UNIVERSITETET I BERGEN

TEKSTFILER OG DATAREPRESENTASJON

INF100

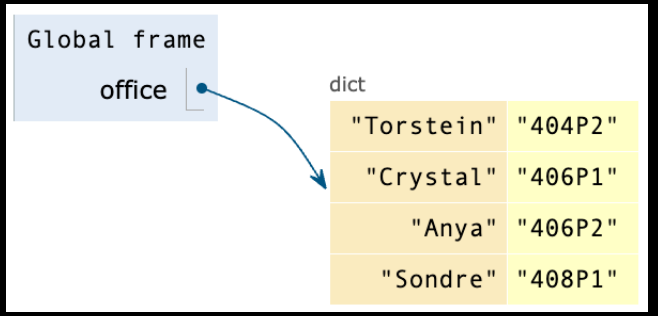
VÅR 2025

Torstein Strømme

OPPSLAGSVERK

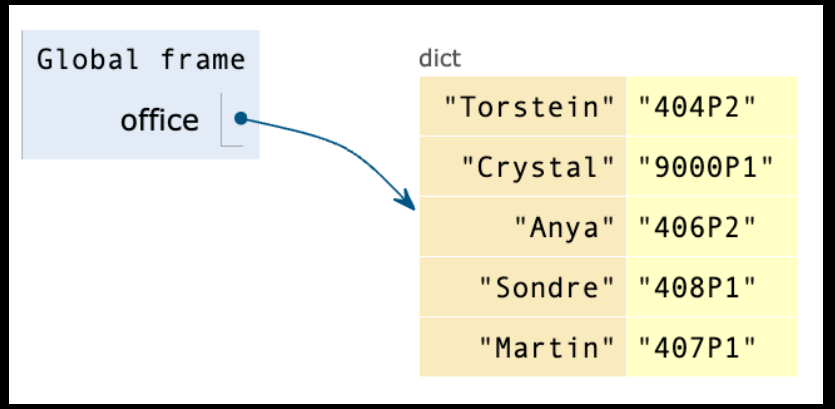
OPPSLAGSVERK

```
# Et oppslagsverk er som en telefonkatalog:
# man kan slå opp på en nøkkel, og få vite dens verdi

office = {
    'Torstein': '404P2',
    'Crystal': '406P1',
    'Anya': '406P2',
    'Sondre': '408P1',
}

print('Torstein sitt kontor er', office['Torstein'])
```



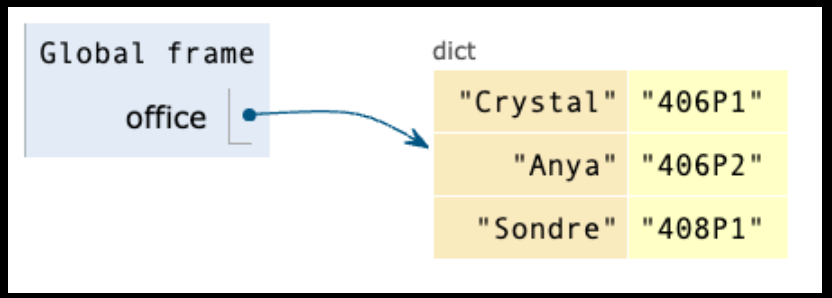
OPPSLAGSVERK

```
office = {
    'Torstein': '404P2',
    'Crystal': '406P1',
    'Anya': '406P2',
    'Sondre': '408P1',
}

office['Martin'] = '407P1' # Legg til ny nøkkel/verdi
office['Crystal'] = '9000P1' # Crystal bytter kontor
```



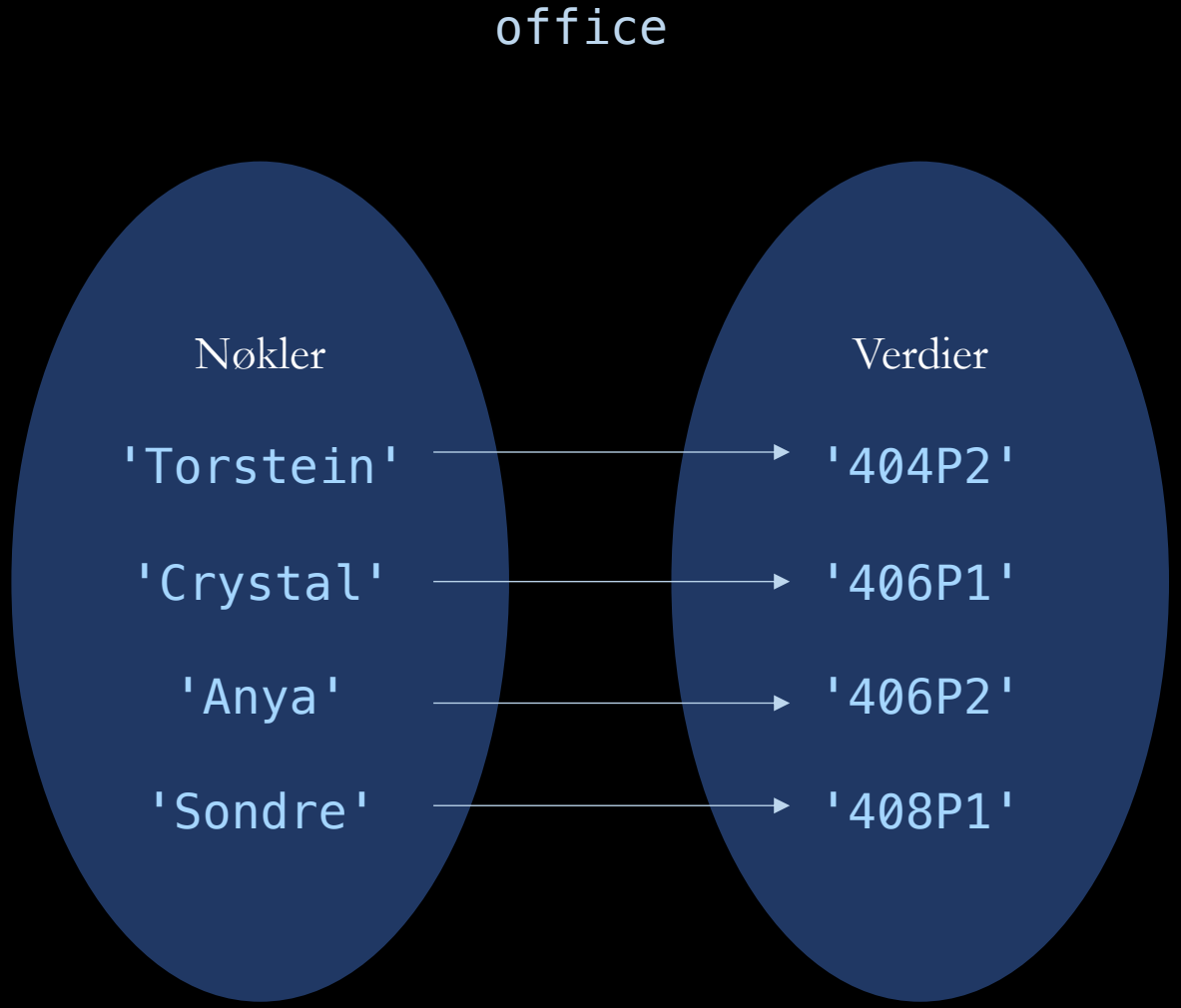
OPPSLAGSVERK

```
office = {
    'Torstein': '404P2',
    'Crystal': '406P1',
    'Anya': '406P2',
    'Sondre': '408P1',
}

office.pop('Torstein') # Fjern nøkkel
```



