

Arbeitspaket (AP) 1: Advanced Data Collection Methods

Persönliche Angaben (bitte ergänzen)

Vorname:

Nachname:

Immatrikulationsnummer:

Modul: Data Science

Prüfungsdatum / Raum /
Zeit: 11.11.2024 / Raum: SF O3.54 / 8:00 – 11:45

Erlaubte Hilfsmittel:
w.MA.XX.DS.24HS (Data Science)
Open Book, Eigener Computer, Internet-Zugang

Nicht erlaubt ist der Einsatz beliebiger Formen von generativer
KI (z.B. Copilot, ChatGPT)
sowie beliebige Formen von Kommunikation oder
Kollaboration mit anderen Menschen.

Bewertungskriterien

(max. erreichbare Punkte: 48)

Kategorie	Beschreibung	Punkteverteilung
Code nicht lauffähig oder Ergebnisse nicht sinnvoll	Der Code enthält Fehler, die verhindern, dass er ausgeführt werden kann (z.B. Syntaxfehler) oder es werden Ergebnisse ausgegeben, welche nicht zur Fragestellung passen.	0 Punkte
Code lauffähig, aber mit gravierenden Mängeln	Der Code läuft, aber die Ergebnisse sind aufgrund wesentlicher Fehler unvollständig (z.B. grundlegende Fehler beim Einlesen der Daten). Nur geringer Fortschritt erkennbar.	25% der max. erreichbaren Punkte
Code lauffähig, aber mit mittleren Mängeln	Der Code läuft und liefert teilweise korrekte Ergebnisse, aber es gibt grössere Fehler (z.B. die Datentypen der eingelesenen Daten entsprechen nicht den Anforderungen gemäss Fragestellung). Die Ergebnisse sind nachvollziehbar, aber unvollständig oder ungenau.	50% der max. erreichbaren Punkte
Code lauffähig, aber mit minimalen Mängeln	Der Code läuft und liefert ein weitgehend korrektes Ergebnis, aber kleinere Fehler (z.B. Spaltenname falsch geschrieben, TimeStamp nicht korrekt formatiert) beeinträchtigen die Vollständigkeit des Ergebnisses.	75% der max. erreichbaren Punkte

Kategorie	Beschreibung	Punkteverteilung
Code lauffähig und korrekt	Der Code läuft einwandfrei und liefert das korrekte Ergebnis ohne Mängel.	100% der max. erreichbaren Punkte

Python Libraries und Settings

```
In [1]: # Libraries
import os
import re
import json
import requests
import feedparser
import pandas as pd
import yfinance as yf
from zipfile import ZipFile
from bs4 import BeautifulSoup
from datetime import datetime
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from pandas import json_normalize
from lxml import html as lxml_html

from pycoingecko import CoinGeckoAPI
cg = CoinGeckoAPI()

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')

# Show current working directory
print(os.getcwd())
```

u:\Lektionen\DS_HS2024\MEP_HS2024\Block_02\Loesungen_LN_II\AP01

Vorbereitung (Dieser Teil wird nicht bewertet!)

1.) Starten Sie eine GitHub Codespaces Instanz auf Basis Ihres Forks des folgenden GitHub Repositories:

GitHub-Repository: https://github.com/mario-gellrich-zhaw/data_ingestion

Aufgaben (Dieser Teil wird bewertet!)

Hinweise zu den folgenden Aufgabenstellungen:

- Beachten Sie auch die zu jeder Aufgabenstellung zugehörigen Details zu Aufgabenstellung.
- Lösen Sie jede Aufgabe mit Hilfe von Python Code. Integrieren Sie den Python Code in die Code-Zellen der jeweiligen Aufgabe.
- Stellen Sie die Lösung(en) jeder Aufgabe ganz oder in Teilen (z.B. erste 5 Zeilen) eines DataFrames im Jupyter Notebook dar.

- Wenn in den Aufgaben in Bezug auf das Datum vom 'aktuellen Tag' gesprochen wird, ist der Tag der Durchführung dieses Leistungsnachweises gemeint.

Aufgabe (1): Importieren Sie jedes der über den folgenden Codezellen angegebenen Files in ein eigenes pandas DataFrame.

Details zur Aufgabenstellung:

- Sie finden die Files im Order 'Data'.
- Berücksichtigen Sie die Ablage der Files beim Einlesen (z.B. Pfad zum Einlesen von 'BigmacPrice.csv' is: './Data/BigmacPrice.csv').
- Zeigen Sie mit Hilfe eines geeigneten Python Befehls die Dimension des pandas DataFrames (Anzahl Zeilen und Spalten) im Jupyter Notebook an.
- Stellen sie die erstellten DataFrames entweder ganz oder in Teilen (z.B. die ersten 5 Zeilen) im Jupyter Notebook dar.

(max. erreichbare Punkte: 4)

Importieren von 'BigmacPrice.csv'

```
In [2]: # Import File 'BigmacPrice.csv' in eine Pandas DataFrame
bigmac = pd.read_csv('./Data/BigmacPrice.csv',
                     sep=',',
                     encoding='utf-8')

# Anzeigen der Anzahl Zeilen und Spalten im DataFrame
print(bigmac.shape)

# Anzeigen des DataFrames
bigmac.head()
```

(1946, 6)

```
Out[2]:
```

	date	currency_code	name	local_price	dollar_ex	dollar_price
0	2000-04-01	ARS	Argentina	2.50	1	2.50
1	2000-04-01	AUD	Australia	2.59	1	2.59
2	2000-04-01	BRL	Brazil	2.95	1	2.95
3	2000-04-01	GBP	Britain	1.90	1	1.90
4	2000-04-01	CAD	Canada	2.85	1	2.85

Importieren von 'CarData.xml'

```
In [3]: # Import File 'CarData.xml'
cars = pd.read_xml('./Data/CarData.xml',
                   encoding='utf-8')

# Anzeigen der Anzahl Zeilen und Spalten im DataFrame
print(cars.shape)

# Anzeigen des DataFrames
cars.head()
```

(1000, 10)

Out[3]:

	car_id	make	type	year	hp	mileage	price	doors	color
0	1	Chevrolet	Sedan	1991	158	25040	17145.80	3	Orange
1	2	BMW	Wagon	1991	465	137449	29727.51	5	Red
2	3	Kia	Van	2001	189	202308	73312.00	4	Red
3	4	Toyota	SUV	2011	117	123661	22342.98	5	Black
4	5	Nissan	SUV	2004	339	256696	53094.74	5	Blue



Importieren von 'ChickenData.xlsx'

In [4]:

```
# Import File 'ChickenData.xlsx'
chickens = pd.read_excel('./Data/ChickenData.xlsx',
                         sheet_name='Chickens')

# Anzeigen der Anzahl Zeilen und Spalten im DataFrame
print(chickens.shape)

# Anzeigen des DataFrames
chickens.head()
```

(100, 4)

Out[4]:

	chicken_id	weight	breed	eggs_per_year
0	1	2435.836917	New Hampshire Red	158
1	2	2157.180545	Australorp	273
2	3	2558.418035	New Hampshire Red	153
3	4	2679.200608	Plymouth Rock	268
4	5	2556.855145	Australorp	360

Importieren von 'HotelData.json'

In [5]:

```
# Import File 'HotelData.json'
hotels = pd.read_json('./Data/HotelData.json')

# Anzeigen der Anzahl Zeilen und Spalten im DataFrame
print(hotels.shape)

# Anzeigen des DataFrames
hotels.head()
```

(456, 5)

Out[5]:

	type	generator	copyright	timestamp	features
0	FeatureCollection	overpass-turbo	The data included in this document is from www...	2024-10-27 16:52:02+00:00	{'type': 'Feature', 'properties': {'@id': 'rel...}}
1	FeatureCollection	overpass-turbo	The data included in this document is from www...	2024-10-27 16:52:02+00:00	{'type': 'Feature', 'properties': {'@id': 'rel...}}
2	FeatureCollection	overpass-turbo	The data included in this document is from www...	2024-10-27 16:52:02+00:00	{'type': 'Feature', 'properties': {'@id': 'rel...}}
3	FeatureCollection	overpass-turbo	The data included in this document is from www...	2024-10-27 16:52:02+00:00	{'type': 'Feature', 'properties': {'@id': 'way...}}
4	FeatureCollection	overpass-turbo	The data included in this document is from www...	2024-10-27 16:52:02+00:00	{'type': 'Feature', 'properties': {'@id': 'way...}}

Aufgabe (2): Extrahieren Sie aus dem File 'BookStore.html' die Preise der Bücher mit Hilfe eines geeigneten XPath.

Details zur Aufgabenstellung:

- Sie finden das HTML-File 'BookStore.html' im Order 'Data'.
- Komplettieren Sie für das Lösen der Aufgabe den unfertigen XPath im folgenden Python Code.
- Beachten Sie, dass nur ein einziger XPath für das Extrahieren aller Preise notwendig ist.
- Stellen Sie die extrahierten Preise im Jupyter Notebook dar.
- Tipp: Für das Testen des XPath können Sie diesen Online XPath-Tester verwenden:
<http://xpathe.com>

(max. erreichbare Punkte: 4)

In [6]:

```
# Import File 'BookStore.html'
with open('./Data/BookStore.html', 'r', encoding='utf-8') as file:
    html_content = file.read()

# Parsen des HTML-Contents
tree = lxml_html.fromstring(html_content)

# XPath zum Extrahieren der Buchpreise (bitte komplettieren)
try:
    prices = tree.xpath('/html/body/div/section/div/div/p[2]/text()')
except:
    prices = [] # Ausgabe einer leeren Liste, falls der XPath nicht korrekt ist

# Anzeigen der extrahierten Buchpreise
print("Extracted Book Prices:", prices)
```

Extracted Book Prices: ['\$10.99', '\$9.99', '\$12.99']

Aufgabe (3): Erstellen Sie ein pandas DataFrame mit den Daten aus der folgenden Tabelle (1), und speichern Sie dieses als XML-File ab.

Details zur Aufgabenstellung:

- Speichern Sie das erstellte XML-File als 'SmartphoneData.xml' im Ordner 'Data'.
- Stellen Sie das erstellte DataFrame im Jupyter Notebook dar.

Tipp: Beim Erstellen eines Dataframes wird typischerweise zuerst ein Python dictionary erstellt und dieses anschliessend in ein DataFrame umgewandelt.

Tabelle (1): Smartphone Data

Marke	Modell	Baujahr	Farbe	Preis
Apple	iPhone 12	2020	Schwarz	799
Samsung	Galaxy S21	2021	Weiß	999
Google	Pixel 5	2020	Grün	699
OnePlus	8 Pro	2020	Blau	899

(max. erreichbare Punkte: 4)

```
In [7]: # Erstellen des DataFrames aus der gegebenen Tabelle
data = {
    'Marke': ['Apple', 'Samsung', 'Google', 'OnePlus'],
    'Modell': ['iPhone 12', 'Galaxy S21', 'Pixel 5', '8 Pro'],
    'Baujahr': [2020, 2021, 2020, 2020],
    'Farbe': ['Schwarz', 'Weiß', 'Grün', 'Blau'],
    'Preis': [799, 999, 699, 899]
}

df = pd.DataFrame(data)

# Abspeichern des DataFrames als XML-Datei
df.to_xml('./Data/SmartphoneData.xml', index=False)

# Anzeigen des DataFrames
df
```

	Marke	Modell	Baujahr	Farbe	Preis
0	Apple	iPhone 12	2020	Schwarz	799
1	Samsung	Galaxy S21	2021	Weiß	999
2	Google	Pixel 5	2020	Grün	699
3	OnePlus	8 Pro	2020	Blau	899

Aufgabe (4): Erstellen Sie ein ZIP-File aus den folgenden genannten Files, und speichern Sie dieses im Order 'Data' ab.

Details zur Aufgabenstellung:

- Verwenden Sie die folgenden Files aus dem Ordner 'Data' als Input für das ZIP-File:
 - BigmacPrice.csv
 - BookStore.html
 - CarData.xml
 - ChickenData.xlsx
 - HotelData.json
- Speichern Sie das ZIP-File als 'MyData.zip' im Ordner 'Data' ab.

(max. erreichbare Punkte: 4)

```
In [8]: # Erstellen eines leeren Zip-Archivs
zipObj = ZipFile('./Data/MyData.zip', 'w')

# Hinzufügen der Dateien zum Zip-Archiv
zipObj.write('./Data/BigmacPrice.csv')
zipObj.write('./Data/BookStore.html')
zipObj.write('./Data/ChickenData.xlsx')
zipObj.write('./Data/HotelData.json')

# Schliessen des Zip-Archivs
zipObj.close()
```

Aufgabe (5): Beziehen Sie Daten zu Cryptowährungen mit Hilfe der Coingecko Web API.

Details zur Aufgabenstellung:

- Verwenden Sie die folgenden Bezeichnungen (Ids) der Cryptowährungen für die Server-Anfragen: **binancecoin, cardano, polkadot, solana**.
- Fragen Sie folgende Daten ab (Währung = EUR):
 - Marktkapitalisierung (include_market_cap='true')
 - 24 Stunden Volumen (include_24hr_vol='true')
 - 24 Stunden Veränderung (include_24hr_change='true')
 - Datum des letzten Updates (include_last_updated_at='true')
- Wandeln Sie die Daten in ein DataFrame um und stellen Sie dieses im Jupyter Notebook dar.
- Tipp: data = cg.get_price() speichert die Daten in ein Python dictionary. Dieses kann mit df = pd.DataFrame(data) in ein DataFrame umgewandelt werden.

