Filekezelés és Pandas

Filekezelés

Miért fontos?

- A mérnöki munka során a nagy adathalmazokat nyakran kapunk, és kell őket kezelnünk
- Memóriában nem tudunk nagy méretű adatokathalmazokat tárolni *Megoldás*: fájlokba mentés
- szenzorokból adatokat beolvasott adatokat file-okba logoljuk

Leggyakoribb fájlformátumok:

1. **INI (Initialization File)**:
    - Az INI fájl egy egyszerű szöveges fájlformátum, amelyet gyakran konfigurációs fájlok tárolására használnak Windows alapú rendszerekben.
    - Az INI fájlok hierarchikus struktúrában tárolják az adatokat, kulcs-érték párokat használva, amelyek szekciókba rendezhetők.
    - A fájl kiterjesztése általában ".ini".
2. **JSON (JavaScript Object Notation)**:
    - JSON egy könnyen olvasható, adatcsereformátum, amelyet gyakran használnak a webes alkalmazásokban.
    - Általában strukturált adatokat tárol, például objektumokat és listákat.
    - Az adatokat párokban tárolja, amelyek kulcs-érték párként vannak meghatározva.
    - A fájl kiterjesztése általában ".json".
3. **YAML (régen: Yet Another Markup Language, most: YAML Ain't Markup Language)**:
    - YAML egy ember- és gépelolvasásra is alkalmas adatátviteli formátum, amely könnyen olvasható és írható.
    - Támogatja a hierarchikus adatstruktúrákat és a listákat.
    - A YAML-t gyakran konfigurációs fájlokban és adatleíróban használják, különösen fejlesztői környezetekben.
    - A fájl kiterjesztése általában ".yaml" vagy ".yml".
4. **CSV (Comma-Separated Values)**:
    - CSV egy egyszerű és gyakran használt táblázatformátum, amely szöveges fájlban tárolja a táblázatos adatokat.
    - Az adatokat vesszők vagy más elválasztók választják el egymástól.
    - A CSV fájlok rendkívül elterjedtek, különösen adatimportálás és -exportálás során.
    - A fájl kiterjesztése általában ".csv".
5. **Feather**:
    - Feather egy bináris tárolási formátum, amely táblázatos adatok tárolására szolgál.
    - Feather fájlok gyors beolvasást és írást tesznek lehetővé, különösen Python és R nyelven.
    - Egyik előnye, hogy a fájlok közvetlenül a memóriába olvashatók, így nagy adatmennyiségek hatékony kezelésére alkalmasak.

- A fájl kiterjesztése általában ".feather".

Fileok írása és olvasása:

Pythonban a `with open("file_nev", "tipus")` , függvénnyel nyithatjuk meg a saját fájlainkat,</br>
A következő lehetőségeink vannak file megnyitásra:

1. **"r"**: Olvasási mód. Megnyitja a fájlt olvasásra. Hibaüzenetet dob, ha a fájl nem létezik.
2. **"w"**: Írási mód. Megnyitja a fájlt írásra. Új fájlt hoz létre, ha nem létezik, vagy felülírja</br> a fájlt, ha már létezik.
3. **"a"**: Hozzáfűző mód. Megnyitja a fájlt írásra, adatot fűzve a fájl végéhez. Új fájlt hoz létre,</br> ha nem létezik.
4. **"r+"**: Olvasási és írási mód. Megnyitja a fájlt olvasásra és írásra egyaránt. Hibaüzenetet dob,</br> ha a fájl nem létezik.
5. **"w+"**: Írási és olvasási mód. Megnyitja a fájlt olvasásra és írásra. Új fájlt hoz létre, ha nem létezik,</br> vagy felülírja a fájlt, ha már létezik.

