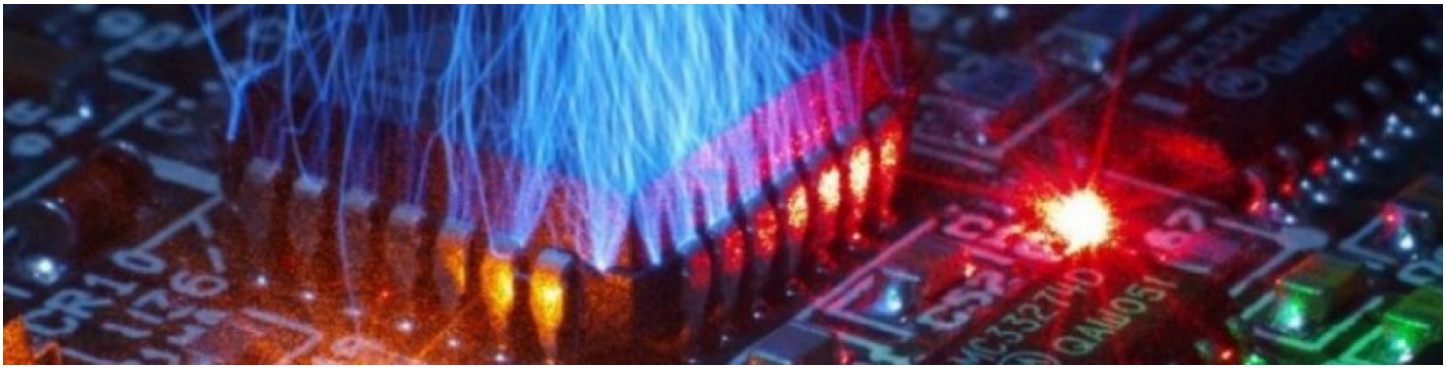
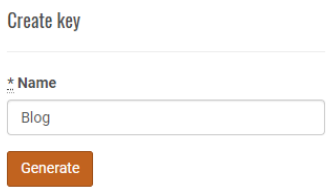


Auslesen von Wetterdaten

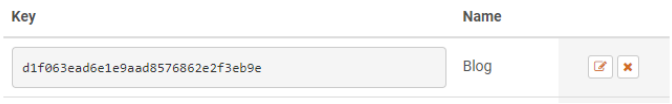
In dieses Beispiel möchte ich zeigen, wie Wetterdaten abrufen und mittels eines Python-Programms weiter verarbeitet werden können. Auf diese Weise kann z. B. das aktuelle Wetter auf einer Webseite oder dem Bildschirm ausgegeben werden. Als Quelle für die Wetterdaten soll die Datenbank von OpenWeatherMap genutzt werden.

Den Dienst nutzen:

Um den Dienst nutzen zu können, ist ein API-Schlüssel notwendig, für den ein Account bei OpenWeatherMap benötigt wird. Sowohl der Account, als auch der API-Schlüssel sind umsonst. Sobald der Account angelegt worden ist, kann man über das Menü *API keys* seines Profils einen neuen API-Schlüssel erzeugen (der gezeigte Schlüssel ist nur ein Testschlüssel).



Der neue API-Schlüssel steht dann sofort zur Verfügung und kann direkt verwendet werden.



Mit dem Gratisabonnement können Informationen zum aktuellen Wetter oder einer Vorhersage für 5 Tage im 3 h Intervall abgerufen werden. Weitere Dienste sind dann kostenpflichtig.

Aktuelle Wetterdaten können nach der Anmeldung und nach dem Erzeugen eines API-Schlüssels mit der folgenden Anfrage abgerufen werden:

```
1. http://api.openweathermap.org/data/2.5/weather?q=Stadt&APPID=Schlüssel
```

Suchen ...

MEHR ÜBER MICH

MEINE BÜCHER

THEMEN

1-WIRE (1) ALLGEMEIN (2) ASSEMBLER (2) AVR (8) AXI (1) C/C++ (11) DOCKER (1) ELEKTROTECHNIK (1) EMBEDDED LINUX (4) FPGA (1) GENETIC ALGORITHM (1) I2C (1) IR (1) LINUX (1) MACHINE LEARNING (1) OPENCV (1) PYTHON (2) RASPBERRY PI (4) SD (3) SPI (3) TIMER (1) USART (1) VHDL (1) XMEGA (8) ZYNQ (1)

DIVERSES

Die Werte *Stadt* und *Schlüssel* werden mit der abzufragenden Stadt und dem angelegten API-Schlüssel ersetzt. Das Ergebnis der Abfrage sind die Daten im JSON-Format, die dann ausgewertet werden können.