(max. erreichbare Punkte: 8)

```
In [9]: # Beziehen der Daten mit Hilfe der Coingecko Web API
data = cg.get_price( ids='binancecoin, cardano, polkadot, solana',
                     vs_currencies=['eur'],
                     include_market_cap='true',
                     include_24hr_vol='true',
                     include_24hr_change='true',
                     include_last_updated_at='true' )
```

```
# Speichern der Daten in einem pandas DataFrame
df = pd.DataFrame(data)

# Anzeigen des DataFrames
df
```

Out[9]:

		binancecoin	cardano	polkadot	solana
	eur	5.984500e+02	6.787110e-01	5.270000e+00	2.261300e+02
	eur_market_cap	8.730478e+10	2.428624e+10	7.580904e+09	1.067309e+11
	eur_24h_vol	2.003432e+09	3.767955e+09	1.032446e+09	1.158916e+10
	eur_24h_change	8.714643e-01	-4.647025e+00	2.661393e+00	9.809026e+00
	last_updated_at	1.731843e+09	1.731843e+09	1.731843e+09	1.731843e+09

Aufgabe (6): Beziehen Sie Börsendaten zur Tesla-Aktie mit Hilfe der Yahoo-Finance Web API.

Details zur Aufgabenstellung:

- Die Tesla Aktie hat das Kürzel TSLA.
- Definieren Sie '2023-01-01' als Startdatum für die Abfrage der Börsendaten.
- Verwenden Sie die Python Library 'datetime' für die Erstellung des Enddatums (aktueller Tag).
- Speichern Sie die Börsendaten in einem pandas DataFrame ab.
- Stellen Sie die Daten der fünf aktuellsten Börsentage im Jupyter Notebook dar.
- Hinweis: Die neusten Daten aus der Yahoo-Finance Web API sind jeweils vom Vortag.

(max. erreichbare Punkte: 8)

```
In [10]: # Erstellen eines Datumsobjekts mit dem aktuellen Datum und der aktuellen Uhrzeit
today = datetime.now().strftime("%Y-%m-%d")
print('Today is:', today)

# Beziehen der Tesla Aktienkurse von Yahoo Finance
data = yf.download('TSLA', start = '2023-01-01', end = today, progress=False)

# Anzeigen der Anzahl Zeilen und Spalten im DataFrame
print('Dimension:', data.shape)

# Anzeigen des DataFrames mit den Daten der fünf aktuellsten Börsentage
data.tail()
```

Today is: 2024-11-17
Dimension: (472, 6)

Out[10]:

	Price	Adj Close	Close	High	Low	Open	Volume
	Ticker	TSLA	TSLA	TSLA	TSLA	TSLA	TSLA
	Date						
2024-11-11 00:00:00+00:00		350.000000	350.000000	358.640015	336.000000	346.299988	210521600
2024-11-12 00:00:00+00:00		328.489990	328.489990	345.839996	323.309998	342.739990	155726000
2024-11-13 00:00:00+00:00		330.239990	330.239990	344.600006	322.500000	335.850006	125405600
2024-11-14 00:00:00+00:00		311.179993	311.179993	329.980011	310.369995	327.690002	120726100
2024-11-15 00:00:00+00:00		320.720001	320.720001	324.679993	309.220001	310.570007	114128900



Aufgabe (7): Beziehen Sie mit Hilfe der Overpass Turbo Web API Daten zu den Tankstellen in der Schweiz.

Details zur Aufgabenstellung:

- Tankstellen sind mit dem key:value Paar "amenity"="fuel" in der OpenStreetMap Map-Feature Übersicht angegeben.
- Speichern Sie die Daten in einem File 'petrol_station.json' im Order 'Data' ab.
- Lesen Sie die Daten zusätzlich in ein pandas DataFrame und zeigen Sie dieses ganz oder in Teilen (z.B. erste 5 Zeilen) an.
- Tipp: Auf der Webseite: <https://overpass-turbo.eu> können Sie Overpass Turbo Abfragen testen.

(max. erreichbare Punkte: 8)

In [11]:

```
# Overpass API URL
url = 'http://overpass-api.de/api/interpreter'

# Overpass turbo Abfrage-String
query = f"""
    [out:json];
    area["ISO3166-1"="CH"][admin_level=2];
    node ["amenity"="fuel"] (area);
    out;"""

# Web API Anfrage
r = requests.get(url, params={'data': query})
data = r.json()['elements']

# Speichern der Daten in einer JSON-Datei
with open('./Data/petrol_station.json', 'w') as json_file:
    json.dump(data, json_file)

# Speichern der Daten in einem Pandas DataFrame und Anzeigen des DataFrames
```

```
df = json_normalize(data)
df.head(5)
```

```
Out[11]:
```

	type	id	lat	lon	tags.amenity	tags.brand	tags.brand:wikidata	t
0	node	734402	46.690599	7.668189		fuel	Eni	Q565594
1	node	1228781	46.878598	7.542326		fuel	BP	NaN
2	node	5702030	46.680546	7.827412		fuel	Migrol	Q1747771
3	node	5702071	46.683415	7.844559		fuel	Avia	Q300147
4	node	8082505	46.683817	7.664427		fuel	NaN	NaN

5 rows × 196 columns

Aufgabe (8): Beziehen und Visualisieren Sie die aktuellen News-Feed (RSS-Feed) Daten der New York Times.

Details zur Aufgabenstellung:

- Verwenden Sie diese URL zum RSS-Feed der New York Times Welt-Nachrichten:
<https://rss.nytimes.com/services/xml/rss/nyt/World.xml>.
- Beziehen Sie die RSS-Feeds und extrahieren sie die Titel der Artikel aus den RSS-Feed Einträgen.
- Zeigen Sie die Titel der RSS-Feed Einträge im Jupyter Notebook an.
- Erstellen Sie eine Graphik mit einer Word-Cloud der Titel der RSS-Feed Einträge und stellen Sie diese im Jupyter Notebook dar.

(max. erreichbare Punkte: 8)

```
In [12]: # Parsen des RSS-News Feed von der New York Times World News Webseite
feed = feedparser.parse("https://rss.nytimes.com/services/xml/rss/nyt/World.xml")

# Extrahieren der RSS-Feed Einträge
feed_entries = feed.entries

# Initialisieren des WordCloud-Texts
wordcloud_text = ""

# Loop für Anzeige des Artikel-Titels
print("Article Titles:")
for entry in feed.entries:

    # Extrahieren des Artikel-Titels
    article_title = entry.title

    # Anhängen des Artikel-Titels an den WordCloud-Text
    wordcloud_text += f"{article_title} "

    # Anzeige des Artikel-Titels
    print(f"{article_title}")

# Generieren der WordCloud
wordcloud = WordCloud(width=700,
```

```
height=400,  
background_color='black').generate(wordcloud_text)  
  
# Anzeigen der WordCloud  
plt.figure(figsize=(8, 6))  
plt.imshow(wordcloud, interpolation='bilinear')  
plt.axis('off') # Hide the axes  
plt.show()
```

Article Titles:

‘No Use for Hatred’: A Village Seeks to Move On From a U.S. Massacre
Russia Launches Major Attack on Power Grid, Ukraine Says
Haiti Has Big Problems and Few Solutions
Israeli Strikes in Central and Northern Gaza Kill More Than 30 People
Biden and Xi Meet, Delivering Messages Seemingly Intended for Trump
At a Ukraine Maternity Clinic, Deaths From an Airstrike Shatter a Community
China Stabbing Kills at Least 8 People
Young Gazans Reach Global Audiences With Videos of Everyday Life in War
Bitter Infighting Divides Russian Opposition
Vladimir Shklyarov, Star Russian Ballet Dancer, Dies at 39
South Africa Police Try Siege Tactics on Illegal Mining, Igniting Debate
As Xi and Biden Meet, Trump and Uncertainty Loom Large
North Korea Blasts Unbearable Sounds To South Korea
Ten Newborns Die in India Hospital Fire
Barnard’s Star Finally Has a Planet, and Possibly More
Why Trump’s Space for Deal-Making in the Middle East Has Shrunk
Man Hiding Tarantulas, Centipedes and Ants Is Stopped From Boarding Flight
Satisfying vs. Productive
Conversations With Murray Sinclair
Israel Strikes Near Beirut as Two Medics Killed in South Lebanon
Democrats’ Message at COP29 Climate Talks: Don’t Panic
Typhoon Man-yi Pounds the Philippines With Rain and High Winds
Biden Discusses With Allies ‘Dangerous’ Cooperation Between Russia and North Korea
Iran Told U.S. That It’s Not Trying to Kill Trump
Trump Will Encounter a Very Different Middle East in His Second Term
Princess Yuriko, Oldest Member of Japan’s Imperial Family, Dies at 101
Taylor Swift to Tour in Canada After Trudeau Issues Plea on Social Media
Putin Talks With German Chancellor, Breaking Ice With the West
A Whole New Ballgame
Gaza War Strains Europe’s Efforts at Social Cohesion
A Tiny Gladiator Tells of the Reach of Roman Empire Celebrity
Nuclear Power Was Once Shunned at Climate Talks. Now, It’s a Rising Star.
Moldova Uncovers Fraudulent Scheme to Get Criminals Off Interpol Blacklist
Israel Pounds Area Near Beirut Amid Signs of a Widening Offensive
Sri Lankan Leader’s Leftist Coalition Wins Elections
Court Hears New York Times Case Against European Commission Over Vaccine Deal
Engineers Discover a 132-Year-Old Message in a Bottle in a Scottish Lighthouse
Lebanese Fear That Wherever the Displaced Go, Israeli Bombs Will Follow
Can These Ex-Hollywood Chimps Find a Place Among the Apes?
Fire in Retirement Home in Spain Leaves at Least 10 Dead
New Zealand Parliament Suspended as Maori Lawmakers Perform Haka to Protest Bill
Why North Korea Is Building Drones
Friday Briefing
In Moscow, Trump’s Victory Is Welcomed, But Warily
Argentina Mulls Exiting Paris Climate Deal
Stress From Fireworks Killed Baby Red Panda, Zoo in Scotland Says
Murray Sinclair, 73, Who Reframed Indigenous Relations in Canada, Dies
Yiannis Boutaris, Vintner, Animal Defender and Greek Mayor, Dies at 82
Elon Musk Met With Iran’s U.N. Ambassador, Iranian Officials Say
Friday Briefing: Dismay Over Trump’s Picks
France-Israel Soccer Match is Overshadowed by Amsterdam Attack
New York Joins a Global City Club, With a Deal on Congestion Pricing
Chemist Identifies Mystery ‘Blobs’ Washing Up in Newfoundland
Pope Francis Wants to Save the Environment. He Can Start With a Tree.
Israel Strikes Humanitarian Zone in Gaza
Norway Apologizes for Forced Assimilation of Sami and Other Minorities
Israel Bombs Targets in Syria, and Keeps Up Strikes in Lebanon
Investigators Assess if Netanyahu’s Aides Forged Oct. 7 Phone Records

Typhoon Usagi Slams Into the Philippines Thursday Briefing



Jupyter notebook --footer info-- (please always provide this at the end of each notebook)

```
In [13]: import os
import platform
import socket
from platform import python_version
from datetime import datetime

print('-----')
print(os.name.upper())
print(platform.system(), '|', platform.release())
print('Datetime:', datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
print('Python Version:', python_version())
print('IP Address:', socket.gethostname())
print('-----')
```

NT
Windows | 11
Datetime: 2024-11-17 12:36:40
Python Version: 3.12.4
IP Address: 172.27.5.229

Input and Output in Python, Formatting of Strings & Dates

Libraries and settings

```
In [1]: # Libraries
import os
import re
import pytz
import json
import folium
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt

from datetime import date
from datetime import time

from zipfile import ZipFile
from bs4 import BeautifulSoup

from PyPDF2 import PdfReader

from reportlab.lib.units import inch
from reportlab.lib.colors import blue
from reportlab.lib.pagesizes import LETTER
from reportlab.pdfgen.canvas import Canvas

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')

# Show current working directory
print(os.getcwd())

```

/home/pfrei/Dokumente/GitHub/data_ingestion/01_Input_Output_and_Formatting_Python

Read & write data from/to a database

Most data driven companies store their data in database management system. **SQLite** is a lightweight relational database management system (RDBMS). With python you can connect to a SQLite DB and make requests using SQL.

