Projektdokumentation
zur
DoIT der Klasse WI2512

Wetteranalyse in Python und Speicherung in einer SQL-Datenbank

Studierender
Severin Fröhlin
Barbarastraße 9
76694 Forst

Fachdozent
[redacted]
Schulleiter der Fachschule für Angewandte Informatik

Bildungsstätte
SRH Berufliche Rehabilitation
Bonhoeferstr. 1
69123 Heidelberg

Inhaltsverzeichnis

1 Einleitung ... 3
1.1 Beweggründe und Motivation ... 3
1.2 Projektbeschreibung und -ziel ... 3
2 Ist-Analyse ... 3
2.1 Arbeits- und Projektumgebung ... 4
3 Projektplanung ... 4
3.1 Kostenplanung ... 4
4 Projektdurchführung ... 5
4.1 Durchführungszeitraum ... 5
4.2 Arbeitsvorbereitung ... 5
4.2.1 XAMPP ... 5
4.2.2 MySQL Workbench ... 5
4.2.3 IDE -PyCharm ... 5
4.2.4 Python Virtual Environment / pip- Paketmanager ... 6
4.2.5 Requirements ... 6
4.3 Erlernen der Programmiersprache Python ... 6
4.4 Download und Analyse der Rohdaten ... 7
4.4.1 Historische Messdaten ... 7
4.4.2 Wetter Forecast ... 8
4.4.3 Stationslexikon ... 9
4.4.4 Luftqualität ... 10
4.5 GeoPy ... 11
4.5.1 Fehler und Programmabbruch ... 11
4.6 ftplib ... 11
5 SQL ... 12
5.1 Datenbankmodellierung ... 12
5.2 Insert ... 13
5.3 Datenbankabfragen ... 14
6 Pandas – Python and Data Science ... 14
6.1 Dataframes und DataframeGroupBy ... 14
6.2 Feather ... 15
6.3 Graphen plotten ... 15
7 Grafische Benutzeroberfläche ... 16
8 Programmstruktur ... 16
8.1 Programmier- und Designfehler ... 17

8.1.1 Programmfehler bei Datentypen ... 17
8.1.2 Hartcodierte Pfade ... 18
8.1.3 Globale Variablen ... 18
9 Projektabschluss ... 18
9.1 Soll-Ist-Vergleich ... 18
9.1.1 Zeitplanung ... 19
9.1.2 Pflichten- und Lastenheft ... 19
9.2 Vergleich zwischen Java und Python ... 20
9.3 Probleme und Hürden ... 21
9.3.1 Recherche, Debugging und Tests ... 21
9.3.2 Defektes Notebook ... 21
9.3.3 Fehlende Versionskontrolle ... 21
9.4 Fazit und Ausblick ... 21
10 Selbstständigkeitserklärung ... 22
Anhänge ... 23
11 Übersicht Programme und Funktionen ... 23
11.1 ../Python ... 23
11.1.1 ./funktionsUndVariablensammlung.py ... 23
11.1.2 ./main.py ... 24
11.1.3 ./pip_install_packages.bat ... 24
11.1.4 ./pip_update_packages.bat ... 24
11.1.5 ./sqlInterface.py ... 25
11.2 ./Python/Auswertung ... 26
11.2.1 ./histotischeMesswerte.py ... 26
11.2.2 ./plotting.py ... 26
11.2.3 ./stationsermittlung.py ... 27
11.2.4 ./wetterVorraussage.py ... 28
11.3 ./Python/Datenbeschaffung ... 28
11.3.1 ./ftpDownloadUndUpdateMessungen.py ... 28
11.3.2 ./sqlImportStationenV4.py ... 29
11.4 ./Python/Übungen ... 30
11.5 ./Python/Attic ... 30
12 Installation von Python ... 30
12.1 Windows Umgebungsvariablen hinterlegen ... 30
12.2 Pip Paketmanager installieren ... 32
13 Gesprächsnotizen ... 32

14 Glossar 33
15 Codeverzeichnis 34
16 Tabellenverzeichnis 34
17 Abbildungsverzeichnis 35
18 Verweise 35

1 Einleitung

1.1 Beweggründe und Motivation

Vor Jahren habe ich auf dem Youtubekanal des Chaos Computer Clubs einen Vortrag von David Kriesel gesehen.[1] Für diesen wurden von „Spiegel Online“ mittels Data Mining[2] Artikel und verschiedene Informationen automatisiert heruntergeladen. Herr Kriesel stellte dar, wie er die Artikel anhand verschiedener Parameter analysiert und gruppiert hat. Die Ergebnisse und deren Schlussfolgerungen präsentierte er anhand von Grafiken und Schaubildern.
Aufgrund dieses Vortrages fing ich an mich für Data Mining und Data Science zu interessieren. Über die Jahre habe ich mehrere Vorträge zu diesem Thema angesehen. Im Zuge dessen wurde von mehreren Seiten betont, dass sich die Programmiersprache Python perfekt für solche Aufgaben eignet. Die DoIT schien mir als perfektes Umfeld, mich mehr mit diesem Thema zu beschäftigen, sowie mehr über die Herangehensweise und die Programmiersprache Python zu lernen.

Im Vorfeld zur DoIT informierte ich mich, wo man geeignete Data Lakes[3] erhält. Für mich standen als relevante Datengrundlagen die Blockchain der Kryptowährung Bitcoin oder historische Wetterdaten des Deutschen Wetterdienstes, kurz DWD, zur Auswahl. Da mich das Thema Wetter mehr interessierte, habe ich mich dafür entschieden.

1.2 Projektbeschreibung und -ziel

Ziel des Projektes ist es, mittels der Programmiersprache Python und einer MySQL Datenbank historische Wetterdaten des DWD herunterzuladen und zu speichern. Dazu muss die Struktur der Wetterdaten analysiert werden. Daraus ergibt sich dann der Datenbankaufbau mit den einzelnen Tabellen und deren Relationen.

Die erhaltenen Daten sollen anschaulich, einfach und verständlich aufbereitet werden, um aus ihnen die gewünschten Informationen ableiten und auslesen zu können. Als Messwerte sollen unter anderem jeweils das Tagesmittel der Temperatur, Windstärke, Sonnenstunden, Niederschlagsmengen und die Schadstoffbelastung der Luft auswertbar sein.

Das Programm soll mit einer grafischen Benutzeroberfläche betrieben werden, in der die ausgewerteten Graphen dargestellt werden können.

2 Ist-Analyse

Meine Programmierkenntnisse beschränken sich auf die in den Unterrichtsfächern behandelten Themen in den Bereichen Java, SQL und den Grundkenntnissen der Programmierung. Es war

[1] (Kriesel, 2016)
[2] 14 Glossar, Data Mining
[3] 14 Glossar, Data Lake

notwendig die Programmiersprache Python zu erlernen, um in dieser den Quellcode schreiben zu können. Des Weiteren wird eine IDE [4] benötigt.

Im Vorfeld zur DoIT habe ich mich bereits über passende Datenquellen informiert. Der DWD stellt alle historischen Wetterdaten auf deren OpenData Server „opendata.dwd.de“ zur Verfügung. Die aktuellen Daten zu Wetter und Luftqualität können über das Python Packet „deutschland“ abgerufen werden.

2.1 Arbeits- und Projektumgebung

Als Arbeitsmittel stehen mein privates Notebook mit Administratorrechten, sowie die nachfolgenden Programme zur Verfügung:

- MS Office 2019 Professional Plus
  - OneNote: Festhalten von Notizen
  - Word: Ausarbeitung der Dokumentation
  - PowerPoint: Ausarbeitung der Präsentation
- PyCharm
  - Entwicklungsumgebung für die Ausarbeitung des Quellcodes
- XAMPP
  - Lokaler SQL-Server und FTP-Zugang für Datenbeschaffung
- Oracle MySQL Workbench
  - Administrieren der Datenbank auf dem SQL-Server
- drawIO
  - Erstellen des Entity Relationship Modells (ERD)

3 Projektplanung

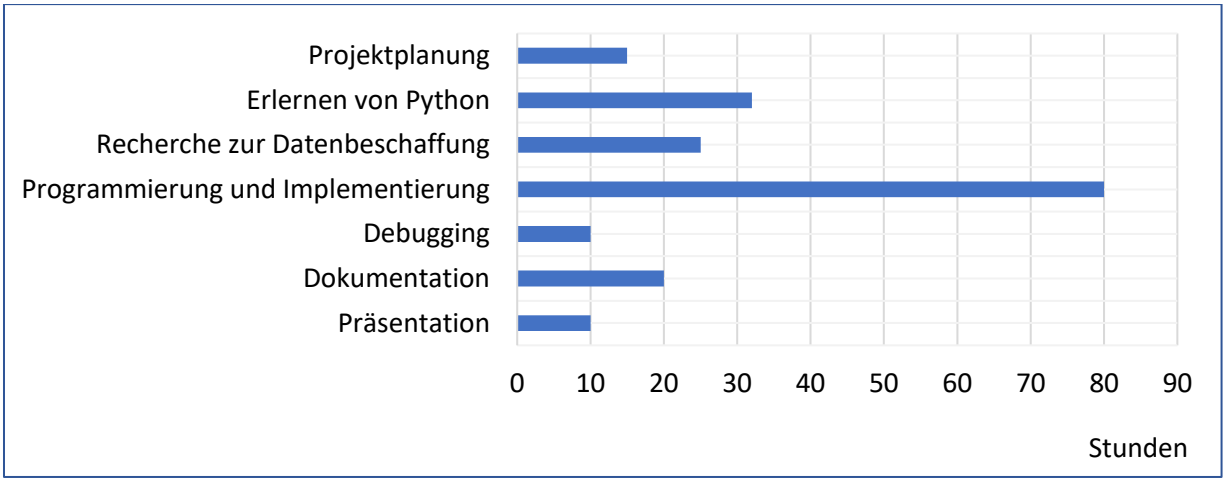

Abbildung 1 - Zeitplanung

3.1 Kostenplanung

Für die Preiskalkulation wurde der durchschnittliche Stundensatz für freiberufliche IT-Fachkräfte herangezogen.[5] Da mir die Strom- und Betriebskosten der SRH nicht bekannt sind, wurde nur die Arbeitszeit zur Berechnung verwendet.

[4] 14 Glossar, IDE - Integrated Development Environment
[5] (GULP, 2022)

Stundenlohn	75,00 €
Arbeitszeit	192,00 h
Gesamtpreis netto	14.400,00 €
+ 19,00% MwSt.	2.736,00 €
Gesamtpreis brutto	17.136,00 €

Tabelle 1 - Preiskalkulation

4 Projektdurchführung

4.1 Durchführungszeitraum

Das Projekt wurde im Rahmen der DoIT durchgeführt. Die reine Bearbeitungszeit des Projekts sind 6 Wochen mit jeweils 32 Stunden. Die Zeiträume teilen sich wie folgt auf:

Projektfindung und Erstellung des Projektantrags	27.06.2022 bis 01.07.2022
Projektbearbeitung	04.07.2022 bis 22.07.2022
Projektbearbeitung	22.08.2022 bis 09.09.2022
Abgabe Quellcode	09.09.2022
Abgabe Dokumentation und Präsentation	30.09.2022
Präsentation	04.10.2022 bis 07.10.2022

Tabelle 2 - Durchführungszeitraum

4.2 Arbeitsvorbereitung

Um mit dem Projekt anfangen zu können, musste zuerst die Infrastruktur auf dem Notebook hergestellt werden. Hierzu mussten die nachfolgenden Programme installiert werden.

4.2.1 XAMPP

Downloadlink: https://www.apachefriends.org/download.html
Durch die Installation von XAMPP können verschiedene Dienste auf dem PC gehostet werden. Für dieses Projekt wird der MySQL Server benötigt. Dieser Dienst ist über die IP 127.0.0.1 und den Port 3306 erreichbar.

Mithilfe des Programmes FileZilla kann man sich mit einem FTP-Server verbinden und je nach Zugangsberechtigungen Dateien hoch- und/ oder herunterladen. Die Verbindung wird standardmäßig über den Port 21 hergestellt.

4.2.2 MySQL Workbench

Downloadlink: https://dev.mysql.com/downloads/workbench/
Grundsätzlich könnte phpMyAdmin von dem Programm XAMPP genutzt werden, jedoch bietet MySQL Workbench eine einfachere und übersichtlichere Administration des MySQL-Servers. Für mich waren die semantische Überprüfung der SQL-Befehle und die Menüführung ein großer Vorteil.

