable(0, 0)

    entry_font = font.Font(size=15)
    entry = Entry(root, justify="right", font=entry_font)
    entry.grid(row=0, column=0, columnspan=4,
               sticky=N + W + S + E, padx=5, pady=5)

    cal_button_bg = '#FF6600'
    num_button_bg = '#4B4B4B'
    other_button_bg = '#DDDDDD'
    text_fg = '#FFFFFF'
    button_active_bg = '#C0C0C0'

    num_button = partial(Button, root, fg=text_fg, bg=num_button_bg,
                         padx=10, pady=3,
activebackground=button_active_bg)
    cal_button = partial(Button, root, fg=text_fg, bg=cal_button_bg,
                         padx=10, pady=3,
activebackground=button_active_bg)

    button7 = num_button(text='7', bg=num_button_bg,
                         command=lambda: get_input(entry, '7'))
    button7.grid(row=2, column=0, pady=5)

    button8 = num_button(text='8', command=lambda: get_input(entry, '8'))
    button8.grid(row=2, column=1, pady=5)

    button9 = num_button(text='9', command=lambda: get_input(entry, '9'))
    button9.grid(row=2, column=2, pady=5)

    button10 = cal_button(text='+', command=lambda: get_input(entry, '+'))
    button10.grid(row=4, column=3, pady=5)

    button4 = num_button(text='4', command=lambda: get_input(entry, '4'))
    button4.grid(row=3, column=0, pady=5)

    button5 = num_button(text='5', command=lambda: get_input(entry, '5'))
    button5.grid(row=3, column=1, pady=5)

    button6 = num_button(text='6', command=lambda: get_input(entry, '6'))
    button6.grid(row=3, column=2, pady=5)

    button11 = cal_button(text='-', command=lambda: get_input(entry, '-'))
    button11.grid(row=3, column=3, pady=5)

    button1 = num_button(text='1', command=lambda: get_input(entry, '1'))
    button1.grid(row=4, column=0, pady=5)

    button2 = num_button(text='2', command=lambda: get_input(entry, '2'))
    button2.grid(row=4, column=1, pady=5)

    button3 = num_button(text='3', command=lambda: get_input(entry, '3'))
    button3.grid(row=4, column=2, pady=5)

    button12 = cal_button(text='*', command=lambda: get_input(entry, '*'))
    button12.grid(row=2, column=3, pady=5)

    button0 = num_button(text='0', command=lambda: get_input(entry, '0'))
    #button0.grid(row=5, column=0, columnspan=2, padx=3, pady=5, sticky=N +
S + E + W)
    button0.grid(row=5, column=0,  pady=5)
    button13 = num_button(text='.', command=lambda: get_input(entry, '.'))
    button13.grid(row=5, column=1, pady=5)

    button14 = Button(root, text='/', fg=text_fg, bg=cal_button_bg,
padx=10, pady=3,
                      command=lambda: get_input(entry, '/'))
    button14.grid(row=1, column=3, pady=5)

    button15 = Button(root, text='<-', bg=other_button_bg, padx=10, pady=3,
                      command=lambda: backspace(entry),
activebackground=button_active_bg)
    button15.grid(row=1, column=0, columnspan=2,
                  padx=3, pady=5, sticky=N + S + E + W)

    button16 = Button(root, text='C', bg=other_button_bg, padx=10, pady=3,
                      command=lambda: clear(entry),
activebackground=button_active_bg)
    button16.grid(row=1, column=2, pady=5)

    button17 = Button(root, text='=', fg=text_fg, bg=cal_button_bg,
padx=10, pady=3,
                      command=lambda: calc(entry),
activebackground=button_active_bg)
    button17.grid(row=5, column=3, pady=5)

    button18 = Button(root, text='^', fg=text_fg, bg=cal_button_bg,
padx=10, pady=3,
                      command=lambda: get_input(entry, '**'))
    button18.grid(row=5, column=2, pady=5)
    def quit():
        exit['command'] = root.quit()
    exit = Button(root, text='Quit', fg='white', bg='black', command=quit,
height=1, width=7)
    exit.grid(row=6, column=1)

    root.mainloop()


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

31. Sudoku Solver

This is a script to solve a 9x9 sudoku matrix using Python.

How to use it?

1. edit app.py to add your sudoku matrix. (Fill `0` for empty cells.)

For example,

```
[[8, 1, 0, 0, 3, 0, 0, 2, 7],
[0, 6, 2, 0, 5, 0, 0, 9, 0],
[0, 7, 0, 0, 0, 0, 0, 0, 0],
[0, 9, 0, 6, 0, 0, 1, 0, 0],
[1, 0, 0, 0, 2, 0, 0, 0, 4],
[0, 0, 8, 0, 0, 5, 0, 7, 0],
[0, 0, 0, 0, 0, 0, 0, 8, 0],
[0, 2, 0, 0, 1, 0, 7, 5, 0],
[3, 8, 0, 0, 7, 0, 0, 4, 2]]
```

Run the script.

```
python3 app.py
```

This will give you output on the console. The output will contain the input sudoku matrix and the solved sudoku matrix.

