1 Behandle strukturert tekstfil

I denne delen skal vi leke oss med en strukturert tekstfil, slik vi kunne fått for eksempel etter en liste over personer som har fått et registreringsskjema, eller etter å ha eksportert et regneark med studentkarakterer.

1.1 Lese en enkel (men strukturert) tekstfil

Før vi kan importere biblioteker, må vi installere dem. Se detaljerte instruksjoner om hvordan du installerer python biblioteker i filen `installere_pakker_vscode.pdf`.

[1]:
```
# Importer biblioteket vi trenger
import matplotlib.pyplot as plt
```

Eksempel `allergier.txt`: en strukturert tekstfil som inneholder på en strukturert måte alle deltakerne til en hendelse, med deres tilsvarende matrestriksjoner.

Vi oppfordrer deg til å se på tekstfilen først ved å åpne den med din vanlige teksteditor (det kan være Visual Studio Code).

- Vi ser at hver linje har samme struktur: `navn:allergi1, allergi2, allergi3`.
- Vi kan dele linjen mellom det som er før `:` (navnet) og det som er etter `:`(allergiene).
- Vi kan bruke steng metoden `split()`, med argumentet `:` som skilletegn, for å dele linjen mellom det som står før skilletegnet `:` og det som står etter. Dette vil gi oss en liste med 2 strenger, der det første elementet er navnet og det andre en streng som representerer deres allergier.

[2]:
```
# Lese filen som en vanlig tekst fil
with open("allergier.txt", mode="rt") as f:
    linjer  = f.read().split("\n")

# Se den første linjen
print(linjer[0])
# Del linjen mellom det som står før ":" og det som står etter
print(linjer[0].split(":"))
```

```
Jakob:
['Jakob', '']
```

- Vi ser i cellen ovenfor at linjen 0, tilsvarer 'Jakob', og vi ser at Jakob har ingen allergier.
- I cellen nedenfor ser vi på en annen linje, som tilsvarer 'Aksel', og vi ser at Aksel har flere allergier.
- Vi ser at hver allergi er skilt fra hverandre med `,` (komma mellomrom). Vi kan bruke `split()` igjen, med argumentet `,` som skilletegn, for å dele linjen hver gang vi ser `,`. Dette vil gi oss en liste med flere strenger, der hvert elementet representerer en allergi.

[3]:
```
# Se en annen linje, som har flere allergier
print("Hele linje:", linjer[6])

# Del linjen mellom det som står før ":" og det som står etter
delt_linje = linjer[6].split(":")
# Navnet er det første elementet
navn = delt_linje[0]
# Og allergiene (som en lang streng) er det andre elementet
streng_allergier = delt_linje[1]
# Del strengen igjen, inn i en liste av strenger, der hvert element
# tilsvarer én allergi.
```

```
liste_allergier = streng_allergier.split(", ")

print(f"Navn: {navn}. Allergier: {liste_allergier}")
```

```
Hele linje: Aksel:svinn, egg, nøtter
Navn: Aksel. Allergier: ['svinn', 'egg', 'nøtter']
```

1.1.1 Eksempel 1: Finn navnene og deres tilhørende allergier

I dette eksempelet ønsker vi å tolke hver linje i filen for å trekke ut hvert navn sammen med deres tilsvarende allergier. Vi må derfor gjenta det vi har gjort i de forrige cellene, men for hver enkelt linje i filen, og lagre den uttrekkede informasjonen på en måte som gjør fremtidige Python-operasjoner enkle:

- Gjenta det vi har nettopp gjort for hver linje i filen.
- Lagre navn-allergier parene i en ordbok, der navnene er nøklene og listene av allergier er verdiene.
- Vi kan forestille oss at vi kunne gjenbruke denne ordboken for flere behandlinger senere i programmet vårt (Finn navnene på personene har ingen allergi. Beregn gjennomsnittlig antall allergier per person. Finn personen som har flest allergier. Finn navnene på personene som har en bestemt allergi. Osv.)