File írása:

In [2]:
```
with open('data.txt', 'w') as f:
    data = 'valami adat, amit a file-ba írunk'
    f.write(data)
```

File olvasása:

Egyben:

In [3]:
```
with open('data.txt', 'r') as f:
    data = f.read()
```

Soronként: (`readline()` függvénnyel; figyelni kell, hogy nagyobb file-oknál nem mindig működik)

In [4]:
```
file = open("data.txt","r")
line = file.readline()
print(line)
```

```
valami adat, amit a file-ba írunk
```

File-ok elérése:

Adott mappa adatainak listázása:</br>

- Szükséges eszközeink az "os" könyvtárban találhatjuk
- Amikor az `alisdir()` vagy `scandir()` függvénynek egy '.'-ot adunk argumentumként, akkor a függvény a jelenlegi munkakönyvtárban lévő fájlokat és könyvtárakat fogja felsorolni, visszaadva azokat abszolút elérési útvonalukkal együtt.

File-ok olvasása root-ban: (`scandir()` -rel)

In [ ]:
```
import os

entries = os.scandir("/")

for entry in entries:
    print(entry.name)
```

File-ok olvasása jelenlegi mappában (`scandir()`-rel):

In [ ]:
```
entries = os.scandir(".")

for entry in entries:
    print(entry.name)
```

File-ok olvasása jelenlegi mappában (`listdir()`-rel):

In [ ]:
```
entries = os.listdir(".")

print(entries)
```

Almappák listázása:

- `os.scandir()` -t használhatjuk

In [ ]:
```
basepath = '/'
for entry in os.scandir(basepath):
    if os.path.isdir(os.path.join(basepath, entry)):
        print(entry)
```

File tulajdonságok elérése:

In [ ]:
```
with os.scandir('/') as dir_contents:
    for entry in dir_contents:
        info = entry.stat() #.stat metodus informaciot gyujt minden konyvtarrol
        print(info.st_mtime) #az st_mtime kiirja a modositasi idot (UNIX timestamp), olvashat
```

Könyvtár létrehozása:

Az ***os*** osztályunk `mkdir()` metódusával hozhatunk létre új könyvtárakat

In [ ]:
```
os.mkdir('example_directory/')
```

Nézzük meg, hogy létrejött-e:

In [ ]:
```
entries = os.scandir("./example_directory/")

for entry in entries:
    print(entry.name)
# nem kapunk errort mert létezik a mappa amit létrehoztunk, de nincs benne semmi
```

Szűrés file név alapján:

az `endswith()` függvényt használhatjuk ***wildcard*** helyett (wildcard: helyettesítő karakter)

In [ ]:
```
for entry in os.scandir('/'):
    if entry.name.endswith('.sys'):
        print(entry.name)
```

wildcard-dal:

In [12]:
```
import fnmatch

for file in os.scandir('.'):
    if fnmatch.fnmatch(file.name, 'd*.csv'):
        print(file.name)
```

Fájlok bezárása:</br>

A fájlokat a .close() függvényünkkel zárhatjuk be. (mindig meg kell tenni)

In [ ]:
```
f.close()
```

Egyéb fájlműveletek:

1. **`os.remove(path)`** :
   - Ez a parancs törli a megadott elérési útvonalon lévő fájlt.
   - A `path` paraméter az eltávolítani kívánt fájl elérési útvonala.
2. **`os.rename(src, dst)`** :
   - Ez a parancs átnevezi vagy áthelyezi a megadott forrásfájlt vagy könyvtárat az új cél elérési útvonalra.
   - A `src` paraméter az eredeti fájl vagy könyvtár elérési útvonala, a `dst` pedig az új elérési útvonal.
3. **`shutil.copy(src, dst)`** :
   - Ez a parancs másol egy fájlt a megadott forrás és cél elérési útvonalak között.
   - A `src` paraméter az eredeti fájl elérési útvonala, a `dst` pedig az új fájl helye.
4. **`shutil.copy2(src, dst)`** :
   - Ez a parancs hasonló a `shutil.copy()` parancshoz, azonban megőrzi az eredeti fájl metaadatokat, például a létrehozási és módosítási időt.
   - A `src` paraméter az eredeti fájl elérési útvonala, a `dst` pedig az új fájl helye.
5. **`shutil.copytree(src, dst)`** :
   - Ez a parancs rekurzívan másol egy teljes könyvtárat, beleértve az összes alkönyvtárat és fájlt, a megadott forrás és cél elérési útvonalak között.
   - A `src` paraméter az eredeti könyvtár elérési útvonala, a `dst` pedig az új könyvtár helye.
6. **`shutil.move(src, dst)`** :
   - Ez a parancs átnevezi vagy áthelyezi egy fájlt vagy könyvtárat a megadott forrás és cél elérési útvonalak között.
   - A `src` paraméter az eredeti fájl vagy könyvtár elérési útvonala, a `dst` pedig az új elérési útvonal.