INPUT =>

```
8 1 0 | 0 3 0 | 0 2 7
0 6 2 | 0 5 0 | 0 9 0
0 7 0 | 0 0 0 | 0 0 0
---------------------
0 9 0 | 6 0 0 | 1 0 0
1 0 0 | 0 2 0 | 0 0 4
0 0 8 | 0 0 5 | 0 7 0
---------------------
0 0 0 | 0 0 0 | 0 8 0
0 2 0 | 0 1 0 | 7 5 0
3 8 0 | 0 7 0 | 0 4 2
```

OUTPUT =>

```
8 1 9 | 4 3 6 | 5 2 7
4 6 2 | 7 5 1 | 3 9 8
5 7 3 | 2 9 8 | 4 1 6
---------------------
2 9 4 | 6 8 7 | 1 3 5
1 5 7 | 9 2 3 | 8 6 4
6 3 8 | 1 4 5 | 2 7 9
---------------------
7 4 5 | 3 6 2 | 9 8 1
9 2 6 | 8 1 4 | 7 5 3
3 8 1 | 5 7 9 | 6 4 2
```

```
def printsudoku(sudoku):
    print("\n\n")
    for i in range(len(sudoku)):
        line = ""
        if i == 3 or i == 6:
            print("---------------------")
        for j in range(len(sudoku[i])):
            if j == 3 or j == 6:
                line += "| "
            line += str(sudoku[i][j])+" "
        print(line)
    print("\n\n")

def findNextCellToFill(sudoku):
    for x in range(9):
        for y in range(9):
            if sudoku[x][y] == 0:
                return x, y
    return -1, -1

def isValid(sudoku, i, j, e):
    rowOk = all([e != sudoku[i][x] for x in range(9)])
    if rowOk:
        columnOk = all([e != sudoku[x][j] for x in range(9)])
        if columnOk:
            secTopX, secTopY = 3*(i//3), 3*(j//3)
            for x in range(secTopX, secTopX+3):
                for y in range(secTopY, secTopY+3):
                    if sudoku[x][y] == e:
                        return False
            return True
    return False

def solveSudoku(sudoku, i=0, j=0):
    i, j = findNextCellToFill(sudoku)
    if i == -1:
        return True
    for e in range(1, 10):
        if isValid(sudoku, i, j, e):
            sudoku[i][j] = e
            if solveSudoku(sudoku, i, j):
                return True
            sudoku[i][j] = 0
    return False
```

Click here for more details

32. File Encrypt Decrypt

A command-line Python script that can encrypt a given file and also decrypt the encrypted file.

Usage

- Encrypt file

```
$ pipenv run python crypt -e file.txt
```

- Decrypt file

```
$ pipenv run python crypt -d file.enc
```

```
import os
import argparse

from cryptography.fernet import Fernet

class Crypt:

    def __init__(self):

        # can be generated Fernet.generate_key()
        # if generated, save it below
        self.key = b'oBa5LeeJt1r4BmNyJXb6FHd1U21GMshH9Pqu_J-HzNQ='
        self.fernet = Fernet(self.key)

    def encrypt(self, input_file_path):
        """
        Encrypt a file
        """

        # split the file and take only the file name
        base_name = os.path.basename(input_file_path).split('.')
[0].split('-')[-1]

        # creates a file name with extension .enc
        output_file = f"{base_name}.enc"
        if os.path.exists(output_file):
            print(f'Encrypted File already exists')
        else:
            with open(input_file_path, 'rb') as i:
                input_data = i.read()

            encrypted = self.fernet.encrypt(input_data)

            with open(output_file, 'wb') as o:
                o.write(encrypted)
            print(f'Encrypted file: {output_file}\n')
    def decrypt(self, input_file_path, output_file_ext='txt'):
        """
        Decrypt an already encrypted file
        """

        # split the file and take only the file name
        base_name = os.path.basename(input_file_path).split('.')
[0].split('-')[-1]
        output_file = f'{base_name}.{output_file_ext}'
        with open(input_file_path, 'rb') as f:
            file_data = f.read()

        decrypted = str(self.fernet.decrypt(file_data), 'utf-8')

        with open(output_file, 'w') as o:
            o.write(decrypted)
        print(f'Decrypted file: {output_file}\n')


if __name__ == '__main__':
    crypt = Crypt()

    parser = argparse.ArgumentParser(
        description=__doc__,
        formatter_class=argparse.RawDescriptionHelpFormatter)
    parser.add_argument('-e', '--encrypt',
                        help='Encrpyt the file')
    parser.add_argument('-d', '--decrypt',
                        help='Decrypt the file')

    args = parser.parse_args()

    if args.encrypt:
        print(f'Input file: {args.encrypt}')
        crypt.encrypt(args.encrypt)
    elif args.decrypt:
        print(f'Input file: {args.decrypt}')
        crypt.decrypt(args.decrypt)
```

Click here for more details

33. Location Of Adress

This script will convert your address to coordinates.

```
import geocoder
t=input("enter the location:")
g = geocoder.arcgis(t)
print(g.latlng)
```

34. Automated Email

Automated email python script You can now send emails to multiple people at once easily with only a few clicks using Smtplib module in Python

Requirement: Python version 3, Smtplib, and JSON

```
pip install smtplib
pip install json
```