4.2.3 IDE -PyCharm

Downloadlink: https://www.jetbrains.com/pycharm/download/#section=windows
Von meinem Fachbetreuer wurde die IDE Jupyter empfohlen. Nach etwas Recherche fiel die Entscheidung allerdings auf die IDE PyCharm. Diese ist vom Unternehmen JetBrains, von welchem auch die IDE IntelliJ stammt, welche wir im Java Unterricht verwenden. PyCharm und IntelliJ gleichen vom optischen Aufbau und den Tastenkombinationen. Jupyter hat zwar standardmäßig mehr Pakete vorinstalliert, diese können allerdings in PyCharm schnell nachinstalliert und in die Programme importiert werden.

4.2.4 Python Virtual Environment / pip- Paketmanager

Um Python in der Kommandozeile nutzen zu können, muss bei der Installation von PyCharm die Python Virtual Environment, kurz „python venv“, mit der Checkbox „add bin to PATH“ ausgewählt werden. Das bedeutet, dass in den Umgebungsvariablen der Betriebssystemeinstellungen eine Variable mit dem Pfad zur Python Virtual Environment hinterlegt wird. Des Weiteren sollte bei der PyCharm Installation der Pip Paketmanager mit installiert werden. Durch diesen können zusätzlich Pakete projektunabhängig zur IDE hinzugefügt werden. Für das Hinterlegen der Umgebungsvariablen sind Administratorrechte nötig und im Anschluss muss ein Neustart des Betriebssystems durchgeführt werden. In manchen Fällen funktioniert das nicht und die Umbegungsvariablen müssen manuell in den Systemeigenschaften hinterlegt und pip manuell installiert werden. Das Vorgehen hierfür wird im Anhang näher erläutert.[6]

4.2.5 Requirements

Für viele Pakete oder Projekte sind sogenannte „requirements“ angegeben. Requirements sind die Anforderungen, die erfüllt werden müssen, damit ein Programm in vollem Umfang läuft. Diese sind meist Textdateien und enthalten eine Liste mit Python Paketen mit der minimalen Versionsnummer. Im Projektordner mit den Quellcodes liegt eine solche Datei und auch eine Installations- sowie Update-Batch Datei.

Requirements:	In Python 3.10.6 bereits enthaltene Pakete:
- pandas~=1.4.3 - requests~=2.28.1 - geopy~=2.2.0 - deutschland~=0.1.9 - pip~=21.3.1 - wheel~=0.37.1 - lxml~=4.9.1 - setuptools~=60.2.0 - future~=0.18.2 - matplotlib~=3.5.3 - mysql-connector-python~=8.0.30 - numpy~=1.23.2 - feather-format~=0.4.1 - python-dateutil~=2.8.2	- os - ftplib - zipfile - json

Tabelle 3 - Requirements

4.3 Erlernen der Programmiersprache Python

Um möglichst viele grundlegende Bereiche kennen zu lernen, habe ich die viele Projekte des Java Unterrichts in Python nachprogrammiert. So kam ich automatisch mit Datenstrukturen, Ein- und Ausgabe von Daten, Kontrollstrukturen, Listen und Objektorientierung in Berührung. Der Umstieg von der strikten Deklaration von Variablen und Methoden in Java zu der Deklaration in Python war ungewohnt. Anfangs benannte ich die Variablen und Funktionen ähnlich dem Aufbau in Java, wie beispielsweise „def static_final_MAX_ANZAHL_GERAETE()“.

Bei den späteren Projekten kam mir die Erkenntnis, dass Python sich zwar gut für die objektorientierte Programmierung eignet, es allerdings für dieses Projekt nicht von Nöten ist. Somit sind viele Programme in dem Projektordner „Übungen/02_OOP“ nicht lauffähig, da hier kein Debugging[7] mehr

[6] 12 Installation von Python
[7] 14 Glossar, Debugging

betrieben wurde. Die in Python üblichen „snake_case“ Notation war für mich ungewohnt. Daher habe ich mich für die „camelCase“ Notation entschieden.

4.4 Download und Analyse der Rohdaten

4.4.1 Historische Messdaten

Die Beschaffung der historischen Messdaten[8] des DWD war schwieriger als gedacht. Die Verbindung zum FTP-Server via FileZilla war nicht möglich. Da keine Informationen aufzufinden waren, wie die Verbindung aufzubauen ist, habe ich die Dateien des OpenData Servers mit dem Programm "HTTrack Website Copier" (Downloadlink: https://www.httrack.com/) heruntergeladen. Meine Annahme war, dass ein automatisierter Download nur über einen kostenpflichtigen Zugang möglich sei. Dies stellte sich später allerdings als falsche Vermutung heraus.

Bei einem Telefonat mit dem DWD hat mir der Mitarbeiter erklärt, wie ich am einfachsten auf den FTP-Server zugreifen kann. Der kostenpflichtige Zugriff betrifft nur die Wettervorhersage über alle Stationen. Die Datenhistorien können kostenlos über das Python Package „ftputil“ heruntergeladen werden.

Die Messdaten sind im Unterordner „climate_environment/CDC/observations_germany/climate/daily/kl/historical/“ zu finden. Hier ist für jede StationsID ein eigenes Zip-Archiv vorhanden. In diesen sind jeweils mehrere Textdokumente mit beispielsweise Minimal-, Maximal- und fehlenden Werten. Für dieses Projekt ist allerdings nur die Datei „produkt_klima_tag_*.txt“ interessant. Der Stern ist ein Platzhalter für den Messzeitraum und die Stationsnummer. Diese Datei hat immer eine Headline mit der StationsID und Messwertbezeichnung. Die einzelnen Werte sind durch Semikola getrennt.

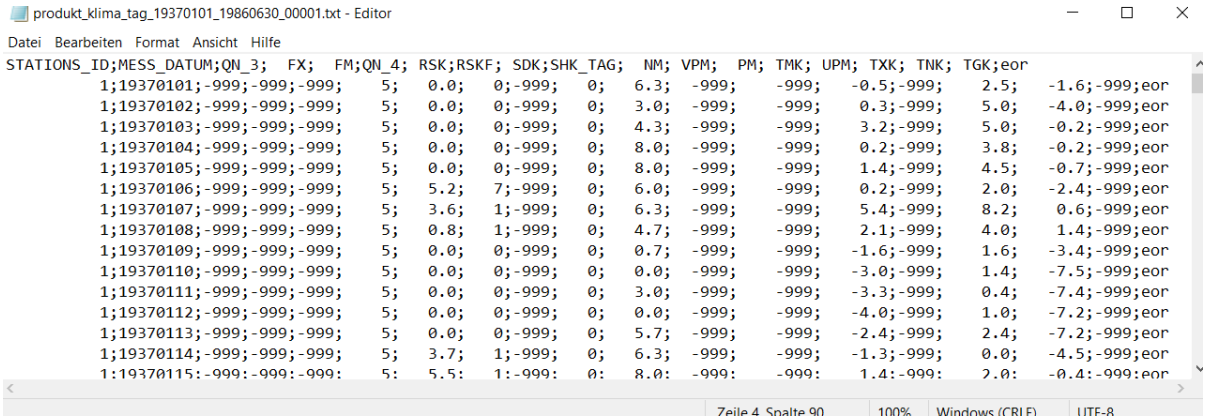

Abbildung 2 - Rohdatenformat der historischen Messungen

Messwert:	Bedeutung:	Format/ Maßeinheit:
STATIONS_ID	Stationsidentifikationsnummer	Integer
MESS_DATUM	Datum	yyyymmdd
QN_3	Qualitätsniveau der nachfolgenden Spalten	Für diesen Projektrahmen erstmal nicht relevant
FX	Tagesmaximum Windspitze	Meter pro Sekunde
FM	Tagesmittel Windgeschwindigkeit	Meter pro Sekunde
QN_4	Qualitätsniveau der nachfolgenden Spalte	Für diesen Projektrahmen erstmal nicht relevant
RSK	Tägliche Niederschlagshöhe	Millimeter

[8] (Deutscher Wetterdienst, 2022)

RSKF	Niederschlagsform	code SQL-Tabelle „doit_projekt.rskf“
SDK	Tägliche Sonnenstundendauer	Stunden
SHK_TAG	Tageswert Schneehöhe	Zentimeter
NM	Tagesmittel des Bedeckungsgrades	„1/8“
VPM	Tagesmittel des Dampfdrucks	Hektopascal
PM	Tagesmittel des Luftdrucks	Hektopascal
TMK	Tagesmittel der Temperatur	Grad Celsius
UPM	Tagesmittel der relativen Feuchte	Prozent
TXK	Tagesmaximum der Lufttemperatur in 2m Höhe	Grad Celsius
TNK	Tagesminimum der Lufttemperatur in 2m Höhe	Grad Celsius
TGK	Minimum der Lufttemperatur am Erdboden in 5cm Höhe	Grad Celsius
eor	Ende data record	

Tabelle 4 - Legende Messwert[9]

4.4.2 Wetter Forecast

Ursprünglich war geplant den Forecast über die „deutschland.dwd“ -API herunterzuladen. Mir wurde im Telefonat mit dem DWD jedoch erklärt, dass der Zugriff über einen API-Token nur kostenpflichtig eingerichtet wird. Der kostenlose Zugriff auf die aktuellen Wetterdaten von Stationen mit der Kennungs-ID „SY“ ist mittels einem URL Request über https://dwd.api.bund.dev/ möglich.[10]

In dem Pull down Menü „/stationOverviewExtended“ ist der Aufbau des Request herauszulesen. Die Stations-IDs müssen als Query[11] der URL angehängt werden. Als Antwort erhält man einen Bytecode[12], der zu einem JSON[13] umgewandelt werden muss, um auf die Daten zugreifen zu können. Bei der Abfrage mit dieser URL https://dwd.api.proxy.bund.dev/v30/stationOverviewExtended?stationIds=10865,G005 erhält man ein Dictionary mit folgendem Inhalt der folgenden Abbildung. Das Dictionary wurde für diesen Screenshot in die IDE kopiert, um die einzelnen Schlüsselwörter zusammenfalten zu können. Auf der ersten Ebene sind die Stations-IDs. Innerhalb des Forecasts findet sich noch einmal der UNIX- Zeitstempel in Millisekunden für den Beginn der Aufnahme. Dieser ist immer um 00:00:00 Uhr des aktuellen Tages. Der „timeStep“, bzw. die Schrittweite, beträgt 3.600.000 Millisekunden oder 60 Minuten. Unter „temperature“ befinden sich 241 Einzelwerte in der Einheit 0,1°C[14]. Unter dem Schlüsselwort „days“ finden sich noch einmal die minimalen und maximalen Tageswerte. Der gesamte Forecast hat eine Reichweite von 10 Tagen in die Zukunft.

Im Programm „../wetterVorraussage.py“[15] werden diese Daten aufbereitet und für die Grafikerstellung vorbereitet. In der Funktion „konvertiereDwdApiAbfrageZuForecastTage“ wurde jedoch die Umrechnung von z.B. 200 °C/10 in 25,0°C noch nicht implementiert. Dies führt dazu, dass in den Graphen die Werte um den Faktor 10 zu hoch sind.