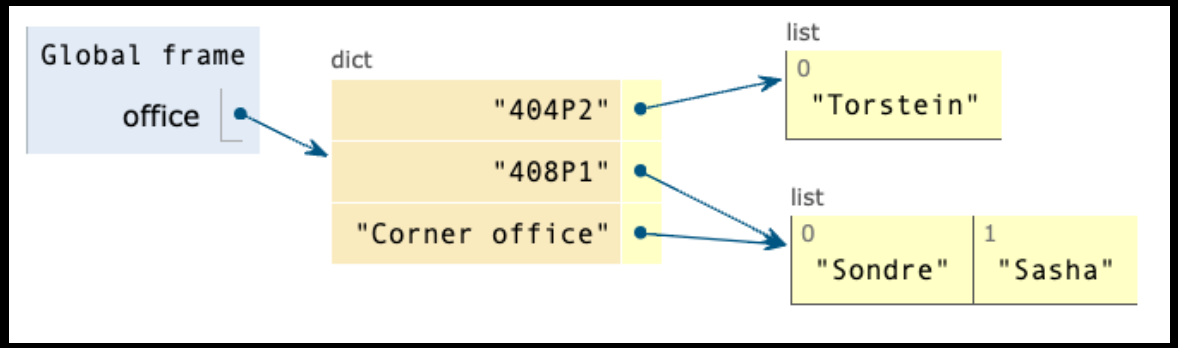
OPPSLAGSVERK

```
office = {
    'Torstein': '404P2',
    'Crystal': '406P1',
    'Anya': '406P2',
    'Sondre': '408P1',
}
```



OPPSLAGSVERK

```
office = {
    '404P2': ['Torstein'],
    '408P1': ['Sondre', 'Sasha'],
}

office['Corner office'] = office['408P1'] # alias
```



OPPSLAGSVERK vs FUNKSJON

Oppslagsverk:

- Kan utvides med nye nøkler/verdier dynamisk
- Kan endre verdi til eksisterende nøkler
- Raskere

```
office = {
    'Torstein': '404P2',
    'Crystal': '406P1',
    'Anya': '406P2',
    'Sondre': '408P1',
}


print('Torsteins kontor:',
      office['Torstein'])
```

```
def office(key):
    if key == 'Torstein': return '404P2'
    elif key == 'Anya': return '406P2'
    elif key == 'Crystal': return '406P1'
    elif key == 'Sondre': return '408P1'


print('Torsteins kontor:', office('Torstein'))
```

OPPSLAGSVERK vs LISTE MED TUPLER

Oppslagsverk:

- Maksimum én verdi per nøkkel
- Rekkefølge betyr ingenting
- Raskere

```
office = {
    'Torstein': '404P2',
    'Crystal': '406P1',
    'Anya': '406P2',
    'Sondre': '408P1',
}


print('Torsteins kontor:',
      office['Torstein'])
```

```
office = [
    ('Torstein', '404P2'),
    ('Anya', '406P2'),
    ('Crystal', '408P1'),
    ('Sondre', '408P1'),
]

def get_key(key, li):
    for k, value in li:
        if k == key:
            return value

print('Torsteins kontor:',
      get_key('Torstein', office))
```

OPPSLAGSVERK vs VARIABLER

Oppslagsverk:

- Ingen "forbudte ord"
- Kan bruke andre typer som nøkler (int/float/bool etc.)

```
office = {
    'Torstein': '404P2',
    'Crystal': '406P1',
    'Anya': '406P2',
    'Sondre': '408P1',
}

print('Torsteins kontor:',
      office['Torstein'])
```

```
Torstein = '404P2'
Anya = '406P2'
Crystal = '408P1'
Sondre = '408P1'

print('Torsteins kontor:', Torstein)
```

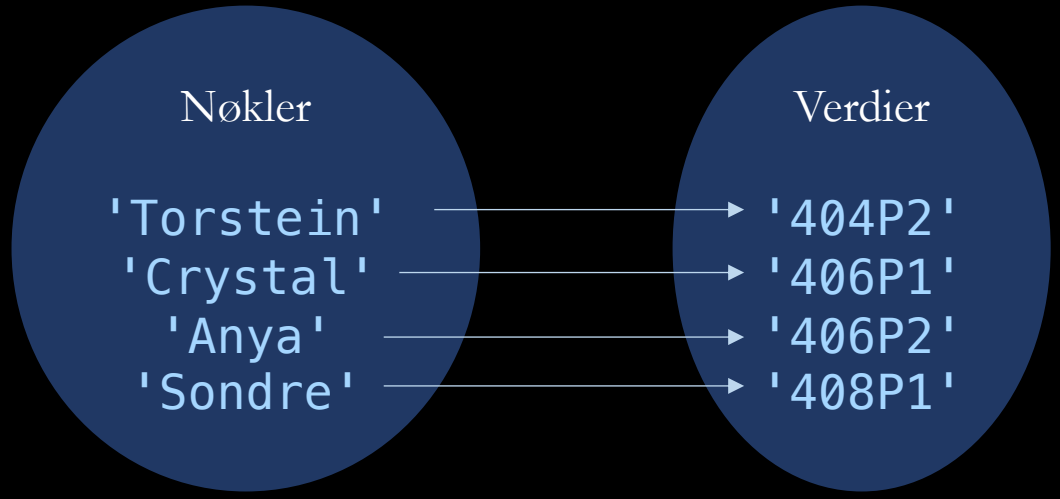
Variabler ER faktisk nøkler i et oppslagsverk!