Can be run easily using command prompt (Python automated_email.py) -> login as you would for your Gmail account(same email and password) -> find your way with the intuitive user-friendly menu (your passwords and emails are only stored on your local device and no one has access to your information otherwise!)

```python
from smtplib import SMTP as smtp
import json

def sendmail(sender_add, reciever_add, msg, password):
    server = smtp('smtp.gmail.com:587')
    server.starttls()
    server.login(sender_add, password)
    server.sendmail(sender_add, reciever_add, msg)
    print("Mail sent succesfully....!")


group = {}
print('\t\t ......LOGIN.....')
your_add = input('Enter your email address :')
password = input('Enter your email password for login:')
print('\n\n\n\n')
choice = 'y'
while(choice != '3' or choice != 'no'):
    print("\n 1.Create a group\n2.Message a group\n3.Exit")
    choice = input()
    if choice == '1':
        ch = 'y'
        while(ch != 'n'):
            gname = input('Enter name of group :')
            group[gname] = input('Enter contact emails separated by a
single space :').rstrip()
            ch = input('Add another....y/n? :').rstrip()
        with open('groups.json', 'a') as f:
            json.dump(group, f)
    elif choice == '2':
        gname = input('Enter name of group :')
        try:
            f = open('groups.json', 'r')
            members = json.load(f)
            f.close()
        except:
            print('Invalid group name. Please Create group first')
            exit
        members = members[gname].split()
        msg = input('Enter message :')
        for i in members:
            try:
                sendmail(your_add, i, msg, password)
            except:
                print("An unexpected error occured. Please try again
later...")
                continue
    else:
        break
```

35. Artificial Intelligence Chatbot

What is an AI bot?

A chatbot (also known as a talkbot, chatterbot, Bot, IM bot, interactive agent, or Artificial Conversational Entity) is a computer program or an artificial intelligence which conducts a conversation via auditory or textual methods.

How to use it?

Start by running the below command on your UNIX terminal or Windows CMD:

```
$ python bash.py
```

```
import sys
try:
    import aiml
except ImportError:
    print('[!] Failed to import the module')
    try:
        select = raw_input('[*] Attempt to auto-install aiml? [Y/n')
    except KeyboardInterrupt:
        print('\n[!] User Cancel')
        sys.exit(5)
    if select.strip().lower()[0] == 'y':
        print('[*] Trying to Install aiml... ')
        sys.stdout.flush()
        try:
            import pip
            pip.main(['install', '-q', 'aiml'])
            import aiml
            print('Finished')
        except Exception:
            print('Something happens PLease check your internet
connection')
            sys.exit(5)

    elif select.strip().lower()[0] == 'n':
        print('[*] User cancel Auto-install')
        sys.exit(5)


kern = aiml.Kernel()
kern.learn('load.xml')
kern.respond('load aiml b')

while True:
    print(kern.respond(raw_input('Type your Message >>')))
```

Click here for more details

36. Bitcoin Price GUI Application

Tells the current price of bitcoin using Python's Tkinter library and using the free Luno API. Press refresh to get the latest price.

How to use

```
> _python bitcoin-price.py_
```

or

```
> _python3 bitcoin-price.py_
```

The default pair is Bitcoin to Malaysian Ringgit, to change that, click here to find out what tickers are available and change `XBTMYR` in line 9.

```
import tkinter as tk
from tkinter import ttk
import urllib.request
import json
import time

def get_luno():
    # to change ticker pair, look at here
https://api.mybitx.com/api/1/tickers
    req = urllib.request.urlopen("https://api.mybitx.com/api/1/ticker?
pair=XBTMYR")
    x = json.loads(req.read().decode("utf-8"))
    req.close()
    return x

def refresh_price():
    aLable.configure(text="Ask price: RM " + get_luno()["ask"])
    bLable.configure(text="Time: " +
        str(time.strftime("%Y-%m-%d %H:%M:%S",
        time.gmtime(get_luno()["timestamp"]/1000 + 28800))))

win = tk.Tk()
win.title("Bitcoin price in MYR")

aLable = ttk.Label(win, text="Ask price: RM " + get_luno()["ask"])
aLable.grid(column=0, row=0, padx=8, pady=4)

bLable = ttk.Label(text="Time: " +
        str(time.strftime("%Y-%m-%d %H:%M:%S",
        time.gmtime(get_luno()["timestamp"]/1000 + 28800))))
bLable.grid(column=0, row=1, padx=8, pady=4)

action = ttk.Button(win, text="Refresh", command=refresh_price)
action.grid(column=0, row=2, padx=8, pady=4)

win.mainloop()
```

37. Codechef Automated Submission

A simple script to submit your code on [https://www.codechef.com] using selenium.

```
from selenium import webdriver
import getpass
import time

username = "username"
password = getpass.getpass("Password:")

problem = 'TEST'

code = """
#include <iostream>

int main(void) {
char c, d=10;
while(std::cin.get(c) && (c!='2' || d!='4') && std::cout.put(d))
d=c;
}
"""

browser = webdriver.Firefox()

browser.get('https://www.codechef.com')
```

```
nameElem = browser.find_element_by_id('edit-name')
nameElem.send_keys(username)

passElem = browser.find_element_by_id('edit-pass')
passElem.send_keys(password)

browser.find_element_by_id('edit-submit').click()

browser.get("https://www.codechef.com/submit/" + problem)

time.sleep(20)

browser.find_element_by_id("edit_area_toggle_checkbox_edit-
program").click()

inputElem = browser.find_element_by_id('edit-program')
inputElem.send_keys(code)

browser.find_element_by_id("edit-submit").click()
```

38. Checksum

This script can generate checksums from md5, sha1, sha224, sha256, sha384, and sha512. Additionally, for another layer of secret, it can create signed checksums using HMAC and a provided secret. Lastly, to provide actual value to the script it can also verify if a checksum matches the file it was generated from.