Pandas (Panel Data and Python Data Analysis):

Mire jó? Mikor használjuk?

- A pandas könyvtár egyszerű és hatékony módot kínál ***fájlműveletek végrehajtására***, ***adattisztításra***, ***szűrésre***, ***statisztikai elemzésekre*** és ***adatvizualizációk létrehozására***.
- A könyvtár számos funkciót és eszközt kínál statisztikai műveletekre és adatmanipulációra
- Célunk, hogy megkönnyítsük az adatmanipulációt és -elemzés folyamtát, és csak a lényegi információra kelljen fókuszálnunk

Telepítése és használata Python környezetünkbe:

Telepítés hagyományos Python környezetben:</br> `pip install pandas` *Telepítés* **Anaconda** *környezetben:* `conda install -c conda-forge pandas`

A `pandas` függvényeit az alábbi paranccsal érhetjük el:

In [15]:
```
import pandas as pd
```

Pandas alapfogalmak:

Series:

- egydimenziós adatszerkezet
- rugalmas eszköz adatok tárolására és kezelésére
- számos beépített eszközzel rendelkezik
- létrehozhatjuk mind listából, mind szótárból
- létrehozó '*konstruktora*' a `pd.Series(adat)`

Listából létrehozva:

In [16]:
```
data = [10, 20, 30, 40, 50]
series = pd.Series(data)
```

Szótárból létrehozva:

In [17]:
```
data = {'a': 10, 'b': 20, 'c': 30, 'd': 40, 'e': 50}
series = pd.Series(data)
```

Lehetőségünk van a szótár csak néhány elemének importjára

In [ ]:
```
myvar = pd.Series(data, index = ['a', 'b'])

print(myvar)
```

Hozzáférés elemekhez az index vagy az egyedi címkék alapján:

In [ ]:
```
print(series[0])  # Az első elem értéke
print(series['b'])  # Az 'b' címkéjű elem értéke
```

Szűrés:

In [ ]:
```
print(series[series > 25])  # Minden elem, amely nagyobb, mint 25
```

Statisztikai műveletek:

In [ ]:
```
print(series.mean())  # Átlag számítása
print(series.max())   # Maximális érték
print(series.min())   # Minimális érték
```

Vizualizáció:

In [ ]:
```
import matplotlib.pyplot as plt
series.plot()  # Diagram létrehozása
```

Dataframe:

- kétdimenziós adatszerkezet

- többdimenziós adatok tárolására és kezelésére alkalmas
- tartalmazza a sorokat és oszlopokat, amelyek címkézettek lehetnek
- számos beépített funkcióval rendelkezik az adatok manipulálására és elemzésére
- könnyen létrehozható, például listákból vagy szótárakból
- létrehozó '*konstruktora*' a `pd.DataFrame(adat)`

Szótárból létrehozva:

In [50]:
```
data = {'Name': {'a': 'John', 'b': 'Anna', 'c': 'Peter', 'd': 'Linda'},
        'Age': {'a': 25, 'b': 30, 'c': 35, 'd': 40},
        'City': {'a': 'New York', 'b': 'Paris', 'c': 'Berlin', 'd': 'Tokyo'}}
df = pd.DataFrame(data)
```

Listából létrehozva:

