# SITE CONNECTIVITY CHECKER / WEBSITE MONITORING PROJECT GURUKUL

#### **About website monitor in python project:**

Sometimes we go to a website only to find it down or under maintenance. This can get frustrating real quick when we need some crucial information from the website and that is when a website monitor can help. We simply create a python program to notify us with the status of the website through email

#### **Website monitor in python project:**

To implement the project, we need to use libraries that already are in python by simply importing.

#### **Project Prerequisites:**

This project is for beginners, hence no prerequisite knowledge is required

#### **Download code:**

You can download the source code from the given link: <add link>

#### **Project File Structure:**

Below is the flow of the project.

1. Importing necessary libraries
2. Declaring functions to send emails and monitor website
3. Read the inputs

##### **Importing necessary libraries:**

#Import libraries

import urllib.request

import smtplib,time, hashlib

**Code Explanation:**

* **import urllib.request:** To open a website and obtain the status, we use urllib’s request.
* **smtplib:** To establish an SMTP connection with gmail, we use the library.
* **time:** To create the delay to monitor between, we use time.
* **hashlib:** To create hashcodes using which we can observe any changes in the website.

##### **Declaring functions to send emails and monitor website:**

#Function to send email

def send\_email(email\_string):

#Fill credentials for sender's email and receiver's email

email\_from = 'enter sender\'s email'

password = 'enter password'

email\_to = 'enter reciever\'s email'

#Enter subject line

subject = "Status of website"

#Connect to gmail's smtp server

smtp\_server = smtplib.SMTP\_SSL("smtp.gmail.com", 465)

#Login to the gmail server

smtp\_server.login(email\_from, password)

#Content of email

message = f"Subject: {subject}\n\n{email\_string}"

#Send email through the smtp server

smtp\_server.sendmail(email\_from, email\_to, message)

#Close the server connection

smtp\_server.close()

**Code explanation:**

* **def send\_email(email\_string):** Declare the function send\_email() with the parameter email\_string.
* **Email\_from, password, email\_to:** Enter email details of the sender and the receiver’s email
* **subject:** Enter subject of the email. It is optional, but a good practise to keep one.
* **smtp\_server = smtplib.SMTP\_SSL("smtp.gmail.com", 465):** Connect to the smtp server of gmail on port 465 using smtplib.SMTL\_SSL. SSL is a security certificate
* **smtp\_server.login:** Connect to the server by logging in using the credentials.
* **Message:** Create a message to send in the body of the email
* **smtp\_server.sendmail:** Send the email using sendemail() with the parameters: sender’s email, receiver’s email and the message to send and close the server connection using close()

#Monitor the website

def monitor\_website():

#Run the loop to keep monitoring

while True:

#Visit the website to know if it is up

status = urllib.request.urlopen(input\_website).getcode()

#If it returns 200, the website is up

if status != 200:

#Call email function

send\_email("The website is down")

else:

send\_email("The website is up")

#Open url and create the hash code

response = urllib.request.urlopen(input\_website).read()

current\_hash = hashlib.sha224(response).hexdigest()

#Revisit the website after time delay

time.sleep(time\_delay)

#Visit the website after delay, and generate the new website

response = urllib.request.urlopen(input\_website).read()

new\_hash = hashlib.sha224(response).hexdigest()

#Check the hash codes

if new\_hash != current\_hash:

send\_email("The website changed")

**Code explanation:**

* **def monitor\_website():** Declare the function monitor\_website() to monitor the website.
* **while True:** To keep monitoring until the user terminates the program, create an infinite while loop
* **status = urllib.request.urlopen(input\_website).getcode():** Open the website the user wishes to monitor and using getcode(), obtain the status of the website. If it returns 200, it implies the website is up and not otherwise.
* **if status != 200:** If the status code is not 220, send an email stating the website being down or so on.
* **current\_hash = hashlib.sha224(response).hexdigest():** Create a hash code for the retrieved website using SHA224.
* **time.sleep(time\_delay):** Wait for a time interval before rechecking the website and obtaining its status
* **response, new\_hash:** Revisit the website and create a new hash code
* **new\_hash != current\_hash:** Check if the hash codes are similar and send an email if they are not.