Examples:

Generate a sha1 checksum

```
python checksum.py -H sha1 -f test.txt -g
# b29d28bc5239dbc2689215811b2a73588609f301
```

Generate a signature

```
python checksum.py -f test.txt -s secret
# 3YYMCthY4hFxQj1wPF3uAg==
```

Verify a checksum

```
python -H sha1 -f test.txt -v b29d28bc5239dbc2689215811b2a73588609f301
```

Verify a signature

```
python -f test.txt -s secret -v 3YYMCthY4hFxQj1wPF3uAg==
```

```
import os
import sys
import hmac
import base64
import hashlib
import argparse

def checksum(hash, seed=None):
    hashs = {
        "md5": hashlib.md5,
        "sha1": hashlib.sha1,
        "sha224": hashlib.sha224,
        "sha256": hashlib.sha256,
        "sha384": hashlib.sha384,
        "sha512": hashlib.sha512
    }
    method = hashs.get(hash, hashlib.md5)()
    if seed is not None:
        method.update(seed.encode("utf-8"))
    else:
        method.update(os.urandom(32))
    return method.hexdigest()

def sign(hash, message, secret):
    hashs = {
        "md5": hashlib.md5,
        "sha1": hashlib.sha1,
        "sha224": hashlib.sha224,
        "sha256": hashlib.sha256,
        "sha384": hashlib.sha384,
        "sha512": hashlib.sha512
    }
    method = hashs.get(hash, hashlib.md5)()
    digest = hmac.new(secret.encode("utf-8"),
                msg=message.encode(),
                digestmod=hashs.get(hash, hashlib.md5)).digest()
    signature = base64.b64encode(digest).decode("utf-8")
    return signature

def verify(hash, input, check, secret=None):
    challenge = None
    if secret is not None:
        challenge = sign(hash, input, secret)
    else:
        challenge = checksum(hash, input)
    return "Valid! :D" if challenge == check else "Invalid :("

def main():
    description = "Checksum tool to generate, sign, and verify"
    parser = argparse.ArgumentParser(description=description)
    parser.add_argument("-g", "--generate", dest="generate",
        action="store_true", help="Generates checksum")
    parser.add_argument("-s", "--sign", dest="sign", default=None,
        help="Signs input using HMAC")
    parser.add_argument("-H", "--hash", dest="hash", default="md5",
        help="Hash method (md5, sha1, sha224, sha256, sha384, sha512)")
    parser.add_argument("-v", "--verify", dest="verify", default=None,
        help="Checksum or signature used to verify against file / stdin")
    parser.add_argument("-f", "--file", dest="file",
        type=argparse.FileType("r"), default=sys.stdin,
        help="File / stdin to create checksum, make signature, or verify
from")
    arguments = parser.parse_args()

    if arguments.verify is not None:
        if not arguments.file:
            print("Missing input to generate checksum from")
            sys.exit(1)
        if arguments.sign is not None:
            print(verify(arguments.hash, arguments.file.read(),
                         arguments.verify, arguments.sign))
            return
        else:
            print(verify(arguments.hash, arguments.file.read(),
                         arguments.verify))
            return
    elif arguments.generate:
        if not arguments.file:
            print("Missing input to generate checksum from")
            sys.exit(1)
        print(checksum(arguments.hash, arguments.file.read()))
        return
    elif arguments.sign is not None:
        if not arguments.file:
            print("Missing input to generate checksum from")
            sys.exit(1)
        print(sign(arguments.hash, arguments.file.read(), arguments.sign))
        return
    print("Missing function (-g, -s, -v)")
    sys.exit(1)

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

39. Cryptocurrency Converter

A simple GUI of a cryptocurrency converter implemented in Python using PyQt. The UI was designed using Qt Creator.

Requirement

```
Python 3.xx PyQt5 requests
```

Usage

To start converting your cryptocurrency to USD, EUR, or other cryptocurrencies type:

```
$ virtualenv crypto-env
$ source crypto-env/bin/activate
$ pip3 install -r requirements.txt
$ python CryptoConverter.py
```

```
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from MainWindow import Ui_MainWindow
import json
import requests


class MainWindow(QMainWindow, Ui_MainWindow):
    def __init__(self, *args, **kwargs):
        super(MainWindow, self).__init__(*args, **kwargs)
        self.setupUi(self)
        self.show()
        # Vars
        self.new_label = '0'
        self.cur1 = 'BTC'
        self.cur2 = 'USD'
        self.result = ''
        # Connect buttons
        for n in range(0, 10):
            getattr(self, 'pushButton_n%s' %
n).clicked.connect(self.digit_pressed)
        self.pushButton_n10.clicked.connect(self.decimal_point)
        self.pushButton_del.clicked.connect(self.del_digit)
        self.pushButton_convert.clicked.connect(self.convert_fun)
        self.comboBox.activated[str].connect(self.currencies1)
        self.comboBox_2.activated[str].connect(self.currencies2)

    def digit_pressed(self):
        button = self.sender()
        self.new_label = self.label_1.text() + button.text()
        if '.' in self.new_label:
            self.label_1.setText(str(self.new_label))
        else:
            self.label_1.setText(str(int(self.new_label)))

    def decimal_point(self):
        if '.' in self.label_1.text():
            pass
        else:
            self.label_1.setText(self.label_1.text() + '.')

    def del_digit(self):
        self.new_label = self.new_label[:-1]
        self.label_1.setText(self.new_label)

    def currencies1(self, item1):
        self.cur1 = item1
        # print(self.cur1)

    def currencies2(self, item2):
        self.cur2 = item2
        # print(self.cur2)