[9] (Deutscher Wetterdienst, 2022)
[10] (dwd.api.bund, 2022)
[11] 14 Glossar, Bytecode
[12] 14 Glossar, JSON
[13] 14 Glossar, Query
[14] (Github DeutscherWetterdienst, 2022)
[15] 11.2.4 ./wetterVorraussage.py

```
def json():
    return {
        "10865": {
            "forecast1": {
                "stationId": "10865",
                "start": 1662501600000,
                "timeStep": 3600000,
                "temperature": [...],
                "temperatureStd": [...],
                "windSpeed": null,
                "windDirection": null,
                "windGust": null,
                "icon": [...],
                "precipitationTotal": [...],
                "precipitationProbablity": null,
                "precipitationProbablityIndex": null
            },
            "forecast2": {...},
            "forecastStart": null,
            "days": [...],
            "warnings": [],
            "threeHourSummaries": null
        },
        "G005": {
            "forecast1": {...},
            "forecast2": {...},
            "forecastStart": null
```

Abbildung 3 - Ergebnis des URL Request

4.4.3 Stationslexikon

Um für die gewünschten Regionen Auswertungen durchführen zu können, reichen die historischen Messungen nicht aus. Diese enthalten die Stationsidentifikationsnummer und man kann mithilfe des vom DWD bereitgestellten Stationslexikon den Standort ermitteln. Unter folgendem Link kann es angezeigt oder heruntergeladen werden kann:

https://www.dwd.de/DE/leistungen/klimadatendeutschland/statliste/statlex_rich.txt?view=nasPublication&nn=16102

Dort erhält man ein Textdokument mit allen Stationen des DWD alphabetisch sortiert. Leider sind die einzelnen Daten der Stationen nicht so leicht trennbar wie bei den historischen Messdaten. Nach mehreren gescheiterten Versuchen die einzelnen Zeilen durch einen Algorithmus zu trennen, erfolgt die Trennung nach dem Zeichenindex innerhalb einer Zeile. Eine dynamische Lösung wäre besser, alle Versuche führten allerdings zu unsinnigen Daten. Falls die Datei einmal anders formatiert sein sollte, funktioniert die gesamte Stationsimportierung und -update nicht mehr und man muss diese Trennung der Werte überarbeiten.

statLex.txt - Editor

Datei Bearbeiten Format Ansicht Hilfe

STAT_NAME	STAT_ID	KE	STAT	BR_HIGH	LA_HIGH	HS	HFG_NFG	BL	BEGINN	ENDE
Aach	1	KL	02783	47.841	8.849	478		BW	01.01.1937	30.06.1986
Aach	1	RR	70191	47.841	8.849	478		BW	01.01.1912	30.06.1986
Aach/Hegau	10771	PE	10771	47.85	8.85	480		BW	27.02.1951	15.06.2000
Aachen	3	EB	02205	50.7827	6.0941	202	803100	NW	01.01.1951	31.03.2011
Aachen	3	FF	02205	50.7827	6.0941	202	803100	NW	01.01.1937	31.03.2011
Aachen	3	KL	02205	50.7827	6.0941	202	803100	NW	01.01.1891	31.03.2011
Aachen	3	MI	02205	50.7827	6.0941	202	803100	NW	28.04.1993	07.10.2008
Aachen	3	MN	10501	50.7827	6.0941	202	803100	NW	01.10.2008	03.04.2012
Aachen	3	PE	3	50.7827	6.0941	202	803100	NW	11.02.1951	10.02.2011
Aachen	3	RR	80310	50.7827	6.0941	202	803100	NW	01.01.1891	31.03.2011
Aachen	3	SO	02205	50.7827	6.0941	202	803100	NW	01.01.1951	31.03.2011
Aachen	3	SY	10501	50.7827	6.0941	202	803100	NW	01.04.1950	01.04.2011
Aachen	3	TU	02205	50.7827	6.0941	202	803100	NW	01.04.1950	31.03.2011
Aachen (Kläranlage)	2	RR	80313	50.807	6.1	138	803130	NW	01.01.1951	31.12.2006
Aachen-Brand	4	RR	80316	50.768	6.121	243		NW	01.01.1951	31.10.1979
Aachen-Brand (KA)	14152	MN	H808	50.749	6.186	213		NW	11.03.2012	31.08.2022
Aachen-Brand (KA)	14152	RR	87411	50.749	6.186	213		NW	11.11.2012	30.08.2022

Zeile 1, Spalte 1 | 100% | Windows (CRLF) | ANSI

Abbildung 4- Stationslexikon

Messwert:	Bedeutung:	Format und nähere Informationen:
STAT_NAME	Stationsname	
STAT_ID	Stationsidentifikationsnummer	
KE	Kennung => Art der Station	Näheres ist in der SQL Tabelle „doit_projekt.messwerte_kennung“ beschrieben
STAT	Stationskennung	
BR_HIGH	Breitengrad	Für die Standortermittlung
LA_HIGH	Längengrad	Für die Standortermittlung
HS	Stationshöhe auf Höhe des Klimagartens bezogen (Ausnahme: KE=SO: Höhe des Aufstellungsortes des Sonnenscheinschreibers)	Meter über NN
HFG_NFG	Flussgebiet (Haupt- und Nebenflussgebiet)	Wurde in diesem Projekt nicht beachtet
BL	Bundesland	
BEGINN	Datum der ersten Messung	tt.mm.yyyy
ENDE	Datum der letzten Messung	tt.mm.yyyy

Abbildung 5 - Legende Stationslexikon[16]

4.4.4 Luftqualität

Gleich wie bei dem Wetter Forecast können die aktuellen Daten nicht über das „deutschland“- Paket heruntergeladen werden. Man muss diese auch mittels URL Request downloaden. Man erhält auch hier einen Bytecode, der zu einem JSON umgewandelt werden muss, um die Daten auswerten zu können. Leider fehlte die Zeit, dieses Feature zu implementieren. Die Dokumentation zu dieser Abfrage ist in den Quellen verlinkt.[17]

[16] (Deutscher Wetterdienst, 2022)
[17] (luftqualitaet.api.bund.dev, 2022)

4.5 GeoPy

GeoPy ist ein Python Paket, durch das man sehr viele Berechnungen und Auswertungen auf Grundlage von Geodaten tätigen kann. Hierfür muss zuerst aus dem Paketordner „geopy.geocoders“ die Klasse „Nominatim“ importiert werden. In einem Objekt dieser Klasse wird der „user_agent“ und optional ein „timeout“, bzw. eine Reaktionszeit, definiert. Der „user_agent“ ist nichts anderes als der gewünschte Kartendienst. In diesem Projekt wurde OpenStreetMap verwendet, um jegliche Urheberfragen zu umgehen.

```
from geopy.geocoders import Nominatim as nm
geolocator = nm(timeout=25, user_agent="https://www.openstreetmap.org/#map")
```

Formel 1 - GeoPy Geolocator

Über dieses Objekt können nun mit der Funktion „geolocator.reverse“ aus den Breiten- und Längengraden im Stationslexikon die Postleizahlen und weitere ortsbezogene Daten auswerten. Umgekehrt kann man mit der Funktion „geolocator.geocode“ auch aus Postleizahl oder Ortsnamen die Breiten- und Längengrade ermitteln.

Mit Hilfe der Funktion „geopy.distance.great_circle“ kann man mit einer aus dem Stationslexikon generierten Tabelle die Distanz von dem eingegeben Punkt auf der Karte zu allen Standorten der Tabelle in Kilometer oder Meilen errechnen. Diese Funktion ist für eine dynamische standortspezifische Wetterdatenanalyse unabdingbar.

4.5.1 Fehler und Programmabbruch

Wenn bei der Ermittlung der Stationsdaten, während des Imports der Stationen, die Geodaten zu den einzelnen Stationen abgerufen werden, kann es zum Absturz des Programms führen. Meine Vermutung ist, dass es sich um eine Blockierung durch den DNS-Server handelt. Bei diesem Vorgang werden sehr viele DNS-Anfragen in sehr kurzer Zeit getätigt, wodurch vermutlich der DNS-Server zur Vorbeugung einer DDOS-Attacke die IP-Adresse sperrt. In der SRH das des Öfteren geschehen. Wenn ich über einen Hotspot meines Handys mit dem DNS Server von Cloudflare das Programm startete, kam es zu keinen Fehlern.

4.6 ftplib

Auf dieses Python Paket bin ich durch das Telefonat mit dem DWD aufmerksam geworden. Mit einem Objekt der Klasse FTP kann man mit Schlüsselwort „with“ eine Verbindung zu einem FTP- Server herstellen. Durch „with“ öffnet es sich innerhalb eines Kontextmanagers, wodurch alles nach Ablauf des Skripts automatisch geschlossen und getrennt wird.
Wenn der Kontextmanager nicht genutzt wird, muss man die Verbindung manuell wieder trennen.

Ein Download via FTP hat verschiedene Schritte

- Es wird ein Objekt der Klasse „FPT“ mit der Host URL erzeugt.
- Danach wird der Login durchgeführt.
- Mit der Funktion „cwd()“ wechselt man das gewünschte Verzeichnis.
- Alle Inhalte des Pfades werden an eine Liste übergeben.
- Diese Liste wird iteriert.
- Mit der Funktion „retrbinary()“ wird die jeweilige Datei auf einen definierten Pfad geschrieben.

5 SQL

Zuerst muss mit dem Account des Serveradministrators ein neuer Benutzer und eine neue Datenbank erstellt werden. Im Anschluss müssen dem Nutzer die entsprechenden Rechte für die Datenbank zugewiesen werden. Das gesamte SQL-Skript „CREATE_user_database_tables.sql“ befindet sich im Unterordner SQL in der Projektabgabe

```
CREATE USER 'weatheradm'@'localhost' IDENTIFIED BY 'geheim123';
CREATE DATABASE doit_projekt;
GRANT DELETE, INSERT, SELECT, UPDATE ON doit_projekt . * TO 'weatheradm'@'localhost';
```

Formel 2 - Erstellen von SQL- Datenbank, -Benutzer und Rechtevergabe

5.1 Datenbankmodellierung

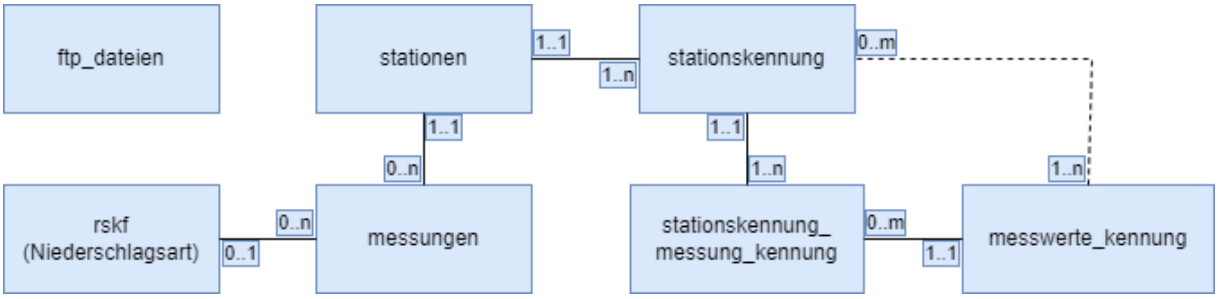

Abbildung 6 - Entity Relationship Diagram, aktuell

Diese Datenbank ist umfangreicher modelliert, als aktuell in diesem Projekt benötigt. Dadurch ist die Datenbank jedoch ordentlicher und übersichtlicher. Ein weiterer Vorteil ist, dass Daten, die einmal gesammelt wurden, für spätere Auswertungen und Analysen bereits importiert sind. Die Abkürzung „PK“ steht hier für „Primary Key“.

Reihenfolge:	Tabelle:	Attribute:	Datentyp:
1.	ftp_dateien	- Dateinamen	- varchar(50) PK
2.	rskf	- Niederschlagsform - Beschreibung	- tinyint(4) PK - varchar(400)
3.	messwerte_Kennung	- Kennung_id - Beschreibung	- varchar(6) - varchar(400)
4.	Stationskennung_ messung_kennung	- id - kennung_id - stationskennung - datum_beginn - datum_ende	- int(11) PK - varchar(6) - varchar(5) - date - date
5.	Stationen	- stations_id - plz - ort - landkreis - bundesland - land - breitengrad - laengengrad - stationshoehe	- int(11) PK - varchar(5) - varchar(40) - varchar(40) - varchar(30) - varchar(20) - float - float - int(11)
6.	Stationskennung	- stationskennung	- varchar(6) PK

		- stations_id	- varchar(5)
7.	Messungen	- messung_id - stations_id - mess_datum - qn_3 - fx - fm - qn_4 - rsk - rskf - sdk - shk_tag - nm - vpm - pm - tmk - upm - txk - tnk - tgk	- varchar(16) PK - date - int(4) - float - float - int(4) - float - float - float - int(4) - float - float - float - float - float - float - float - float

Tabelle 5 - Datenbank Tabellen

Des Weiteren ist die Tabelle „ftp_dateien“ unabhängig und steht in keiner Relation zu den anderen Tabellen. Dies sollte noch korrigiert werden, da ein Eintrag dieser Tabelle immer genau einer Station zugeordnet werden kann. Eine Station kann aber vielen ftp_Dateien zugeordnet werden.

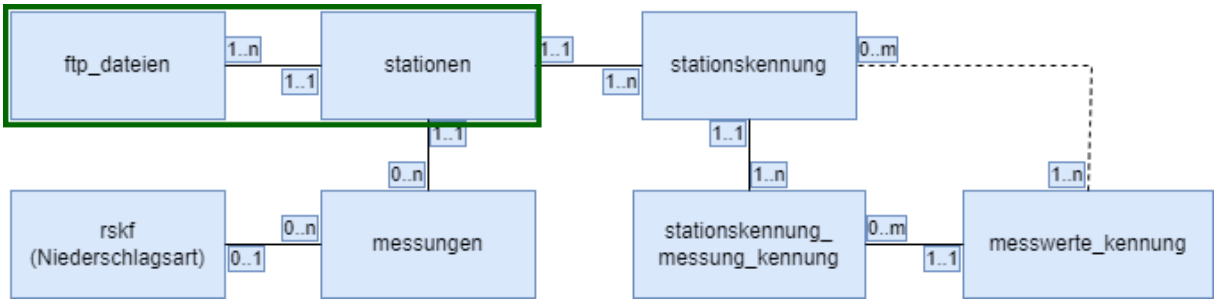

Abbildung 7 - Entity Relationship Diagram, verbessert

Die Tabelle „ftp_dateien“ müsste zusätzlich umstrukturiert werden. Der Aufbau der Attribute könnte so aussehen:

ID int PK not null,
dateinamen varchar(50)
stations_id int(11) sekondary Key not null;

5.2 Insert

Um von verschiedenen Modulen einen einheitlichen Insert zu gewährleisten, werden die Insertbefehle zentralisiert im sqlInterface.py generiert und zum Server gesendet.
Es wäre besser, wenn in dem Insertbefehl nur die Attribute und danach für jede Zeile die Werte gesammelt importiert werden würden. Dies würde Ressourcen des SQL-Servers schonen und auch die Programmgeschwindigkeit erhöhen.