In [46]:
```
data = {'Name': ['John', 'Anna', 'Peter', 'Linda'],
        'Age': [25, 30, 35, 40],
        'City': ['New York', 'Paris', 'Berlin', 'Tokyo']}
df = pd.DataFrame(data)
```

Adatok elérése a `.loc[hely]` függvénnyel történik:

In [47]:
```
print(df.loc[0])
```

```
Name         John
Age            25
City     New York
Name: 0, dtype: object
```

Akár többet is lekérhetünk

In [48]:
```
print(df.loc[[0, 1]])
```

```
   Name  Age      City
0  John   25  New York
1  Anna   30     Paris
```

Saját indexelés alapján is elérhetjük:

In [51]:
```
print(df.loc["a"])
```

```
Name         John
Age            25
City     New York
Name: a, dtype: object
```

Kulcs alapján értéket is módosíthatunk:

In [53]:
```
df.loc[df['Name'] == 'John', 'Age'] = 45
print(df)
```

```
       Name   Age      City
a      John  45.0  New York
b      Anna  30.0     Paris
c     Peter  35.0    Berlin
d     Linda  40.0     Tokyo
John    NaN  45.0       NaN
```

Példa: Drónadatok feldolgozása

Az alábbiakban adott a University of Zürich (UZH) Robotics and Perception Group laborjának egy mérése. Ebben a mérésben egy önvezető drón trajektóriáját figyeltük meg és a feladatunk, hogy a mozgást kinematikai szempontból elemezzük:

1. Töltsük le, és tároljuk el egy DataFrame-ben a mérési adatokat!
2. Jelenítsük meg a mért adatokat egy interaktív (plotly) ábrán!
3. Numerikus deriválás segítségével számítsuk ki a helyzetkoordinátákból a drón sebességét!
4. Hasonlítsuk össze a kapott értékeket a táblázatban található - szenzorosan mért - sebességértékekkel!

Megoldás: 1. Adatok letöltése

Megjegyzés: Ha a letöltés során nem működik a `git` *parancs, akkor a linkre kattintva csak töltsük le a mappát!*

In [ ]:
```
# A git paranccsal egyszerűen letölthetjük az adatokat
! git clone https://github.com/uzh-rpg/mh_autotune/
```

```
fatal: destination path 'mh_autotune' already exists and is not an empty directory.
```

Szükséges importok:

In [ ]:
```
! pip install plotly
! pip install pandas
```

```
Requirement already satisfied: plotly in c:\users\mbenc\anaconda3\lib\site-packages (5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\mbenc\anaconda3\lib\site-packages
(from plotly) (8.0.1)
Requirement already satisfied: pandas in c:\users\mbenc\anaconda3\lib\site-packages (1.5.3)
Requirement already satisfied: numpy>=1.21.0 in c:\users\mbenc\anaconda3\lib\site-packages (f
rom pandas) (1.23.5)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\mbenc\anaconda3\lib\site-pa
ckages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\mbenc\anaconda3\lib\site-packages (fr
om pandas) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\mbenc\anaconda3\lib\site-packages (from p
ython-dateutil>=2.8.1->pandas) (1.16.0)
```

Letöltés után már készen is állunk arra, hogy beolvassuk és egy `DataFrame` változóban eltároljuk az adathalmazunkat.

In [ ]:
```
import pandas as pd

# Fileokat egyszerűen beolvashatunk közvetlen DataFrame-be is:
df = pd.read_csv("mh_autotune/resources/trajectories/spiral.csv")

# ZIP esetén:
# df = pd.read_csv("mh_autotune-main/mh_autotune-main/resources/trajectories/spiral.csv")

df
```

