TEKSTFILER OG CSV

INF100

VÅR 2026

Crystal Chang Din

Med bidrag fra: Torstein Strømme

HISTORIETIME: EN COMPUTER

When the Computer Wore a Skirt

Computer at her work with microscope and the Friden calculating machine. (LRC-1952-B701_P-74753)
NASA

1952

https://www.nasa.gov/history/langleys-computers-1935-1970/

Dorothy Vaughan

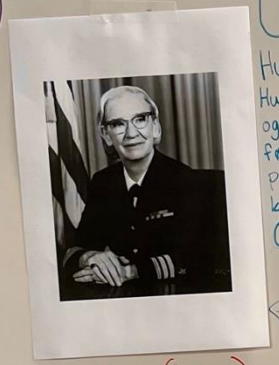
Katherine Johnson

Mary W. Jackson

Kvinnelige Pionerer

Ada Lovelace (1815-1852)

- Hun regnes som verdens første dataprogrammerer. På 1840-tallet innså hun at Charles Babbages "analytiske maskin" kunne brukes til mer enn bare kalkulasjoner, og skrev den første algoritmen beregnet for en maskin.



Grace Hopper (1906-1992)

Hun er kjent som "Queen of Code". Hun var en amerikansk informatiker og kontreadmiral som utviklet den første kompilatoren (en oversetter fra programmeringsspråk til maskinkode) og var sentral i utvikling av Cobol.

Hedy Lamarr (1914-2000)

Hun var en skuespiller og en oppfinner som, under andre verdenskrig, utviklet en frekvenshopp-teknologi. Denne teknologien la grunnlag for moderne trådløs kommunikasjon som Wifi, Bluetooth og GPS.

Margaret Hamilton (1936)

Hun var en matematikar som ledet programvareteamet MIT som utviklet den "ombord-programvaren" som de brukt i Apollo-månefordene

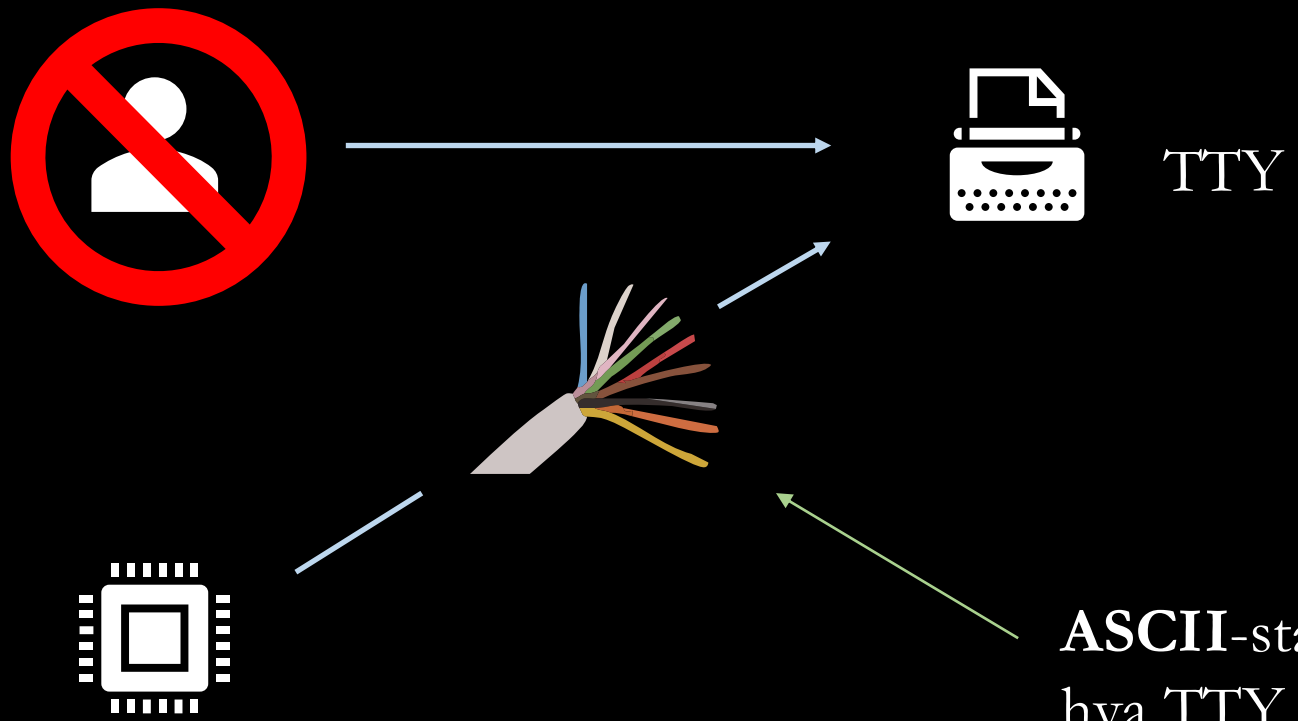
ECMA-kvinnene (1946)

6 kvinner (Fran Bilas, Jean Bartik, Ruth Lichterman, Kay McNulty, Betty Snyder og Marlyn Wescoff). De programmerte den første hel-elektroniske, programmerbare datamaskinen, ENIAC. De utviklet de første metodene for programmering. Arbeidet deres la grunnlaget for moderne programmering og viste at programmering var en egen, avansert del av teknologien.