    # Live data from API
    def api(self, cur1, cur2):
        api_link = "https://min-api.cryptocompare.com/data/pricemulti?
fsyms={}&tsyms={}".format(cur1, cur2)
        resp = requests.get(api_link)
        # print(r.status_code)
        data = json.loads(resp.content)
        # print(data)
        var = data[self.cur1][self.cur2]
        return var

    def convert_fun(self):
        try:
            if len(self.new_label) == 0:
                self.label_1.setText('0')
                self.label_2.setText('0')
            if '.' in self.new_label:
                self.result = float(self.new_label) * self.api(self.cur1,
self.cur2)
                self.result = round(self.result, 2)
                self.label_2.setText(str(self.result))
            else:
                self.result = int(self.new_label) * self.api(self.cur1,
self.cur2)
                self.result = round(self.result, 2)
                self.label_2.setText(str(self.result))
        except (KeyError, ValueError):
            pass
        except requests.exceptions.ConnectionError:
            print('Please verify your internet connection!')


if __name__ == '__main__':
    app = QApplication([])
    app.setApplicationName("CryptoConverter")
    window = MainWindow()
    app.exec_()
```

40. Cryptocurrency Prices

This program gets the live price of cryptocurrencies.

Requirements

Install the required libraries:

```
$ pip install requests bs4 colorama
```

After that run with:

```
$ python cryptocurrency-prices.py
```

```
#!python3
# -*- coding: utf-8 -*-

import requests
from bs4 import BeautifulSoup
from colorama import init, Fore, Back, Style
import sys
import os

#get the price
def get_price():
    #response from the url
    response = requests.get(url)

    #soup object of the html content
    soup = BeautifulSoup(response.content,'html.parser')

    #for bitcoin
    if asset == 'btc':
        price = soup.find('span',{'class':'price'}).text #bitcoin works
faster with the price class

    #for other altcoins
    else:
        price = soup.find('span',{'class':'woobJfK-Xb2EM1W1o8yoE'}).text
#other altcoins only work with this class

    return float(price.replace(",",""))

#asset choice
asset = input('Abbreviation of the asset: ')
url = 'https://cryptowat.ch/assets/' + asset
```

```
#catching the NoneType AttributeError error for coins that cant be found
try:
    price = get_price()

except AttributeError:
    print("The asset doesn't exist or it's not supported!")
    sys.exit()

#visual
if sys.platform == 'win32':
    os.system('cls')
else:
    os.system('clear')

#since the last price must be something from the start its set to 0
price = 0

#loop
while True:

    #getting the price
    last_price = price
    price = get_price()

    #coloring the price according to the change
    if price > last_price:
        color = Fore.GREEN
    elif last_price > price:
        color = Fore.RED
    else:
        color = Style.RESET_ALL

    #printing the price
    print('$ ',end='')
    print(color + str(price) + Style.RESET_ALL)
```

41. Colored Image to Black & White Image Converter

A simple Python script that takes a colored image filename as an argument and converts it into a grayscale image and saves the output image file. It shows the basic usage of the Pillow library.

Libraries Required

1. Pillow (PIL) `$pip install Pillow`

Usage

1. Go to the script's folder and open the command prompt.
2. Run command : `$python bw_convert.py <image_file_name>`

Example

`$python bw_convert.py sample_image.jpg` `$python bw_convert.py sample_image.png`

```
import sys
from PIL import Image
from PIL.ExifTags import TAGS

image_file = sys.argv[1]
image_name = image_file.split(".")[0]

try:
    image = Image.open(image_file)
except IOError:
    print("Error in loading image!!")
    sys.exit(1)

bw_image = image.convert('L')
bw_image.save("bw_"+image_name+".png")
```

42. CricBuzz Score Update

This Python script, based on web scraping, fetches the score of the recent cricket matches using the Beautiful soup library to scrape the data.

How to run

Type in this command in the terminal.

```
python3 cricbuzz_scrap.py
```

It will display score updates of all the recent matches.

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

quote_page = 'http://www.cricbuzz.com/cricket-match/live-scores'
page = urlopen(quote_page)
soup = BeautifulSoup(page,'html.parser')

update=[]

for score in soup.find_all('div',attrs={'class':'cb-col cb-col-100 cb-lv-main'}):
    s=score.text.strip()
    update.append(s)

print('-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-')
```

```
for i in range(len(update)):
    print(i+1),
    print(update[i])
print('-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-*-*-*-*-*-*-*')
```

43. CSV to Excel

This programme writes the data in any Comma-separated value file (such as: .csv or .data) to a Excel file.

Requirements

Install the required libraries:

```
$ pip install openpyxl
```

After that run with:

```
$ python main.py
```

Example input file

```
#!python3
# -*- coding: utf-8 -*-

import openpyxl
import sys

#inputs
print("This programme writes the data in any Comma-separated value file
(such as: .csv or .data) to a Excel file.")
print("The input and output files must be in the same directory of the
python file for the programme to work.\n")

csv_name = input("Name of the CSV file for input (with the extension): ")
sep = input("Seperator of the CSV file: ")
excel_name = input("Name of the excel file for output (with the extension):
")
sheet_name = input("Name of the excel sheet for output: ")

#opening the files
try:
    wb = openpyxl.load_workbook(excel_name)
    sheet = wb.get_sheet_by_name(sheet_name)

    file = open(csv_name,"r",encoding = "utf-8")
```

```
except:
    print("File Error!")
    sys.exit()

#rows and columns
row = 1
column = 1

#for each line in the file
for line in file:
    #remove the \n from the line and make it a list with the seperator
    line = line[:-1]
    line = line.split(sep)

    #for each data in the line
    for data in line:
        #write the data to the cell
        sheet.cell(row,column).value = data
        #after each data column number increases by 1
        column += 1

    #to write the next line column number is set to 1 and row number is
increased by 1
    column = 1
    row += 1

#saving the excel file and closing the csv file
wb.save(excel_name)
file.close()
```