Write data to a database

```
In [2]: # Create a new db
conn = sqlite3.connect('data/example_sqlite.db')

# Close connection to db
conn.close()
```

```
In [3]: # Open connection to db
conn = sqlite3.connect('data/example_sqlite.db')

# Define variables and data types for the (empty) table
conn.execute('''CREATE TABLE IF NOT EXISTS COMPANY
                (ID INT PRIMARY KEY      NOT NULL,
                 NAME          TEXT      NOT NULL,
                 AGE           INT       NOT NULL,
                 CITY          CHAR(50),
                 SALARY        REAL);''')

# Read data from a file and write to data frame
data = pd.read_excel("data/db_data.xlsx", sheet_name = "Sheet1")
print(data)

# Write data to the data base table named 'COMPANY'
data.to_sql('COMPANY', conn, if_exists='replace')

# Commit the changes to the table
conn.commit()

# Close connection to db
conn.close()
```

	NAME	AGE	CITY	SALARY
0	Peter	25	Berlin	85000
1	Mary	30	London	95000
2	Anne	24	Zürich	125000
3	Joe	28	New York	115000
4	Mike	27	Paris	75000
5	Sue	26	London	95000

Query the database using SQL and write result to a pandas data frame

```
In [4]: # Connection to db
conn = sqlite3.connect("data/example_sqlite.db")

# Read data
df_sub = pd.read_sql("SELECT * FROM COMPANY WHERE AGE <= 26",
                      con=conn,
                      index_col=[ 'index' ])
print(df_sub)

# Close connection to db
conn.close()
```

	NAME	AGE	CITY	SALARY
index				
0	Peter	25	Berlin	85000
2	Anne	24	Zürich	125000
5	Sue	26	London	95000

Read & write data from/to files

Common data/file formats

Data formats in information technology may refer to:

- Data type, constraint placed upon the interpretation of data in a type system
- Signal (electrical engineering), a format for signal data used in signal processing
- Recording format, a format for encoding data for storage on a storage medium
- **File format, a format for encoding data for storage in a computer file**
- Container format (digital), a format for encoding data for storage by means of a standardized audio/video codecs file format
- Content format, a format for representing media content as data
- Audio format, a format for encoded sound data
- Video format, a format for encoded video data

Wikipedia: https://en.wikipedia.org/wiki/Data_format

This section provides common **file formats** a data scientist or a data engineer must be aware of. Later, we'll see how to read these file formats in Python.

List with common file formats explained in this notebook:

- CSV
- TXT
- JSON
- XML
- HTML
- ZIP
- XLSX
- PDF
- Image files (e.g. JPEG)

CSV (comma separated value)

- A comma-separated values (CSV) file is a delimited text file.
- Each line of the file is a data record.
- Each record consists of one or more fields, separated by a separator (default = comma).
- The use of the comma as a field separator is the source of the name for this file format.
- The separator can also be user-defined, e.g. you can also use a semicolon instead of a comma.
- A CSV file typically stores tabular data (numbers and text).

```
In [5]: # Read data from .csv-file using pandas
data = pd.read_csv("data/example.csv", sep=";")

# Print the header info of data (first five rows)
print(data.head(5))

# Write data to csv
data.to_csv("data/example_write.csv", sep=";")
```

	chicken_id	weight	breed	eggs_per_year
0	1	2728.854920	New Hampshire Red	158
1	2	2323.761365	Australorp	273
2	3	2635.062034	New Hampshire Red	153
3	4	2603.985152	Plymouth Rock	268
4	5	3079.394487	Australorp	360

TXT (plain text)

- In Plain Text file format, everything is written in plain text
- Usually, this text is in unstructured form and there is no meta-data associated with it
- The TXT file format can easily be read by any program

```
In [6]: # Open a connection to the text-file
text_file = open("data/example.txt",
                 "r",
                 encoding='utf-8')

# Read data from .txt file
lines = text_file.read()

# Show type
print(type(lines))

# Print the data
print(lines)
```

<class 'str'>
Dorothy lived in the midst of the great Kansas prairies, with Uncle Henry, who was a farmer, and Aunt Em, who was the farmer's wife. Their house was small, for the lumber to build it had to be carried by wagon many miles. There were four walls, a floor and a roof, which made one room; and this room contained a rusty looking cookstove, a cupboard for the dishes, a table, three or four chairs, and the beds. Uncle Henry and Aunt Em had a big bed in one corner, and Dorothy a little bed in another corner. There was no garret at all, and no cellar—except a small hole dug in the ground, called a cyclone cellar, where the family could go in case one of those great whirlwinds arose, mighty enough to crush any building in its path. It was reached by a trap door in the middle of the floor, from which a ladder led down into the small, dark hole.

```
In [7]: # Write data to .txt
lines = ['Dorothy lived in the midst of the great Kansas prairies',
         'with Uncle Henry, who was a farmer ...']

with open('data/example_write.txt', 'w') as f:
    f.writelines(lines)

# Check whether file exists
files = [f for f in os.listdir('data') if re.match('example_write.txt', f)]
print(files)

['example_write.txt']
```

JSON (JavaScript Object Notation)

- JSON is a syntax for storing and exchanging data
- JSON is text, written with JavaScript object notation

```
In [8]: # Read data
with open('data/example.json', 'r') as f:
    data = json.load(f)
print(data)

# Read data to a data frame using the pandas library
data = pd.read_json("data/example.json")

# Print the data
print(data)
```

{'firstName': 'John', 'lastName': 'Smith', 'age': 27, 'address': {'streetAddress': '21 2nd Street', 'city': 'New York', 'state': 'NY', 'postalCode': '10021-3100'}}	firstName lastName age address
streetAddress	John Smith 27 21 2nd Street
city	John Smith 27 New York
state	John Smith 27 NY
postalCode	John Smith 27 10021-3100

```
In [9]: # Write data to .json
data.to_json('data/example_write.json')

# Check whether the file exists
files = [f for f in os.listdir('data') if re.match('example_write.json', f)]
print(files)
```

['example_write.json']

XML (extensible markup language)

- XML stands for extensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

```
In [10]: # First option: reading the xml file with BeautifulSoup
bs = BeautifulSoup(open('data/example.xml'), 'html.parser')
print(bs.prettify())

# Second option: using pandas and convert the xml file to a data frame
data = pd.read_xml("data/example.xml")
print("-----")
print(data[["name", "price"]])
```

```

<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
  <food>
    <name>
      Belgian Waffles
    </name>
    <price>
      $5.95
    </price>
    <description>
      Two of our famous Belgian Waffles with plenty of real maple syrup
    </description>
    <calories>
      650
    </calories>
  </food>
</breakfast_menu>

-----
          name  price
0  Belgian Waffles  $5.95

```

```

In [11]: # Second option: reading xml using .read_xml() from pandas
data = pd.read_xml("data/example.xml")
print(data)

# Write data to .xml
data.to_xml('data/example_write.xml')

# Check whether file exists
files = [f for f in os.listdir('data') if re.match('example_write.xml', f)]
print(files)

```

	name	price	description \
0	Belgian Waffles	\$5.95	\n Two of our famous Belgian Waffles with pl...
			description \
0			calories
0			650

['example_write.xml']

HTML (hyper text markup language)

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

```

In [12]: # Read data from .html
filename = 'data/example.html'
html = open(filename, "r").read()
print(html)

```

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body><h1>My First Heading</h1>
<p>My first paragraph.</p></body>
</html>
```

```
In [13]: # Write data to .html (taking the html-file from above)
with open('data/example_write.html', 'w') as f:
    f.writelines(html)

# Check whether file exists
files = [f for f in os.listdir('data') if re.match('example_write.html', f)]
print(files)

['example_write.html']
```

ZIP (archive file format)

- ZIP is an archive file format that supports lossless data compression
- A ZIP file may contain one or more files or directories that may have been compressed
- The ZIP file format permits a number of compression algorithms, though DEFLATE is the most common
- The name "zip" (meaning "move at high speed") was suggested by R. Mahoney
- They wanted to imply that their product would be faster than ARC and other compression formats of the time

```
In [14]: # Pandas supports zip file reads
data = pd.read_csv("data/archive.zip", sep=";")
data.head(5)
```

```
Out[14]:   chicken_id      weight      breed  eggs_per_year
0           1  2728.854920  New Hampshire Red        158
1           2  2323.761365       Australorp        273
2           3  2635.062034  New Hampshire Red        153
3           4  2603.985152       Plymouth Rock        268
4           5  3079.394487       Australorp        360
```

```
In [15]: # Create an empty Zip-archive
zipObj = ZipFile('data/example_write.zip', 'w')

# Add selected files to the zip archive
zipObj.write('data/example.csv')
zipObj.write('data/example.html')
zipObj.write('data/example.json')

# Close the Zip-archive
zipObj.close()
```

```
# Check whether zip-file exists
files = [f for f in os.listdir('data') if re.match('example_write.zip', f)]
print(files)

['example_write.zip']
```

XLSX (Microsoft Excel Open XML file format)

- It is an XML-based file format created by Microsoft Excel
- The XLSX format was introduced with Microsoft Office 2007
- In XLSX data is organized under the cells and columns in a sheet
- Each XLSX file may contain one or more sheets
- A single workbook can contain multiple sheets

```
In [16]: # Read data from an example .xlsx-file
data = pd.read_excel("data/example.xlsx", sheet_name = "sheet1")

# Print the data
data.head(5)
```

Out[16]:

	chicken_id	weight	breed	eggs_per_year
0	1	2728.854920	New Hampshire Red	158
1	2	2323.761365	Australorp	273
2	3	2635.062034	New Hampshire Red	153
3	4	2603.985152	Plymouth Rock	268
4	5	3079.394487	Australorp	360

```
In [17]: # Write data to xlsx
data.to_excel('data/example_write.xlsx', sheet_name = "sheet1")

# Check whether file exists
files = [f for f in os.listdir('data') if re.match('example_write.xlsx', f)]
print(files)

['example_write.xlsx']
```

PDF (portable document format)

- PDF is a file format developed by Adobe in the 1990s to present documents, including text formatting and images, in a manner independent of application software, hardware, and operating systems
- Based on the PostScript language, each PDF file encapsulates a complete description of a fixed-layout flat document, including the text, fonts, vector graphics, raster images and other information needed to display it

```
In [18]: # Reading metadata
reader = PdfReader("data/example.pdf")
meta = reader.metadata
print(len(reader.pages))

# All of the following could be None!
```

```

print(meta.author)
print(meta.creator)
print(meta.producer)
print(meta.subject)

# Extract text
page = reader.pages[0]
print('\n')
print(page.extract_text())

# Number of pages
print('\n')
print(f'Number of pages in PDF: {len(reader.pages)}')

```

```

1
Gellrich Mario (gell)
Acrobat PDFMaker 21 für Excel
Adobe PDF Library 21.11.71
None

```

```

This is a PDF-File
Table 1:
Pet Color
Dog brown
Cat yellow
Bird green
Dog black
Cat gray
Bird green
Dog red
Cat blue

```

Number of pages in PDF: 1

```

In [19]: # Create a canvas
canvas = Canvas("data/example_write.pdf", pagesize = LETTER)

# Set font to Times New Roman with 36-point size
canvas.setFont("Times-Roman", 36)

# Draw blue text one inch from the left and ten inches from the bottom
canvas.setFillColor(blue)
canvas.drawString(1 * inch, 10 * inch, "This is a PDF file ...")

# Save the PDF file
canvas.save()

# Check whether file exists
files = [f for f in os.listdir('data') if re.match('example_write.pdf', f)]
print(files)

```

```
[ 'example_write.pdf' ]
```

Image file formats

- Image files consists of pixels and meta-data associated with it
- Usual image files are 3-dimensional, having RGB values

- Image files can also be 2-dimensional (grayscale) or 4-dimensional (having intensity)
- Each image consists one or more frames of pixels
- Each frame is made up of two-dimensional array of pixel values
- Pixel values can be of any intensity
- Meta-data associated with an image, can be an image type (.png) or pixel dimensions
- The different formats (JPEG, PNG, TIFF, GIF, ...) are used to organize and store digital images in a different way

```
In [20]: # Read image
image = plt.imread('data/example.jpeg')

# Plot image
plt.figure(figsize=(6,4))
plt.imshow(image)
```

Out[20]: <matplotlib.image.AxesImage at 0x72c5695ee360>



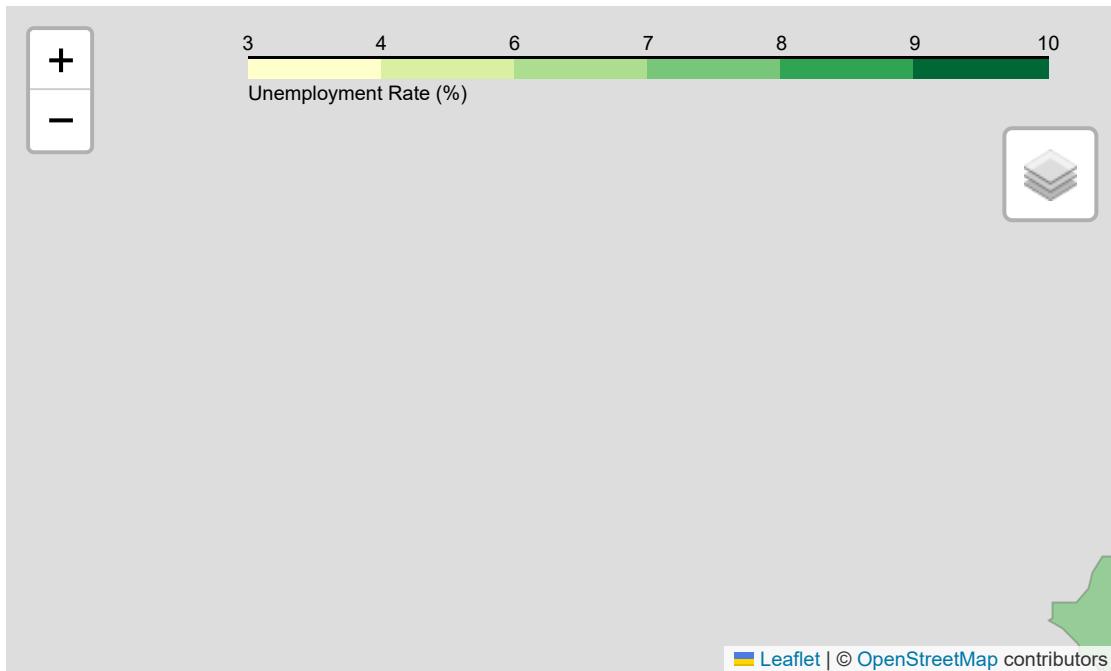
Spatial data