[4]:
```
# Initialiser en tom ordbok
navn_og_allergier = {}

for linje in linjer:
    # Del hver linje ved ":" for å separere navnet fra allergiene.
    delt_linje = linje.split(":")

    # Navnet er det første elementet og allergiene det andre
    navn = delt_linje[0]
    streng_allergier = delt_linje[1]

    # Del strengen igjen, inn i en liste av strenger, der hvert element
    # tilsvarer én allergi.
    liste_allergier = streng_allergier.split(", ")

    # Legge til en navn-allergi par i ordboka med navnet som nøkkel og
    # liste av allergier som verdi
    navn_og_allergier[navn] = liste_allergier

print(navn_og_allergier)
```

```
{'Jakob': [''], 'Noah': ['svinn'], 'Emil': ['gluten'], 'Lukas': [''], 'Isak':
['gluten'], 'Filip': [''], 'Aksel': ['svinn', 'egg', 'nøtter'], 'Theodor': [''],
'Ludvig': [''], 'Oskar': [''], 'Liam': ['melk', 'fisk'], 'Johannes': [''],
'Kasper': ['kjøtt', 'melk', 'laktose'], 'Magnus': [''], 'Tobias': ['kjøtt'],
'Henrik': [''], 'Mathias': ['kjøtt', 'svinn', 'egg'], 'Viktor': [''], 'Ulrik':
['svinn', 'gluten'], 'Matheo': ['nøtter', 'fisk'], 'Adam': ['svinn'], 'Gustav':
[''], 'Nora': [''], 'Emma': [''], 'Olivia': ['svinn'], 'Ella': [''], 'Sofie':
[''], 'Leah': [''], 'Frida': ['gluten', 'fisk'], 'Iben': [''], 'Sara':
['nøtter'], 'Maja': [''], 'Ingrid': [''], 'Alma': ['kjøtt', 'laktose'], 'Selma':
['svinn', 'melk'], 'Emilie': [''], 'Ada': [''], 'Astrid': [''], 'Hedda':
['kjøtt'], 'Anna': [''], 'Amalie': [''], 'Ellinor': ['gluten'], 'Aurora': [''],
'Hedvig': [''], 'Eva': [''], 'Mia': ['gluten', 'nøtter']}
```

1.1.2 Eksempel 2: Allergi teller: tell hvor mange personer har en gitt allergi

I dette eksempelet ønsker vi å bruke filen til å beregne antall personer som har hver allergi, for å få en bedre ide om proporsjonen av hver type mat vi bør kjøpe. Det betyr at i motsetning til det forrige eksempelet, trenger vi ikke å ta hensyn til navnene, vi er bare interessert i allergiene i seg selv. Også her bør vi trekke ut og lagre dataene for å lette fremtidige Python-operasjoner.

- Iterer over linjene, men fokuser kun på allergiene
- Lagre allergiene i én stor liste
- Bruk "allergi_teller" vi implementerte i en tidligere video for å telle antall personer med allergi for hver allergi.

[5]:
```
# Initialiser en tom liste
alle_allergiene = []

for linje in linjer:
    # Del hver linje ved ":" for å separere navnet fra allergiene.
    delt_linje = linje.split(":")

    # Navnet er det første elementet og allergiene det andre
    streng_allergier = delt_linje[1]

    # Liste av allergier for denne personen
    liste_allergier = streng_allergier.split(", ")

    # Slå sammen de allergiene så langt med den nåværende liste av allergier
    # Vi bruker den sekvens operasjonen "+"
    alle_allergiene = alle_allergiene + liste_allergier
```

Koden nedenfor kommer fra oppgavene `mod4-del1-2-oppgaver-løsning`. Den teller antallet personer som har en gitt restriksjon.

[6]:
```
def restriksjon_teller(restriksjoner):
    """
    Telle hvor mange personer har en gitt restriksjon.

    Returnere en ordbok som teller antallet personer med en gitt
    restriksjon for hver restriksjon i listen.
    """
    # Opprette en ordbok som kommer til å telle hvor mange personer
    # har en gitt restriksjon
    teller = {}

    # Hent ut hver mat restriksjon i listen over restriksjoner
    for mat in restriksjoner:
        # Hvis vi allerede har funnet denne restriksjonen, øker vi antallet
        if mat in teller:
            teller[mat] = teller[mat] + 1
        # Ellers, oppretter vi en ny nøkkel, og initialiserer verdien til 1
        else:
            teller[mat] = 1
    return teller
```

```
teller = restriksjon_teller(alle_allergiene)

print(teller)
```