Was ist eine JSON-Datei?:

Eine JSON-Datei (*JavaScript Object Notation*) ist eine Datei, die Daten, bzw. Datenstrukturen in einer kompakten Datenstruktur abgelegt. Sie ermöglicht es Daten effizient zwischen verschiedenen Anwendungen, wie z. B. das Internet oder andere Computersysteme auszutauschen. Sämtliche Daten sind als Klartext hinterlegt und können demnach auch von Menschen gelesen werden.

Eine JSON-Datei besteht aus einer Struktur, die mit Schlüssel/Werte-Paaren gefüllt ist. Einzelne Elemente können auch wieder als Struktur aus Schlüssel/Werte-Paare bestehen.

```
{
    "cod": 200,
    "coord": {
        "lat": 50.99,
        "lon": 7.41
    },
    "dt": 1555226604,
    "id": 2930216,
    "main": {
        "humidity": 86,
        "pressure": 1023,
        "temp": 275.76,
        "temp_max": 278.71,
        "temp_min": 272.59
    },
    "name": "Engelskirchen",
    "sys": {
        "country": "DE",
        "id": 1271,
        "message": 0.0048,
        "sunrise": 1555216704,
        "sunset": 1555266158,
        "type": 1
    },
}
```

Durch diesen einfachen Aufbau lassen sich JSON-Dateien von verschiedenen Interpretern leicht einlesen.

Einlesen der Daten mittels Python:

Zum Einlesen der Daten sollen die Python-Module *requests* und *json* genutzt werden.

```
$ python3 -m pip install requests json
```

Anschließend wird ein neues Python-Skript geöffnet und die beiden Module importiert. Über die *get*-Methode des Modul *requests* wird dann eine neue Anfrage abgesendet.

```
import json
import requests

Response = requests.get("http://api.openweathermap.org/data/2.5/weather?q=Engelskirchen8
```

Die Antwort der Anfrage wird in einem Response-Objekt mit dem Namen *Response* gespeichert und kann über die *json*-Methode dieses Objektes in eine JSON-Struktur geparsed werden. Über die *dumps*-Methode des *json*-Moduls kann das Ergebnis dann in der Konsole ausgegeben werden.

```
WeatherData = Response.json()
print(json.dumps(WeatherData, indent = 4, sort_keys = True))
```

Das Resultat ist die JSON-Struktur mit den vollständigen Wetterdaten des jeweiligen Tages:

```
{
    "base": "stations",
    "clouds": {
        "all": 76
    },
    "cod": 200,
    "coord": {
        "lat": 50.99,
        "lon": 7.41
    },
    "dt": 1555226604,
    "id": 2930216,
    "main": {
        "humidity": 86,
        "pressure": 1023,
        "temp": 275.76,
        "temp_max": 278.71,
        "temp_min": 272.59
    },
```


- Registrieren
- Anmelden
- Feed der Einträge
- Kommentar-Feed
- WordPress.org

```
    "name": "Engelskirchen",
    "sys": {
        "country": "DE",
        "id": 1271,
        "message": 0.0048,
        "sunrise": 1555216704,
        "sunset": 1555266158,
        "type": 1
    },
    "visibility": 8000,
    "weather": [
        {
            "description": "broken clouds",
            "icon": "04d",
            "id": 803,
            "main": "Clouds"
        }
    ],
    "wind": {
        "speed": 0.5
    }
}
```


Natürlich kann die Ausgabe jetzt noch etwas aufgehübscht werden, indem die einzelnen Elemente ausgelesen und angezeigt werden.

```
print("Aktuelles Wetter fuer den {} UTC in {}".format(datetime.utcfromtimestamp(WeatherD
print("Luftfeuchtigkeit: {}%".format(WeatherData["main"]["humidity"]))
print("Luftdruck: {} hpa".format(WeatherData["main"]["pressure"]))
print("Temperatur: {}° C".format(str(WeatherData["main"]["temp"] - 273)))
print("Max. Temperatur: {}° C".format(str(WeatherData["main"]["temp_max"] - 273)))
print("Min. Temperatur: {}° C".format(str(WeatherData["main"]["temp_min"] - 273)))
print("Windgeschwindigkeit: {} m/s".format(WeatherData["wind"]["speed"]))
print("Windrichtung: {}°".format(WeatherData["wind"]["deg"]))
```

