Auftrag für Leistungsnachweis

|  |  |  |  |
| --- | --- | --- | --- |
| **Fach** | Programming-  Tools | **Bearbeitungszeit** | ca 7h |
| **Titel** | REST-Logger | |  |
| **Studiengruppe** | BWI-A20 | **Anzahl Aufgaben** | Einzelarbeit |
| **Abgabe-Datum** | 2.12.21 / 17:00 | **Bewertung** | Geprüft wird der Code nach vorgegebenen von Funktionalen und Qualitäts-Kriterien.  Anhand eines Fachgespräches wird überprüft, ob der Student den Code und die darunter liegenden theoretischen Grundlagen verstanden hat. (Gewicht: 100%) |

|  |  |
| --- | --- |
| **Erlaubte Hilfsmittel** | * Alles inkl. Google, PyCharm, … **ausser 1:1 Abschreiben oder Kopieren** ohne Deklaration als Kommentar im Code * Keine Real-World oder virtuelle Unterstützung durch eine andere Person (Selbstständigkeit) |
| **Auftrag** | * Design und Implementation einer eigenen, allgemeinen Logger Klasse * Entwickeln einer CLI Test-Applikation, welche Daten via einen REST-Auf- ruf von einem Web-Service abfragt (JSON respones) und nach verschie- denen Strategien in einem csv-File loggen kann. |

|  |
| --- |
| **Aufgabenstellung: REST Logger Class** |
| Daten in einem File zu loggen ist in verschiedensten Bereichen eine häufige Teilaufgabe.  Sie entwickeln eine **eigene, allgemein verwendbare** Logger Klasse, welche Informationen strukturiert in Form eines CSV-Files (Welches von Excel weiterverarbeitet werden könnte) abspeichert. Erstellen Sie ebenfalls im gleichen File eine CLI-[Applikation](https://hfu-moodle.ch/mod/resource/view.php?id=9181), welche Sie zum Testen verwenden können.  Als Basis für das Test-Programm nehmen Sie [Weather Logger](https://raw.githubusercontent.com/walter-rothlin/Source-Code/master/Python_WaltisExamples/Code_05_DataLogger/WeatherLogger_01.py). Lösen Sie einen eigenen Gratis-Token (Current Weather Data) unter <https://openweathermap.org/api>  Dabei soll das Log-File einen Header aus zwei Zeilen haben:   1. Kommentar mit Filename und Start-Time, wann mit Loggen begonnen wurde (XML Syntax) 2. Eine Titel-Zeile, welche die Spalten beschriftet   Die Logeinträge müssen mindestens folgende Elemente enthalten:   * Time-Stamp (Format ist eine Property der Klasse) * Log-Level (DEBUG, INFO, WARNINGS, ERROR, CRITICAL)   Weiter müssen folgende Properties gesetzt werden können (über Setter **und** Initializer-Overloading):   * Delimiter (Default: „|“) * File-Path und File-Name * Anzahl Einträge, ab wann „gescrolled“ wird (alte Einträge gelöscht werden sollen) * Aufzeichnungs-Strategie (Fixed Slices, OnlyChanges) * Im Initializer muss gewählt werden können, ob neues File oder an bestehendes angehängt wird.   Für die Test-Applikation sind folgende Settings zu implementieren:   * Sample-Time * URL für REST-Call * Float und Integer Log-Daten werden ausserhalb der Log-Klasse gerundet und formatiert |

|  |
| --- |
| * + - * 1. **Kriterien (je 1 Punkt)** |
| * + - 1. Lauffähiger Code abgegeben **(2 Punkte)** Alles in einem File (ausnahmsweise für diese Aufgabenstellung) Filename: Vorname\_Nachname\_A19\_DS.py (z.B. Rea\_Vogel\_A19\_DS.py)       2. CLI Applikation schreibt ein Log-File **(2 Punkte)**       3. Für den Weather REST Service wurde ein eigener Token verwendet       4. Eine eigene, reusable Klasse mit **einfachem** **Interface** implementiert **(4 Punkte)**       5. Nur absolut Notwendiges ist public **(2 Punkte)**       6. Kommentare in Form von doc\_strings sind enthalten       7. Log-File enthält eine Kommentar-Zeile mit XML-Syntax       8. Log-File enthält eine Headerzeile (Spalten-Bezeichnungen) Log-Entries enthalten formatierten Time-Stamp und Level       9. Scrolling Strategie implementiert       10. Anzahl Zeilen für Scrollbereich definierbar       11. ChangesOnly implementiert       12. Append / New implementiert       13. Delimiter via \_\_init\_\_ setzbar (mit Default-Wert)       14. Strategie via \_\_init\_\_ setzbar (mit Default-Wert)       15. Scrolling area via \_\_init\_\_ setzbar (mit Default-Wert) |