Um es im Programmdesign etwas zu vereinfachen wurde für jede zu importierende Zeile, ein eigener Insertbefehl generiert. Diese werden dann zum Schluss gemeinsam „commited“. Durch dieses Vorgehen entstand bei der späteren Implementierung der Updatefunktion, der Vorteil, dass die neuen

Datensätze importiert werden können. Die bereits importierten Datensätze werden zwar vorbereitet, aber beim Commit- Statement werden diese nicht berücksichtigt, da für diese Datensätze das Execute-Statement nicht ausgeführt wird.

5.3 Datenbankabfragen

Im Testbetrieb hat sich herausgestellt, dass die Abfrage von der Tabelle „messungen“ bis zu 25 Minuten dauern kann. Dies ist keine akzeptable Geschwindigkeit und es musste nach einer alternativen Möglichkeit gesucht werden Daten geordnet und schnell abrufbar zu speichern.

Nach etwas Recherche bin ich auf das Dateiformat „Feather“ [18] [19] gestoßen. Mit diesem Datentyp können Tabellen schnell und einfach aus Python heraus generiert und gelesen werden. Die Abfrage aus der Tabelle „messungen.feather“ benötigt circa 30 Sekunden, was um ein Vielfaches schneller als vom SQL-Server ist. Wenn ich zukünftig wieder mit großen Datenmengen arbeiten sollte, werde ich direkt dieses Format in der Planung berücksichtigen.

6 Pandas – Python and Data Science

Pandas ist eine 2008 veröffentlichte Python Bibliothek, die auf NumPy aufbaut und dieses erweitert. NumPy bietet die Unterstützung von mehrdimensionalen Arrays und Möglichkeit des Vektorisierens[20]. Pandas erweitert diese Funktionen nochmals. Dadurch kann man verschiedene Tabellen oder Arrays zum Beispiel verbinden, filtern oder plotten. Pandas ist dabei sehr schnell. Die einzelnen Arrays müssen nicht durch iteriert werden, weil man mit Indizes arbeitet.

6.1 Dataframes und DataframeGroupBy

Ein Dataframe ist ein Objekt der Klasse pandas.DataFrame. Es ist eine Tabelle mit Spalten- und Zeilenindizes. Der Vorteil von Dataframes ist, dass man den einzelnen Spalten Namen geben kann. Der Zeilenindex kann frei gewählt werden. Dadurch erhält man eine Freiheit, durch die man große Datenmengen übersichtlich und verständlich sortieren kann. Man kann auch zwei Dataframes konkatenieren oder verflechten/sortiermischen. Im Englischen „concat“ und „merge“. Dies ist zu vergleichen mit dem Join von SQL. Das neue Dataframe enthält nur die Daten der gewünschten Schnittmenge.

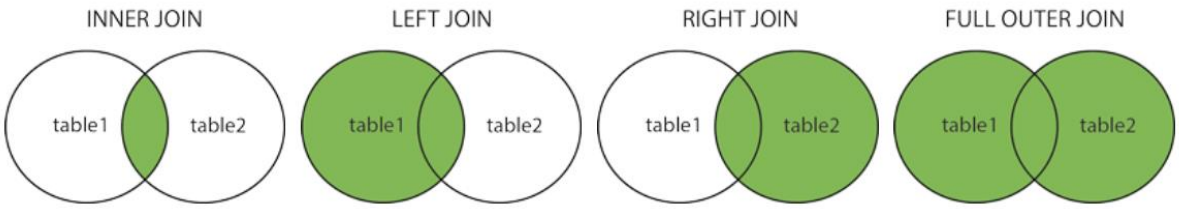

Abbildung 8 - SQL- Join [21]

In der folgenden Codezeile werden insgesamt drei Dataframes zusammen sortiert. Die Funktion muss von innen nach außen aufgelöst werden. Zuerst wird ein Dataframe aus „dfStationsMessungKennung“ und „dfStationskennung“ bei der Schnittmenge „fsv.attributeStationskennun[0]“ erstellt. Im Anschluss wird das gerade erstellte Dataframe noch mit „dfStationen“ über die Schnittmenge

[18] (Morpheus Tutorials, 2022)
[19] 6.2 Feather
[20] 14 Glossar, Mehrdimensionale Arrays und Vektorisieren
[21] (w3school.com, 2022)

„fsv.attributeStationskennung[1]“ verknüpft. Man fügt alle drei Dataframes zu einem großen zusammen.

```
pd.merge(pd.merge(dfStationsMessungKennung,
                  dfStationskennung, on=fsv.attributeStationskennung[0]),
         dfStationen, on=fsv.attributeStationskennung[1])
```

Formel 3 - pd.merge

Das Objekt DataframeGroupBy ist ein nach einer Spalte oder Zeile gruppiertes Dataframe. Hier wurde es genutzt, um die ausgewerteten historischen Messdaten nach Datum zu sortieren/ gruppieren. Dadurch konnte für jeden Tag einzeln der Minimal-, Maximal und Mittelwert ermittelt werden.

6.2 Feather

Feather ist ein portables Dateiformat um Dataframes zu speichern. Es wird in den Sprachen Python und R verwendet. Es wurde entwickelt, um eine möglichst schnelle Lese- und Schreibrate zu haben. Diese werden nur durch die Festplattengeschwindigkeit beschränkt. Es sollte „leicht wie eine Feder“ sein. Die Feather Datei benötigt mit 0,716GB nur circa 23% des von der SQL-Datenbank mit 3,02GB benötigten des Festplattenspeichers. Die Leserate ist, wie oben bereits beschrieben, circa 50-mal so schnell.[22]

6.3 Graphen plotten

Pandas bietet auch die Funktion Graphen zu plotten. Hierzu muss allerdings die Bibliothek „matplotlib“ installiert werden. Im Falle, dass das Paket fehlt, weißt die IDE einen darauf hin.

```
df.plot(y=[df.columns[i], df.columns[i + 1],
           df.columns[i + 2]], rot="45",
        ylabel=messung[1], grid=True,title=title)
```

Formel 4 - Plotting

Die Funktion „plot()“ benötigt als Parameter einen Werte für die y und/ oder x Achse. Hier können auch mehrere Datenreihen in einer Liste übergeben werden. Der Parameter „rot“ gibt die Rotation der X- Achsen Beschriftung an. „ylabel“ und „xlabel“ sind die Achsenbezeichnung. „Grid“ regelt, ob ein Raster hinter dem Diagramm sein soll. Title ist die Überschrift des Graphen. Standardmäßig wird der Graph als Liniendiagramm erstellt. Falls man dies nicht möchte, muss mit dem Parameter „kind: str“ das passende Diagramm gewählt werden. Insgesamt können elf verschiedene Diagramme erstellt werden. Die populärsten sind Linien-, Balken-, Kuchendiagramme und Histogramme.

Mit der Funktion plt.show() werden die erstellten Graphen in je einer eigenen GUI dargestellt.

[22] 5.3 Datenbankabfragen

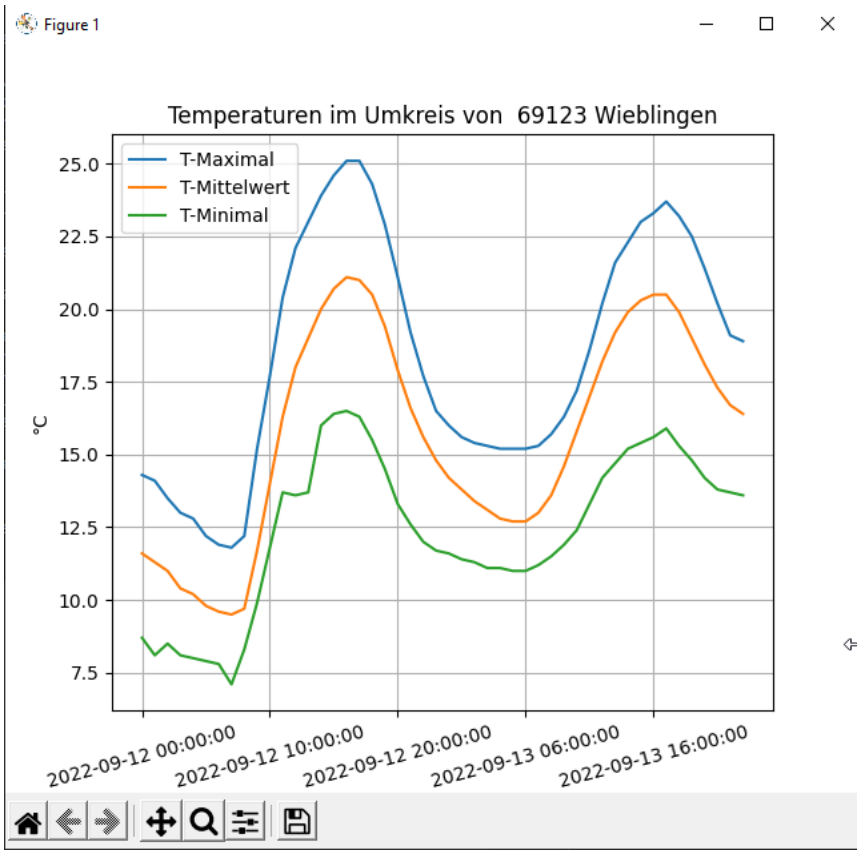

Abbildung 9 - Temperaturkurve

7 Grafische Benutzeroberfläche

Lediglich die Graphen werden in einer GUI dargestellt. Dort kann man zum Beispiel zoomen, filtern und die Graphen speichern.

Geplant war das Menü ebenfalls in einer GUI mit dem Paket TKinter zu realisieren. Dieses Framework bietet viele einsteigerfreundliche Funktionen. Zusätzlich sollte noch eine interaktive Karte[23] implementiert werden, mit der man nach Postleizahlräumen filtern kann. Leider fehlte hierfür noch die Zeit und die Menüführung wurde in einer kleinen konsolenbasierten Benutzeroberfläche realisiert.

8 Programmstruktur

Der Quellcode wurde in verschiedene Bereiche und Ordner unterteilt.

Pfad:	**Inhalt:**
./Python	- Projektordner mit allen enthaltenen relevanten Dateien - Projektübergreifende Skripte, die in mehreren Bereichen benötigt werden
./Python/Attic	- Nicht mehr benötigte Dateien und Ordner
./Python/Auswertung	- Unterprogramme, die in der Datenbank und Feather Datei liegenden Daten auswerten

[23] (suche.postleizahl.org, 2022)

./Python/Datenbeschaffung	- Alle Programme für den Download der Rohdaten, deren Aufschlüsselung und Import
./Python/venv	- Enthält alle Packages und Libraries - Dieses Verzeichnis ist nur vorhanden, wenn innerhalb von PyCharm unter „Python Packages“ Pakete installiert wurden
./Python/Übungen	- Alle nachprogrammierten Übungen aus dem Java- Unterricht. Diese sind nicht relevant für das Hauptprogramm der Wetteranalyse
./External Libraries	- Die über pip installierten globalen Packages und Libraries. Diese sind über alle Projekte verfügbar.