Eine GUI mit PyQt zum Anzeigen der Daten:

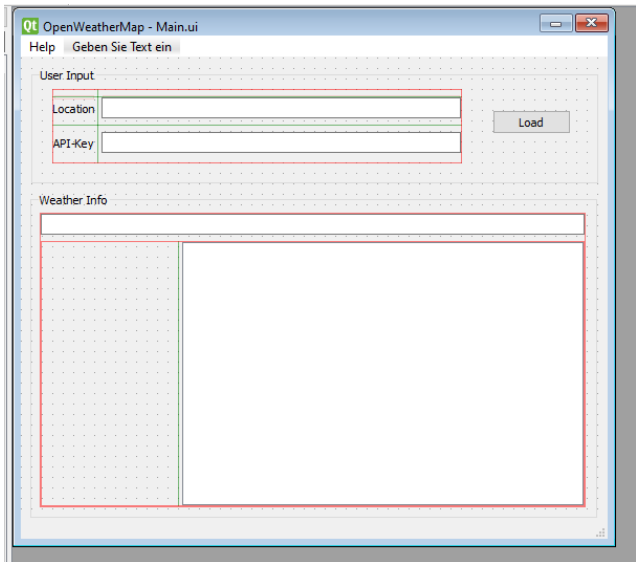
Statt über die Konsole können die Daten auch mit Hilfe einer Qt-GUI angezeigt werden. Mit dem Modul *PyQt* ist es möglich das Framework Qt in einer Pythonumgebung zu nutzen. Die Installation des Moduls kann mit Hilfe von *pip* durchgeführt werden:

```
$ pip install --user PyQt5
```

Zu allererst muss das Aussehen der Oberfläche designed werden. Leider liefert das Modul PyQt keinen Designer, etc. mit, sodass man zusätzlich noch den Qt Creator oder den Qt Designer installieren muss. Ein netter Vorteil von PyQt ist es allerdings, dass geschriebene (oder mit dem Designer entworfene) Qt UIs direkt eingebunden werden, sodass man ganz einfach bestehende Anwendungen nach Python portieren kann.

Für die Anwendung habe ich eine recht einfache GUI entworfen, die die folgenden Elemente beinhaltet:

- Ein Eingabefeld für den Standort
- Ein Eingabefeld für den API-Schlüssel
- Einen Knopf um die Ausgabe zu starten
- Ein Ausgabefeld für den ausgelesenen Standort und das Land
- Ein Ausgabefeld für ein wetterabhängiges Bild
- Eine Ausgabetabelle für die Wetterdaten



Die erstellte GUI muss am Ende als *.ui*-Datei gespeichert werden (hier heißt sie *Main.ui*). Diese UI-Datei wird nachher von dem Python-Skript eingelesen, entpackt und als GUI angezeigt.

In dem Python-Skript muss dann zuerst ein neues *QApplication*-Objekt erzeugt werden, welches dann als Hauptfenster genutzt werden kann:

```
from PyQt5 import QtWidgets, uic
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *

class Main(QtWidgets.QMainWindow):
    def __init__(self):
        super().__init__()

if(__name__ == "__main__"):
    app = QApplication(sys.argv)
    App = Main()
    sys.exit(app.exec_())
```

Dieses Hauptfenster kann nun mit der entworfenen GUI erweitert werden. Die erzeugte GUI wird anschließend mit der *show()*-Methode angezeigt:

```
uic.loadUi("app/ui/Main.ui", self)
self.show()
```

