

Daniel Lubig  
E-mail = [Daniel.lubig@tu-dresden.de](mailto:Daniel.lubig@tu-dresden.de)

# Airport Operations - Anwendung zur Datenanalyse

Dresden, 19.05.2021

# Airport Operations - Anwendung zur Datenanalyse

Installation von Software zur Nutzung von Python

**Fragen/Probleme bei der Installation?**

# Airport Operations - Anwendung zur Datenanalyse

## Python vs. Excel

### Python

Free-to-use + Standardsoftware zur Datenverarbeitung

Keine Software bedingten Limitierungen zur verwendeten Dateigröße

Schnelle Handhabung großer Datenmenge

Volle Kontrolle über alle Bildelemente

Kein direkter Zugriff auf Zellen

Aufwendige Veranschaulichung von Ergebnissen

### Excel

Schnelle Veranschaulichung der Ergebnisse

Direkter Zugriff auf Zellen

Kostenpflichtig (für nicht Studenten)

Begrenzt auf ~1,05 Millionen Datenzeilen

Lange Rechendauer für komplexe Berechnungen

Aufwendige Anpassung von Diagrammen

# Airport Operations - Anwendung zur Datenanalyse

## csv-Datensatz

- csv = comma-seperated values
- kapazitätssparende Methode zur Speicherung größerer Datenmengen