44. Current City Weather

Uses a GET request to retrieve details on your current weather details within your city, simply insert the name of any city and it will provide details such as current temperature, current wind speed, and current weather type.

```
import requests


def get_temperature(json_data):
    temp_in_celcius = json_data['main']['temp']
    return temp_in_celcius

def get_weather_type(json_data):
    weather_type = json_data['weather'][0]['description']
    return weather_type

def get_wind_speed(json_data):
    wind_speed = json_data['wind']['speed']
    return wind_speed
```

```
def get_weather_data(json_data, city):
    description_of_weather = json_data['weather'][0]['description']
    weather_type = get_weather_type(json_data)
    temperature = get_temperature(json_data)
    wind_speed = get_wind_speed(json_data)
    weather_details = ''
    return weather_details + ("The weather in {} is currently {} with a
temperature of {} degrees and wind speeds reaching {} km/ph".format(city,
weather_type, temperature, wind_speed))


def main():
    api_address = 'https://api.openweathermap.org/data/2.5/weather?
q=Sydney,au&appid=a10fd8a212e47edf8d946f26fb4cdef8&q='
    city = input("City Name : ")
    units_format = "&units=metric"
    final_url = api_address + city + units_format
    json_data = requests.get(final_url).json()
    weather_details = get_weather_data(json_data, city)
    # print formatted data
    print(weather_details)

main()
```

45. Directory Organizer

Organizes the files in the given directory according to their type(common extensions). Unrecognized extensions are kept in the parent directory itself while others are moved to respective new directories like 'Pictures' etc.

python main.py [-h] path_to_the_directory

```
#!/usr/bin/python3

import argparse
import os


def path():
    parse = argparse.ArgumentParser(
        add_help=True, description="Organize your files to different
directories according to their type")
    parse.add_argument('directory_path', type=str, default='./',
                       help="The absolute path to the directory")
    return parse.parse_args().directory_path


documents = ['.log', '.txt', '.doc', '.docx', '.md', '.pdf', '.wps']
picture = ['.png', '.jpg', 'jpeg', '.bmp']
```

```
music = ['.mp3', '.wav']
compressed = ['.zip', '.rar', '.tar', '.gz', '.bz2', '.xz']
video = ['.3gp', '.mov', '.mp4', '.mkv', '.srt', '.avi']
web = ['.html', .'.css', '.js']
source = ['.py', '.c', '.cpp', '.java',]


directories = [path() + '/Compressed', path() + '/Documents',
               path() + '/Pictures', path() + '/Music', path() + '/Video',
path() + '/Web', path() + '/Source-codes',]

print("This will organize your files to different directories according to
their type!!")
print("Are you sure you want to continue? (y/n)")
flag = input('>>>')
if flag.lower() == 'y':
    try:
        for d in directories:
            os.mkdir(d)
    except FileExistsError:
        pass

    for files in os.listdir(path()):
        dot = (files.rfind('.'))
        if dot is not 0 and dot is not -1:
            if files[dot:].lower() in music:
                os.rename(path() + '/' + files, path() + '/Music/' + files)
            if files[dot:].lower() in picture:
                os.rename(path() + '/' + files, path() + '/Pictures/' +
files)
            if files[dot:].lower() in documents:
                os.rename(path() + '/' + files, path() + '/Documents/' +
files)
            if files[dot:].lower() in compressed:
                os.rename(path() + '/' + files, path() +
                          '/Compressed/' + files)
            if files[dot:].lower() in video:
                os.rename(path() + '/' + files, path() + '/Video/' + files)
            if files[dot:].lower() in web:
                os.rename(path() + '/' + files, path() + '/Web/' + files)
            if files[dot:].lower() in source:
                os.rename(path() + '/' + files, path() + '/Source-codes/' +
files)

    for d in directories:
        if os.listdir(d) is None:
            os.removedirs(d)
else:
    print("Exiting")
    os.sys.exit(0)
```

46. Excel Files Merger

A simple script that Excel files with a similar table structure from a given path as input and creates a unified excel workbook.

Libraries Required

1. openpyxl `$pip install openpyxl`

Usage

A sample script 'Combine excel files into 1.py' has been provided to show the usage of the Excel Merger. When the script is run, it will ask for the name of the Unified Workbook and the path of the folder containing the excel files that need to be merged.

```
from openpyxl import load_workbook
from openpyxl import Workbook
import os


# Read data from active worksheet and return it as a list
def reader(file):
    global path
    abs_file = os.path.join(path, file)
    wb_sheet = load_workbook(abs_file).active
    rows = []
    # min_row is set to 2, ignore the first row which contains headers
    for row in wb_sheet.iter_rows(min_row=2):
        row_data = []
        for cell in row:
            row_data.append(cell.value)
        rows.append(row_data)
    return rows


# You can replace these with your own headers for the table
headers = ['Nume', 'Prenume', 'Titlu', 'Editura', 'Cota', 'Pret', 'An']
# Unified excel name
workbook_name = input('Unified Workbook name ')
book = Workbook()
sheet = book.active
# Specify path
path = input('Path: ')
# Get all files from folder
files = os.listdir(path)
for file in files:
    rows = reader(file)
    for row in rows:
        sheet.append(row)
    book.save(filename=workbook_name)
```

47. Extended IP address info

View extended info about your public IP address from the terminal.