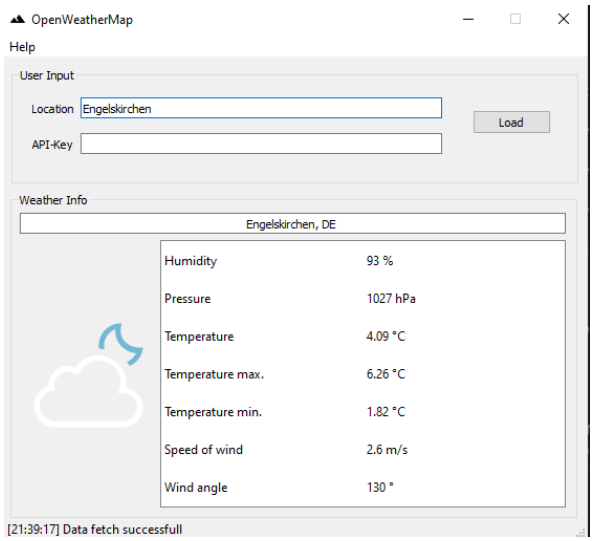
Was jetzt noch fehlt sich die Slots für den Knopf und die Textfelder:

```
def __init__(self):
    super().__init__()

    self.__Location = str()
    self.__APIKey = str()
    self.__Timer = QTimer(self)
    self.__Timer.timeout.connect(self.__loadData)

    uic.loadUi("app/ui/Main.ui", self)

    self.pushButton_LoadData.clicked.connect(self.__loadData)
    self.lineEdit_Location.textChanged.connect(self.locationChangedEvent)
    self.lineEdit_APIKey.textChanged.connect(self.apiChangedEvent)

def locationChangedEvent(self, text):
    self.__Location = text

def apiChangedEvent(self, text):
    self.__APIKey = text

def __loadData(self):
    if(not(self.__Timer.isActive())):
        self.__Timer.start(10000)

    Response = requests.get("http://api.openweathermap.org/data/2.5/weather?q={}&APPID={

    if(Response.ok):
        self.statusBar().showMessage("[{}] Data fetch successfull".format(datetime.now()
        self.__WeatherData = Response.json()

        self.__refreshData()
    else:
        self.statusBar().showMessage("[{}] Error during data fetch!".format(datetime.now
```

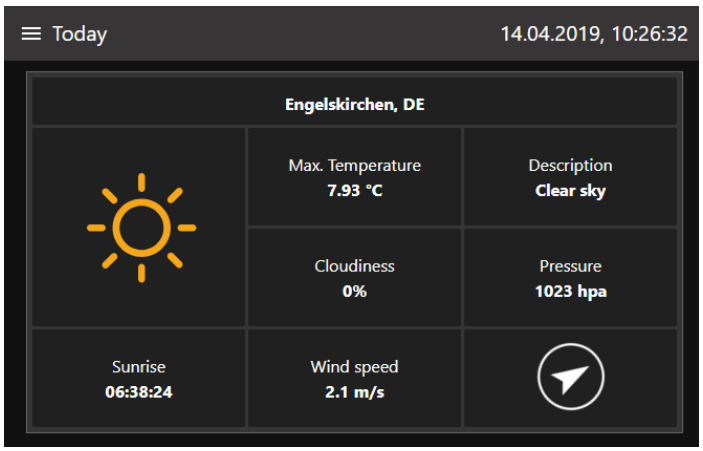
Über den Timer werden die Daten periodisch neu abgefragt, sodass immer die aktuellsten Daten angezeigt werden. Anschließend werden die Daten in der *__refreshData()*-Methode formatiert in die Tabelle eingetragen und das passende Wetterbild geladen.

```
    def __refreshData(self):
        self.tableWidget_Data.setItem(0, 0, QTableWidgetItem("Humidity"))
        self.tableWidget_Data.setItem(0, 1, QTableWidgetItem("{} %".format(self.__Weathe
        self.tableWidget_Data.setItem(1, 0, QTableWidgetItem("Pressure"))
        self.tableWidget_Data.setItem(1, 1, QTableWidgetItem("{} hPa".format(self.__Weat
        self.tableWidget_Data.setItem(2, 0, QTableWidgetItem("Temperature"))
        self.tableWidget_Data.setItem(2, 1, QTableWidgetItem("{:.2f} °C".format(self.__W
        self.tableWidget_Data.setItem(3, 0, QTableWidgetItem("Temperature max."))
        self.tableWidget_Data.setItem(3, 1, QTableWidgetItem("{:.2f} °C".format(self.__W
        self.tableWidget_Data.setItem(4, 0, QTableWidgetItem("Temperature min."))
        self.tableWidget_Data.setItem(4, 1, QTableWidgetItem("{:.2f} °C".format(self.__W
        self.tableWidget_Data.setItem(5, 0, QTableWidgetItem("Speed of wind"))
        self.tableWidget_Data.setItem(5, 1, QTableWidgetItem("{} m/s".format(self.__Weat

        if("deg" in self.__WeatherData["wind"]):
            self.tableWidget_Data.setItem(6, 0, QTableWidgetItem("Wind angle"))
            self.tableWidget_Data.setItem(6, 1, QTableWidgetItem("{} °".format(self.__We
        else:
            self.tableWidget_Data.setItem(6, 0, QTableWidgetItem("Wind angle"))
            self.tableWidget_Data.setItem(6, 1, QTableWidgetItem("{} °".format(0)))

        self.lineEdit_LocationOut.setText("{}, {}".format(self.__WeatherData["name"], se
        self.label_WeatherImage.setPixmap(QPixmap.fromImage(QImage(GetImage(self.__Weath
```