OPPSLAGSVERK

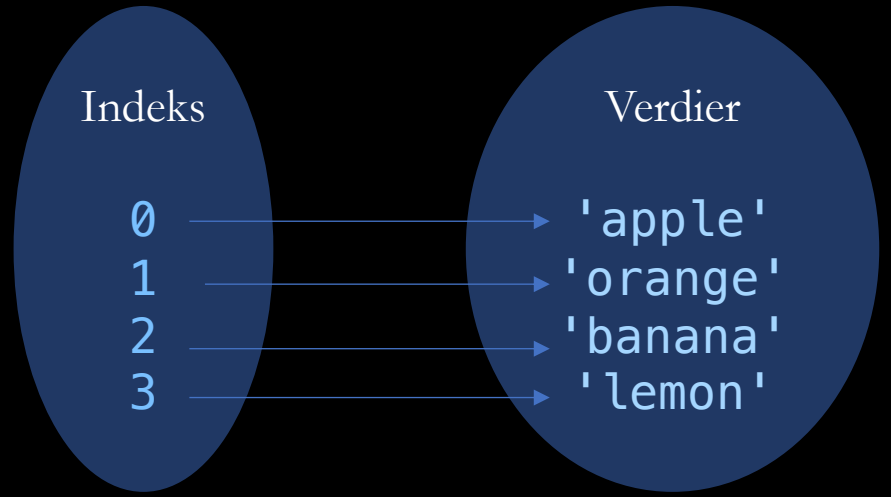
- En samling med “variabler”
- Kan bruke ulike typer som nøkler
  - int, float, str, bool, etc.
  - (men ikke noe som muterbart)
- Oppslagsverk er muterbare

OPPSLAGSVERK vs LISTER

```
office = {
    'Torstein': '404P2',
    'Crystal': '406P1',
    'Anya': '406P2',
    'Sondre': '408P1',
}
```

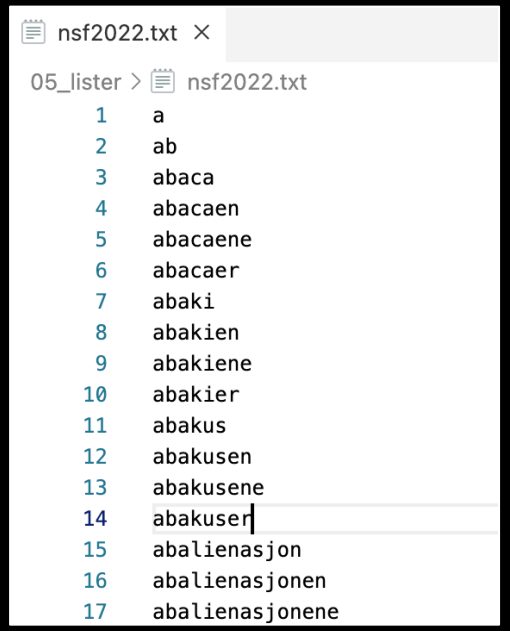



```
fruits = [
    'apple',
    'orange',
    'banana',
    'lemon',
]
```



FINN VANLIGSTE BOKSTAV

- Gitt en tekstfil: skriv ut hvilke bokstaver den inneholder og hvor mange av hver bokstav




```
'a'  -> 619290
'\n' -> 879902
'b'  -> 180822
'c'  -> 12238
'e'  -> 1587179
'n'  -> 934403
'r'  -> 836100
'k'  -> 471437
...
```

JSON

En streng som oppretter oppslagsverk og lister med data

https://dogapi.dog/api/v2/facts?limit=3

{"data":[{"id":"8fa94033-0160-4f27-a6b0-009e77bef24c","type":"fact","attributes":{"body":"Dogs have lived with humans for over 14,000 years. Cats have lived with people for only 7,000 years."}},{"id":"c898f9c5-0172-4a18-b67c-227b27ab4045","type":"fact","attributes":{"body":"More than one in three U.S. families owns a dog."}},{"id":"c6fe1512-611d-44b1-972a-e44d15967a7e","type":"fact","attributes":{"body":"The names of 77 ancient Egyptian dogs have been recorded. The names refer to color and character, such as Blackie, Ebony, Good Herdsman, Reliable, and Brave One."}}]}

```
{
  "data": [
    {
      "id": "8fa94033-0160-4f27-a6b0-009e77bef24c",
      "type": "fact",
      "attributes": {
        "body": "Dogs have lived with humans for over 14,000 years."
      }
    },
    {
      "id": "c898f9c5-0172-4a18-b67c-227b27ab4045",
      "type": "fact",
      "attributes": {
        "body": "More than one in three U.S. families owns a dog."
      }
    },
    {
      "id": "c6fe1512-611d-44b1-972a-e44d15967a7e",
      "type": "fact",
      "attributes": {
        "body": "The names of 77 ancient Egyptian dogs have been recorded."
      }
    }
  ]
}
```

JSON

- Strukurert data lagret som en streng/tekstfil
- Brukes ofte for kommunikasjon
- Støtter de vanligste datatypene skrevet (nesten) akkurat som i Python:
  - dict
  - list
  - str
  - int
  - float
  - bool
  - None

Men støtter ikke «trailing comma»

Men du må bruke " ", ikke ' '

Noen få unntak (NaN og infinity)

Men bruker lowercase

Men kaller den for «null»

Alle nøkler i JSON må være strenger

JSON støtter ikke kommentarer

JSON

Fra streng til data

```python
import json

json_string = '{"foo": 42, "bar": 95}'
data = json.loads(json_string)

print(data['foo']) # printer 42
```