```
In [21]: # Read spatial data
url = ("https://raw.githubusercontent.com/python-visualization/folium/master/exa
state_geo = f"{url}/us-states.json"
state_unemployment = f"{url}/US_Unemployment_Oct2012.csv"
state_data = pd.read_csv(state_unemployment)

# Create choropleth map
m = folium.Map(location=[48, -102], zoom_start=3)
folium.Choropleth(
    geo_data=state_geo,
    name="choropleth",
    data=state_data,
    columns=["State", "Unemployment"],
    key_on="feature.id",
    fill_color="YlGn",
    fill_opacity=0.7,
```

```
    line_opacity=0.2,  
    legend_name="Unemployment Rate (%)",  
) .add_to(m)  
  
folium.LayerControl().add_to(m)  
  
# Show map  
m
```

Out[21]:



String formatting in Python

%-formatting

In [22]:

```
# Example (1):  
name = "Eric"  
print("Hello, %s." % name)  
  
# Example (2):  
name = "Eric"  
age = 74  
print("Hello, %s. You are %s." % (name, age))  
  
# Example (3):  
s1 = "Peter"  
s2 = "Mary"  
s3 = "%s and %s are living together." % (s1, s2)  
s4 = "{} and {} are living together.".format(s1, s2)  
print(s3)  
print(s4)  
  
# Example (4):  
first_name = "Eric"  
last_name = "Idle"  
age = 74  
profession = "comedian"  
affiliation = "Monty Python"
```

```
print("Hello, %s %s. You are %. You are a %. You were a member of %. " %
      (first_name, last_name, age, profession, affiliation))
```

```
Hello, Eric.  
Hello, Eric. You are 74.  
Peter and Mary are living together.  
Peter and Mary are living together.  
Hello, Eric Idle. You are 74. You are a comedian. You were a member of Monty Python.
```

str.format()

```
In [23]: # Example (1):
name = 'Peter'
age = 45
print("Hello, {}. You are {}".format(name, age))

# Example (2):
print("Hello, {}{}. You are {}{}".format(age, name))

# Example (3):
person = {'name': 'Peter', 'age': 45}
print("Hello, {}{}. You are {}{}".format(
    name=person['name'], age=person['age']))

# Example (4):
person = {'name': 'Eric', 'age': 74}
print("Hello, {}{}. You are {}{}".format(**person))

# Example (5):
first_name = "Eric"
last_name = "Idle"
age = 74
profession = "comedian"
affiliation = "Monty Python"
print(("Hello, {}{} {}. You are {}{}. " +
       "You are a {}{}. You were a member of {}{}.".")
      .format(first_name=first_name, last_name=last_name, age=age,
             profession=profession, affiliation=affiliation))
```

```
Hello, Peter. You are 45.  
Hello, Peter. You are 45.  
Hello, Peter. You are 45.  
Hello, Eric. You are 74.  
Hello, Eric Idle. You are 74. You are a comedian. You were a member of Monty Python.
```

In order to print the format use the syntax: `print(" {:.2f} ".format(3.1415926))`

Number	Format	Output	Description
3.1415926	{:.2f}	3.14	Format float 2 decimal places
3.1415926	{:+.2f}	+3.14	Format float 2 decimal places with sign
-1	{:+.2f}	-1.00	Format float 2 decimal places with sign
2.71828	{:.0f}	3	Format float with no decimal places

5	{:0>2d}	05	Pad number with zeros (left padding, width 2)
5	{:x<4d}	5xxx	Pad number with x's (right padding, width 4)
10	{:x<4d}	10xx	Pad number with x's (right padding, width 4)
1000000	{:,}	1,000,000	Number format with comma separator
0.25	{:.2%}	25.00%	Format percentage
10000000000	{:.2e}	1.00e+09	Exponent notation
13	{:10d}	13	Right aligned (default, width 10)
13	{:<10d}	13	Left aligned (width 10)
13	{:^10d}	13	Center aligned (width 10)

```
In [24]: # Examples of number formatting with str.format()
print("{:.2f}".format(3.1415926))
```

```
print("{:.2%}".format(0.25))

print("{:0>8d}".format(42))

print("{:,}".format(1000000000))

print("{:.2e}".format(1000000000))
```

```
3.14
25.00%
00000042
1,000,000,000
1.00e+09
```

f-strings

```
In [25]: # Example (1):
name = "Eric"
age = 74
print(f"Hello, {name}. You are {age}.")
```

```
# Example (2):
print(F"Hello, {name}. You are {age}.")
```

```
# Example (3):
print(f"The result is: {15 * 15 / 10}")
```

```
# Example (4):
print(f"{name.lower()} is funny.")
```

```
# Example 4:
name = "Eric"
profession = "comedian"
affiliation = "Monty Python"
message = (f"Hi {name}. "
           f"You are a {profession}. "
           f"You were in {affiliation}.")
print(message)
```

```
Hello, Eric. You are 74.  
Hello, Eric. You are 74.  
The result is: 22.5  
eric is funny.  
Hi Eric. You are a comedian. You were in Monty Python.
```

Floating point values have the f suffix. We can also specify the precision: the number of decimal places. The precision is a value that goes right after the dot character.

```
In [26]: # Format floats  
pi = 3.1415926535897932384626433832795028841971693993751058209749445923078164062  
  
# No specified format  
print(f'{pi}')  
  
# Zero decimal places  
print(f'{pi:.0f}')  
  
# Eight decimal places  
print(f'{pi:.8f}')  
  
# Two decimal places  
print(f'{pi*10000:.2f}')
```

3.141592653589793
3
3.14159265
31415.93

The width specifier sets the width of the value. The value may be filled with spaces or other characters if the value is shorter than the specified width.

```
In [27]: # Format width  
for x in range(1, 11):  
    print(f'{x:2} | {x**2:3} | {x**5:6}')  
  
# Format width and with and decimals  
print('\n')  
for x in range(1, 11):  
    print(f'{x:5.2f} | {x**2:6.2f} | {x**5:9.2f}')
```

1	1	1
2	4	32
3	9	243
4	16	1024
5	25	3125
6	36	7776
7	49	16807
8	64	32768
9	81	59049
10	100	100000

1.00	1.00	1.00
2.00	4.00	32.00
3.00	9.00	243.00
4.00	16.00	1024.00
5.00	25.00	3125.00
6.00	36.00	7776.00
7.00	49.00	16807.00
8.00	64.00	32768.00
9.00	81.00	59049.00
10.00	100.00	100000.00

By default, the strings are justified to the left. We can use the > character to justify the strings to the right. The > character follows the colon character.

```
In [28]: # Justify string
s1 = '123'
s2 = '1234'
s3 = '12345'
s4 = '123456'

print(f'{s1:>10}')
print(f'{s2:>10}')
print(f'{s3:>10}')
print(f'{s4:>10}')
```

123
1234
12345
123456

The example displays a formatted current datetime. The datetime format specifiers follow the : character.

```
In [29]: # Format_datetime
import datetime

now = datetime.datetime.now()
print(f'{now:%Y-%m-%d %H:%M:%S}')
```

2025-11-07 15:32:47

Numbers can have various numeric notations, such as decadic or hexadecimal.

```
In [30]: # Numeric notations
a = 300
```

```
# hexadecimal
print(f"{a:x}")

# octal
print(f"{a:o}")

# scientific
print(f"{a:e}")
```

```
12c
454
3.000000e+02
```

Basic date types in Python

```
In [31]: # Use the dir() function to get a list containing all objects a module.
print(dir(datetime))
```

```
['MAXYEAR', 'MINYEAR', 'UTC', '__all__', '__builtins__', '__cached__', '__doc__',
 '__file__', '__loader__', '__name__', '__package__', '__spec__', 'date', 'datetime',
 'datetime_CAPI', 'time', 'timedelta', 'timezone', 'tzinfo']
```

Date object to represent a date

```
In [32]: from datetime import date

# Date object to represent a date
a = date(2022, 3, 7)
print(a)

# Get current date
today = date.today()
print("Current date =", today)
```

```
2022-03-07
Current date = 2025-11-07
```

Time object to represent a time

```
In [33]: from datetime import time

# time(hour = 0, minute = 0, second = 0)
a = time()
print("a =", a)

# time(hour, minute and second)
c = time(hour = 11, minute = 34, second = 56)
print("c =", c)

# time(hour, minute and second)
b = time(11, 34, 56)
print("b =", b)
```

```
a = 00:00:00
c = 11:34:56
b = 11:34:56
```

Datetime object to represent a datetime

```
In [34]: from datetime import datetime

# datetime(year, month, day)
a = datetime(2022, 12, 31)
print(a)

# datetime(year, month, day, hour, minute, second)
b = datetime(2022, 12, 31, 23, 59, 59)
print(b)
```

2022-12-31 00:00:00
2022-12-31 23:59:59

Timedelta object

```
In [35]: from datetime import datetime

# Using date objects to calculate the time delta
t1 = date(year = 1984, month = 12, day = 31)
t2 = date(year = 2022, month = 12, day = 31)
t3 = t2 - t1
print(t3)
print("type of t3 =", type(t3))

# Using datetime objects to calculate the time delta
t4 = datetime(year = 1984, month = 12, day = 31, hour = 12)
t5 = datetime(year = 2022, month = 12, day = 31, hour = 4)
t6 = t5 - t4
print(t6)
print("type of t6 =", type(t6))
```

13879 days, 0:00:00
type of t3 = <class 'datetime.timedelta'>
13878 days, 16:00:00
type of t6 = <class 'datetime.timedelta'>

Format datetime using strftime()

```
In [36]: from datetime import datetime

# Current date and time
now = datetime.now()

# H:M:S format
t = now.strftime("%H:%M:%S")
print("time:", t)

# mm/dd/YY H:M:S format
s1 = now.strftime("%m/%d/%Y, %H:%M:%S")
print("s1:", s1)

# dd/mm/YY H:M:S format
s2 = now.strftime("%d/%m/%Y, %H:%M:%S")
print("s2:", s2)
```

```
time: 15:32:47
s1: 11/07/2025, 15:32:47
s2: 07/11/2025, 15:32:47
```

Handling time zone

```
In [37]: from datetime import datetime

# Local time
local = datetime.now()
print("Local:", local.strftime("%m/%d/%Y, %H:%M:%S"))

# NY time
tz_NY = pytz.timezone('America/New_York')
datetime_NY = datetime.now(tz_NY)
print("New York:", datetime_NY.strftime("%m/%d/%Y, %H:%M:%S"))
```

Local: 11/07/2025, 15:32:47
New York: 11/07/2025, 09:32:47

Jupyter notebook --footer info-- (please always provide this at the end of each notebook)

```
In [38]: import os
import platform
import socket
from platform import python_version
from datetime import datetime

print('-----')
print(os.name.upper())
print(platform.system(), '|', platform.release())
print('Datetime:', datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
print('Python Version:', python_version())
print('-----')

-----
POSIX
Linux | 6.8.0-87-generic
Datetime: 2025-11-07 15:32:47
Python Version: 3.12.12
-----
```

Writing data to and reading data from a database using Python

Libraries and settings

```
In [1]: # Libraries
import os
import sqlite3
import fnmatch
import pandas as pd
import matplotlib.pyplot as plt

# Define settings for graphics
plt.style.use('dark_background')

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')

# Show current working directory
print(os.getcwd())

c:\Users\Phil\Documents\GitHub\data_ingestion\02_SQLite_database_Python
```

Create sqlite data base

```
In [2]: # Create data base
conn = sqlite3.connect('apartment_database.db')
cursor = conn.cursor()

# Show dbs in the directory
flist = fnmatch.filter(os.listdir('.'), '*.db')
for i in flist:
    print(i)

apartment_database.db
```

Create empty SQL-table in the database

```
In [3]: cursor.execute('''CREATE TABLE IF NOT EXISTS apartments_table (OrderId VARCHAR(5
                                                               Rooms DECIMAL(8,
                                                               Area INT(8),
                                                               Price DECIMAL(8,
                                                               # Confirm changes to the table
                                                               conn.commit()''')  
'''
```

Read data from file to data frame

```
In [4]: # Read data
df = pd.read_csv('apartments_data_prepared.csv',
```

```

        sep=',',
        encoding='utf-8')[['web-scraper-order', 'rooms', 'area', 'price']

# Calculate new variable
df['price_per_m2'] = df['price'] / df['area']

print(df.shape)
df.head(5)

```

(1008, 5)

Out[4]:

	web-scraper-order	rooms	area	price	price_per_m2
0	1662023695-433	3.5	122.0	3180.0	26.065574
1	1662023745-820	2.5	78.0	3760.0	48.205128
2	1662023742-807	5.5	115.0	2860.0	24.869565
3	1662023804-1290	3.5	74.0	2165.0	29.256757
4	1662023739-771	5.5	195.0	6900.0	35.384615

Write data to the SQL-table in data base

In [5]:

```
df.to_sql(name = 'apartments_table',
          con = conn,
          index = False,
          if_exists = 'replace')
```

Out[5]: 1008

Query the SQL-table

In [6]:

```
# Query the SQL-table
cursor.execute('''SELECT * FROM apartments_table
                  WHERE Rooms >= 2.0
                  AND Price > 1000''')

df = pd.DataFrame(cursor.fetchall(),
                   columns=['OrderId', 'Rooms', 'Area', 'Price', 'Price_per_m2'])
print(df)
```

	OrderId	Rooms	Area	Price	Price_per_m2
0	1662023695-433	3.5	122.0	3180.0	26.065574
1	1662023745-820	2.5	78.0	3760.0	48.205128
2	1662023742-807	5.5	115.0	2860.0	24.869565
3	1662023804-1290	3.5	74.0	2165.0	29.256757
4	1662023739-771	5.5	195.0	6900.0	35.384615
..
878	1662023749-858	5.5	123.0	2550.0	20.731707
879	1662023739-792	3.5	70.0	1465.0	20.928571
880	1662023745-835	2.5	56.0	1870.0	33.392857
881	1662023783-1112	3.5	70.0	2190.0	31.285714
882	1662023749-853	3.5	96.0	2260.0	23.541667

[883 rows x 5 columns]

Plot histogram of rental prices

```
In [7]: plt.figure(figsize=(6,4))
df['Price_per_m2'].plot.hist(grid=True,
                           bins=20,
                           rwidth=0.9,
                           color='greenyellow')
plt.title('Appartment price per m2 (CHF)')
plt.xlabel('Price per m2')
plt.ylabel('Frequency')
plt.grid(linestyle='-', linewidth=0.1)

conn.close()
```



Jupyter notebook --footer info-- (please always provide this at the end of each notebook)

```
In [8]: import os
import platform
import socket
from platform import python_version
from datetime import datetime

print('-----')
print(os.name.upper())
print(platform.system(), '|', platform.release())
print('Datetime:', datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
print('Python Version:', python_version())
print('-----')
```

NT
Windows | 11
Datetime: 2025-11-09 16:53:08
Python Version: 3.13.9

CoinGecko REST API: extracting crypto currency data

Libraries and settings

```
In [1]: # Libraries
import os
import locale
import pandas as pd
locale.setlocale(locale.LC_ALL,"")

import matplotlib.pyplot as plt

# Define settings for graphics
plt.style.use('dark_background')

from pycoingecko import CoinGeckoAPI
cg = CoinGeckoAPI()

# Settings
import warnings
warnings.filterwarnings("ignore")

# Current working directory
print(f'Current working directory: {os.getcwd()}')
```

```
Current working directory: /home/pfrei/Dokumente/GitHub/data_ingestion/03_CoinGecko_WebAPI
```

Get data

```
In [2]: # Simple price endpoint with the required parameters
cg.get_price(ids='bitcoin', vs_currencies='usd')
```

```
Out[2]: {'bitcoin': {'usd': 100208}}
```

```
In [3]: # Multiple arguments (USD)
cg.get_price(ids=['bitcoin', 'near', 'ethereum', 'dogecoin'], vs_currencies='usd')
```

```
Out[3]: {'bitcoin': {'usd': 100178},
         'dogecoin': {'usd': 0.161942},
         'ethereum': {'usd': 3228.47},
         'near': {'usd': 2.35}}
```

```
In [4]: # Multiple arguments (USD & EUR)
cg.get_price(ids=['bitcoin', 'near', 'ethereum', 'dogecoin'], vs_currencies=['usd'])
```

```
Out[4]: {'bitcoin': {'usd': 100178, 'eur': 86560},
         'dogecoin': {'usd': 0.161942, 'eur': 0.139928},
         'ethereum': {'usd': 3228.47, 'eur': 2789.61},
         'near': {'usd': 2.35, 'eur': 2.03}}
```

```
In [5]: # Pass optional parameters as defined in the API doc (https://www.coingecko.com/)
data = cg.get_price( ids='bitcoin, near, ethereum, dogecoin',
                     vs_currencies='usd',
                     include_market_cap='true',
                     include_24hr_vol='true',
                     include_24hr_change='true',
                     include_last_updated_at='true')
data
```

```
Out[5]: {'bitcoin': {'usd': 100178,
                     'usd_market_cap': 1998453423623.9656,
                     'usd_24h_vol': 84447825282.7734,
                     'usd_24h_change': -2.5534005551223045,
                     'last_updated_at': 1762525902},
          'dogecoin': {'usd': 0.161942,
                     'usd_market_cap': 24564397439.4144,
                     'usd_24h_vol': 2001991207.3112733,
                     'usd_24h_change': 0.09182864524757609,
                     'last_updated_at': 1762525895},
          'ethereum': {'usd': 3228.47,
                     'usd_market_cap': 389530575205.93695,
                     'usd_24h_vol': 36473967607.96456,
                     'usd_24h_change': -3.0552440699142105,
                     'last_updated_at': 1762525903},
          'near': {'usd': 2.35,
                    'usd_market_cap': 3009740702.218562,
                    'usd_24h_vol': 863003667.3925748,
                    'usd_24h_change': 18.249270707311727,
                    'last_updated_at': 1762525895}}
```

```
In [6]: # Extract single values
names = list(data.keys())
print(names)
print("-----")

# Price of the first element in List
print(names[0])
print(data[names[0]]["usd"])
```

```
['bitcoin', 'dogecoin', 'ethereum', 'near']
-----
bitcoin
100178
```

Extract and plot data

```
In [7]: # Keys from dictionary
names = list(data.keys())

# Get values from dictionary and format to 2 decimal places
values = []
for i in range(len(names)):
    vals = data[names[i]]["usd"]
    formatted_val = f'{vals:.2f}'
    values.append(formatted_val)

# Create DataFrame
df = pd.DataFrame({
```

```

        'Name': names,
        'Value': values})

# Convert 'Value' column to numeric
df['Value'] = pd.to_numeric(df['Value'])

# Sort DataFrame by 'Value' in descending order
df = df.sort_values(by='Value', ascending=False)

print(df)

      Name      Value
0   bitcoin  100178.00
2  ethereum    3228.47
3      near      2.35
1  dogecoin     0.16

```

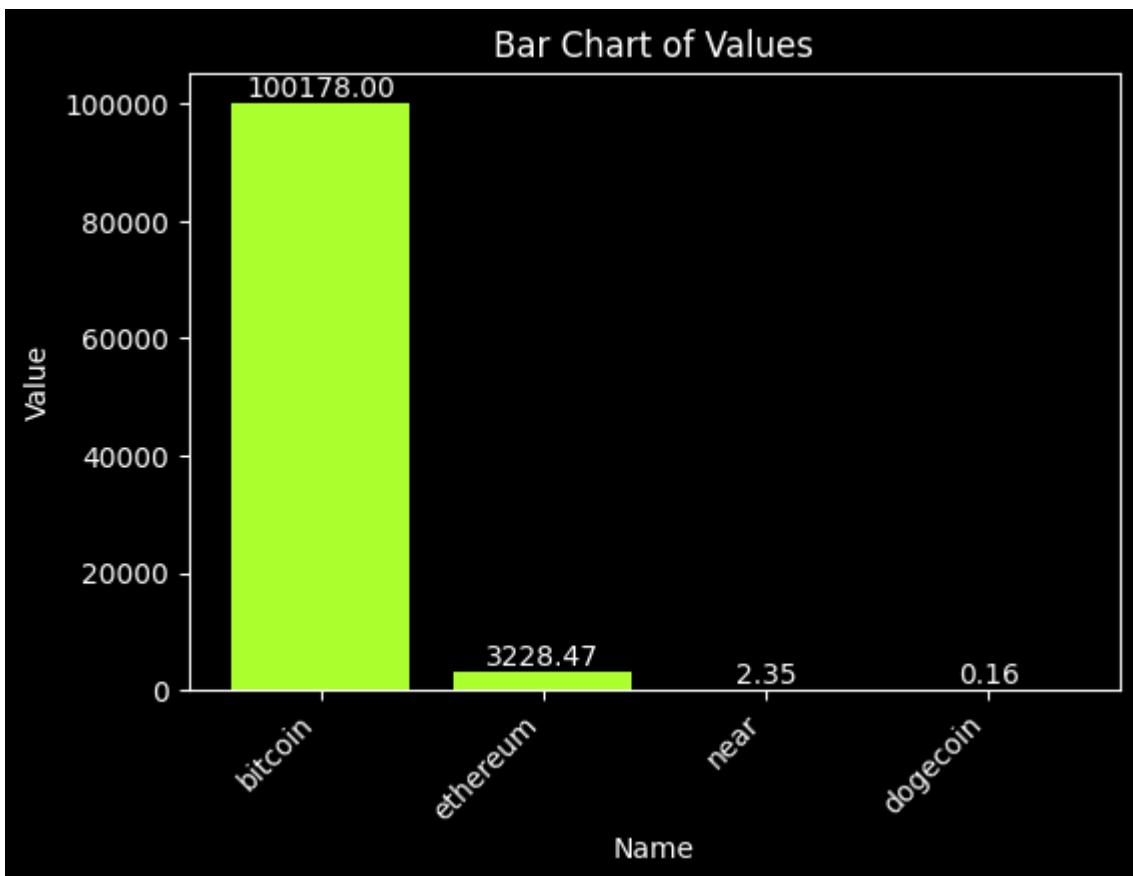
In [8]:

```

# Create bar chart
plt.figure(figsize=(6, 4))
bars = plt.bar(df['Name'], df['Value'], color='greenyellow')
plt.xlabel('Name')
plt.ylabel('Value')
plt.title('Bar Chart of Values')
plt.xticks(rotation=45, ha='right')

# Add values on top of bars
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2,
              yval + 0.05, f'{yval:.2f}',
              ha='center',
              va='bottom')
plt.show()

```



Jupyter notebook --footer info-- (please always provide this at the end of each notebook)

In [9]:

```
import os
import platform
import socket
from platform import python_version
from datetime import datetime

print('-----')
print(os.name.upper())
print(platform.system(), '|', platform.release())
print('Datetime:', datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
print('Python Version:', python_version())
print('-----')
```

```
-----
POSIX
Linux | 6.8.0-87-generic
Datetime: 2025-11-07 15:31:45
Python Version: 3.12.12
-----
```

Stock market analysis using data from the Yahoo-Finance Web API

Libraries and settings

```
In [1]: # Libraries
import os
import ta
import fnmatch
import pandas as pd
import yfinance as yf
from datetime import datetime
import matplotlib.pyplot as plt

# Define settings for graphics
# plt.style.use('dark_background')

# Settings
import warnings
warnings.filterwarnings("ignore")

# Current working directory
print(os.getcwd())
```

/home/pfrei/Dokumente/GitHub/data_ingestion/05_Yahoo_Finance_WebAPI

Get data

List of stock market symbols: <https://finance.yahoo.com/lookup>

```
In [2]: # Retrieve the Microsoft stock data from Yahoo finance
today = datetime.now().strftime("%Y-%m-%d")
print('Today is:', today)

data = yf.download('MSFT', start = "2022-01-01", end = today, progress=False)
data.tail()
```

Today is: 2025-11-07

```
Out[2]:      Price     Close      High      Low      Open      Volume
              Ticker      MSFT      MSFT      MSFT      MSFT      MSFT
              Date
2025-10-29  541.549988  546.270020  536.729980  544.940002  36023000
2025-10-30  525.760010  534.969971  522.119995  530.479980  41023100
2025-10-31  517.809998  529.320007  515.099976  528.880005  34006400
2025-11-03  517.030029  524.960022  514.590027  519.809998  22374700
2025-11-04  514.330017  515.549988  507.839996  511.760010  20958700
```

Calculate technical indicators

```
In [3]: # Function to compute Bollinger Bands
def BBANDS(data, n):
    MA = data.Close.rolling(window=n).mean()
    SD = data.Close.rolling(window=n).std()
    data['MiddleBand'] = MA
    data['UpperBand'] = MA + (2 * SD)
    data['LowerBand'] = MA - (2 * SD)
    return data