Tabelle 6 – Projektstruktur

Eine detailliertere Beschreibung der Programme und deren Funktionen befindet sich im Anhang.[24]

8.1 Programmier- und Designfehler

8.1.1 Programmfehler bei Datentypen

```
Verbindung zum SQL Server erfolgreich.
dtypes der Spalten von dfStationen:
Index:  STATIONS_ID      int64
Spalten:
 PLZ                int64
ORT               object
LANDKREIS         object
BUNDESLAND        object
LAND              object
BREITENGRAD      float64
LAENGENGRAD      float64
STATIONSHOEHE      int64
dtype: object
dtypes der Spalten von dfStationskennung:
Index:  STATIONSKENNUNG      object
Spalten:
 STATIONS_ID    int64
dtype: object
dtypes der Spalten von dfStationsMessungKennung:
Index:  ID      object
Spalten:
 KENNUNG_ID         object
STATIONSKENNUNG     int64
DATUM_BEGINN       object
DATUM_ENDE         object
dtype: object
```

Formel 5 - Stationslisten Datentypen

In der letzten Phase des Projektes schlich sich ein gravierender Fehler in den Programmcode. In der Funktion „erstelleStationenDataFrames()“ im Modul ./Auswertung/stationsermittlung.py werden die Datentypen, in Pandas „dtype“ genannt, der Spalten innerhalb des Dataframes „dfStationsMessungKennung“ nicht korrekt konvertiert. Somit ist der „dtypes“ unterschiedlich und die Tabellen können nicht mit der Funktion „merge“ zusammengefügt werden. Der Datentyp des Indexes des dfStationskennung ist ein andere, wie der von der Spalte Stationskennung im dfStationsMessungKennung. Der Datentyp dieser Spalte muss zu String geändert werden, damit Pandas den dtype „object“ erkennt.

Mit dem in **Fehler! Verweisquelle konnte nicht gefunden werden.** gezeigten Befehl sollte der Datentyp richtig zugewiesen werden. Aus Unachtsamkeit wurde die Zeile 49 der Abbildung in den Quellcode geschrieben. Eigentlich sollte die Zuweisung wie in Zeile 51+52 erfolgen. Nach der Änderung dieses Fehlers, funktioniert das Programm wieder ohne Probleme.

```
48  # Programmierfehler:
49  # dfStationen[fsv.attributeStationen[0]] = dfStationen[fsv.attributeStationen[0]].astype(str)
50  # Diese Zeile sollte eigentlich eingefügt werden:
51  dfStationsMessungKennung[fsv.attributeStationMessungKennung[2]] = \
52      dfStationsMessungKennung[fsv.attributeStationMessungKennung[2]].astype(str)
```

Formel 6 - Datentypzuweisung in Pandas

[24] 11 Übersicht Programme und Funktionen

8.1.2 Hartcodierte Pfade

```
        os.remove(f"archiv/{wertedatei}")
        os.remove(dateiname)

        print(f"{datensatzZähler} Datensätze wurden importiert. ")
        dateienZähler += 1
    for datei in os.listdir():
        if datei == "archiv":
            os.rmdir("archiv")
```

Abbildung 10 - hartcodierte Strings

In dem Modul „ftpDownloadUndUpdateMessungen“ wurde der Pfad hart in den Code geschrieben, obwohl in der Variable „archivpfad“ der Pfad enthalten ist. Diese hart codierten Strings müssen durch einen F-String mit der oben genannten Variablen ersetzt werden.

8.1.3 Globale Variablen

Im Entwicklungsprozess ist leider die Tatsache missachtet worden, dass alle nicht in einer Funktion enthaltenen Variablen „public“ sind. Das bedeutet sie können von allen Modulen gelesen und geändert werden. Als Lösungsansatz sollten die unveränderbaren Variablen, im englischen „immutable“, von einer Funktion umschlossen werden. Der folgende Codeausschnitt veranschaulicht noch einmal dieses Problem und dessen Lösungsansatz. Aus Zeitgründen konnte für diesen Designfehler kein Refactoring mehr durchgeführt werden.

```
sqlTabelleFtpDateien = "ftp_dateien"

def getSqlTabelleFtpDateien():
    return "ftp_dateien"
```

Formel 7 - Lösungsansatz für public Variablen

Diese Funktionen arbeiten wie ein Getter eines Objektes. Wenn diese aufgerufen werden, erhält man die Variable als Rückgabewert. Da diese durch die Programmierung nur einen Wert zurückgeben, sind die Variablen geschützt, also immer noch „public“, aber „immutable“.

9 Projektabschluss

9.1 Soll-Ist-Vergleich

Soll	Ist	Soll erfüllt?
Erlernen des Umgangs mit großen Datenmengen	Automatischer Download, Verarbeitung und effiziente Speicherung	Erfüllt
Erlernen von Python	Erfolgreiche Programmierung des Projekts in Phyton	Erfüllt
Abruf der Wetterdaten über API	Abruf der Vorhersage über get request, da der Zugang über deutschland.api nicht möglich war	Erfüllt
Historische Wetterdaten	Herunterladen über FTP-Download	Erfüllt

Datenspeicherung auf MySQL Server	Speicherung der Daten auf MySQl Server und im effizienteren Feather Format	Erfüllt
Anschauliche und verständliche Aufbereitung der Daten	Möglichkeit zur Erstellung von Graphen für verschiedene Messwerte, Standorte und Zeiten	Erfüllt
Vorgegebene Messwerte: - Temperatur - Wind - Sonnenstunden - Niederschlag - Schadstoffbelastung	Auswertbare Messwerte: - Temperatur - Wind - Sonnenstunden - Niederschlag	Größtenteils erfüllt
Aufbereitung: - Erstellung von Graphen - Ablesen von Trends - Wetterprognosen	- Möglichkeit zur Erstellung von Graphen - Bedingte Ablesbarkeit von Trends - Abrufung Wetterprognosen vom DWD	Größtenteils erfüllt
Grafische Benutzeroberfläche (GUI)	- Konsolenbasiertes Menü - Darstellung der Graphen in einer GUi	Teilweise erfüllt.

Tabelle 7 - Projektziele Soll-Ist-Vergleich

9.1.1 Zeitplanung

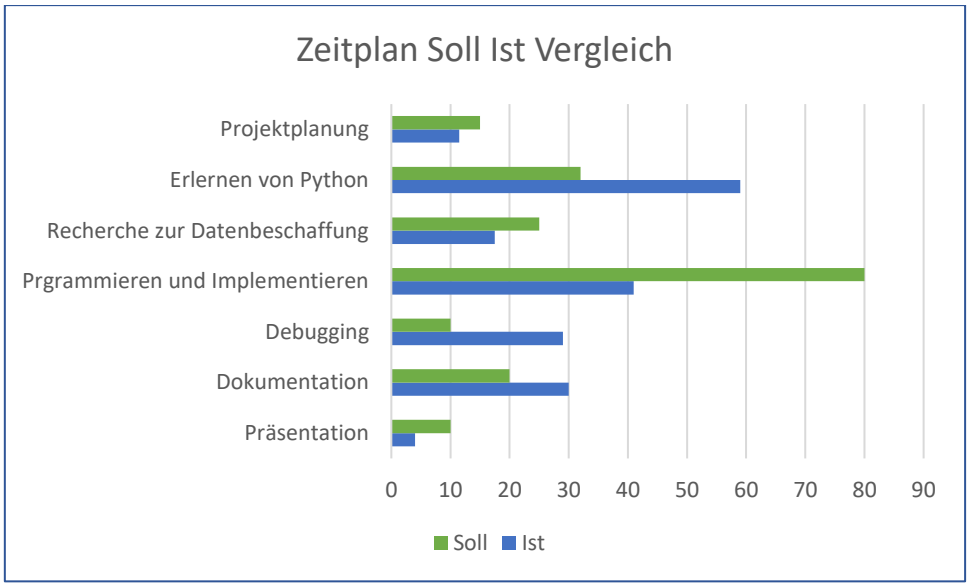

Abbildung 11 - Zeitplan Soll-Ist-Vergleich

9.1.2 Pflichten- und Lastenheft

Da das Ja-Ankreuzen des Feldes „Projektdokumentation“ der missverständlichen Formulierung des Projektantrages geschuldet war und ich dies dem Fachdozenten direkt nach Abgabe des Antrags mitteilte, habe ich kein Pflichten- und Lastenheft erstellt.

Im Vorfeld wurde uns mitgeteilt, dass eine Projektdokumentation Pflicht sei. Das Pflichten- und Lastenheft ist im Projektantrag als Projektdokumentation ausgewiesen. Hier sollte klarer dargestellt werden, dass sich das Pflichten- und Lastenheft von der Pflichtdokumentation unterscheidet. Hier noch zur Veranschaulichung der entsprechende Ausschnitt:[25]

[25] (Landmann, 2018)

Projektdokumentation (Lasten-/Pflichtenheft mind. 10 Seiten)	Ja ☒	Nein ☐
Kundenhandbuch (mind. 5 Seiten)	Ja ☐	Nein ☒

9.2 Vergleich zwischen Java und Python

In der nachfolgenden Tabelle werden nur die mir aufgefallen Vorteile aufgelistet. Die Vorzüge der einen Sprache sind oft die Nachteile der anderen.

Java	**Python**
Durch gezwungene Deklaration von Datentypen, Sichtbarkeit und Klassen- oder Instanzvariablen ist der Einstieg einfacher. Lernende finden sich einfacherer zurecht.	Durch die gezwungene Einrückung von Funktionen und Kontrollstrukturen ist der Code sehr übersichtlich.
Gettern und Settern lassen sich einfacher erstellen und haben für Anfänger eine übersichtliche Struktur.	Der Aufruf von Getter und Setter ist durch die sogenannten Properties einfacher.
Durch die einfachere Deklaration von z.B. final, public und private Variablen, lassen sich Klassen leichter „sicher“ und übersichtlicher konzipieren.	Da Python nicht streng typisiert ist, können Konstruktoren und deren Aufruf vielseitiger implementiert werden. Man kann nur einen Konstruktor pro Klasse implementieren. Durch integrierte Kontrollstrukturen können die Parameter nach Typen neu sortiert werden.
Mittels Labels kann in verschachtelten Schleifen die äußerste Schleife unterbrochen werden.	Funktionsparameter können ähnlich wie bei einem Dictionary zugeordnet werden. Man schreibt den Namen des Parameters und danach ein Gleichheitszeichen gefolgt von dem Parameterwert in die Parameterübergabe.
	Durch optionale Funktionsparameter wird keine Funktionsüberladung benötigt. Dies verkürzt den Quellcode und macht ihn übersichtlicher.
	Durch die ausgeschriebenen Keywords liest sich der Quellcode fast wie Englisch.
	Die Syntax ist einfacherer und kürzer wie zum Beispiel bei Texteingaben: var = str(input(„Bitte Text eingeben“))

Tabelle 8 - Vergleich Java und Python

Trotz der größeren Zahl an Vorteilen von Python würde ich Java als erst zu lernende Sprache empfehlen. In ihr ist meiner Meinung nach das Erlernen von grundsätzlichen Programmierkonzepten und -paradigmen einfacher. Man hat mehr Richtlinien als in Python. Diese Freiheit kann schnell zur Falle werden, was durch die globalen Variablen in diesem Projekt verdeutlicht wird. Man muss sich während der Planungsphase mehr Gedanken über die Architektur und das Programmdesign machen. Nur so können strukturierte und sichere Programme und Projekte entstehen. Man muss festhalten, dass jede Programmiersprache ihre Stärken und Schwächen hat. Somit ergeben sich von selbst deren optimale Einsatzgebiete.

9.3 Probleme und Hürden

9.3.1 Recherche, Debugging und Tests

Die im Vorfeld geschätzte Zeit für Recherche hat nicht ausgereicht. In den geplanten 32 Stunden für das Erlernen von Python konnte ich nur die Grundlagen erarbeiten. Es fehlte jedoch tiefergehendes Wissen über die verschiedenen Module, Kontextmanager und Bibliotheken. Der zeitliche Mehraufwand musste bei anderen Projektphasen gekürzt werden. Somit schlichen sich dann natürlich viele Fehler ein, die bei den Tests auffielen. Teilweise benötigte ich für das Debugging kleiner Fehler mehrere Stunden.

9.3.2 Defektes Notebook

Während der IKT-Woche musste ich mein Notebook aufgrund eines Displaydefektes zur Reparatur einsenden. Zuvor wurde mir seitens des Supports aus Datenschutzgründen geraten alle Daten zu sichern und das Notebook auf Werkseinstellung zu setzen. Der Reparaturservice würde das Notebook sowieso zurücksetzen. Somit habe ich alle Daten gesichert und nach dem Urlaub den Laptop meiner Frau verwendet. Diese benötigte ihn jedoch schnellst möglichst wieder selbst. Als mein Notebook nach dem Sommerurlaub wieder ankam, gingen zwei volle Tage aufgrund der Einrichtung des Betriebssystems, der Installation aller Programme und dem Portieren des Projekts verloren. Dabei hatte ich Probleme bei der Installation von Python und dem Pip Paketmanager. Die Anleitung zum Lösen dieser Schwierigkeiten befindet sich in Form einer Schritt-für-Schritt Anweisung im Anhang.[26]

9.3.3 Fehlende Versionskontrolle

Im Vorfeld zum Projekt hatte ich nicht die Möglichkeit mich ausführlich mit einer passenden Form der Versionskontrolle auseinanderzusetzen. Das Verständnis für Systeme wie Git ist zwar vorhanden, jedoch fehlte das Wissen zur praktischen Anwendung innerhalb der IDE. Somit war es oft schwer den Überblick zu behalten, was in welchem Programm bearbeitet oder geändert wurde. Gerade auch durch das defekte Notebook und den damit verbundenen Portierungen der Projektunterlagen, kam es zu Verwechslungen des aktuellen Standes der Programme.