	Unnamed: 0	t	p_x	p_y	p_z	q_w	q_x	q_y	q_z
0	0	0.000000	0.750000	2.000000	-2.020000	0.707107	-7.536130e-26	3.320919e-25	-0.707107
1	1	0.008737	0.749994	1.999997	-2.019764	0.707001	1.613351e-02	-6.279024e-03	-0.707001
2	2	0.017475	0.749948	1.999977	-2.019056	0.706683	3.226219e-02	-1.255616e-02	-0.706683
3	3	0.026212	0.749826	1.999924	-2.017880	0.706153	4.838119e-02	-1.882954e-02	-0.706153
4	4	0.034950	0.749588	1.999819	-2.016241	0.705412	6.448568e-02	-2.509727e-02	-0.705412
...	...	...	...	...	...	...	...	...	...
3776	3776	32.992700	-16.243308	-10.728720	3.965199	0.616491	-3.988703e-01	2.369371e-01	0.636162
3777	3777	33.001500	-16.362791	-10.571990	3.960519	0.616502	-3.988709e-01	2.370199e-01	0.636120
3778	3778	33.010200	-16.482262	-10.414652	3.955796	0.616513	-3.988715e-01	2.371027e-01	0.636078
3779	3779	33.019000	-16.601721	-10.256708	3.951031	0.616524	-3.988720e-01	2.371855e-01	0.636036
3780	3780	33.027700	-16.721167	-10.098159	3.946224	0.616535	-3.988726e-01	2.372684e-01	0.635994

3781 rows × 19 columns

Amit fontos észrevenni, hogy ez a sok adat önmagában semmitmondó, és nehezen lehet összefoglalni, hogy a mi a mérés **információtartalma**. Annak érdekében, hogy "okosabbak is legyünk" az adatokból jellemzően több statisztikai mutatót is számolunk, ábrázoljuk az adatokat (akár térben és időben is), illetve a számunkra megfelelő módon transzformáljuk.

Mivel itt egy mérésről van szó, ezért érdemes ábrázolni az adatokat!

Megoldás: 2. Az adatok ábrázolása

In [ ]:
```
# Notebook-kompatibilitás
from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)

# Plotly beolvasása
import plotly.express as px

# Az egyes tengelyek a dataframe egyes oszlopai
fig = px.line_3d(df, x='p_x', y='p_y', z='p_z')
fig.show()
```

p_

Megoldás 3. Sebességek ábrázolása

Numerikus deriválás segítségével könnyen számíthatóak a sebességek: $v_i[k] = \dot{x}_i[k] = \frac{x_i[k]-x_i[k-1]}{\Delta t}$

In [ ]:
```
import numpy as np

df["p_x_d"] = (df["p_x"]-df["p_x"].shift(1))/(df["t"]-df["t"].shift(1))
df["p_y_d"] = (df["p_y"]-df["p_y"].shift(1))/(df["t"]-df["t"].shift(1))
df["p_z_d"] = (df["p_z"]-df["p_z"].shift(1))/(df["t"]-df["t"].shift(1))
df["p_v_abs"] = np.sqrt(df["p_x_d"].values**2+df["p_y_d"].values**2+df["p_z_d"].values**2)
df
```

	Unnamed: 0	t	p_x	p_y	p_z	q_w	q_x	q_y	q_z
0	0	0.000000	0.750000	2.000000	-2.020000	0.707107	-7.536130e-26	3.320919e-25	-0.707107
1	1	0.008737	0.749994	1.999997	-2.019764	0.707001	1.613351e-02	-6.279024e-03	-0.707001
2	2	0.017475	0.749948	1.999977	-2.019056	0.706683	3.226219e-02	-1.255616e-02	-0.706683
3	3	0.026212	0.749826	1.999924	-2.017880	0.706153	4.838119e-02	-1.882954e-02	-0.706153
4	4	0.034950	0.749588	1.999819	-2.016241	0.705412	6.448568e-02	-2.509727e-02	-0.705412
...	...	...	...	...	...	...	...	...	...
3776	3776	32.992700	-16.243308	-10.728720	3.965199	0.616491	-3.988703e-01	2.369371e-01	0.636162
3777	3777	33.001500	-16.362791	-10.571990	3.960519	0.616502	-3.988709e-01	2.370199e-01	0.636120
3778	3778	33.010200	-16.482262	-10.414652	3.955796	0.616513	-3.988715e-01	2.371027e-01	0.636078
3779	3779	33.019000	-16.601721	-10.256708	3.951031	0.616524	-3.988720e-01	2.371855e-01	0.636036
3780	3780	33.027700	-16.721167	-10.098159	3.946224	0.616535	-3.988726e-01	2.372684e-01	0.635994