Am Ende sieht die GUI dann folgendermaßen aus:



Nachfolgend ist noch eine GUI abgebildet, die ich mit Node-RED entworfen habe und die für den offiziellen Raspberry Pi optimiert worden ist. Es wird das Wetter inkl. UV-Werte, eine Vorhersage für 5 Tage sowie eine Regen-, bzw. Wolkenkarte angezeigt. Die Daten werden zudem in einer SQL-Datenbank mitgeloggt.



Das Python-Skript gibt es in meinem GitHub-Repository zum Download.

19 KOMMENTARE

henner sagt:

29. April 2013 um 12:31 Uhr

hi daniel,
coole python-anwendung! eine kleine anmerkung: latitude (Geo_Lat) ist der geographische Breitengrad, longitude (Geo_Long) der Längengrad….
gruß henner

Antworten

Kampi sagt:
29. April 2013 um 12:46 Uhr

Hey Henner,

VIELEN DANK für den Hinweis :)
Da habe ich die beiden durcheinander geworfen (dachte ich wüsste es noch)…..sehr sehr dummer Fehler….im Zweifel immer Google fragen!
Dank dir! Ich korrigiere es heute Abend direkt.

Btw: Kommt jetzt die Tage (hoffentlich) als Anleitung:

http://www.roboternetz.de/community/threads/61732-Adafruit-Thermodrucker?p=577485#post577485

Gruß
Daniel

Antworten

Andreas sagt:
13. April 2019 um 16:34 Uhr

Die Idee die Wetterdaten mit einem Python-script zu lesen ist sehr gut, allerdings habe ich das gleiche Problem und exakt die gleichen Fehlermeldungen wie
enshiro vom 10. November 2017 um 18:25 Uhr
Leider gibt es dazu eine Antworten.
Ich habe danach eine die Webseite geändert in
Baum = urllib.urlopen(„https://www.yahoo.com/news/weather/“).read()
Dann kommt schon ein Fehler bei der Zeile
Baum = parseString(Baum)
Ich komme leider nicht zu einem funktionierenden Script.
Meine Frage wäre noch, ob das Script auch unter Python3 lauffähig ist ?

Antworten

Kampi sagt:
13. April 2019 um 20:53 Uhr

Hallo Andreas,

das Skript ist für Python3 entwickelt worden, aber leider sieht es so aus, als ob Yahoo die API geändert hat. Mittlerweile würde ich auch eher zu OpenWeatherMap tendieren:

https://openweathermap.org/

Der API-Key ist umsonst und du sendest die Anfrage http://api.openweathermap.org/data/2.5/weather?q=Stadt&APPID=APIKey ab und bekommst die Wetterdaten im JSON-Format zurück.

https://openweathermap.org/current

Gruß
Daniel

Antworten

Rolf Klinger sagt:
27. November 2019 um 11:27 Uhr

Vielen Dank für die gute Beschreibung und die links für die Wetterdaten. Leider finde ich nicht den link für das Python-Script zur GUI in GitLab. Vielen Dank im voraus für einen Hinweis.