The python script runs `curl` with the following parameters

`curl -H "Accept: application/json" [https://ipinfo.io/json](https://ipinfo.io/json)`

Run program

```
python extended_ip_address_info.py
```

Output

The output should be in the form of the following:

```
{
  "ip": "xxx.xxx.xxx.xxx",
  "city": "A_city",
  "hostname": "host.isp-website.com",
  "region": "A_region",
  "country": "Country code",
  "loc": "coordinates",
  "org": "AS-number ISP-name",
  "postal": "postal-code",
  "timezone": "Europe/City",
  "readme": "https://ipinfo.io/missingauth"
}
```

```
#!/bin/python
# -*- coding: utf-8 -*-

# Using curl to get data from https://ipinfo.io/json
# Template from pycurl documentation
# http://pycurl.io/docs/latest/quickstart.html#examining-response-headers

import pycurl #curl library
import certifi #HTTP over TLS/SSL library
from io import BytesIO #Buffered I/O implementation using an in-memory
bytes buffer.

#set header, '--header' or -H
header = ['Accept: application/json']

buffer = BytesIO()
c = pycurl.Curl() #curl
c.setopt(c.HTTPHEADER, header) #header
c.setopt(c.URL, 'https://ipinfo.io/json') #URL
c.setopt(c.WRITEDATA, buffer)
```

```
c.setopt(c.CAINFO, certifi.where()) # SSL certificates
c.perform()
c.close()

body = buffer.getvalue()
# Body is a byte string.
# We have to know the encoding in order to print it to a text file
# such as standard output.
print(body.decode('iso-8859-1'))
```

48. Excel to Python List of List Converter

A simple tool that reads an excel file and any corresponding sheet, and converts it to a python list of list data structure.

Libraries Required

1. xlrd `$pip install xlrd`

Usage

A sample script `excel_to_list_usage.py` has been provided to show the usage of the ExcelToList. It reads the excel and its sheet, and prints the list of lists.

```
import xlrd
import sys

class ExcelToList():

    def __init__(self, file, sheet):
        self.file = file
        self.sheet = sheet

    def convert(self):
        converted_list = []
        inputexcel = xlrd.open_workbook(self.file)
        inputsheet = inputexcel.sheet_by_name(self.sheet)
        numberofrows = inputsheet.nrows
        numberofcols = inputsheet.ncols
        start_row,start_col = 0,0
        for current_row in range(start_row,numberofrows):
            currentlist = []
            for current_col in range(start_col,numberofcols):

currentlist.append(inputsheet.cell(current_row,current_col).value)
            converted_list.append(currentlist)
        return converted_list
```

Click here for more details

49. File Explorer Dialog Box in Python

Open file explorer dialog box UI to select files using Python.

1. Using tkinter

Example using tkinter:

```
$ python select_file_tk.py
```

```python
import tkinter as tk
from tkinter import filedialog

root = tk.Tk()
root.withdraw()

file_path = filedialog.askopenfilename()
print(file_path)
```

2. Using PyQt

Install PyQt5

Example using PyQt5:

```
$ python select_file_pyqt.py
```

```python
from PyQt5.QtWidgets import QFileDialog, QApplication
from PyQt5 import QtWidgets


def select_files(directory_location=None):
    qtapp = QApplication([directory_location])
    qtwgt = QtWidgets.QWidget()
    filenames, _ = QFileDialog.getOpenFileNames(qtwgt)
    return filenames


def main():
    filenames = select_files()
    print("You selected:\n", "\n".join(filename for filename in filenames))


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

50. File-Sharing-Bot

File-Sharing telegram Bot developed in Python It is like a centralized file repository where authorized users can share files and files are available to all the users. Functionalities and commands can be seen using the/help command. This bot can be directly hosted on Heroku.

```
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters
import logging
import os
import telegram
import shutil
# Enable logging
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %
(message)s',
                    level=logging.INFO)

logger = logging.getLogger(__name__)

#list of authorized users
#create a list of telegram usernames to authorise them, 0th username is
admin.
username_list = []
# Define a few command handlers. These usually take the two arguments bot
and
# update. Error handlers also receive the raised TelegramError object in
error.
def start(bot, update):
    """Send a message when the command /start is issued."""
    reply = "Welcome to World of Automation. \nI am a bot developed by a
Lazy Programmer.\nSend /help command to see what i can do."
    update.message.reply_text(reply)


def help(bot, update):
    """Send a message when the command /help is issued."""
    admin = update.message.from_user.username
    if admin == username_list[0]:
        reply = '''Send /get folder_name/file_name.extension to receive a
file.
                \nSend /ls folder_name to show list of files.
                \nSend /put folder_name/file_name.extension to upload last
sent file.
                \nSend /mkdir folder_name to create a Folder.
                \nSend /remove folder_name/filename.extension to delete a
file.
                \nSend /adduser username to give access.
                \nSend /removeuser username to revoke access.
                \nSend /showuser to show list of users
                '''
    else:
        reply = '''Send /get folder_name/file_name.extension to receive a
```