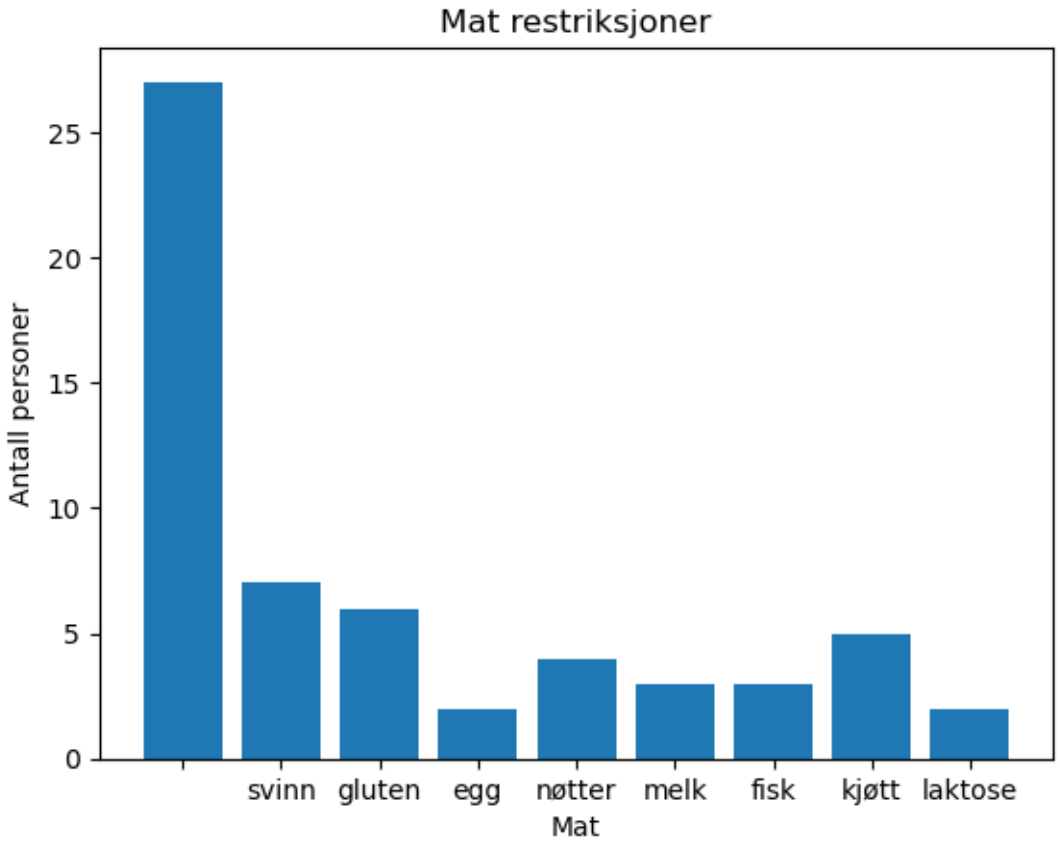
```
{'': 27, 'svinn': 7, 'gluten': 6, 'egg': 2, 'nøtter': 4, 'melk': 3, 'fisk': 3,
'kjøtt': 5, 'laktose': 2}
```

1.2 Lage en figur

Vi gjenbruker vår kode som lager en figur med antall personer som har en gitt allergi

[7]:
```
fig, ax = plt.subplots()

# Syntaksen: plt.bar(x-verdiene, y-verdiene)
ax.bar(x=list(teller.keys()), height=teller.values())
# Sett x-akse etikett
ax.set_xlabel("Mat")
# Sett y-akse etikett
ax.set_ylabel("Antall personer")
# Sett tittelen til figuren
ax.set_title("Mat restriksjoner")
# Lagre figuren
fig.savefig("mat_restriksjoner_figur")
```



1.3 Bruk data fra en CSV-fil

En CSV er et filformat som brukes til å representere strukturert tabellinformasjon, som f.eks regneark. CSV-filer ("Comma-Separated Values") er vanlige for å lagre og utveksle data mellom ulike programvareprogrammer, spesielt når det kommer til enkle datasett.

Eksempel `karakterer.csv`: en tabell med karakterene til hver student og hvert emne.

Vi oppfordrer deg til å se på filen først ved å åpne den med din vanlige regneark programvare (e.g. Office spreadsheets) for å se at det er en vanlig regneark men også en enkel teksteditor (det kan være Visual Studio Code) for å se at data egentlig er en strukturert tekst.

Merk at det finnes Python-pakker som forenkler lesing og skriving av regnearkfiler. Men i dette kurset ønsker vi å legge vekt på hvordan grunnleggende metoder også kan gjøre jobben.