Antworten

Kampi sagt:
27. November 2019 um 14:44 Uhr

Hallo Rolf,

die GUI ist auch kein Python-Skript, sondern ein Flow für Node-Red. Aktuell ist es auch noch nicht bei Git zu finden aber du kannst dir die aktuelle Version (noch nicht komplett fertig) stattdessen über meine Dropbox runterladen.

Gruß
Daniel

Antworten

Rolf Klinger sagt:
2. Dezember 2019 um 10:11 Uhr

Hallo Daniel,
entschuldige, wenn ich mich erst heute melde, aber so schnell habe ich nicht mit eine Antwort von Dir gerechnet. Dafür aber vielen Dank.
Mit Node-Red habe ich mich bis heute noch nicht beschäftigt, werde ich aber zum Anlass nehmen, mich hier einzulesen. Nützlich war für mich aber der link zu OpenWeatherMap, von wo ich den API-Key erhalten habe und mir jetzt die aktuelle Außentemperatur für meine Geo-Daten zur weiteren Verarbeitung in einem Python-Script verwenden kann.
Nochmals vielen Dank
Gruß
Rolf

Antworten

Äd Franzis sagt:
8. April 2020 um 14:19 Uhr

Vielen lieben Dank, Daniel

tolle Arbeit, wirklich schön geworden.
Bisher bin ich allerdings noch am Anfang und gebe nur ein paar Daten auf eine LCD-Display aus.
Aber leider habe ich ein Problem mit der Windrichtung:
Bei dem Befehl
print(„Windrichtung: {}°“.format(WeatherData[„wind“][„deg“]))
bekomme ich die Fehlermeldung, dass „deg“ in dem JSON-Struktur unbekannt ist, d.h.
KeyError: ‚deg‘.
Allerdings finde ich „deg“ auch nicht in deiner JSON-Beispieldatei oben.
Aber woher bekommst du dann die Windrichtung?

Liebe Grüße,
Äd

Antworten

Kampi sagt:
8. April 2020 um 17:18 Uhr

Hallo Äd,

ggf. gibt es bei dir keine genauen Infos darüber, weshalb das Feld leer ist. Du kannst dir die Dokumentation des JSON-Objektes hier

https://openweathermap.org/current

anschauen und dort ist „deg" Objekt drin. Die Beispieldatei enthält nicht alle Elemente und ist gekürzt, da sie nur als Anschauungsbeispiel dienen soll :). Wenn du ein komplettes JSON-Objekt brauchst, kannst du das Beispiel von openweathermap nehmen.

Gruß
Daniel

Antworten

Äd Franzis sagt:
9. April 2020 um 01:47 Uhr

Hallo Daniel,

vielen Dank, für deine Schnelle Antwort.
Ja, habe nun mit zig Städten ausprobiert: Bei denen aus meiner Umgebung fehlt die Windrichtung. Erst dachte ich noch, es liegt daran, weil die Windgeschwindigkeit derzeit unter 1 km/h liegt. Aber dann fand ich auch Orte, wo ebenfalls bei sehr wenig Wind (unter 1 km/h) die Richtung trotzdem angezeigt wurde.
Schade. Die Windrichtung hätte mich sehr interessiert.

Liebe Grüße
Äd

Antworten

Kampi sagt:
9. April 2020 um 06:57 Uhr

Hallo Äd,

du nutzt den selben API-Call wie ich in meinem Beispiel? Sonst kannst du ggf. mal bei denen nachfragen ob einfach keine Daten vorhanden sind oder ob du einen Fehler in deinem Aufruf machst. Ich kann mir da leider keinen Reim drauf machen und bekomme es für mich auch nicht nachgestellt :/

Gruß
Daniel

Antworten

Äd Franzis sagt:
9. April 2020 um 22:48 Uhr

Hallo Daniel,

ja, der Aufruf ist gleich (halt mit meiner ID und meinem Ort). Da es mit manchen Orten funzt und mit anderen nicht, denke ich, dass es nicht von allen Orten geliefert wird.
Aber das Coolste: während ich das schreibe habe ich die API nochmal auprobiert: Jetzt ist mehr Wind („speed":2.54) und das Feld „deg" existiert.

Ich beobachte das mal und berichte dann.