Datensatz zur Übung → <https://github.com/DanielLubigTUD/AirportOperations2021>

### Schritt 1: Umwandeln der csv-file in ein bearbeitungsfähiges Dateiformat

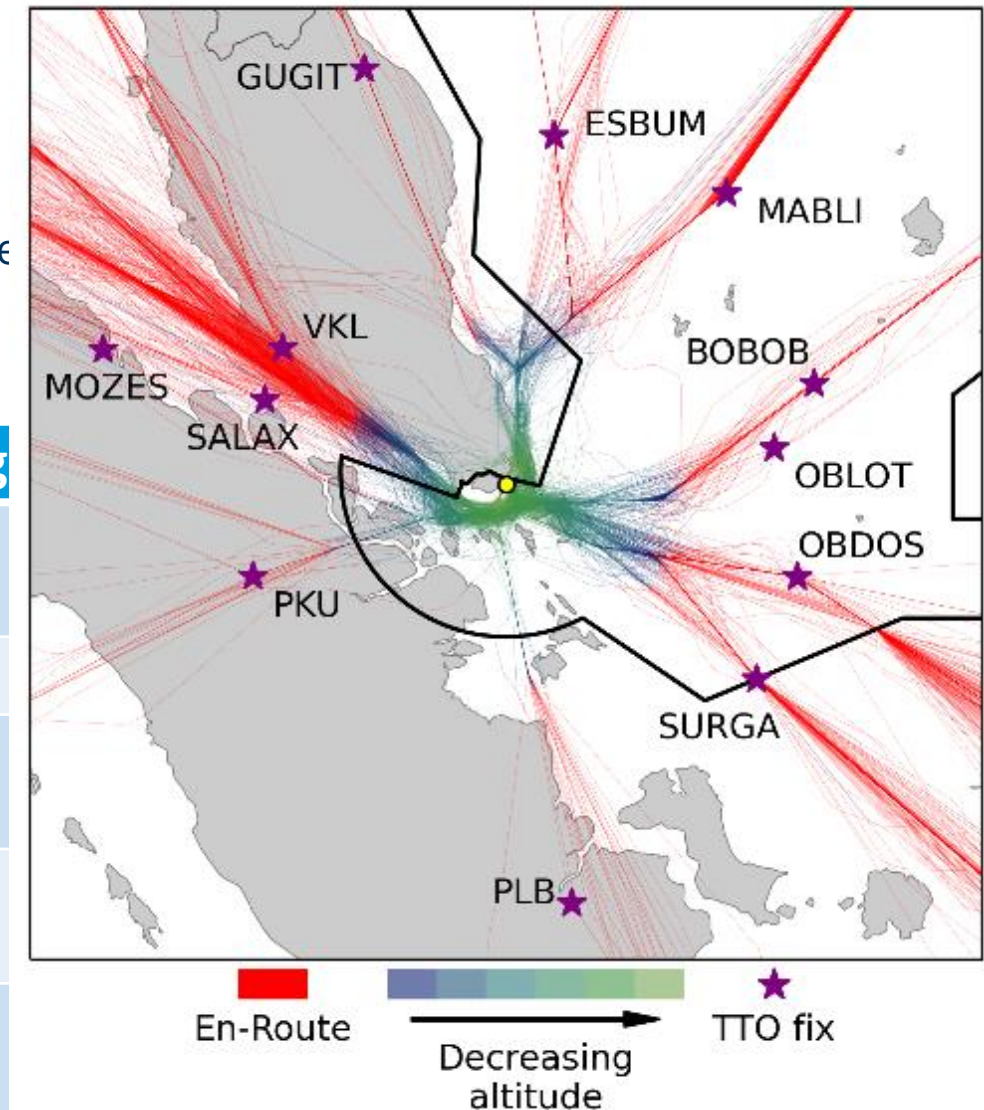
Excel → „Text in Spalten“

Python → Initiieren eines JupyterLab notebooks und Installation notwendiger python packages/libraries

# Airport Operations - Anwendung zur Python package/librarys

- Sind nicht automatisch installiert → Installation/Importieren
- Speicherplatz niedrig halten

Name	Library or Package
Pandas	Library
Matplotlib	Library
NumPy (Numerical Python)	Library
SciPy (Scientific Python)	Library
SimPy (Simulation Python)	Library
Cartopy	Package



Erstellen von Karten

# Airport Operations - Anwendung zur Datenanalyse

## Installation/Import von packages/librarys

### Importierungsbefehl

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

### Installationsbefehl (nicht Teil dieser Übung)

#### COLAB

```
!apt-get -qq install python-cartopy python3-cartopy  
!pip install Cartopy
```

#### JupyterLab

```
!pip install Cartopy
```

- Vorher weitere Schritte notwendig

# Airport Operations - Anwendung zur Datenanalyse

## Einlesen der csv-file als pandas DataFrame

### Einlesen der Dateien als DataFrame mithilfe von pandas:

Für csv-Dateien:

```
dataset_EDDM = pd.read_csv('Path on PC' or 'web link')
```

Für Excel-Dateien → `pd.read_excel('Path on PC' or 'web link')`

Für pickle-Dateien → `pd.read_pickle('Path on PC' or 'web link')`

### Informationen zum DataFrame:

```
dataset_EDDM.info()
```

### Ermittlung der Datentypen im Datenset:

```
dataset_EDDM.dtypes
```

- Übersicht zu Datentypen der einzelnen Spalten

# Airport Operations - Anwendung zur Datenanalyse

## wichtige Datentypen

Typ	Verwendung	Beispiel	Besonderheiten
String	Textbausteine	`Hallo`, `1`	Durch Anführungsstriche definiert
Integer	Ganze Zahlen	1	
Float	Kommazahlen	1.34	
Datetime/ Timestamp	Zeitpunkte	01.04.2019 12:34:23	können durch unterschiedliches Format definiert sein
Timedelta	Zeitdifferenzen	0 days 01:20:21	
Object	Kombination mehrerer Datentypen	23cdf563	Nicht für Berechnungen geeignet Ähnlichkeit mit Strings

- Python wandelt Datenformate automatisch um!

Bsp: float – integer = float **oder** timestamp – timestamp = timedelta



# Airport Operations - Anwendung zur Datenanalyse

## Umwandeln von Datentypen

### String

```
df['col'].astype(str)
```

df = Name des DataFrame

col = Name der Spalte

### Float

```
df['col'].astype(float)
```

### Integer

```
df['col'].astype(int)
```

### Timestamp

```
pd.to_datetime(df['col'])
```

### Timedelta

```
pd.to_timedelta(df['col'])
```

# Airport Operations - Anwendung zur Datenanalyse

## Speichern des Datensatzes

### Als Excel-Datei speichern

```
df.to_excel('Path on PC.xlsx')
```

### Als csv-Datei speichern

```
df.to_csv('Path on PC.csv')
```

### Als Pickle-Datei (pkl) speichern

```
df.to_pickle('Path on PC.pkl')
```

- Schnelles einlesen der Daten
  - Speicherplatzsparend
  - Datentypen bleiben erhalten
- Nicht unabhängig von Programmiersprachen!