1.3.1 Lese csv-filen som en vanlig tekstfil

[8]:
```
with open("karakterer.csv", mode="rt") as f:
    linjer  = f.read().split("\n")
```

1.3.2 Hent ut listen av fag

Listen av fag er den første raden, men uten den første kolonnen.

[9]:
```
# Ta den første linjen
linje0 = linjer[0]
print(linje0)

# Del strengen inn i en liste av strenger
delt_linje0 = linje0.split(";")
print(delt_linje0)

# Hent ut alt bortsett fra det første elementet
liste_fag = linjer[0].split(";")[1:]
# Det gir oss listen av fag
print(liste_fag)
```

```
;Kemi;Engelsk;Historie;Fransk;Informatikk;Litteratur;Matematikk;Biologi;Fysikk;G
ym;Filosofi;Kunst;Musikk
['', 'Kemi', 'Engelsk', 'Historie', 'Fransk', 'Informatikk', 'Litteratur',
'Matematikk', 'Biologi', 'Fysikk', 'Gym', 'Filosofi', 'Kunst', 'Musikk']
['Kemi', 'Engelsk', 'Historie', 'Fransk', 'Informatikk', 'Litteratur',
'Matematikk', 'Biologi', 'Fysikk', 'Gym', 'Filosofi', 'Kunst', 'Musikk']
```

1.3.3 Hent ut listen av student

Listen av studenter er den første kolonnen, men uten den første raden.

[10]:
```
# Opprett en tom liste av students navn
liste_navn = []

# Iterer over radene men fra den andre raden
for linje in linjer[1:]:

```

```
    # Del strengen inn i en liste av strenger
    delt_linje = linje.split(";")

    # Hent ut det første elementet
    navn = delt_linje[0]

    # Legg til navnet på slutten av listen
    liste_navn.append(navn)

print(liste_navn)
```

```
['Jakob', 'Noah', 'Emil', 'Lukas', 'Isak', 'Filip', 'Aksel', 'Theodor',
'Ludvig', 'Oskar', 'Liam', 'Johannes', 'Kasper', 'Magnus', 'Tobias', 'Henrik',
'Mathias', 'Viktor', 'Ulrik', 'Matheo', 'Adam', 'Gustav', 'Nora', 'Emma',
'Olivia', 'Ella', 'Sofie', 'Leah', 'Frida', 'Iben', 'Sara', 'Maja', 'Ingrid',
'Alma', 'Selma', 'Emilie', 'Ada', 'Astrid', 'Hedda', 'Anna', 'Amalie',
'Ellinor', 'Aurora', 'Hedvig', 'Eva', 'Mia', '']
```

1.3.4 Skriv en ny fil med bare informasjon om en gitt student

La oss si at vi vil nå sende en epost til hver student men karakterene sine, men uten de andre karakterene. Da må vi opprette en ny fil, som har bare informasjon om en gitt student.

[11]:
```
student = "Jakob"

def finn_student_linje(linjer, navn):
    """
    Finn linjen i filen som tilsvarer studenten.

    Hvis studenten ikke er i filen, blir "" returnert.
    """
    student_linje = ""
    for linje in linjer:
        if navn in linje:
            student_linje = linje
    return student_linje

# Skriv linjen (+ linjen med fag) i en ny fil
with open(f"{student}_karakterer.csv", mode="wt") as f:
    # Den første linjen med fag er den samme som før
    f.writelines(linjer[0] + "\n")

    # Og vi bare skrive linjen som tilsvarer studenten
    student_linje = finn_student_linje(linjer, student)
    f.writelines(student_linje + "\n")
```

1.3.5 Lag en figur

Vi gjenbruker vår kode fra `mod4-del2-figurer` som lager en figur med karakteren for hvert fag.

[12]:
```python
# Listen av fag er den første raden, men uten den første kolonnen.
liste_fag = linjer[0].split(";")[1:]
liste_karakterer = student_linje.split(";")[1:]

# Hver karakter er en streng av en flyttall, ikke en flyttall!
# Vi må konvertere først!
for i in range(len(liste_karakterer)):
    liste_karakterer[i] = float(liste_karakterer[i])

# Nå kan vi gjenbruke vår kode som lager en figur
# (Vi trenger en større figur, så derfor bruker vi "figsize")
fig, ax = plt.subplots(figsize=(15, 5))
ax.bar(liste_fag, liste_karakterer)
ax.set_xlabel("Fag")
ax.set_ylabel("Karakter")
ax.set_title(f"{student} karakterer")
fig.savefig(f"{student}_karakterer")
```