JSON

Fra data til streng

```python
import json

data = {'foo': 42, 'bar': 95}
json_string = json.dumps(data)

print(repr(json_string)) # '{"foo": 42, "bar": 95}'
```

- En CSV-fil: ren tekst som representerer tabell-data

```
personnummer,medisin,startdato
01010111111,kake,2022-08-22
22020222222,fluor,2022-01-11
01010111111,tran,2022-06-01
```


	A	B	C
1	personnummer	medisin	startdato
2	1010111111	vitaminbjørner	22/08/2022
3	22020222222	fluortabletter	11/01/2022
4	1010111111	tran	01/06/2022

- En CSV-fil: ren tekst som representerer tabell-data

```
personnummer,medisin,startdato
01010111111,kake,2022-08-22
22020222222,fluor,2022-01-11
01010111111,tran,2022-06-01
```

- I Python: tabell-data representert som streng

```
table_string = '''
personnummer,medisin,startdato
01010111111,kake,2022-08-22
22020222222,fluor,2022-01-11
01010111111,tran,2022-06-01
'''
```

- En CSV-fil: ren tekst som representerer tabell-data

```
personnummer,medisin,startdato
01010111111,kake,2022-08-22
22020222222,fluor,2022-01-11
01010111111,tran,2022-06-01
```

- I Python: tabell-data representert som liste av strenger

```python
table_string_list = [
    'personnummer,medisin,startdato',
    '01010111111,kake,2022-08-22',
    '22020222222,fluor,2022-01-11',
    '01010111111,tran,2022-06-01',
]
```

- En CSV-fil: ren tekst som representerer tabell-data

```
personnummer,medisin,startdato
01010111111,kake,2022-08-22
22020222222,fluor,2022-01-11
01010111111,tran,2022-06-01
```

- I Python: tabell-data representert som 2D-liste

```python
table_2d_list = [
    ['personnummer', 'medisin', 'startdato'],
    ['01010111111', 'kake', '2022-08-22'],
    ['22020222222', 'fluor', '2022-01-11'],
    ['01010111111', 'tran', '2022-06-01'],
]
```

- En CSV-fil: ren tekst som representerer tabell-data

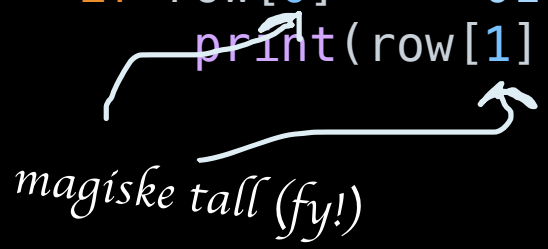
```
personnummer,medisin,startdato
01010111111,kake,2022-08-22
22020222222,fluor,2022-01-11
01010111111,tran,2022-06-01
```

- I Python: tabell-data representert som liste av oppslagsverk

```
table_dict_list = [
    {'personnummer ': '01010111111', 'medisin': 'kake', 'startdato': '2022-08-22'},
    {'personnummer ': '22020222222', 'medisin': 'fluor', 'startdato': '2022-01-11'},
    {'personnummer ': '01010111111', 'medisin': 'tran', 'startdato': '2022-06-01'}
]
```

- Liste av lister: hvilke medisiner er registrert på person 01010111111?

```
table_2d_list = [
    ['personnummer', 'medisin', 'startdato'],
    ['01010111111', 'kake', '2022-08-22'],
    ['22020222222', 'fluor', '2022-01-11'],
    ['01010111111', 'tran', '2022-06-01'],
]

for row in table_2d_list:
    if row[0] == '01010111111':
        print(row[1])
```



- Liste av oppslagsverk: hvilke medisiner er registrert på person 01010111111?

```
table_dict_list = [
    {'personnummer': '01010111111', 'medisin': 'kake', 'startdato': '2022-08-22'},
    {'personnummer': '22020222222', 'medisin': 'fluor', 'startdato': '2022-01-11'},
    {'personnummer': '01010111111', 'medisin': 'tran', 'startdato': '2022-06-01'}
]


for row in table_dict_list:
    if row['personnummer'] == '01010111111':
        print(row['medisin'])
```

FRA CSV TIL LISTE AV OPPSLAGSVERK

- Fra CSV til streng

```
from pathlib import Path
table_string = Path('org.csv').read_text()
```

```
table_string = '''
personnummer,medisin,startdato
01010111111,kake,2022-08-22
22020222222,fluor,2022-01-11
01010111111,tran,2022-06-01
'''
```

- Fra CSV til streng
- **Fra streng til liste av linjer**
  - Øverste linje er header
  - Øvrige linjer er data

```
from pathlib import Path
table_string = Path('org.csv').read_text()

lines = table_string.splitlines()
header_line = lines[0]
data_lines = lines[1:]
```

```
lines = [
    'personnummer,medisin,startdato',
    '01010111111,kake,2022-08-22',
    '22020222222,fluor,2022-01-11',
    '01010111111,tran,2022-06-01',
]
```

```
data_lines = [
    '01010111111,kake,2022-08-22',
    '22020222222,fluor,2022-01-11',
    '01010111111,tran,2022-06-01',
]
```

```
header_line = 'personnumer,medisin,startdato'
```

- Fra CSV til streng
- **Fra streng til liste av linjer**
  - Øverste linje er header
  - Øvrige linjer er data

```
from pathlib import Path
table_string = Path('org.csv').read_text()

lines = table_string.splitlines()
header_line = lines[0]
data_lines = lines[1:]
headers = header_line.split(',')
```

```
lines = [
    'personnummer,medisin,startdato',
    '01010111111,kake,2022-08-22',
    '22020222222,fluor,2022-01-11',
    '01010111111,tran,2022-06-01',
]
```

```
data_lines = [
    '01010111111,kake,2022-08-22',
    '22020222222,fluor,2022-01-11',
    '01010111111,tran,2022-06-01',
]
```

```
header_line = 'personnumer,medisin,startdato'
```

```
headers = ['personnummer', 'medisin', 'startdato']
```

- Fra CSV til streng
- Fra streng til liste av linjer
  - Øverste linje er header
  - Øvrige linjer er data