```
file.
                \nSend /ls folder_name to show list of files.
                \nSend /put folder_name/file_name.extension to upload last
sent file.
                \nSend /mkdir folder_name to create a Folder.
                '''
    update.message.reply_text(reply)


def get(bot, update):
    """Send requested file."""
    username = update.message.from_user.username
    if(username not in username_list):
        update.message.reply_text("You are not Authorized.")
        return
    file = update.message.text.split(" ")[-1]
    if(file == "/send"):
        update.message.reply_text("Invalid File name.")
    else:
        reply = "Findind and Sending a requested file to you. Hold on..."
        update.message.reply_text(reply)
        path = os.getcwd()+'/'+file
        if (os.path.exists(path)):

bot.send_document(chat_id=update.message.chat_id,document=open(path, 'rb'),
timeout = 100)
        else:
            update.message.reply_text("File not Found.")

def ls(bot, update):
    """Show files in requested directory."""
    username = update.message.from_user.username
    if(username not in username_list):
        update.message.reply_text("You are not Authorized.")
        return
    file = update.message.text.split(" ")[-1]
    if(file == "/show"):
        update.message.reply_text("Invalid Directory name.")
    else:
        reply = "Findind and Sending a list of files to you. Hold on..."
        update.message.reply_text(reply)
        path = os.getcwd()+'/'+file
        if (os.path.exists(path)):
            update.message.reply_text(os.listdir(path))
        else:
            update.message.reply_text("Directory not Found.")

def put(bot, update):
    f = open(str(os.getcwd())+"/file", "r")
    file_id = f.read()
    f.close
    if file_id == "":
        update.message.reply_text("You didn't upload file.")
    else:
            new_file = bot.get_file(file_id)
            message = update.message.text.split(" ")
            path = message[-1]
            if len(path) < 1:
                update.message.reply_text("Enter Path correctly.")
            else:
                new_file.download(os.getcwd()+'/'+path)
                update.message.reply_text("File Stored.")


def mkdir(bot, update):
    message = update.message.text.split(" ")
    if len(message) < 1 or message[-1] == "/mkdir":
        update.message.reply_text("Invalid Syntax. Refer syntax in help
section.")
        return
    path = os.getcwd() + "/" + message[-1]
    os.mkdir(path)
    update.message.reply_text("Folder Created.")


def echo(bot, update):
    """Echo the user message."""
    if update.message.document:
        file_id = update.message.document.file_id
        f = open(str(os.getcwd())+"/file", "w")
        f.write(file_id)
        f.close
        update.message.reply_text("Received.Now send file name and location
to store. using /put command")
    else:
        reply = "Invalid Input."
        update.message.reply_text(reply)

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

def add_user(bot, update):
    admin = update.message.from_user.username
    if admin == username_list[0]:
        username = update.message.text.split(" ")[-1]
        username_list.append(username)
        update.message.reply_text("User added.")
    else:
        update.message.reply_text("You are not Authorized.")

def show_user(bot, update):
    admin = update.message.from_user.username
    if admin == username_list[0]:
        update.message.reply_text(username_list)
    else:
        update.message.reply_text("You are not Authorized.")
```

```
def remove_user(bot, update):
    admin = update.message.from_user.username
    if admin == username_list[0]:
        username = update.message.text.split(" ")[-1]
        username_list.remove(username)
        update.message.reply_text("User Removed.")
    else:
        update.message.reply_text("You are not Authorized.")

def remove(bot, update):
    admin = update.message.from_user.username
    if admin == username_list[0]:
        filename = update.message.text.split(" ")[-1]
        os.remove(os.getcwd()+ "/" + filename)
        update.message.reply_text("File Removed.")
    else:
        update.message.reply_text("You are not Authorized.")

def rmdir(bot, update):
    admin = update.message.from_user.username
    if admin == username_list[0]:
        filename = update.message.text.split(" ")[-1]
        shutil.rmtree(os.getcwd()+ "/" + filename)
        update.message.reply_text("Folder Removed.")
    else:
        update.message.reply_text("You are not Authorized.")

def main():
    """Start the bot."""
    # Create the EventHandler and pass it your bot's token.
    TOKEN = os.environ['TOKEN']
    updater = Updater(TOKEN)

    # Get the dispatcher to register handlers
    dp = updater.dispatcher

    # on different commands - answer in Telegram
    dp.add_handler(CommandHandler("start", start))
    dp.add_handler(CommandHandler("help", help))
    dp.add_handler(CommandHandler("get", get))
    dp.add_handler(CommandHandler("ls", ls))
    dp.add_handler(CommandHandler("put", put))
    dp.add_handler(CommandHandler("mkdir", mkdir))

    #admin functionalities
    dp.add_handler(CommandHandler("adduser", add_user))
    dp.add_handler(CommandHandler("showuser", show_user))
    dp.add_handler(CommandHandler("removeUser", remove_user))
    dp.add_handler(CommandHandler("remove", remove))
    dp.add_handler(CommandHandler("rmdir", rmdir))

    # on noncommand i.e message - echo the message on Telegram
    dp.add_handler(MessageHandler(Filters.document, echo))
    # log all errors
    dp.add_error_handler(error)

    # Start the Bot
    updater.start_polling()

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


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

Click here for more details

Summary

In this article, I have discussed 50 Python scripts or 50 mini-projects that you can create to brush up on your Python skills. I hope these will help you learn something new. Clap 👏 on this post to learn more posts from this series of posts soon.

Thank you for reading this article, don't forget to follow me on ***Medium*** *or* ***Twitter*** *to read more articles like this. You can also share this story with your friends if you find it helpful for others.*

Thank you for reading!

Buy Me A Coffee