|  |
| --- |
| **Beispiel eines Log-Files** |
| #<Name>G:\log.txt</Name> <Date> 2019-08-29 17:05:59 </Date> Timestamp;Level;q [%];p\_Low [bar];ShutOff-Valve;Water Pump   |  |  | | --- | --- | | 2019-08-29 | 18:07:01;INFO;0;2.34;CLOSED;OFF | | 2019-08-29 | 18:07:15;INFO;10;2.34;CLOSED;OFF | | 2019-08-29 | 18:07:16;INFO;10;2.3;OPEN;ON | | 2019-08-29 | 18:07:19;INFO;10;2.1;OPEN;ON | | 2019-08-29 | 18:07:19;INFO;20.5;2.1;OPEN;ON | | 2019-08-29 | 18:07:58;INFO;30.13;1.34;OFFEN;ON | | 2019-08-29 | 18:07:59;INFO;30.13;1.34;OFFEN;ON | | 2019-08-29 | 18:07:60;INFO;30.13;1.34;OFFEN;ON | | 2019-08-29 | 18:08:01;INFO;30.13;1.34;OFFEN;ON | |

## Mögliche Ausgangslage für das CLI Test-Programm

OpenWeather Web-App: <https://openweathermap.org/city/7287397>

API doc and subscription: <https://openweathermap.org/api>

Get a free API-Key for Current weather data

(Limited to 60 calls/minute or 1Mio calls/month)

Test-API Link:

<http://api.openweathermap.org/data/2.5/weather?q=Wangen%20SZ&appid=3836093dde650898> [eb014e6f27304646](http://api.openweathermap.org/data/2.5/weather?q=Wangen%20SZ&appid=3836093dde650898eb014e6f27304646)

#

# Name: WeatherLogger.py

#

# Description: Polling REST Service and write values to console

#

# Autor: Walter Rothlin

#

# History:

# 03-Dec-2020 Walter Rothlin Initial Version ()

# 10-Oct-2021 Walter Rothlin Adapted for BWI-A20

#

import requests

import json

import time

pollingTime = float(input("Polling-Time [s]:"))

serviceURL = "https://api.openweathermap.org/data/2.5/weather"

appId = "144747fd356c86e7926ca91ce78ce170"

while True:

responseStr = requests.get(serviceURL + "?q=Uster&units=metric&lang=de&ap- pid=" + appId)

jsonResponse = json.loads(responseStr.text)

temp = jsonResponse['main']['temp']

pressure = jsonResponse['main']['pressure']

humidity = jsonResponse['main']['humidity']

lon = jsonResponse['coord']['lon']

lat = jsonResponse['coord']['lat']

cloud = jsonResponse['weather'][0]['description']

print(temp, pressure, humidity, lon, lat, cloud)

time.sleep(pollingTime)

|  |  |
| --- | --- |
|  | **Fachgespräch** |
|  | 1. Auf welcher Zeile wird die **\_\_init\_\_** Methode ihrer Klasse aufgerufen? 2. Auf welcher Zeile wird die **\_\_str\_\_** Methode ihrer Klasse aufgerufen? 3. Wieso beginnen einige Methoden und Instance-Variablen in Ihrer Klasse mit **\_\_**? 4. Zeigen Sie die Stelle im Code, wo Sie das **ChangesOnly** implementiert haben? 5. Zeigen Sie die Stelle im Code, wo Sie das **Scrolling** implementiert haben? |  |