- **Fyll tabell med rader**

```
table_dict = [
    {},
    {},
    {},
]
```

```
from pathlib import Path
table_string = Path('org.csv').read_text()

lines = table_string.splitlines()
header_line = lines[0]
data_lines = lines[1:]
headers = header_line.split(',')

table_dict = []
for line in data_lines:
    row_dict = {}
    # begynn fyll opp row_dict her ...

    # ... ferdig fylt opp row_dict her
    table_dict.append(row_dict)
```

- Fra CSV til streng
- Fra streng til liste av linjer
  - Øverste linje er header
  - Øvrige linjer er data
- Fyll tabell med rader
- **Fra linje til oppslagslagsverk**

```
table_dict = []
for line in data_lines:
    row_dict = {}
    # begynn fyll opp row_dict her ...

    cells = line.split(',')
    for i in range(number_of_cols):
        header = headers[i]
        value = cells[i]
        row_dict[header] = value

    # ... ferdig fylt opp row_dict her
    table_dict.append(row_dict)
```

```
line = '01010111111,kake,2022-08-22'
```

```
cells = ['01010111111', 'kake', '2022-08-22']
```

- Fra CSV til streng
- Fra streng til liste av linjer
  - Øverste linje er header
  - Øvrige linjer er data

- Fyll tabell med rader
- **Fra linje til oppslagslagsverk**

```
number_of_cols = len(headers)

table_dict = []
for line in data_lines:
    row_dict = {}
    # begynn fyll opp row_dict her ...

    cells = line.split(',')
    for i in range(number_of_cols):
        header = headers[i]
        value = cells[i]
        row_dict[header] = value

    # ... ferdig fylt opp row_dict her
    table_dict.append(row_dict)
```

```
number_of_cols = 3
```

```
i = 0
```

```
header = 'personnummer'
```

```
value = '01010111111'
```

```
row_dict['personnummer'] = '01010111111'
```

- Fra CSV til streng
- Fra streng til liste av linjer
  - Øverste linje er header
  - Øvrige linjer er data

- Fyll tabell med rader
- **Fra linje til oppslagslagsverk**

```
number_of_cols = len(headers)

table_dict = []
for line in data_lines:
    row_dict = {}
    # begynn fyll opp row_dict her ...

    cells = line.split(',')
    for i in range(number_of_cols):
        header = headers[i]
        value = cells[i]
        row_dict[header] = value

    # ... ferdig fylt opp row_dict her
    table_dict.append(row_dict)
```

```
number_of_cols = 3
```

```
i = 1
```

```
header = 'medisin'
```

```
value = 'kake'
```

```
row_dict['medisin'] = 'kake'
```

- Fra CSV til streng
- Fra streng til liste av linjer
  - Øverste linje er header
  - Øvrige linjer er data

- Fyll tabell med rader
- **Fra linje til oppslagslagsverk**

```
number_of_cols = len(headers)

table_dict = []
for line in data_lines:
    row_dict = {}
    # begynn fyll opp row_dict her ...

    cells = line.split(',')
    for i in range(number_of_cols):
        header = headers[i]
        value = cells[i]
        row_dict[header] = value

    # ... ferdig fylt opp row_dict her
    table_dict.append(row_dict)
```

```
number_of_cols = 3
```

```
i = 2
```

```
header = 'startdato'
```

```
value = '2022-08-22'
```

```
row_dict['startdato '] = '2022-08-22 '
```

- Fra CSV til streng
- Fra streng til liste av linjer
  - Øverste linje er header
  - Øvrige linjer er data
- Fyll tabell med rader
- **Fra linje til oppslagslagsverk**

```
number_of_cols = len(headers)

table_dict = []
for line in data_lines:
    row_dict = {}
    # begynn fyll opp row_dict her ...

    cells = line.split(',')
    for i in range(number_of_cols):
        header = headers[i]
        value = cells[i]
        row_dict[header] = value

    # ... ferdig fylt opp row_dict her
    table_dict.append(row_dict)
```

```
row_dict = {'personnummer': '01010111111', 'medisin ': 'kake', 'startdato': '2022-08-22'}
```

- Fra CSV til streng
- Fra streng til liste av linjer
  - Øverste linje er header
  - Øvrige linjer er data
- Fyll tabell med rader
- **Fra linje til oppslagslagsverk**

```
number_of_cols = len(headers)

table_dict = []
for line in data_lines:
    row_dict = {}
    # begynn fyll opp row_dict her ...

    cells = line.split(',')
    for i in range(number_of_cols):
        header = headers[i]
        value = cells[i]
        row_dict[header] = value

    # ... ferdig fylt opp row_dict her
    table_dict.append(row_dict)
```

```
table_dict_list = [
    {'personnummer ': '01010111111', 'medisin': 'kake', 'startdato': '2022-08-22'},
    {'personnummer ': '22020222222', 'medisin': 'fluor', 'startdato': '2022-01-11'},
    {'personnummer ': '01010111111', 'medisin': 'tran', 'startdato': '2022-06-01'},
]
```

```
from pathlib import Path

table_as_csv = Path('filename.csv').read_text(encoding='utf-8')
lines = table_as_csv.splitlines()
headers = lines[0].split(',')
data_lines = lines[1:]
number_of_cols = len(headers)

table_dict = []
for line in data_lines:
    row_dict = {}
    cells = line.split(',')
    for i in range(len(headers)):
        header = headers[i]
        value = cells[i]
        row_dict[header] = value
    table_dict.append(row_dict)
```

NOEN MÅ HA GJORT DETTE FØR

Alternativ A

```
from pathlib import Path
from csv import DictReader


with Path('filename.csv').open('rt', encoding='utf-8', newline='') as file:
    reader = DictReader(file, delimiter=',', quotechar='"')

    headers = reader.fieldnames
    data = list(reader)
```

Alternativ B

```python
from pathlib import Path
from csv import DictReader
from io import StringIO

file_string = Path('filename.csv').read_text(encoding='utf-8')
reader = DictReader(StringIO(file_string), delimiter=',', quotechar='"')

headers = reader.fieldnames
data = list(reader)
```

Fungerer også hvis du har en CSV-streng som ikke kommer fra en fil
(men er litt langsommere med store filer)