9.4 Fazit und Ausblick

Dieses Projekt war eine gute Herausforderung. Es vereinte viele Bereiche, die mich sehr interessieren. Besonders hervorzuheben ist der Bereich des Data Mining. Wenn ich zu Beginn des Projekts bereits das Wissen über Pandas Dataframes und deren Umgang gehabt hätte, wäre für die Datenspeicherung kein SQL- Server verwendet worden.

Die aktuell möglichen Auswertungen sind nur ein kleiner Teil des Realisierbaren. Mir ist während der Datenbeschaffung bewusst geworden, dass es beim Data Mining vorrangig um das Sammeln von Daten geht. Das bedeutet, man versucht so viele Informationen wie möglich von den Datenquellen zu speichern. Es ist zweitrangig, ob man bereits eine Verwendung der Daten hat. Wurden sie einmal gesammelt und gespeichert, kann man im Nachgang in aller Ruhe Auswertungen durchführen. Aus diesem Grund habe ich die Datenbank umfangreicher angelegt, als sie zum aktuellen Stand des Projektes benötigt wurde. Man kann dadurch leicht neue Features hinzufügen und nach anderen Merkmalen filtern. Dabei werden die Grenzen nur durch die eigene Fantasie gesteckt. Interessant wird es, wenn man verschiedene Messwerte miteinander vergleicht. Als Beispiel:

- Durchschnittstemperatur und Regentage
- Windgeschwindigkeiten und Regenmenge
- Windgeschwindigkeit und Bodentemperatur
- Art des Schneefalls und Menge über die Jahre

[26] 12 Installation von Python

- Temperaturvergleiche von urbanen und ländlichen Gebieten, Stichwort „Urban Heat Effect“

Obwohl durch dieses Projekt das ganze Thema nur angerissen wurde war es für mich ein Erfolg. Ich konnte mein Wissen breit gefächert erweitern. Ich habe sehr viel über mich und meine Arbeitsweise gelernt.

10 Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe.

Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe. [27]

Heidelberg, den 09.09.2022

Severin Fröhlin

[27] (Genau)

Anhänge

11 Übersicht Programme und Funktionen

Im Folgenden werden die einzelnen Unterprogramme und deren Funktionen näher beschrieben.

- Die Überschriften sind wie ein Pfad aufgebaut.
- Parameter mit dem Zusatz „pa: Datentyp“ erwarten einen bestimmten Datentyp.
- Parameter mit dem Zusatz „pa=None“ sind Optional:
    - Die Liste der Übergabewerte werden von links nach rechts den Funktionsparametern zugeordnet.
- Bei Funktionsaufrufen können die Übergabeparameter ähnlich wie bei einem Dictionary übergeben werden. Als Beispiel:

```
def filtereMessungen(stationen, messwerte, datumBeginn=None, datumEnde=None):...

filtereMessungen(stationen=[2290,"G005"], datumBeginn="2010-01-01",
                 messwerte="TMK", datumEnde="2010-02-01")
```

Abbildung 12 - Funktionsaufruf und Parameterübergabe

11.1 ../Python

11.1.1 ./funktionsUndVariablensammlung.py

Diese Datei ist als Sammlung für alle Variablen und Funktionen gedacht, welche von mehreren Programmen dieses Projekts benötigt werden.

Importierte Module:

- from geopy.geocoders import Nominatim as nm
- import pandas as pd
- import timeit

Globale Variablen: geolocator, sqlTabelleFtpDateien, attributeFtpDateien, sqlTabelleStationskennung, attributeStationskennung, sqlTabelleStationMessungKennung, attributeStationMessungKennung, sqlTabelleMessungen, attributeMessungen, sqlTabelleStationen, attributeStationen, messungenDict

ersetzen (abschnitt: str)

- abschnitt vom Typ String

Der erhaltene String wird nach fest definierten Zeichen durchsucht und ersetzt diese im Anschluss.

Rückgabe: umformatierte String

setZeitstempel ()

aktualisiert die globale Variable „zeitstempel“

Rückgabe: Keine

setSpaltenNamenVonDataFrame (df: pd.DataFrame, liste)

- df Objekt vom Typ Dataframe
- liste eine Liste mit Namen

Die Variable „liste“ wird in ein Dictionary konvertiert. Im Anschluss werden die Spaltennamen des Dataframes mit den Dictionary Inhalten neu benannt.

Rückgabe: Dataframe mit neuen Spaltennamen

datumZuJahrMonatTag (datumUnformatiert: str)

- datumUnformatiert Ein Datum als String im Format yyyy-mm-tt

Konvertiert den erhaltenen Datumsstring zu einer Liste mit Integern.

Rückgabe: Liste [int(Jahr), int(Monat), int(Tag)]

konvertiereDataframe (dataframe: pd.DataFrame)
Sortiert das Dataframe nach Datum. Danach werden jeweils die Minimal-, Maximal- und Mittelwerte ermittelt.

Rückgabe: DataframeGroupBy mit Maximal-, Minimal- und Mittelwert je Messwert

speichereFeather (df: pd.DataFrame, pfad: str)
Speichert ein Dataframe an dem angegeben Pfad.

Rückgabe: Keine

leseFeather (pfad: str)
Liest ein Dataframe vom angegebenen Pfad.

Rückgabe: DataFrame

11.1.2 ./main.py

Dieses Programm ist das Hauptprogramm mit dem Userinterface. Durch die verschiedenen Eingaben wird man durch das Menü geführt. Von hier aus werden alle Pakete und Module aufgerufen. Im Bereich „Auswertungen“ werden zum Beispiel die Module aus dem Unterorder „Auswertung“ aufgerufen.

Importierte Module:

- import sqlInterface
- from funktionsUndVariablensammlung import attributeMessungen as aMess
- import funktionsUndVariablensammlung as fsv
- from Auswertung import *
- import Datenbeschaffung as db

Globale Variablen: plz, ortsName, radius, stationen

11.1.3 ./pip_install_packages.bat

Ein Installationsskript, welches mit pip alle Packages und Libraries aus der „requirements.txt“ downloadet und installiert.

11.1.4 ./pip_update_packages.bat

Ein Installationsskript, welches alle mit pip installierten Python Erweiterungen ausliest und updatet.

11.1.5 ./sqlInterface.py

Dieses Modul stellt eine Verbindung zum SQL- Server her und dient gleichzeitig als Schnittstelle zu diesem. Alle Programme dieses Projekts, die etwas an den SQL-Server senden oder abrufen möchten, haben diese Datei importiert.

Importierte Module:

- - import mysql.connector

Globale Variablen: dbDoitProjekt, datenbank

update (attribute, tabelle: str)

- Attribute: Liste mit Attributnamen der SQL- Tabelle
- Tabelle: Name der SQL- Tabelle

Generiert aus tabelle und der Liste attribute das Updatestatement für den SQL-Server. Dabei spielt es keine Rolle, wie viele Attribute in der Liste enthalten sind. Das Statement passt sich automatisch an.

Rückgabe: SQL-Statement als String

insert (attribute, tabelle: str)

- Attribute: Liste mit Attributnamen der SQL- Tabelle
- Tabelle: Name der SQL- Tabelle

Generiert wie die Funktion update() ein String mit dem SQL-Statement. In dem Falle einen Insertbefehl. Die Länge der Liste attribute ist dabei irrelevant. Auch hier wird der String automatisch an die Anzahl der Listenelemente angepasst.

Rückgabe: SQL-Statement als String

select (tabelle, attribute=None)

- tabelle Name der SQL-Tabelle
- attribute Liste der Attribute für die Rückgabe

Wertet vom SQL-Server die gewünschten Attribute aus.

Rückgabe: Tupel mit Tupeln mit den Zeilen der SQL-Tabelle

executeStatement (statement, parameterListe=None)

- statement Ist das vorbereitete Statement update(), inssert() oder select() Funktionen
- attribute Liste von Attributen, bei einem Selectbefehl nicht nötig

Versucht das vorbereitete Statement an den SQL- Server zu senden.

Rückgabe: Keine

commitStatement ()

Versucht die an den SQL-Server gesendeten Statements zu bestätigen, um sie endgültig auszuführen

Rückgabe: Keine

closeConnection ()

Da der Aufbau zum SQL-Server sich nicht in einem Kontextmanager befindet, muss bei der Beendigung des Programmes die Verbindung getrennt werden.

Rückgabe: Keine

11.2 ./Python/Auswertung

11.2.1 ./histotischeMesswerte.py

In diesem Modul sammeln sich Funktionen, die zur Auswertung der historischen Messdaten benötigt werden.

Importierte Module:

- from datetime import date
- import sqlInterface as sql
- import pandas as pd
- import funktionsUndVariablensammlung as fsv

Globale Variablen: pfad, dfMessungen

erstelleMessungenDataFrames ()
Diese Funktion löst einen „select all“ Befehl für die Tabelle „messungen“ aus. Die Ergebnisse werden in der globalen Variable dfMessungen gespeichert.

Rückgabe: Keine

aktualisiereMessungenFeather ()
Speichert das Dataframe dfMessungen unter dem Pfad, welcher in der globalen variable „pfad“ angegeben ist. Dies dauert circa 25 bis 30 Minuten.

Rückgabe: Keine

filtereMessungen (stationen, messwerte, datumBeginn=None, datumEnde=None)

- stationen Liste mit Integern von Stations_ids
- messwerte Liste mit Strings von Messwerten
- datumBeginn Ein Datum als String im Format yyyy-mm-tt
- datumEnde Ein Datum als String im Format yyyy-mm-tt

Filtert das Dataframe dfMessungen nach den angegebenen Parametern. Während der Rückgabe wird das Dataframe dfGefiltert noch an die Funktion fsv.konvertiereDataframe() übergeben.

Rückgabe: nach Messdatum gruppiertes DataFrameGroupBy

ladeHistorischeMessungen ()
Importiert historische Messdaten aus der, in der globalen Variablen „pfad“ angegebenen, Feather Datei. Ist dies nicht möglich, wird die Funktion aktualisiereMessungenFeather() aufgerufen, um die Messdaten vom SQL- Server zu laden und danach eine Feather Datei zu schreiben.

Rückgabe: Keine

11.2.2 ./plotting.py

Dieses Modul stellt Dataframes als Graphen dar.

Importierte Module:

- import matplotlib.pyplot as plt
- import funktionsUndVariablensammlung as fsv

Globale Variablen: Keine

erstelleGrafikenMessungen(df, ort, auswahl=None)

- df DataFrameGroupBy der vorgefilterten und sortierten Dataframes

- ort Liste aus Geopy.geocoder Ortsdaten des gefilterten Gebietes
- auswahl Auswahl der zu plottenden Werte

Hier wird das Dataframe mit den übergebenen Werten geplottet und die Graphen werden in einem Fenster auf dem Bildschirm dargestellt.

Rückgabe: Keine, öffnet aber die Graphen als GUI-Fenster.

11.2.3 ./stationsermittlung.py

Dieses Modul bündelt Funktionen, die für die Auswahl und Filterung von Stationen zuständig sind.

Importierte Module:

- from geopy import distance
- import sqlInterface as sql
- import pandas as pd
- import funktionsUndVariablensammlung as fsv

Globale Variablen: dfStationen, dfStationskennung, dfStationsMessungKennung, zuSuchenderOrt, plzDictionary, latlonPlz, distanz, featherPfad

ermittleLatLon(ort)

- ort Ortsbezeichnung als Name oder Postleizahl

Diese Funktion ermittelt die Breiten- und Längengrade des Parameters „ort“ und aktualisiert zusätzlich die globalen Variablen „zuSuchenderOrt“, „plzDictionary“, „latlonPlz“.

Rückgabe: String mit Breiten- und Längengrad

erstelleStationenDataFrames()

Aktualisiert die globalen Variablen dfStationen, dfStationskennung, dfStationsMessungKennung mit den Werten der SQL-Abfrage für das jeweilige Dataframe.

