|  |  |
| --- | --- |
|  | import configparser |
|  | except: |
|  | from six.moves import configparser |
|  |  |
|  | import smtplib |
|  | from email.mime.multipart import MIMEMultipart |
|  | from email.mime.text import MIMEText |
|  | import requests |
|  |  |
|  | #2 variable related to temperature API |
|  |
|  |
|  |  |
|  |
|  |
|  |
|  | class EmailSender(): |
|  | #4 initialization |
|  | def \_\_init\_\_(self): |
|  | self.cf = configparser.ConfigParser() |
|  | self.cf.read('./config.ini') |
|  | self.sec = 'email' |
|  |  |
|  | self.email = self.cf.get(self.sec, 'email') |
|  | self.host = self.cf.get(self.sec, 'host') |
|  | self.port = self.cf.get(self.sec, 'port') |
|  | self.password = self.cf.get(self.sec, 'password') |
|  |  |
|  | #5 main function to send email |
|  | def SendEmail(self, recipient): |
|  | title = "Home Sweet Home" |
|  |  |
|  | #6 create a new multipart mime object |
|  | msg = MIMEMultipart() |
|  | msg['Subject'] = '[temperature Notification]' |
|  | msg['From'] = self.email |
|  | msg['To'] = ', '.join(recipient) |
|  |  |
|  | #7 call weather API using requests |
|  | response = requests.request("GET", url, params=querystring) |
|  | result = "" |
|  |  |
|  | json\_data = response.json() |
|  | #print(json\_data) |
|  |  |
|  | #8 loop over each data and check for abnormal weather (rain, snow) |
|  | for i in range(len(json\_data)): |
|  | if(json\_data[i]['temp\_code']['value'] in temp\_dict): |
|  | if(i == 0): |
|  | Print(“the temperature is”) |
|  | else: |
|  | result = "%s in %s hour(s) time. temperature is " % (weather\_dict[json\_data[i]['weather\_code']['value']], i) |
|  |  |
|  | result += '%s%s while the humidity is about %s%s' % (json\_data[i]['temp']['value'], json\_data[i]['temp']['units'], json\_data[i]['humidity']['value'], json\_data[i]['humidity']['units']) |
|  |  |
|  | msgText = MIMEText('<b>%s</b><p>%s</p>' % (title, result), 'html') |
|  | msg.attach(msgText) |
|  |  |
|  | #9 authenticate and send email |
|  | with smtplib.SMTP(self.host, self.port) as smtpObj: |
|  | smtpObj.ehlo() |
|  | smtpObj.starttls() |
|  | smtpObj.login(self.email, self.password) |
|  | smtpObj.sendmail(self.email, recipient, msg.as\_string()) |
|  | return "Success" |
|  |  |
|  | return "Failed" |
|  | break |