##### **Read the inputs:**

#Read the website and read time interval

input\_website = input('Enter the website to monitor: ')

input\_website = 'https://'+input\_website

time\_delay = int(input('Enter the time interval to check website: '))

#Monitoring website

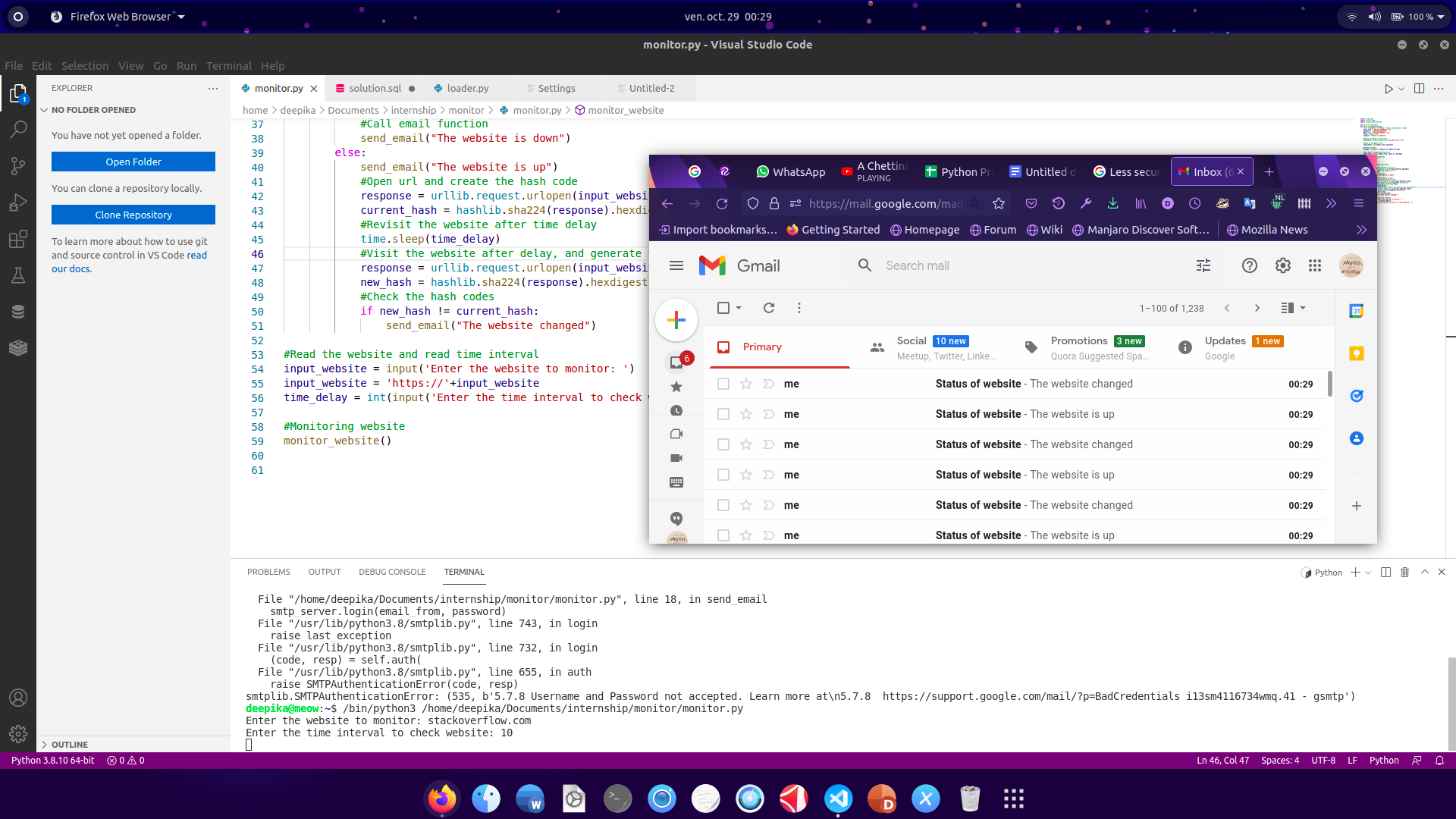
monitor\_website()

**Code explanation:**

* **input\_website:** Read the website to monitor from the user and append ‘https://’ to the given website.
* **time\_delay:** Read the time interval from the user
* **monitor\_website():** Call the function to monitor the website

#### **Project output:**

Run the program and view your output



#### **Summary**

Thus we implemented a website monitor that will notify the user when there occurs a change in the website or when a website that was down, comes up.

Comments-

* You’ve not used the word ProjectGurukul while developing the project.