Rückgabe: Keine

errechneDistanzZuPostleizahl(zeile)

- zeile Zeile aus dem Dataframe dfStationen

Errechnet mithilfe von GeoPy die Distanz zwischen einer Postleizahl und der in „zeile“ übergebenen Position aus deren Breiten- und Längengraden.

ermittleStationenImUmkreis(ort, radius, kennung=None)

- ort Ortsbezeichnung als Name oder Postleizahl
- radius radius in km zu „ort“
- kennung String z.B. „SY“, um nach Kennung filtern zu können

Diese Funktion berechnet anhand einer Postleizahl oder einem Ortsnamen und dem Radius die Distanz zu allen in der Stationsliste enthalten Stationen. Im Anschluss werden die Dataframes dfStationen, dfStationskennung und dfStationsMessungKennung mit der Funktion „merge“ an der Schnittmenge „zusammensortiert oder vermischt“.[28]

Rückgabe: DataFrame aller Stationen innerhalb des Radius

[28] 6.1 Dataframes und DataframeGroupBy

filtereStationenNachKennung(df, kennung)
Rückgabe: Dataframe aller Stationen mit der übergebenen Kennung

11.2.4 ./wetterVorraussage.py

Dieses Modul ist für die Abfrage der aktuellen Wetterdaten zuständig.

Importierte Packages:

- import pandas as pd
- import requests
- import funktionsUndVariablensammlung as fsv
- from datetime import datetime
- import json

Globale Variablen: stationen, zeitstempel, url

abfrageVorbereiten(stationsIdListe)
Formatiert die Liste stationsIdListe zu einem für die apiAbfrage passenden Query String.

Rückgabe: String mit durch Kommata getrennte StationsIds

apiAbfrage(dfStationen)
Aus dem Dataframe dfStationen werden die StationsIds als Liste in der globalen Variable „stationen“ gespeichert. Im Anschluss wird die Funktion „abfrageVorbereiten()“ aufgerufen, um die globale Variable „url“ für den URL Request zu vervollständigen. Der Rückgabewert des Requests ist ein Bytecode, der erst zu einem String konvertiert werden muss. Um diesen String besser auswerten zu können, wird er bei der Rückgabe in ein JSON Format konvertiert.

Rückgabe: JSON

konvertiereDwdApiAbfrageZuForecastStündlich(dwdApiAbfrage)
Zuerst werden aus dem JSON der dwdApiAbfrage zwei Listen erstellt. Die erste enthält die Stationsnamen, die zweite deren Temperaturwerte. Aus diesen beiden Listen wird mithilfe des Zeitstempels und der Schrittweite der Messwerte eine Liste der Zeitpunkte der einzelnen Daten generiert.

Aus diesen drei Listen wird ein Dataframe erzeugt, welches dann für das Plotten vorbereitet wird.

Rückgabe: DataFrame

konvertiereDwdApiAbfrageZuForecastTage(dwdApiAbfrage)
Diese Funktion funktioniert sehr ähnlich der konvertiereDwdApiAbfrageZuForecastStündlich(). Mit dem Unterschied, dass hier aus dem JSON andere Werte für die Berechnung ausgewertet werden. Hier werden nur die Tageswerte und nicht die stündlichen Werte ausgelesen.

Rückgabe: DataFrame

11.3 ./Python/Datenbeschaffung

11.3.1 ./ftpDownloadUndUpdateMessungen.py

Dieses Programm führt den Download, das Entpacken, das Aufbereiten für den SQL-Insert und die Löschung der bearbeiteten Archive der Rohdaten der historischen Wetterdaten durch.

Importierte Module:

- import zipfile
- from ftplib import FTP

- import sqlInterface as sql
- import os
- import funktionsUndVariablensammlung as fsv
- from funktionsUndVariablensammlung import attributeMessungen as aMess

Globale Variablen: host, pfad, benutzer, passwort, bereitsImportierteArchive

updateMessungenDatenbank()

Zuerst stellt die Funktion eine Verbindung zum FTP Server her, der Ordner wird gewechselt und die Namen der enthalten Dateien werden in einer Liste gespeichert. Danach wird jede Datei einzeln mit der Datenbanktabelle „ftp_dateien“ abgeglichen und geprüft, ob sie schon einmal heruntergeladen wurde. Falls ja, wird die Datei übersprungen. Falls nein, wird die Datei, beziehungsweise das Zip-Archiv, auf den lokalen Speicher kopiert. Das Archiv wird geöffnet und das Textdokument mit den Messdaten wird entpackt und geöffnet.

Das Textdokument ist ein sehr langer String, der zuerst in einzelne Zeilen gesplittet werden muss. Die Zeilen werden als Liste an die Funktion messungenImportieren() übergeben und dort in die Datenbank importiert.
Wenn das erfolgreich war, wird der Dateiname über das Executestatement im SQL-Interface der Datenbanktabelle „ftp_dateien“ hinzugefügt.

Danach wird die Datei wieder gelöscht.

Wenn alle Dateinamen des FTP-Servers bearbeitet wurden, wird noch der Ordner „archiv“ gelöscht.

messungenImportieren(zeilen)

- zeilen Liste mit allen Zeilen einer Messungen Textdatei.

Diese Funktion nimmt Zeile für Zeile der übergebenen Liste und zerteilt sie in einzelnen Messwerte, die dann über das sql.executeStatment in die Datenbank „messungen“ importiert werden.

Rückgabe: Integer, zum Aufaddieren der importierten Messungen.

11.3.2 ./sqlImportStationenV4.py

Wenn dieses Modul gestartet wird, lädt sich das Skript das aktuelle Stationslexikon von der Webseite des Deutschen Wetterdienstes über einen get request. Danach werden vom SQL-Server die Tabellen „stationen“ und „stationskennung_messung_kennung“ abgefragt und in einer Variable gespeichert.

Importierte Module:

- import sqlInterface as sql
- import funktionsUndVariablensammlung as fsv
- import requests
- import os

Globale Variablen: downloadURL, stationsLexikon, statLex, bereitsImportierteStationen, bereitsImportierteStationskennungMessungKennung, location

ermittleOrtsDaten(filter: str)

Ermittelt je nach Inhalt des Parameters den entsprechenden Wert aus dem Dictionary der globalen Variablen „location“. Bei der Eingabe „postcode“ erhält man zum Beispiel als Rückgabe die Postleizahl. Falls der Filter nicht in „location“ enthalten ist, ist der Rückgabewert „None“.

Rückgabe: String

updateDatenbankenStationslisten()

Zuerst wird der Inhalt des get-requests in der Datei statLex.txt gespeichert. Danach wird diese Datei geöffnet und bei den Zeilensprüngen in einzelne Strings/ Zeilen unterteilt. Die ersten Zeilen werden übersprungen und danach werden aus jeder Zeile die einzelnen Werte ausgelesen.

Die einzelnen Werte werden den Tabellen zugeordnet und über das SQL Interface in die Datenbank importiert. Zuvor wird der Datensatz immer noch mit den oben abgerufenen Tabellen abgeglichen, ob sie bereits importiert wurden. Falls ja werden die Datensätze geupdatet. Um die referenzielle Integrität zu wahren, werden die Stationen und die Stationskennungen nach den Inserts in einer Liste gespeichert. Somit wird gewährleistet, dass, wenn ein neuer Datensatz mit derselben StationsID ausgelesen wird, nur noch die Tabelle „stationskennung“ oder „stationskennung_messung_kennung“ als update oder insert an den SQL- Server gesendet wird.

11.4 ./Python/Übungen

Dieser Ordner enthält alle Übungsprogramme aus Java, die ich zum Erlenen der Sprache Python nachprogrammiert habe. Da diese für das Projekt direkt keine Relevanz haben, werde ich die Programme auch nicht näher beschreiben. Gerade die späteren Programme, wie die Kontoverwaltung oder die Inventarverwaltung enthalten Fehler, die ich nicht verbessert habe.

11.5 ./Python/Attic

Dieser Ordner enthält nur Programme und Funktionen, die nicht mehr benötigt werden oder verworfen wurden. Sie haben wie die Übungen keine Relevanz für das eigentliche Projekt.

12 Installation von Python

12.1 Windows Umgebungsvariablen hinterlegen

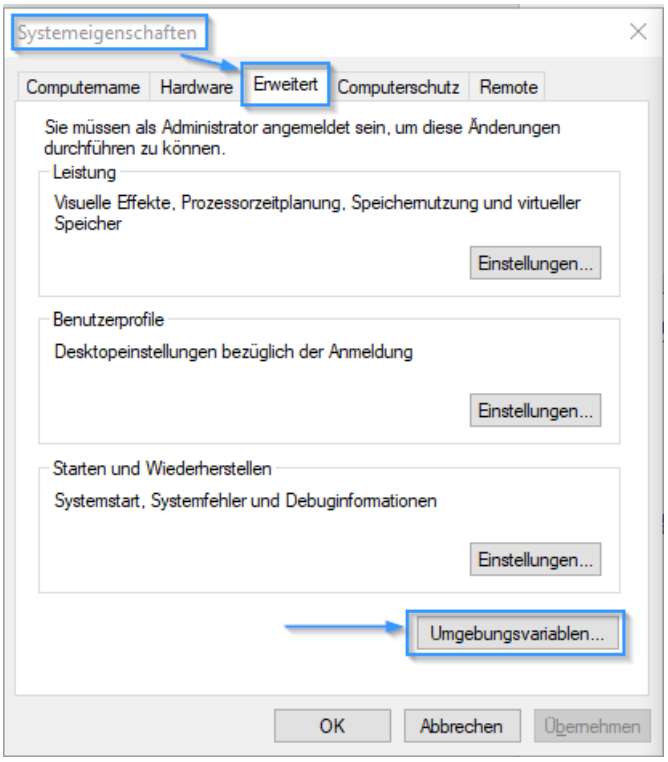

Abbildung 13 - Systemeigenschaften

Um in Windows Umgebungsvariablen hinzufügen zu können benötigt man zuerst Administratorrechte.

Man drückt die Windowstaste und gibt in das Suchfeld „Systemeigenschafen“ ein. Unter dem Register „Erweitert“ findet man ganz unten eine Schaltfläche mit der Bezeichnung „Umgebungsvariablen…“.

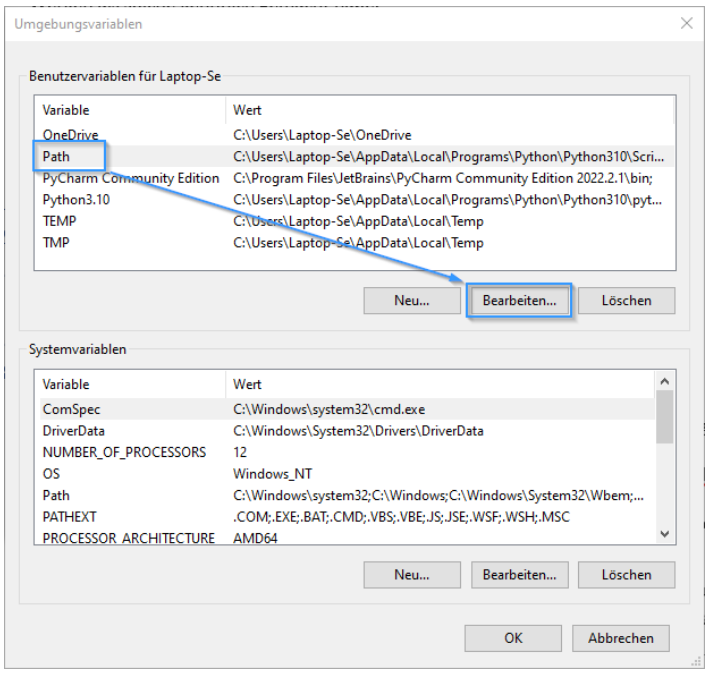

Abbildung 14 - Umgebungsvariablen

In den Umgebungsvariablen markiert man die Variable „Path“ und wählt die Schaltfläche „Bearbeiten…“.