Skríve til streng

```
from pathlib import Path
from csv import DictWriter
from io import StringIO

headers = ...
data = ...

output = StringIO()
writer = DictWriter(output, fieldnames=headers, delimiter=',', quotechar='"')
writer.writeheader()
writer.writerows(data)

output_string = output.getvalue()
output.close()
```

Skrive til fil

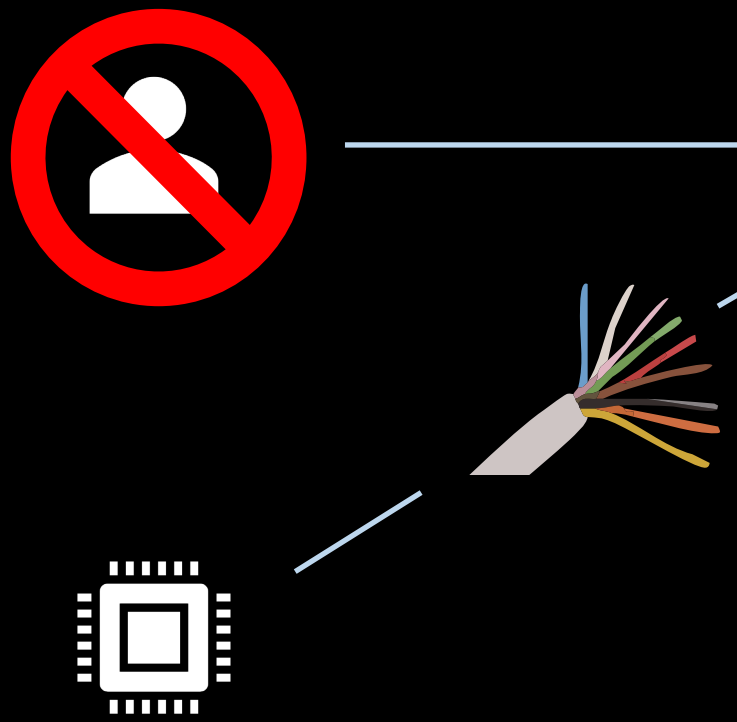
```
from pathlib import Path
from csv import DictWriter


headers = ...
data = ...

with Path('newfile.csv').open('wt', encoding='utf-8', newline='') as file:
    writer = DictWriter(file, fieldnames=headers, delimiter=',', quotechar='"')
    writer.writeheader()
    writer.writerows(data)
```

TEKSTFILER

HISTORIETIME: EN COMPUTER

1954, NACA

HISTORIETIME: EN COMPUTER



TTY

ASCII-standarden angir hva TTY skal gjøre for et gitt signal

ASCII

BEL: Ring med bjellen

LF: line feed (flytt papiret én rad ned)

CR: carriage return (flytt skrivehodet tilbake til starten av arket

SP: Space (flytt skrivehodet én plass frem)

ASCII (1977/1986)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

BS: Backspace (flytt skrivehodet én plass tilbake)

DEL: Slett forrige tegn

TEKSTFIL

- Mange filformater er basert på «ren tekst»
  (innholdet i filen er bare en streng)
- .txt
- .py/.js/.java/.cpp/.bat/.sh
- .csv/.json/.xml/.yaml/
- .html/.css/.tex/.md
- .svg
- …

Andre filformater er *ikke* basert på ren tekst:

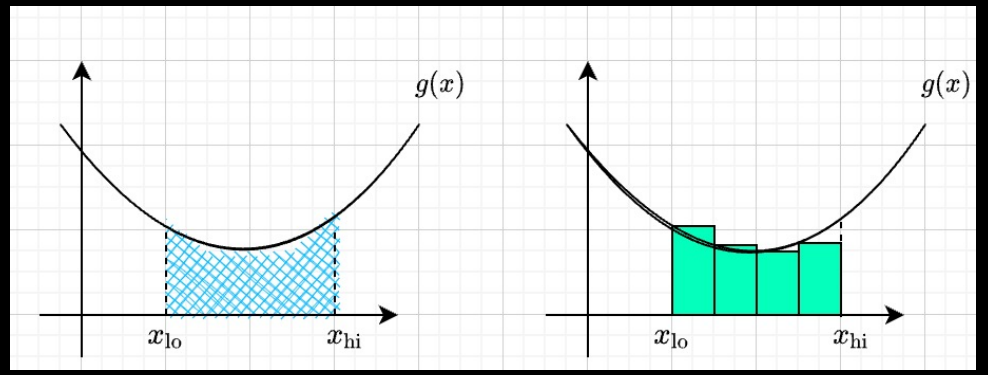
- .docx/.rtf/.pdf
- .xlsx
- .zip
- .mp3/.acc/.mp4
- .png/.jpeg/.gif

TEKSTFIL

- Mange filformater er basert på «ren tekst»
- Filer basert på ren tekst kan håndteres i enhver teksteditor (VSCode, Notepad, TextEdit, vim etc)