# Compute the Bollinger Bands for Google using the 50-day Moving average
BBANDS = BBANDS(data, 50)
BBANDS
```

Out[3]:

	Price	Close	High	Low	Open	Volume	MiddleBand	UpperB
Ticker	MSFT	MSFT	MSFT	MSFT	MSFT			
Date								
2022-01-03	324.504608	327.655138	319.686719	325.086250	28865100		NaN	NaN
2022-01-04	318.940247	324.940796	316.138684	324.582096	32674300		NaN	NaN
2022-01-05	306.696838	316.090267	306.309087	315.886673	40054300		NaN	NaN
2022-01-06	304.273346	308.945831	301.956480	303.565678	39646100		NaN	NaN
2022-01-07	304.428497	306.813198	300.599379	304.535116	32720000		NaN	NaN
...
2025-10-29	541.549988	546.270020	536.729980	544.940002	36023000	513.136599	532.886	
2025-10-30	525.760010	534.969971	522.119995	530.479980	41023100	513.553998	533.473	
2025-10-31	517.809998	529.320007	515.099976	528.880005	34006400	513.825399	533.596	
2025-11-03	517.030029	524.960022	514.590027	519.809998	22374700	514.021399	533.719	
2025-11-04	514.330017	515.549988	507.839996	511.760010	20958700	514.222799	533.718	

964 rows × 8 columns

Plot data and technical indicators

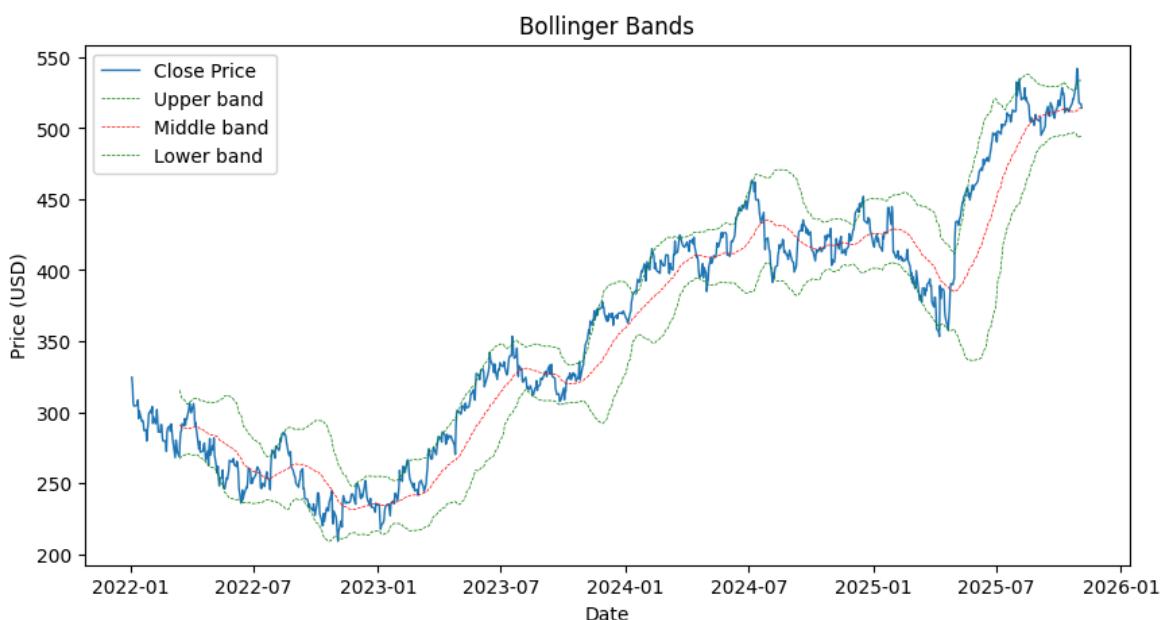
```
In [4]: # Create the plot
plt.figure(figsize=(10, 5))

# Set the title and axis labels
plt.title('Bollinger Bands')
plt.xlabel('Date')
plt.ylabel('Price (USD)')

plt.plot(BBANDS.Close, lw=1.0, label='Close Price')
plt.plot(data['UpperBand'], 'g--', lw=0.5, label='Upper band')
plt.plot(data['MiddleBand'], 'r--', lw=0.5, label='Middle band')
plt.plot(data['LowerBand'], 'g--', lw=0.5, label='Lower band')

# Add a Legend to the axis
plt.legend()

plt.show()
```



Jupyter notebook --footer info-- (please always provide this at the end of each notebook)

```
In [5]: import os
import platform
import socket
from platform import python_version
from datetime import datetime

print('-----')
print(os.name.upper())
print(platform.system(), '|', platform.release())
print('Datetime:', datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
print('Python Version:', python_version())
print('-----')
```

POSIX

Linux | 6.8.0-87-generic
Datetime: 2025-11-07 15:31:57
Python Version: 3.12.12

Getting all available Swiss supermarkets using the overpass Web API

Overpass turbo query to get all available supermarkets in Switzerland

```
In [1]: from pandas import json_normalize
import requests
import json
import folium

# Overpass API URL
url = 'http://overpass-api.de/api/interpreter'

# Overpass turbo query
query = f"""
    [out:json];
    area["ISO3166-1"="CH"][admin_level=2];
    node ["shop"]="supermarket"](area);
    out;"""

# Web API request
r = requests.get(url, params={'data': query})
data = r.json()['elements']

# Save data to file
with open('supermarkets.json', 'w') as json_file:
    json.dump(data, json_file)

# Store data in data frame
df = json_normalize(data)
df.head(5)
```

Out[1]:

	type	id	lat	lon	tags.addr:city	tags.addr:country	tags.addr:hon
0	node	33126515	47.155616	9.037915	Schänis	CH	
1	node	36726161	47.226191	8.980329	Uznach	NaN	
2	node	39768209	47.225154	8.969868	Uznach	NaN	
3	node	39947904	47.376732	8.542161	Zürich	CH	
4	node	48932835	47.375020	8.522895	Zürich	NaN	

5 rows × 278 columns

Plot supermarkets on map

```
In [2]: # Subset of supermarkets by brand
locations = df[["lat", "lon", "tags.brand", "tags.shop"]].loc[df["tags.brand"].isin(locations.head(5))]

# Subsets
locations_coop = df[["lat", "lon", "tags.brand", "tags.shop"]].loc[df["tags.brand"].isin(['Coop'])]
locations_migros = df[["lat", "lon", "tags.brand", "tags.shop"]].loc[df["tags.brand"].isin(['Migros'])]

# Marker symbols
url_01 = 'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master'
url_02 = 'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master'

# Create map
map = folium.Map(location=[locations.lat.mean(),
                           locations.lon.mean()],
                  zoom_start=8,
                  control_scale=True)

# Add Lat/Lon of Coop supermarkets
for i in range(0, len(locations_coop)):
    folium.Marker(location=(locations_coop.iloc[i]['lat'],
                            locations_coop.iloc[i]['lon']),
                  popup=locations_coop.iloc[i]['tags.brand'],
                  icon=folium.features.CustomIcon(url_01, icon_size=(16, 28))).add_to(map)

# Add Lat/Lon of Migros supermarkets
for i in range(0, len(locations_migros)):
    folium.Marker(location=(locations_migros.iloc[i]['lat'],
                            locations_migros.iloc[i]['lon']),
                  popup=locations_migros.iloc[i]['tags.brand'],
                  icon=folium.features.CustomIcon(url_02, icon_size=(16, 28))).add_to(map)

# Plot map
map
```

	lat	lon	tags.brand	tags.shop
1	47.226191	8.980329	Migros	supermarket
2	47.225154	8.969868	Coop	supermarket
3	47.376732	8.542161	Coop	supermarket
4	47.375020	8.522895	Migros	supermarket
6	47.491874	8.706448	Migros	supermarket

Out[2]:



Jupyter notebook --footer info-- (please always provide this at the end of each notebook)

In [3]:

```
import os
import platform
import socket
from platform import python_version
from datetime import datetime

print('-----')
print(os.name.upper())
print(platform.system(), '|', platform.release())
print('Datetime:', datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
print('Python Version:', python_version())
print('-----')
```

```
-----
POSIX
Linux | 6.8.0-87-generic
Datetime: 2025-11-07 15:32:08
Python Version: 3.12.12
-----
```

Getting Text data from RSS-Feeds

Libraries and settings

```
In [1]: # Libraries
import os
import feedparser
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# Settings
import warnings
warnings.filterwarnings("ignore")

# Current working directory
print(f'Current working directory: {os.getcwd()}')
```

Current working directory: /home/pfrei/Dokumente/GitHub/data_ingestion/07_RSS_Feeds

Parse the RSS feed

```
In [2]: # RSS feeds from NZZ website
feed = feedparser.parse("https://www.nzz.ch/startseite.rss")
```

Extract the data

```
In [3]: # Get data from RSS feeds
feed_entries = feed.entries

# Initialize an empty string to store the text
wordcloud_text = ""

# Show first RSS-feeds
print("Article Titles:")
for entry in feed.entries:

    article_title = entry.title
    article_link = entry.link
    article_published_at = entry.published
    article_published_at_parsed = entry.published_parsed