# Airport Operations - Anwendung zur Datenanalyse

## neue beste Freunde

### Stackoverflow

- Frage-Antwort Plattform für Programmierer

Link: <https://stackoverflow.com/>

### Github

- Kollaborative Entwicklungsplattform für Software

Link: <https://github.com/>

### Python interne Hilfsfunktion

- Informationen zu Funktionen und deren Bestandteilen

JupyterLab: Funktionsname + ? → `plt.savefig?`

COLAB: ? + Funktionsname → `?plt.savefig`

### Dokumentationen von librarys/packages

- Informationen zu Funktionen und deren Bestandteilen

Link: Google it!

# Airport Operations - Anwendung zur Datenanalyse

## Aufgabe 1 - Movements pro Tag im Datenzeitraum

Schritt 1: Zeitraum der Daten bestimmen (Start- und Enddatum)

Schritt 2: Alle Tage im Datenzeitraum als Liste ablegen

Schritt 3: Schleife zur Bestimmung der Bewegungen für jeden Tag initiieren

Schritt 3.1: Bestimmen der Grenzen eines Tages

Schritt 3.2: Filtern des Datensatzes nach Arrivals and Departures

Schritt 3.3: Alle Flüge innerhalb der Grenzen von Schritt 3.1 extrahieren

Schritt 3.4: Bestimmen der Anzahl der Starts und Landungen

Schritt 3.5: Speichern des Wertes in einem neuem DataFrame

Schritt 4: Bestimmen der Movements aus der Anzahl der Starts und Landungen

Schritt 5: Balkendiagramm zur Visualisierung erstellen

# Airport Operations - Anwendung zur Datenanalyse

## Aufgabe 2 – Verkehrsreichste Stunde ermitteln

IATA : „IATA defines the Peak Hour as the Busiest Hour in the second busiest day during the average week of the peak month; other authorities use the 30<sup>th</sup> busiest hour, etc. “

**2.A: 2nd busiest day → busiest hour**

**HAUSAUFGABE**

2.B: 30th busiest hour in the dataset

# Airport Operations - Anwendung zur Datenanalyse

## Aufgabe 2A – 30. verkehrsreichste Stunde ermitteln

Schritt 1: Zeitraum der Daten bestimmen (Start- und Enddatum)

Schritt 2: Alle Stunden im Datenzeitraum als Liste ablegen

Schritt 3: Schleife zur Bestimmung der Bewegungen für jede Stunde initiieren

Schritt 3.1: Bestimmen der Grenzen einer Stunde

Schritt 3.2: Filtern des Datensatzes nach Arrivals and Departures

Schritt 3.3: Alle Flüge innerhalb der Grenzen von Schritt 3.1 extrahieren

Schritt 3.4: Bestimmen der Anzahl der Starts und Landungen

Schritt 3.5: Speichern des Wertes in einem neuem DataFrame

Schritt 4: Bestimmen der Movements aus der Anzahl der Starts und Landungen

Schritt 5: Sortieren des Dataframes nach Movements (absteigend)

Schritt 6: Ermitteln der 30. verkehrsreichsten Stunde

# Airport Operations - Anwendung zur Datenanalyse

## Fehlersuche

- Python script läuft so lange bis ein „Error“ auftritt → Python Fehlermeldung taucht auf!
- Teilweise schwer verständliche Fehlermeldungen

### **Tipp: Festlegen einer Testvariable oder Nutzung von Textblöcken zur Eingrenzung der fehlerhaften Codezeile**

`print('checkpoint1')` → Python gibt den Text aus, wenn bis zur Codezeile kein Fehler auftritt

`checkpoint = 1` → Python weist der Variable ‚checkpoint‘ den Wert 1 zu, wenn bis zur Stelle der Codezeile kein Fehler auftritt

→ die Variable kann nach Durchlauf des scripts ausgegeben werden

Daniel Lubig  
E-mail = [Daniel.lubig@tu-dresden.de](mailto:Daniel.lubig@tu-dresden.de)

# Danke für Ihre Teilnahme!

Dresden, 19.05.2021