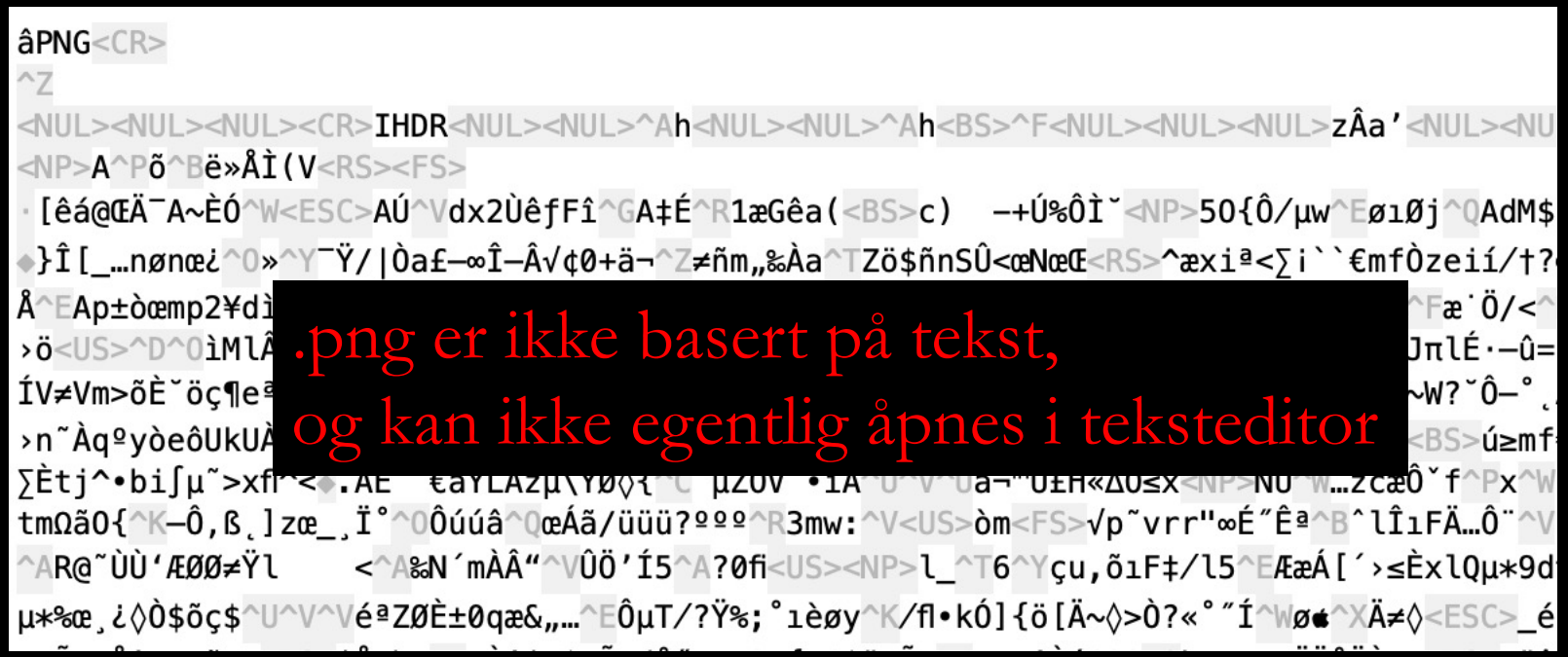
area_under_graph.svg

…<path d="M 10.5 190.5 L 255.95 190.5" fill="none" stroke="rgb(0, 0, 0)" stroke-width="1.5" stroke-miterlimit="10" pointer-events="stroke"/><path d="M 263.82 190.5 L 253.32 195.75 L 255.95 190.5 L 253.32 185.25 Z" fill="rgb(0, 0, 0)" stroke="rgb(0, 0, 0)" stroke-width="1.5" stroke-miterlimit="10" pointer-events="all"/><path d="M 40.5 220.5 L 40.5 20.05" fill="none" stroke="rgb(0, 0, 0)" stroke-width="1.5" stroke-miterlimit="10" pointer-events="stroke"/>…



TEKSTFIL

- Mange filformater er basert på «ren tekst»
- Filer basert på ren tekst kan håndteres i enhver teksteditor (VSCode, Notepad, TextEdit, vim etc)



.png er ikke basert på tekst,
og kan ikke egentlig åpnes i teksteditor

python-circle.png

TEKSTFIL

- Mange filformater er basert på «ren tekst»
- Filer basert på ren tekst kan håndteres i enhver teksteditor (VSCode, Notepad, TextEdit, vim etc)

- Det er dessverre forskjell på rene tekst-filer også ☹
- Forskjellen går på tvers av fil-endelse, og handler om valg av *koding*
- Koding: regler for hvordan skal 0'ere og 1'ere i datamaskinen tolkes som skriftsymboler

ASCII

- 127 ulike tegn
- Gammel standard
- Det eneste som alltid fungerer

- Mangler æøåéü£†ƒ💩

Binary	Oct	Dec	Hex	Glyph 1963	Glyph 1965	Glyph 1967
010 0000	040	32	20	space		
010 0001	041	33	21	!		
010 0010	042	34	22	"		
010 0011	043					
010 0100	044					
010 0101	045					
010 0110	046					
010 0111	047					
010 1000	050					
010 1001	051					
010 1010	052					
010 1011	053					
010 1100	054					
100 0001	101	65	41	A		
100 0010	102	66	42	B		
100 0011	103	67	43	C		
100 0100	104	68	44	D		
100 0101	1					
100 0110	1					
100 0111	1					
100 1000	1					
100 1001	1					
100 1010	1					
100 1011	1					
100 1100	1					
110 0001	141	97	61			a
110 0010	142	98	62			b
110 0011	143	99	63			c
110 0100	144	100	64			d
110 0101	145	101	65			e
110 0110	146	102	66			f
110 0111	147	103	67			g
110 1000	150	104	68			h
110 1001	151	105	69			i
110 1010	152	106	6A			j
110 1011	153	107	6B			k
110 1100	154	108	6C			l

https://en.wikipedia.org/wiki/ASCII#Printable_characters

UTF-8

- Alle unicode –symboler

(144697 ulike tegn akkurat nå)