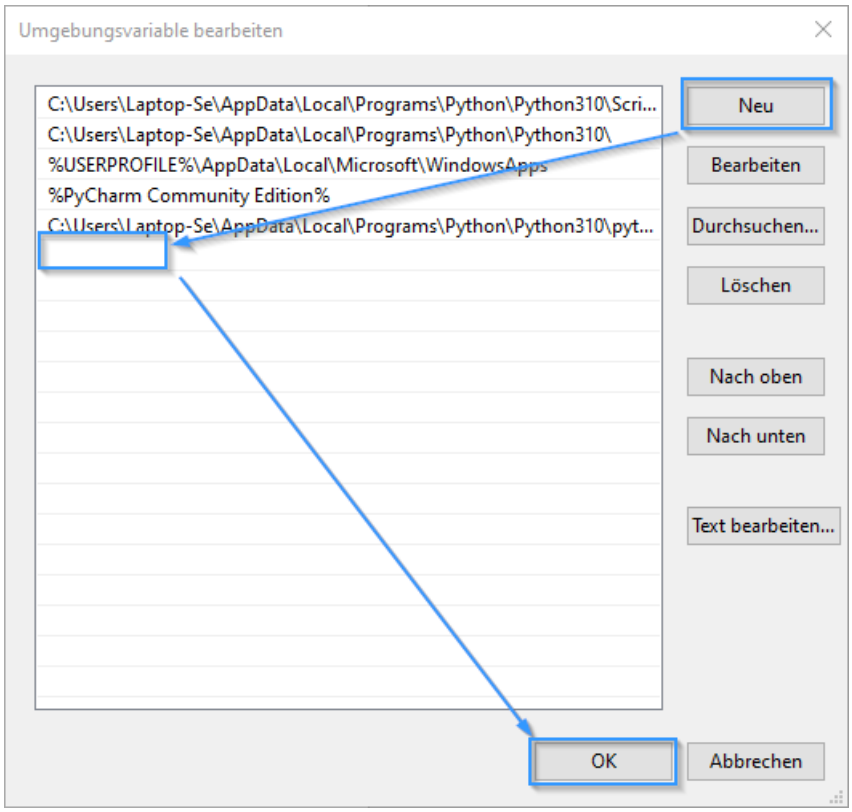

Abbildung 15 - Umgebungsvariablen bearbeiten

Hier muss man auf „Neu“ klicken. Im Anschluss muss im entsprechenden Feld der Pfad zum Pythoninterpreter eingegeben werden.

Nun noch alle Fenster mit „OK“ bestätigen und den PC neu starten.

12.2 Pip Paketmanager installieren

Um Pip zu installieren, muss die Eingabeaufforderung mit Administratorrechten geöffnet werden. Danach muss folgender Befehl eingegeben werden:

```
# curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

Anschließend muss man in den Downloadordner navigieren und diesen Befehl eingeben:

```
# python get-pip.py
```

Um zu überprüfen, ob die Installation erfolgreich war, öffnet man nach einem Neustart wieder die Eingabeaufforderung und führt eine Kreuzprüfung durch. Zuerst wird über Python die Python Version abgefragt und im Anschluss noch einmal über Pip. Wenn bei beiden Befehlen der die gleiche Version ausgegeben wird, war die Installation erfolgreich

```
# python –version

# pip --version
```

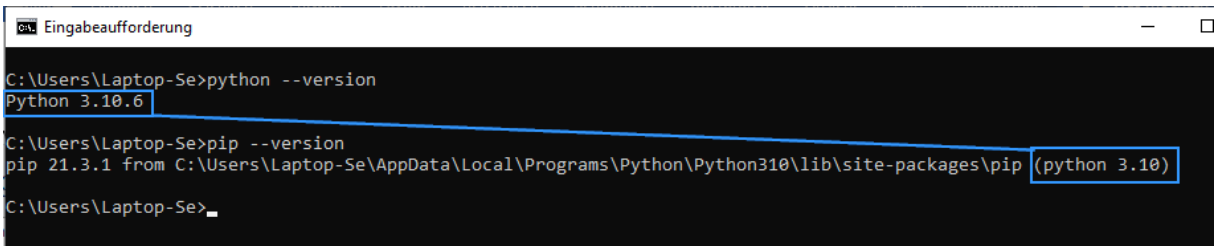

Abbildung 16 - Pip Version

13 Gesprächsnotizen

Telefonat mit dem DWD am 24.08.2022 um 9:00Uhr.

Frage: Wie kann ich direkt die Daten über den FTP OpenData Server abfragen?

Antwort: FileZilla oder ähnliche Programme funktionieren nicht
Am einfachsten ist es die Daten in Python mittels des Packages "ftplib.FTP" herunterzuladen

Frage: Wie kann ich den Forecast über das "deutschland.dwd"-Package in Python Wetterdaten abrufen?

Antwort: Den Zugriff direkt über die API [29] ist nur mittels Kostenpflichtigen Zugang möglich. Die Kosten für die Einrichtung eines API-Tokens würden mir in Rechnung gestellt werden.

[29] 14 Glossar, API

Tipp: Alternativ stellt der DWD den Forecast von allen Stationen mit der Kennungs-ID "SY" kostenlos per URL Request zur Verfügung. Viele Online Wetterdienste nutzen auch nur diese. Der Rückgabewert enthält ein Bytecode, den man zu einem String und danach mittels des Packages "JSON" zu einem Dictionary konvertieren kann.

14 Glossar

Data Mining[30]

Beschreibt einen Prozess zur systematischen Gewinnung, Speicherung und Auswertung von großen Datenmengen.

Data Lake[31]

Data Lake ist die Bezeichnung für einen Sammlung an Rohdaten. Diese müssen erst einmal keine besondere Formatierung besitzen. Es geht eher darum eine große Menge an Daten zu erhalten. Später werden dann in der Datenanalyse die Rohdaten gefiltert, gruppiert und ausgewertet.

IDE - Integrated Development Environment

Eine integrierte Entwicklungsumgebung ist eine Software, die viele Tools für die Softwareentwicklung enthält. Die Funktionen reichen von einem einfachen Texteditor mit Syntaxhighlighting, einem Compiler oder Interpreter über einen Linker bis hin zu Debuggingmodulen.

Debugging

Debugging beschreibt den Prozess der Fehlersuche.

API – Application Programming Interface

Eine API, im Deutschen eine Programmierschnittstelle, bezeichnet eine Anbindung von einer Software zu einer anderen. In diesem Projekt stellt das Programm „sqlInterface.py“ eine Schnittstelle von Python zur SQL-Datenbank dar.

Bytecode

Bytecode ist ein Zwischencode, der von Programmen erstellt wird, um Hardware übergreifend zu funktionieren. Ein Beispiel wären Java und Python. Programme dieser Programmiersprachen sind nicht direkt Maschinencode. Um vom Prozessor verarbeitet werden zu können, benötigen sie einen Interpreter.

JSON – JavaScript Object Notation

Ist ein programmiersprachenunabhängiges Format, um Texte zu speichern oder zu übermitteln.

Query-String

Ist ein Teil einer URL, um dem Host mitzuteilen, was für Daten angefragt werden.

Mehrdimensionale Arrays und Vektorisieren

Mehrdimensionale Arrays sind Matrizen, also Arrays innerhalb von Arrays. Es ist ähnlich wie ein Koordinatensystem. Das erste Array bildet die X- Achse, das zweite Array die Y- Achse. Beim Vektorisieren wird jede Zahl des Arrays zum Beispiel mit drei multipliziert. Zur Veranschaulichung ist das nochmal in der folgenden Programmausgabe dargestellt.

---

[30] (Ronsdorf, 2020)
[31] (Amazon Web Services, 2022)

```
Liste mit drei Listen, Die Trennung durch Kommata:
 listeMitListen =  [[10, 20, 30], [10, 20, 30], [10, 20, 30]]

Array mit zwei Dimensionen. Jedes enthaltene Array
enthält ein weiteres Array.
npArray = np.array(listeMitListen)
 [[10 20 30]
 [10 20 30]
 [10 20 30]]

Vektorisieren:
npArray[0] *= 2 ergibt [20 40 60]
npArray[1] *= 3 ergibt [30 60 90]
npArray[2] *= 4 ergibt [ 40  80 120]

Zugriff auf die zweite Zahl im zweiten Array:
1. Name des Arrays          = npArray
2. Index des ersten Arrays  = [0]
3. Index des zweiten Arrays = [2]
npArray[0][2] = 60
```

Formel 8 - Arrays und Vektorisieren

15 Codeverzeichnis

Formel 1 - GeoPy Geolocator 11
Formel 2 - Erstellen von SQL- Datenbank, -Benutzer und Rechtevergabe 12
Formel 3 - pd.merge 15
Formel 4 - Plotting 15
Formel 5 - Stationslisten Datentypen 17
Formel 6 - Datentypzuweisung in Pandas 17
Formel 7 - Lösungsansatz für public Variablen 18
Formel 8 - Arrays und Vektorisieren 34

16 Tabellenverzeichnis

Tabelle 1 - Preiskalkulation 5
Tabelle 2 - Durchführungszeitraum 5
Tabelle 3 - Requirements 6
Tabelle 4 - Legende Messwert 8
Tabelle 5 - Datenbank Tabellen 13
Tabelle 6 – Projektstruktur 17
Tabelle 7 - Projektziele Soll-Ist-Vergleich 19
Tabelle 8 - Vergleich Java und Python 20

17 Abbildungsverzeichnis

Abbildung 1 - Zeitplanung 4
Abbildung 2 - Rohdatenformat der historischen Messungen 7
Abbildung 3 - Ergebnis des URL Request 9
Abbildung 4- Stationslexikon 10
Abbildung 5 - Legende Stationslexikon 10
Abbildung 6 - Entity Relationship Diagram, aktuell 12
Abbildung 7 - Entity Relationship Diagram, verbessert 13
Abbildung 8 - SQL- Join 14
Abbildung 9 - Temperaturkurve 16
Abbildung 12 - hartcodierte Strings 18
Abbildung 13 - Zeitplan Soll-Ist-Vergleich 19
Abbildung 14 - Funktionsaufruf und Parameterübergabe 23
Abbildung 15 - Systemeigenschaften 30
Abbildung 16 - Umgebungsvariablen 31
Abbildung 17 - Umgebungsvariablen bearbeiten 31
Abbildung 18 - Pip Version 32

18 Verweise

Amazon Web Services. (06. 09 2022). *www.aws.amazon.com/de*. Von https://aws.amazon.com/de/big-data/datalakes-and-analytics/what-is-a-data-lake/ abgerufen

Deutscher Wetterdienst. (2022). *https://opendata.dwd.de*. Abgerufen am 07. 07 2022 von https://opendata.dwd.de/climate_environment/CDC/

Deutscher Wetterdienst. (2022). *https://www.dwd.de/DE/*. Abgerufen am 04. 07 2022 von https://www.dwd.de/DE/leistungen/klimadatendeutschland/stationsliste.html

dwd.api.bund. (2022). *api.bund.dev/*. Abgerufen am 22. 08 2022 von https://dwd.api.bund.dev/

Genau, L. (06. 09 2022). *www.scribbr.de*. Von https://www.scribbr.de/facharbeit/selbststaendigkeitserklaerung/ abgerufen

Github DeutscherWetterdienst. (07. 09 2022). *https://github.com/bundesAPI/dwd-api*. Von https://github.com/bundesAPI/dwd-api: https://github.com/bundesAPI/dwd-api abgerufen

GULP, R. (06. 09 2022). *www.gulp.de*. Von https://www.gulp.de/knowledge-base/stundensaetze/und-er-steigt-weiter-stundensatz-der-it--engineering-freiberufler-erreicht-neuen-hoechstwert.html abgerufen

Koffler, M. (2019). Python Der Grundkurs. In M. Koffler, *Python Der Grundkurs* (S. 464). Bonn: Rheinwerk Computing.

Kriesel, D. (29. 12 2016). *SpiegelMining – Reverse Engineering von Spiegel-Online (33c3)*. Von https://www.youtube.com/c/mediacccde: https://www.youtube.com/watch?v=-YpwsdRKt8Q abgerufen

Landmann, J. (25. 06 2018). DoIT Projektantrag. Heidelberg, Baden- Württemberg.

luftqualitaet.api.bund.dev. (2022). *luftqualitaet.api.bund.dev/*. Abgerufen am 07. 09 2022 von https://luftqualitaet.api.bund.dev/

Microsoft. (08. 08 2022). *www.docs.microsoft.com*. Abgerufen am 07. 09 2022 von https://docs.microsoft.com/de-de/cpp/c-language/multidimensional-arrays-c?view=msvc-170

Morpheus Tutorials. (10. 04 2022). *https://www.youtube.com/c/TheMorpheus407*. Abgerufen am 31. 08 2022 von https://www.youtube.com/watch?v=JX0cHWUBkUQ&t

Ronsdorf, J. (13. 05 2020). *www.news.microsoft.com*. Von https://news.microsoft.com/de-de/microsoft-erklaert-was-ist-data-mining-definition-funktionen/ abgerufen

suche.postleizahl.org. (20. 07 2022). *https://www.suche-postleitzahl.org*. Von https://www.suche-postleitzahl.org/plz-karte-erstellen abgerufen

w3school.com. (07. 09 2022). *https://www.w3schools.com*. Von https://www.w3schools.com/sql/sql_join.asp abgerufen

xampp. (kein Datum). www.apachefriends.org. *https://www.apachefriends.org/download.html*. Von https://www.apachefriends.org/download.html abgerufen