3781 rows × 23 columns

Ezt követően ezt is ábrázoljuk:

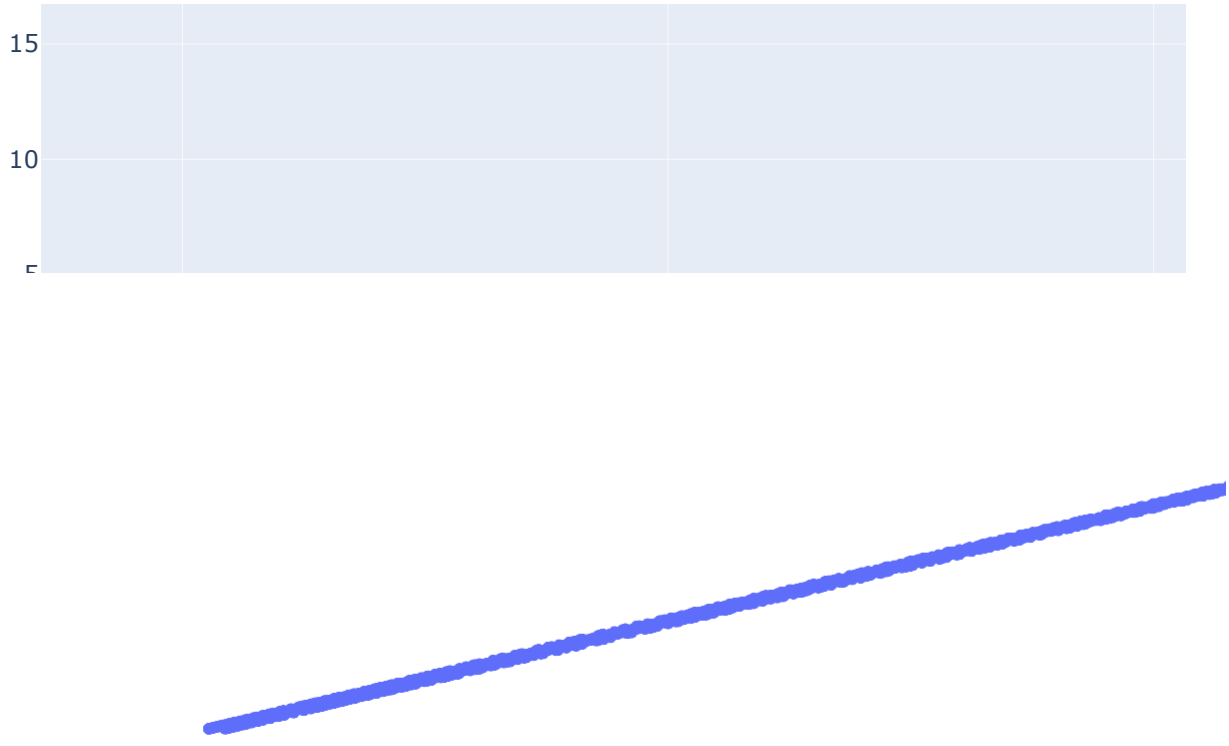
In [ ]:
```
fig = px.scatter_3d(df.iloc[1:,:], x = 'p_x', y = 'p_y', z = 'p_z', color = "p_v_abs")
fig.show()
```

4

p_z

Megoldás: 4. Sebességértékek összehasonlítása

In [ ]:
```
fig = px.scatter(df.iloc[1:,:], x = 'p_x_d', y = 'v_x')
fig.show()
```



Feladat:

1. Számoljuk ki és jelenítsük meg a drón gyorsulását!
2. Határozzuk meg a trajektória görbületét!