Aber ich habe ein anderes – schlimmeres – Problem: Ich habe dein Pythonprogramm (ganz oben, das erste) etwas ergänzt, so dass ich nun ca. alle 10 Minuten Wetterdaten abrufe und auf einem LCD darstellr. Das geht eine Weile gut, dann bekomme ich folgenden Fehler und das Programm stürzt ab:

```
Thu Apr 9 20:17:53 2020
Aktuelles Wetter fuer den 2020-04-09 18:17:53 UTC in Ulm
few clouds
Luftfeuchtigkeit: 28%
Luftdruck: 1022 hpa
Temperatur: 20.36°C
Max. Temperatur: 22.93°C
Min. Temperatur: 17.37°C
Windgeschwindigkeit: 1.5 m/s

Traceback (most recent call last):
File „/usr/lib/python3/dist-packages/urllib3/connection.py“, line 138, in _new_conn
(self.host, self.port), self.timeout, **extra_kw)
File „/usr/lib/python3/dist-packages/urllib3/util/connection.py“, line 98, in create_connection
raise err
File „/usr/lib/python3/dist-packages/urllib3/util/connection.py“, line 88, in create_connection
sock.connect(sa)
OSError: [Errno 101] Network is unreachable

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
File „/usr/lib/python3/dist-packages/urllib3/connectionpool.py“, line 594, in urlopen
chunked=chunked)
File „/usr/lib/python3/dist-packages/urllib3/connectionpool.py“, line 361, in _make_request
conn.request(method, url, **httplib_request_kw)
File „/usr/lib/python3.5/http/client.py“, line 1107, in request
self._send_request(method, url, body, headers)
File „/usr/lib/python3.5/http/client.py“, line 1152, in _send_request
self.endheaders(body)
File „/usr/lib/python3.5/http/client.py“, line 1103, in endheaders
self._send_output(message_body)
File „/usr/lib/python3.5/http/client.py“, line 934, in _send_output
self.send(msg)
File „/usr/lib/python3.5/http/client.py“, line 877, in send
self.connect()
File „/usr/lib/python3/dist-packages/urllib3/connection.py“, line 163, in connect
conn = self._new_conn()
File „/usr/lib/python3/dist-packages/urllib3/connection.py“, line 147, in _new_conn
self, „Failed to establish a new connection: %s“ % e)
requests.packages.urllib3.exceptions.NewConnectionError: : Failed to establish a new connection:
[Errno 101] Network is unreachable

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
File „/usr/lib/python3/dist-packages/requests/adapters.py“, line 423, in send
timeout=timeout
File „/usr/lib/python3/dist-packages/urllib3/connectionpool.py“, line 643, in urlopen
_stacktrace=sys.exc_info()[2])
File „/usr/lib/python3/dist-packages/urllib3/util/retry.py“, line 363, in increment
raise MaxRetryError(_pool, url, error or ResponseError(cause))
requests.packages.urllib3.exceptions.MaxRetryError:
HTTPConnectionPool(host='api.openweathermap.org', port=80): Max retries exceeded with url: /
data/2.5/weather?q=Ulm&APPID=1d94ca3239bedd3b2bc708a259e391f5 (Caused by
NewConnectionError(,: Failed to establish a new connection: [Errno 101] Network is
```

unreachable',))

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
File „/home/pi/Dokumente/autostart/status_myLCD.py", line 132, in
Response = requests.get(„http://api.openweathermap.org/data/2.5/weather?q=Ulm&APPID=1d94ca3239bedd3b2bc708a259e391f5")
File „/usr/lib/python3/dist-packages/requests/api.py", line 70, in get
return request('get', url, params=params, **kwargs)
File „/usr/lib/python3/dist-packages/requests/api.py", line 56, in request
return session.request(method=method, url=url, **kwargs)
File „/usr/lib/python3/dist-packages/requests/sessions.py", line 488, in request
resp = self.send(prep, **send_kwargs)
File „/usr/lib/python3/dist-packages/requests/sessions.py", line 609, in send
r = adapter.send(request, **kwargs)
File „/usr/lib/python3/dist-packages/requests/adapters.py", line 487, in send
raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool(host='api.openweathermap.org', port=80): Max retries exceeded with url: /data/2.5/weather?q=Ulm&APPID=1d94ca3239bedd3b2bc708a259e391f5 (Caused by NewConnectionError(,: Failed to establish a new connection: [Errno 101] Network is unreachable',))
pi@raspberrypi:~ $

Leider bin ich erst Anfänger in Python und kann die Meldungen nicht interpretieren. Kann mir jemand bitte helfen, was ich falsch mache und wie ich es beheben kann?