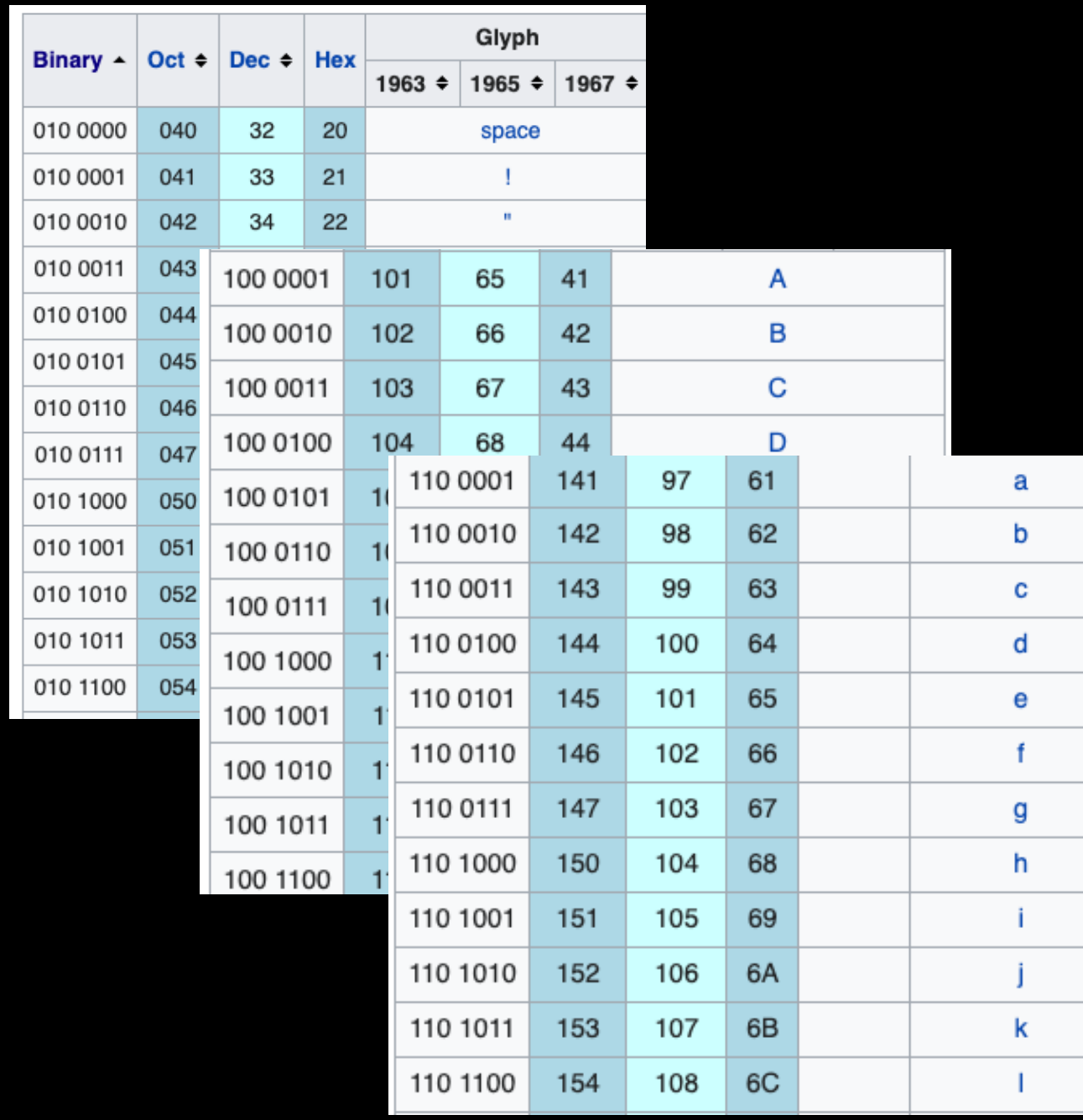
- Bakoverkompatibel med ASCII

- Unicode-verdien til et tegn:

```
ord('A')
```

- Tegnet til en gitt unicode-verdi:

```
chr(105)
```



Binary	Oct	Dec	Hex	Glyph 1963	Glyph 1965	Glyph 1967
010 0000	040	32	20	space		
010 0001	041	33	21	!		
010 0010	042	34	22	"		
010 0011	043					
010 0100	044					
010 0101	045					
010 0110	046					
010 0111	047					
010 1000	050					
010 1001	051					
010 1010	052					
010 1011	053					
010 1100	054					

100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	1			
100 0110	1			
100 0111	1			
100 1000	1			
100 1001	1			
100 1010	1			
100 1011	1			
100 1100	1			

110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l

DATA SOM REN TEKST

CSV

Tabelldata (≤2D)

```
personnummer,medisin,startdato,sluttdato
01010111111,vitaminbjørner,2022-08-22,2022-11-28
22020222222,fluortabletter,2022-01-11,2022-03-03
01010111111,tran,2022-06-01,2022-06-30
```

JSON

Svært fleksibel

```
{"data": [
   {"id": "8fa94033", "type": "fact", "attributes":
      {"body": "Dogs have lived with humans for 14,000 years."}},
   {"id": "c898f9c5", "type": "fact", "attributes":
      {"body": "More than one in three U.S. families owns a dog."}},
   {"id": "c6fe1512", "type": "fact", "attributes":
      {"body": "77 ancient Egyptian dog names are known."}}
   ]
}
```

RECAP: STRENGER ↔ LISTER

```
s = 'Marshall,Rubble,Chase,Rocky,Zuma,Sky'
a = s.split(',')

print(a)
```

```
['Marshall', 'Rubble', 'Chase', 'Rocky', 'Zuma', 'Sky']
```

RECAP: STRENGER ↔ LISTER

```
a = ['Marshall', 'Rubble', 'Chase', 'Rocky', 'Zuma', 'Sky']
s = ';'.join(a)

print(s)
```

```
Marshall;Rubble;Chase;Rocky;Zuma;Sky
```

RECAP: STRENGER ↔ LISTER

```
s = '''\
Marshall,Rubble,Chase
Rocky,Zuma,Sky
'''
a = s.splitlines()

print(a)
```

```
['Marshall,Rubble,Chase', 'Rocky,Zuma,Sky']
```

REFORMATTERING AV CSV

- Du har en CSV-fil med data, men det noen steder mangler informasjon

```
personnummer,medisin,startdato,sluttdato
01010111111,vitaminbjørner,2022-08-22,2022-11-28
22020222222,fluortabletter,2022-01-11,
01010111111,tran,2022-06-01,2022-06-30
```

- Produser en CSV-fil hvor manglende data får en gitt verdi

```
personnummer,medisin,startdato,sluttdato
01010111111,vitaminbjørner,2022-08-22,2022-11-28
22020222222,fluortabletter,2022-01-11,2024-03-08
01010111111,tran,2022-06-01,2022-06-30
```

REFORMATTERING AV CSV

- Du har en CSV-fil med data, men det er på feil format

```
personnummer,medisin,startdato,sluttdato
01010111111,vitaminbjørner,2022-08-22,2022-11-28
22020222222,fluortabletter,2022-01-11,2022-03-03
01010111111,tran,2022-06-01,2022-06-30
```

- Produser en CSV-fil hvor data er på ønsket format

```
personnummer,medisin,dato,endring
01010111111,vitaminbjørner,2022-08-22,1
01010111111,vitaminbjørner,2022-11-28,0
22020222222,fluortabletter,2022-01-11,1
22020222222,fluortabletter,2022-03-03,0
01010111111,tran,2022-06-01,1
01010111111,tran,2022-06-30,0
```