    # Append the title to the wordcloud_text
    wordcloud_text += f"{article_title} "

    # Change and format output
    print(f"{article_title}")
    # print(f"{article_title}[{article_link}]")
    # print(f"Published at {article_published_at}")
```

Article Titles:

KOMMENTAR - Von #DeleteFacebook zum digitalen Seelenstriptease: KI führt zum Ende des Datenschutzes, wie wir ihn kennen

Wo Israel über die Trümmer von Gaza wacht: Zu Besuch bei den Soldaten an der «gelben Linie»

Der Franken ist die stärkste Währung der Welt – aber schlägt er auch das Gold? Die längste bisher publizierte Zeitreihe liefert die Antwort

Giorgia Melonis Migrationspolitik: Nach ersten Erfolgen macht sich Ernüchterung breit

Stadion Zürich: Gegner ziehen ihren Rekurs ans Bundesgericht weiter

Die Insel Bali war ein Paradies. Dann kamen die digitalen Nomaden

Wie das Jassen zum Nationalspiel wurde: Wer verstehen will, wie die Schweiz funktioniert, muss eine Runde mitspielen

Die Zukunft der Computerspiele: Auf Monsterjagd am Hauptbahnhof

Wer hat den Sandkasten erfunden?

Auf dem Marktplatz in Oerlikon treffen sich Männer aus dem Balkan zum Schach. Sie spielen gegen Einsamkeit, Heimweh und einander.

Im Nachbarland herrscht Krieg und hier wird Krieg gespielt: Ein Besuch auf dem Arsoft-Gelände Battleground in Polen.

KOMMENTAR - Lautes Gebrüll, schwer erträgliche Doppelmoral

«De Irrsinn gaad mir nöd in Chopf ine»: Roger Schwabinski kämpft gegen die SRG und für seine grosse Liebe

Deutschland sei der «Puff Europas», sagt die deutsche Bundestagspräsidentin und will ein Sexkaufverbot

Ein Preis aus den Händen Putins: Deutschlands dirigierender Grenzgänger Justus Frantz und sein Bekenntnis zum russischen Regime

Spanien bittet Mexiko um Verzeihung für Schmerz und Ungerechtigkeit der Kolonialzeit

Google Maps aus der Antike: Ein Forschungsteam hat die Strassen des Römischen Reiches neu vermessen

SPONSORED CONTENT - Innovation in der DNA: Wie zwei Partner die Logistikbranche revolutionieren

KOMMENTAR - Treffen mit Donald Trump: Manager übernehmen, wo die Schweizer Diplomatie versagt

KOMMENTAR - Nach einigem Hin und Her soll der private Astronaut Jared Isaacman nun doch die Nasa leiten. Warum nicht gleich?

KOMMENTAR - Ein Geschäft voller falscher Anreize: Die Pflege von Angehörigen ist zu lukrativ

KOMMENTAR - Gut, dass Schweizer Unternehmer bei Trump weibeln – die Hoffnung auf den Supreme Court ist nämlich vergeblich

KOMMENTAR - Frühfranzösisch funktioniert nicht. Doch die Politik ignoriert die Realität – zum Schaden der Schüler

KOMMENTAR - Trumps Honeymoon ist vorbei – doch die Demokraten laufen Gefahr, ihren Bonus wieder zu verspielen

KOMMENTAR - Die Palästinenser sind noch kein Partner für den Frieden im Nahen Osten

KOMMENTAR - Konzernverantwortung: Die Ausrichtung auf die EU ist verfehlt

«More than Honey» war einmal: In Zürich streiten sich Imker und Naturschützer – der Kampf um die Biene ist entbrannt

Die SBB beschaffen bei Siemens 116 neue S-Bahn-Züge – die meisten davon werden im Raum Zürich verkehren

Statt zum Wohnort fährt ein Uber-Chauffeur Frauen nachts an abgelegene Orte. Dort begeht er sexuelle Übergriffe

GASTKOMMENTAR - Ich verstehe die Empörung über die Bezeichnung «Mohr» nicht: Denn ich bin selbst, per Definition des Wortes, ein Mohr

Ein paar wenige Minuten singt Eva Ndoja. Da platzt die Polizei in den Zürcher Nachtclub und verhaftet die bekannte albanische Sängerin

Der Kanton Zürich wollte an den üppigen Steuererträgen aus dem Immobilienhandel teilhaben. Das Verdikt ist deutlich: Kommt nicht infrage

INTERVIEW - «Die angeblichen neuen «Wunderwaffen» bringen Russland strategisch ni

chts. Die USA sollten sie rhetorisch einfach ignorieren», sagt der Experte Trumps Ukraine-Strategie: vom Anspruch auf Vermittlung zur politischen Ernüchterung

Operationsplan Venezuela: Wie die USA das Maduro-Regime stürzen könnten

INTERVIEW - «Endlich trifft der Westen Putin dort, wo es ihm weh tut», sagt ein Sanktionsexperte

GASTKOMMENTAR - Ohne amerikanisches Engagement droht in Nahost ein politisches Fiasco

Putin, Trump, Xi Jinping: warum bösartige Narzissten ganze Bevölkerungsgruppen mi treissen

Amerikas Zölle vor Gericht: Darum geht es im wichtigsten Verfahren seit Donald Trumps Wiederwahl

Der Student Olli H. hat eine Software entwickelt für Cyberkriminelle. Das ist die Geschichte seiner Verurteilung – und die Rolle der Schweiz darin

INTERVIEW - Verändert uns Reisen?

Félix Vallotton nahm seine Zeit kritisch ins Visier. Die Frauen verachtete er bis weilen auch unverhohlen

Nicht nur bei Hibbeligkeit hilfreich – Sport kann viele ADHS-Beschwerden abschwächen

Wohlstand mit Nebenwirkungen: wie volle Kassen und politischer Komfort die Entwicklung der grössten Stadt der Schweiz hemmen

Nissan Micra: ein Franzose in Japan

PODCAST «NZZ QUANTENSPRUNG» - Axolotl können Gliedmassen und Organe nachwachsen lassen. Forscher entschlüsseln, wie – um die Fähigkeit auf den Menschen zu übertragen

INTERVIEW - Ein verlorenes Bein nachwachsen lassen? Der Forscher Michael Levin will es möglich machen. Ein Gespräch

«Axolotl Overkill»: Der Schwanzlurch wird zum Mode-Haustier

Neue Therapien gegen Diabetes und Parkinson – nach 25 Jahren zahlt sich die Beharrlichkeit von Stammzellforschern aus

Newsletter «Quantensprung»: Die grossen Fragen aus Wissenschaft und Technologie auf dem Prüfstand – jeden Freitag in Ihrem Postfach

Diese Männer haben Trump besucht – und sorgen in der Schweizer Wirtschaft für etwas Hoffnung

Mehr Verkehrstote, aber kaum jemand stört sich daran: Auf den Schweizer Strassen sterben plötzlich wieder mehr Menschen

Nur eine kleine Elite gewinnt: McKinsey zeigt, was KI den Firmen wirklich bringt

SERIE - Gesund altern: wie sich normale Veränderungen im Gehirn von einer Demenz oder Parkinson unterscheiden lassen

Polizei durchsucht drei Häuser im Fall des getöteten Fabian aus Güstrow – und verkündet die Verhaftung einer Frau

GASTKOMMENTAR - Wenn der Staat alles mit einem Preisschild versieht, droht Korruption

DER ANDERE BLICK - Vom Musterschüler zum kranken Mann: Die Identitätskrise der deutschen Seele betrifft mehr als nur Politik und Wirtschaft

DER ANDERE BLICK - Die deutsche Stahlindustrie braucht günstige Energie – und keine staatlichen Subventionen

BILDSTRECKE - «Die Leute wollen, dass die Demokraten aufstehen und den Kampf weiterführen»

Mexikos Präsidentin wird auf offener Strasse sexuell belästigt: Frauenorganisationen solidarisieren sich mit ihr, aber sie wird auch kritisiert

Shutdown in den USA: Nach ihren Wahlsiegen wollen die Demokraten mehr – die Regierung warnt vor «riesigem Chaos» an Flughäfen

Eine Billion Dollar für Elon Musk: Die Tesla-Aktionäre stimmen dem grössten Bonus aller Zeiten zu

Schluss mit Excel und Powerpoint: Banker bekommen Konkurrenz durch KI – verlieren sie nun ihre Jobs?

Bitter: Campari hat die Finanzpolizei im Haus

Worldline-Debakel: Das Milliardenloch bei der Schweizer Börsenbetreiberin SIX wird immer grösser

Die Billionen-Wette: 17 Jahre nach der Finanzkrise lösen verbrieftete Kredite wieder Nervosität aus

So erben und vererben Sie richtig - das sind die wichtigsten Punkte

INTERVIEW - «Für die Entwaffnung der Hamas braucht es ein Uno-Mandat», sagt der Schweizer Botschafter in Israel

EU-Skeptiker warnen vor Bahnchaos wie in Deutschland – SBB stellen sich hinter den Bundesrat

Die Stadt Zürich möchte ihre Rolle als Hauptstadt des Schweizer Films ausbauen – mit zusätzlichen 820'000 Franken pro Jahr

ERKLÄRT - Kampf um Tempo 30: Der Kanton Zürich verlangt mehr Mitsprache auf städtischen Strassen – das Volk hat das letzte Wort

Todesursache geklärt: Der Zürcher SVP-Nationalrat Alfred Heer ist an akutem Herzversagen gestorben

Rückschlag für Saudiarabien: Die Olympischen E-Sports-Spiele sind abgesagt

Die Rückkehr zum alten englischen Kick-and-Rush: Fussballklubs versuchen, möglichst viel Chaos im Strafraum zu stiften

2026 Olympia als Langläuferin und 2028 als Triathletin? Gerne, sagt Anja Weber. Was haben die Mäuse mit dem Untergang der Sowjetunion zu tun?

«Der schlimmste Augenblick in meinem Leben war kein Wendepunkt, sondern nur eine kurze Pause», schreibt Malala Yousafzai.

Ein Jahr nach dem Kafka-Jubiläum legt die polnische Altmeisterin Agnieszka Holland mit einem Bionic nach – zu spät?

Wieder einmal behauptet Google, ein Quantencomputer habe einen Supercomputer überflügelt. Wenn sich das nur beweisen liesse.

Warum die Mücken bis in den November munter weiterstechen

Generate a WordCloud

```
In [4]: # Generate the word cloud
wordcloud = WordCloud(width=800,
                      height=400,
                      background_color='black').generate(wordcloud_text)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off') # Hide the axes
plt.show()
```



Jupyter notebook --footer info-- (please always provide this at the end of each notebook)

In [5]:

```
import os
import platform
import socket
from platform import python_version
from datetime import datetime

print('-----')
print(os.name.upper())
print(platform.system(), '|', platform.release())
print('Datetime:', datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
print('Python Version:', python_version())
print('-----')
```

```
-----
POSIX
Linux | 6.8.0-87-generic
Datetime: 2025-11-07 15:32:10
Python Version: 3.12.12
-----
```

Web Scraper Skidata

Data from: <https://www.skiinfo.de/schweiz/skipaesse>

Libraries and settings

```
In [1]: import requests
from bs4 import BeautifulSoup
import pandas as pd
import matplotlib.pyplot as plt

# Define settings for graphics
plt.style.use('dark_background')
```

Web Scraper Skidata

```
In [2]: # Option (1): Send an HTTP request to the URL
# url = 'https://www.skiinfo.de/schweiz/skipaesse'
# response = requests.get(url)
# html_content = response.content

# Option (2): Read html from file
# Read html from file
with open('skipass.html', 'r') as file:
    html_content = file.read()

# Parse the HTML content
soup = BeautifulSoup(html_content, 'html.parser')

# Locate the table and extract values
table = soup.find('table')

# Extract table headers
headers = [header.get_text().replace('\xa0', '') for header in table.find_all('th')]

# Extract table rows
rows = []
for row in table.find_all('tr')[1:]:
    cells = row.find_all('td')
    row_data = [cell.get_text().replace('\xa0', '').strip() for cell in cells]
    rows.append(row_data)

# Create a DataFrame
df = pd.DataFrame(rows, columns=headers)

# Change column names
df.columns = ['Skigebiet',
              'Saisonkarte_Erwachsene',
              'Saisonkarte_Kinder',
              'Tageskarte_Erwachsene',
              'Tageskarte_Kinder',
              'Online_Kaufen']
```

```

# Show dimensions
print('Number of rows:', df.shape)

# Change data types and remove special characters
df['Saisonkarte_Erwachsene'] = pd.to_numeric(df['Saisonkarte_Erwachsene'].str.replace(',', '.'))
df['Saisonkarte_Kinder'] = pd.to_numeric(df['Saisonkarte_Kinder'].str.replace(',', '.'))
df['Tageskarte_Erwachsene'] = pd.to_numeric(df['Tageskarte_Erwachsene'].str.replace(',', '.'))
df['Tageskarte_Kinder'] = pd.to_numeric(df['Tageskarte_Kinder'].str.replace(',', '.'))

# Write to csv
df.to_csv('skiinfo.csv', index=False)

# Show the first 5 rows
df.describe()

```

Number of rows: (147, 6)

	Saisonkarte_Erwachsene	Saisonkarte_Kinder	Tageskarte_Erwachsene	Tageskarte_Kinder
count	146.000000	136.000000	138.000000	135.000000
mean	723.945205	312.911765	49.652899	27.600000
std	977.369805	144.385927	18.083643	9.600000
min	1.000000	0.000000	20.000000	0.000000
25%	340.500000	203.000000	35.000000	22.000000
50%	549.000000	299.000000	44.000000	26.000000
75%	899.000000	399.000000	63.750000	31.000000
max	11449.000000	875.000000	95.000000	55.000000

Histogram of Skiprices

```

In [3]: # Histogramme erstellen
fig, axes = plt.subplots(2, 2, figsize=(7, 6))

df['Saisonkarte_Erwachsene'].plot(kind='hist',
                                   bins=20,
                                   edgecolor='black',
                                   color='greenyellow',
                                   ax=axes[0, 0])
axes[0, 0].set_title('Histogram of Saisonkarte_Erwachsene', fontsize=11)
axes[0, 0].set_xlabel('Price')
axes[0, 0].set_ylabel('Frequency')

df['Saisonkarte_Kinder'].plot(kind='hist',
                               bins=20,
                               edgecolor='black',
                               color='orange',
                               ax=axes[0, 1])
axes[0, 1].set_title('Histogram of Saisonkarte_Kinder', fontsize=11)
axes[0, 1].set_xlabel('Price')
axes[0, 1].set_ylabel('Frequency')

df['Tageskarte_Erwachsene'].plot(kind='hist',

```

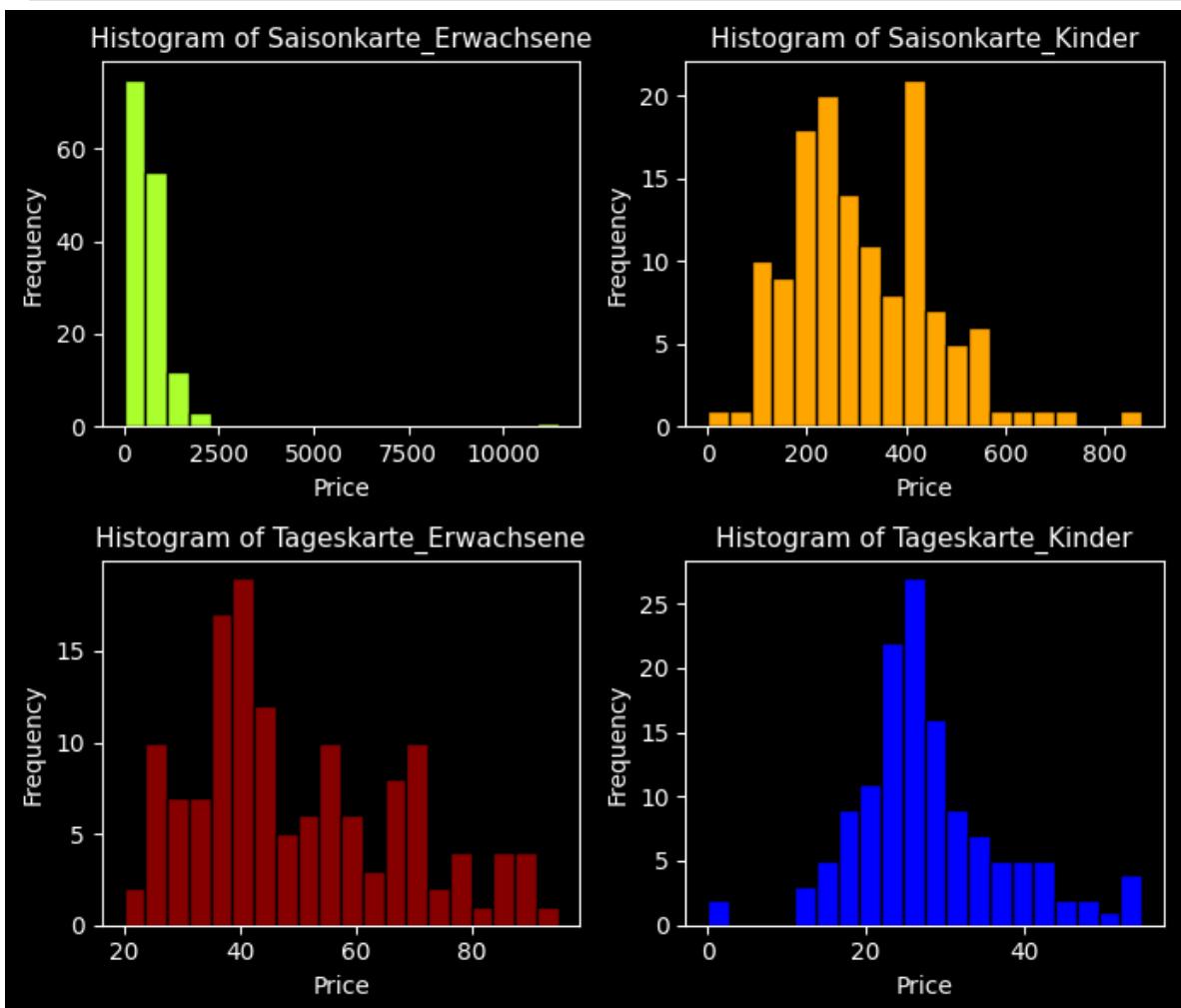
```

        bins=20,
        edgecolor='black',
        color='darkred',
        ax=axes[1, 0])
axes[1, 0].set_title('Histogram of Tageskarte_Erwachsene', fontsize=11)
axes[1, 0].set_xlabel('Price')
axes[1, 0].set_ylabel('Frequency')

df['Tageskarte_Kinder'].plot(kind='hist',
                             bins=20,
                             edgecolor='black',
                             color='blue',
                             ax=axes[1, 1])
axes[1, 1].set_title('Histogram of Tageskarte_Kinder', fontsize=11)
axes[1, 1].set_xlabel('Price')
axes[1, 1].set_ylabel('Frequency')

plt.tight_layout()
plt.show()

```



Jupyter notebook --footer info-- (please always provide this at the end of each notebook)

In [4]:

```

import os
import platform
import socket
from platform import python_version
from datetime import datetime

```

```
print('-----')
print(os.name.upper())
print(platform.system(), '|', platform.release())
print('Datetime:', datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
print('Python Version:', python_version())
print('-----')
```

```
-----
NT
Windows | 11
Datetime: 2025-11-09 16:55:07
Python Version: 3.13.9
-----
```

Web Scraping with Python and BeautifulSoup

Libraries and settings

```
In [2]: import requests
import xlwt
from bs4 import BeautifulSoup
import os

# Get current working directory
print(os.getcwd())

c:\Users\Phil\Documents\GitHub\data_ingestion\08_Web_Scraping_Python_BS4
```

Create workbook

```
In [ ]: wb = xlwt.Workbook()
ws = wb.add_sheet('Emails')
ws.write(0,0,'Emails')
```

Initialize list and counter

```
In [ ]: emailList = []
r = 0
```

Add URL to search for emails

```
In [ ]: # Add url of the page you want to scrape to urlString
urlString = 'https://www.zh.ch/de/bildung/schulen.html'
# urlString='https://www.agvs-upsa.ch/de/verband/mitgliederverzeichnis/liste'
```

Function to extract emails and store them in the list

```
In [ ]: def emailExtractor(urlString):
    getH=requests.get(urlString)
    h=getH.content
    soup=BeautifulSoup(h,'html.parser')
    mailtos = soup.select('a[href^=mailto]')
    for i in mailtos:
        href=i['href']
        try:
            str1, str2 = href.split(':')
        except ValueError:
            break
```

```
emailList.append(str2)
```

emailExtractor(urlString)

```
In [ ]: # Parts of the function ...
getH = requests.get(urlString)
print(getH)
```

<Response [200]>

```
In [ ]: # HTML-document
         h    = getH.content
         print(h[0:1000])
```

b'\n\n\n\n\n\n\n\n\n\n\n\n<!doctype html>\n<html lang="de">\n<head>\n <meta charset="UTF-8"/>\n <meta http-equiv="X-UA-Compatible" content="IE=edge"/>\n <title>Schulen | Kanton Zürich</title>\n <meta content="Kanton Zürich" property="og:site_name"/>\n <meta content="Schulen" property="og:title"/>\n <meta content="Der Kanton Zürich hat eine starke öffentliche Volksschule, ausgezeichnete Mittel- und Berufsfachschulen sowie internationale anerkannte Hochschulen." property="og:description"/>\n <meta content="https://www.zh.ch/content/dam/zhweb/bilder-dokumente/footer/social-media-links/kantonzuerich.png.zhweb-transform/open-graph-teaser-image-large/kantonzuerich.1611224633548.png" property="og:image"/>\n <meta content="https://www.zh.ch/de/bildung/schulen.html" property="og:url"/>\n\n <meta name="description" content="Der Kanton Zürich hat eine starke öffentliche Volksschule, ausgezeichnete Mittel- und Berufsfachschulen sowie internationale anerkannte Hochschulen.">\n

```
In [ ]: # HTML parse tree  
soup = BeautifulSoup(h,'html.parser')  
# print(soup)
```

```
In [ ]: # Links containing emails  
mailtos = soup.select('a[href^=mailto]')  
print(mailtos[0:3])
```

[

schule@adliswil.ch, praesidiumaesch@nassenmatt.ch

[praesidiumaesch](mailto:praesidiumaesch@nassenmatt.ch)<wbr>@nassenmatt.ch
,

[schulverwaltung<wbr>@bruelmatt.ch](mailto:schulverwaltung@bruelmatt.ch)

```
In [ ]: # Get single email  
href = mailtos[0]['href']  
href
```

```
Out[1]: 'mailto:schule@adliswil.ch'
```

```
In [ ]: # Splits the string  
str1, str2 = href.split(':')  
str2
```

```
Out[ ]: 'schule@adliswil.ch'
```

Print scraped emails and save it in an excel sheet

```
In [ ]: for email in emailList:  
    r=r+1  
    ws.write(r,0,email)  
    print(email)  
wb.save('emails.xls')
```

schule@adliswil.ch
praesidiumaesch@nassenmatt.ch
schulverwaltung@bruelmatt.ch
schulverwaltung@schule-aeugst.ch
schulverwaltung@osa.ch
primarschule@stadttaffoltern.ch
verwaltung@schulzweckverband.ch
schule.altikon@bluewin.ch
schulverwaltung@ps-andelfingen.ch
schulverwaltung@sek-andelfingen.ch
schulverwaltung.bachenbuelach@psbb.ch
schulverwaltung@sekbuelach.ch
schulverwaltung@primarschule-bachs.ch
schulverwaltung@baeretswil.org
schulverwaltung@bassersdorf.ch
schulverwaltung@schulebauma.ch
schulleitung@schulebenken.ch
schulverwaltung@primabirmensdorf.ch
schulverwaltung@primarschule-bonstetten.ch
schulverwaltung@sek-bonstetten.ch
schulleitung@schuleboppelsen.ch
schulverwaltung@schulebruetten.ch
bildung@bubikon.ch
schulverwaltung@buchs-zh.ch
primarschule@buelach.ch
sekretariat@schuledachsen.ch
schulverwaltung@primarschule-daegerlen.ch
schulverwaltung@leepuent.ch
schulverwaltung@schule-rotflue.ch
schulverwaltung@schuledaettlikon.ch
primarschule@psdielsdorf.ch
schulverwaltung@sekdielsdorf.ch
schulverwaltung@dietikon.ch
schule@dietlikon.org
sekretariat@schule-dinhard.ch
primarschule@duebendorf.ch
schulverwaltung@sek-ds.ch
schulabteilung@schuleduernten.ch
bildung@egg.ch
schulverwaltung@eglisau.ch
schulverwaltung@schule-elgg.ch
schulverwaltung@sek-elgg.ch
verwaltung@schule-ellikon.ch
schulverwaltung@sekrickenbach.ch
verwaltung@elsauschlatt.ch
info@ps.embra.ch
info@sek-embra.ch
schulverwaltung@erlenbach.ch
bildung@schulefaellanden.ch
sekretariat@schulefehraltorf.ch
schulverwaltung@schule-feuerthalen.ch
schulsekretariat@fischenthal.ch
schulverwaltung@schuleflaachtal.ch
sekretariat@schule-flurlingen.ch
schulverwaltung@schule-glattfelden.ch
schulverwaltung@schulegossau-zh.ch
schulverwaltung@primgreif.ch
sekretariat@schulegrueningen.ch
sekretariat@schule-hagenbuch.ch
schulverwaltung@hausen.zh.ch

schulverwaltung@sekhausen.ch
schulverwaltung@hedingen.ch
schulverwaltung@schule-henggart.ch
schulverwaltung@schule-herrliberg.ch
schulverwaltung@gseh.ch
verwaltung@schule-hettlingen.ch
schulverwaltung@schulehinwil.ch
schule@hittnau.ch
info@schule-hochfelden.ch
schulverwaltung@schulehoeri.ch
bea.pfeifer@hombrechtikon.ch
schulsekretariat@horgen.ch
bildung@ilef.ch
schulverwaltung@primarschulekappel.ch
schulverwaltung@kilchberg.ch
schulverwaltung@campusmoos.ch
schulverwaltung@kloten.ch
sekretariat@schuleknonau.ch
schulverwaltung@sekmaettmi.ch
schulverwaltung@schule-kuesnacht.ch
schule@langnau.zh.ch
schulverwaltung@lindau.ch
info@schule-lufingen.ch
schule@maennedorf.ch
sekretariat@primarschule-marthalen.ch
schulleitung@skmarthalen.ch
schulverwaltung@tagesschule-maschwanden.ch
schule@maur.ch
schulverwaltung@schulemeilen.ch
schulverwaltung@ps-mettmenstetten.ch
schulpflege@moenchaltorf.ch
schulverwaltung@oswueri.ch
schulverwaltung@primarschule-neerach.ch
schulverwaltung@schule-neftenbach.ch
schulverwaltung@primarschule-niederglatt.ch
schule@eduzis.ch
primarschule@niederhasli.ch
monika.manfredi@nuerensdorf.ch
schulverwaltung@schule-oberembrach.ch
info@sek-embra.ch
schulverwaltung@oberengstringen.ch
primarschule@oberglatt.zh.ch
schulverwaltung@sekro.ch
nadja.juon@oberrieden.ch
schulverwaltung@primarobfelden.ch
schulverwaltung@sek-obfelden.ch
schulverwaltung@schule-oetwil.ch
schulverwaltung@psog.ch
schule@opfikon.ch
schulverwaltung@ps-ossingen.ch
schulverwaltung@sekossingen.ch
schulpflege@ps-o.ch
sekretariat@sekuf.ch
schulverwaltung@ps-ottenbach.ch
schule@pfaeffikon.ch
schulverwaltung@schulepfungen.ch
schule.verwaltung@rafz.ch
k.rehberg@web.de
verwaltung@ps-regensdorf.ch
info@sek-regensdorf.ch

praesidium@primarschule-rheinau.ch
schule@richterswil.ch
schulverwaltung@primarschule-rickenbach.ch
schulverwaltung@sekkrickenbach.ch
schulverwaltung@schule-rifferswil.ch
sv@schule-rft.ch
schulverwaltung@psruemlang.ch
schulverwaltung@sekro.ch
schulverwaltung@rueschlikon.ch
schulverwaltung@schule-rueti.ch
schulverwaltung@russikon.ch
schule@schlieren.ch
zora.mangold@schule-schwerzenbach.ch
schulverwaltung@seegraeben.ch
sekretariat@primarschule-seuzach.ch
schulverwaltung@sekseuzach.ch
sekretariat@primarschule-stadel.ch
sekretariat@oberstufe-stadel.ch
sekretariat@schule-staefa.ch
schulverwaltung@schule-stallikon.ch
schulverwaltung@stammheim.ch
schulverwaltung@steinmaur.zh.ch
cornelia.schumacher@schule-thalheim.ch
schulverwaltung@sek-andelfingen.ch
bildung@thalwil.ch
praesidium@schule-truellikon.ch
schulleitung@skmarthalen.ch
daniel.hangartner@pstruttikon.ch
sekretariat@primarschule-turbenthal.ch
schulverwaltung@sektw.ch
schulverwaltung@schule-uetikon.ch
sekretariat@ps-uhwiesen.ch
sekretariat@seku.ch
schulverwaltung@schule-uitikon.ch
schulverwaltung@ps-buel.ch
schulverwaltung@schule-ur.ch
hans.karrer@urdorf.ch
ps@uster.ch
info@sekuster.ch
schule@volketswil.ch
primarschule@waedenswil.ch
schulverwaltung@oswaedenswil.ch
schulverwaltung@schule-wald.ch
kontakt@schule.wallisellen.ch
schule@wangen-bruettisellen.ch
info@schulewehntal.ch
schulverwaltung@schule-weiach.ch
sekretariat@oberstufe-stadel.ch
schulverwaltung@weiningen.ch
sekretariat@oberstufeweiningen.ch
schulverwaltung@schuleweisslingen.ch
verwaltung@schulewettswil.ch
schulverwaltung@wetzikon.ch
sekundarschule@wetzikon.ch
schulverwaltung@swibe.ch
sekretariat@pswila.ch
sekretariat@sekwila.ch
sekretariat@schuwi.ch
schulverwaltung@sektw.ch
verwaltung@schulen-winkel.ch

```
schulverwaltung@zell.ch  
info@schulezollikon.ch  
ksb-glattal.info@zuerich.ch  
ksb-letzi.info@zuerich.ch  
ksb-limmattal.info@zuerich.ch  
ksb-schwamendingen.info@zuerich.ch  
ksb-uto.info@zuerich.ch  
ksb-waidberg.info@zuerich.ch  
ksb-zuerichberg.info@zuerich.ch  
info@schule-zumikon.ch  
schulpflege@win.ch  
info@bi.zh.ch
```

Jupyter notebook --footer info-- (please always provide this at the end of each notebook)

In []:

```
import os  
import platform  
import socket  
from platform import python_version  
from datetime import datetime  
  
print('-----')  
print(os.name.upper())  
print(platform.system(), '|', platform.release())  
print('Datetime:', datetime.now().strftime("%Y-%m-%d %H:%M:%S"))  
print('Python Version:', python_version())  
print('-----')
```

```
-----  
POSIX  
Linux | 6.8.0-87-generic  
Datetime: 2025-11-07 15:32:16  
Python Version: 3.12.12  
-----
```