Die Befehle sind eigentlich gleich wie oben im Beispiel, nur dass ich Response = requests.get(„..."), WeatherData = Response.json() und die Ausgabe-Befehle in eine Schleife gelegt habe (durchführung alle 10 Minuten).

Lieben Gruß
Äd

Antworten

Kampi sagt:
9. April 2020 um 23:05 Uhr

Hallo Äd,

mmh komisch. Du kannst das Skript erweitern, indem du prüfst ob ein bestimmter Schlüssel in dem Dictionary ist.

if(„deg" in self.__WeatherData[„wind"]):

Hab das Beispiel mal entsprechend angepasst.
Zu deinem anderen Problem:
Das sieht nach einem Netzwerkfehler aus, da er keine HTTP-Verbindung mehr aufbauen kann. Irgendwie scheint der Raspberry Pi die Netzwerkverbindung zu verlieren. Nutzt du W-LAN? Probier mal ob ein Kabel das Problem löst.

Gruß
Daniel

Antworten

Äd Franzis sagt:
10. April 2020 um 00:09 Uhr

Vielen lieben Dank für deine Hilfe, Daniel, das ist wirklich klasse.

Ich probiere beides morgen gleich aus. Stimmt, ist über WLAN.

Im Moment lasse ich auf dem Raspi einen Test mit #Cheerlights laufen – ähnliche Abrage auch mit API und JSON (habe das Programm dafür umgebaut). Wenn es ein Netzwerkproblem ist, sollte es dann ja auch auftreten.

Liebe Grüße
Äd

Antworten

Äd Franzis sagt:
10. April 2020 um 22:56 Uhr

N'abend Daniel,

vielen Dank, hattest mit beidem Recht. Es funzt. Schon seit zig Stunden keine Programmabstürze mehr. Super.

Bedenklich finde ich allerdings, dass eine schlechte Internetverbindung ein Programm zum Abstürzen bring. Würde mich interessieren, ob man das nicht abschicken kann.

Liebe Grüße
Äd

Antworten

Äd Franzis sagt:
11. April 2020 um 15:22 Uhr

Hi

habe mich leider zu früh gefreut: jetzt trotz LAN wieder Abstürze.
Glaube nicht, dass es am Internetanschluss liegt: 600 Mbit/s Download, 300 Mbit/s Upload, FTTH.
Bleibt die Frage, ob man den Fehler nicht im Vorfeld abfangen kann, damit das Programm nicht abstürzt.

Liebe Grüße,
Äd

Antworten

Kampi sagt:
11. April 2020 um 17:10 Uhr

Hallo Äd,

das Problem muss nicht unbedingt mit deiner Internetverbindung zusammenhängen. Es kann auch vom Raspberry / W-LAN aus kommen (kurzer Verbindungsabbruch, etc.).
Was du machen könntest ist ein try-catch-Block zu verwenden um damit die Abfrage zu „schützen":

https://www.w3schools.com/python/python_try_except.asp

Dann stürzt dein Programm nicht ab, sondern macht lediglich irgendwas, was du vorher definiert hast, wenn ein Fehler auftritt (z. B. die Verbindung abbricht).

Gruß
Daniel

Antworten

Äd Franzis sagt:
12. April 2020 um 09:00 Uhr

Vielen herzlichen Dank, Daniel.

Super! Ja mit dem Tipp funzt es nun auch im WLAN. Nun läuft die SW schon seit mehreren Stunden auch im WLAN ohne SW-Abstürzte. (wobei jetzt aber auch nur 2 mal der Fehler aufgetreten war).

Frohe Ostern.
LG.
Äd

Antworten

Kampi sagt:
13. April 2020 um 11:43 Uhr

Hallo Äd,

freut mich :)
Dann mal viel Spaß damit und noch ein frohes Osterfest.

Gruß
Daniel

Antworten

SCHREIBE EINEN KOMMENTAR

Deine E-Mail-Adresse wird nicht veröffentlicht. Erforderliche Felder sind mit * markiert

Kommentar *

Name *

E-Mail-Adresse *

Website

Kommentar abschicken

WordPress-Theme: Treville von ThemeZee.