Katherine Johnson (1918-2020)

Hun var en NASA-matematiker (kalt en "menneskelig datamaskin") hvis beregninger var avgjørende for suksessen til de første amerikanske bemannede romfordene.

HISTORIETIME: EN COMPUTER



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

ASCII

BEL: Ring med bjellen

LF: line feed (flytt papiret én rad opp)

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

ASCII

- Standard ASCII: 128 tegn (0–127)
  Utvidet ASCII: 256 tegn (0–255)
- 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)

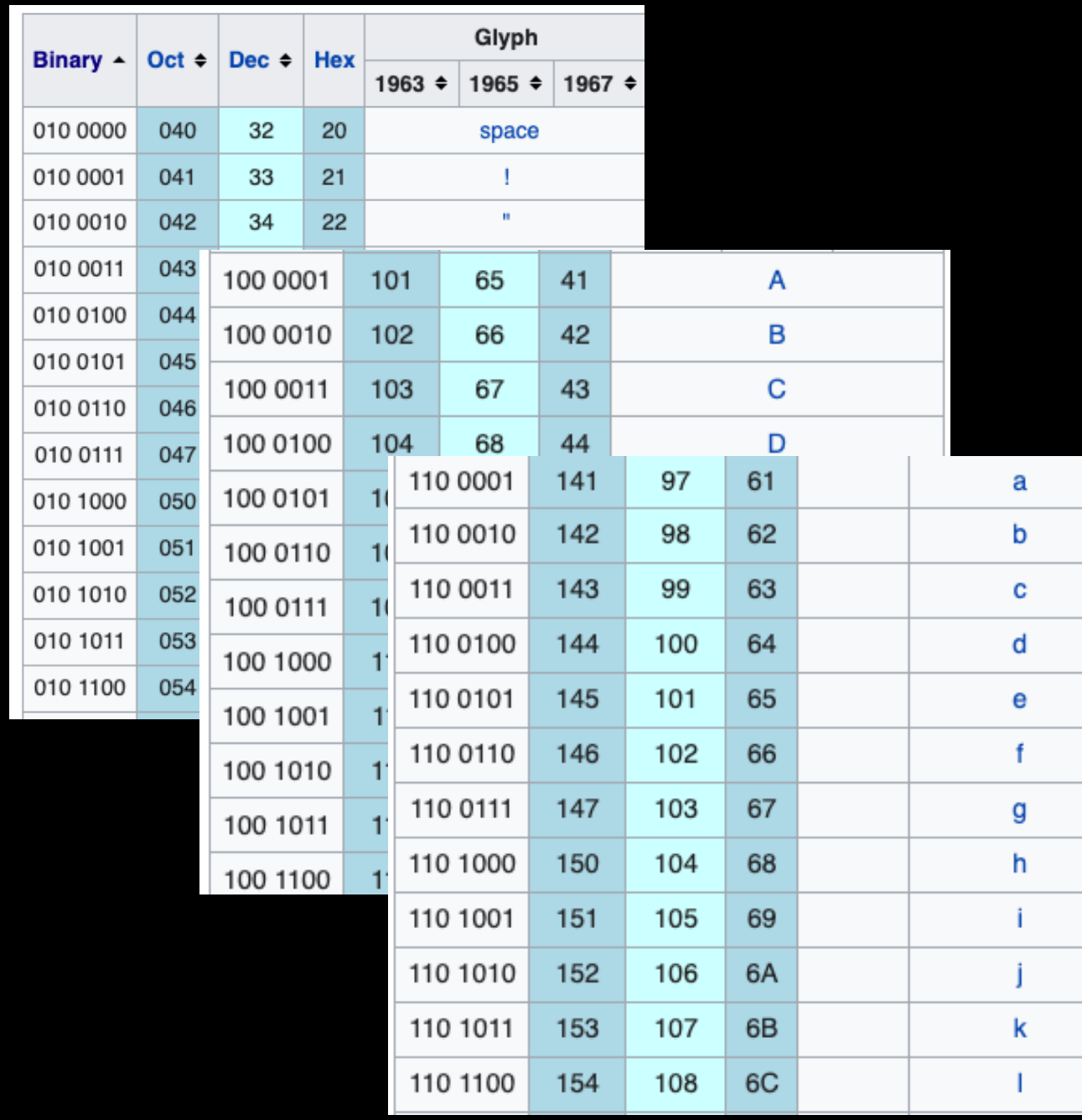
- Bakoverkompatibel med ASCII

- 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

TEKSTFILER

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:

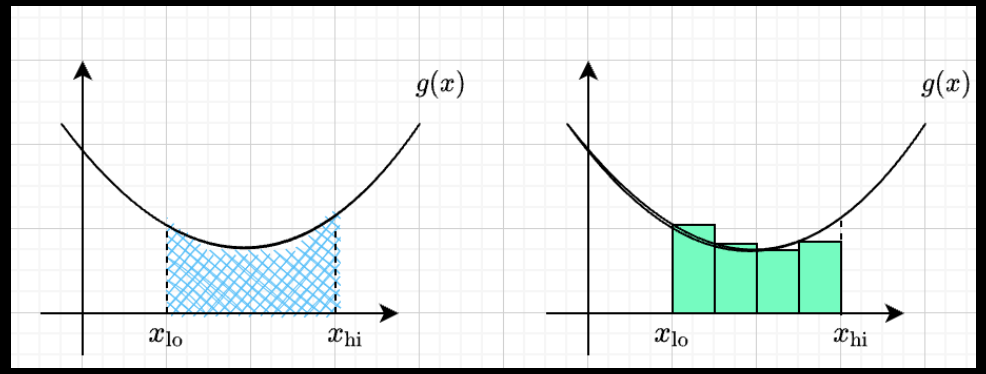
- .docx/.rtf/.pdf
- .xlsx
- .zip
- .mp3/.acc/.mp4
- .png/.jpeg/.gif

TEKSTFIL

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

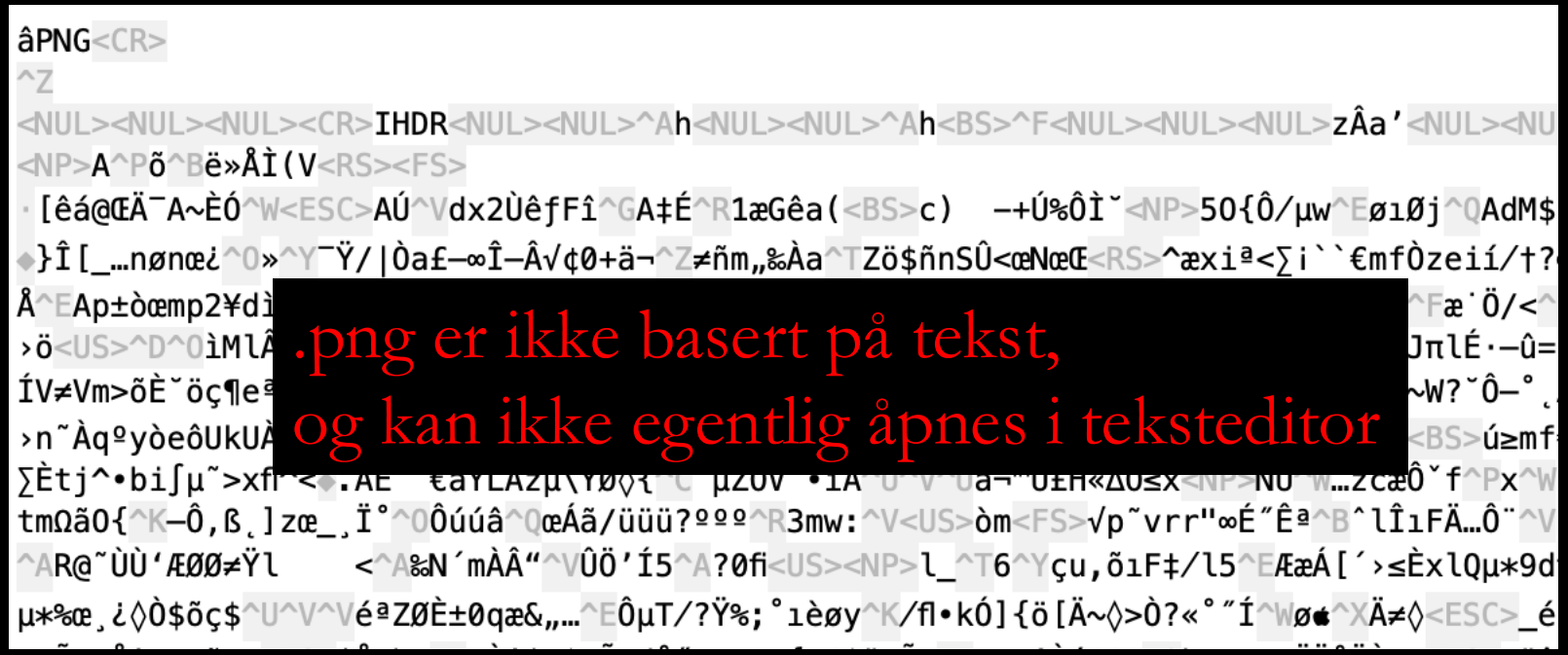
area_under_graph.svg

…<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

FINN VANLIGSTE BOKSTAV

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

nsf2025.txt

```
1    a
2    ab
3    abaca
4    abacaen
5    abacaene
6    abacaer
7    abaki
8    abakien
9    abakiene
10   abakier
```

```
'a'  -> 652904
'\n' -> 922321
'b'  -> 192304
'c'  -> 13075
'e'  -> 1667062
'n'  -> 980049
'r'  -> 878452
'k'  -> 496689
...
```

https://www2.scrabbleforbundet.no/?p=5127 - more-5127

RECAP: STRENGER ↔ LISTER

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

print(a)
```

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

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
```

CSV

Comma separated values

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

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


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

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

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

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

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

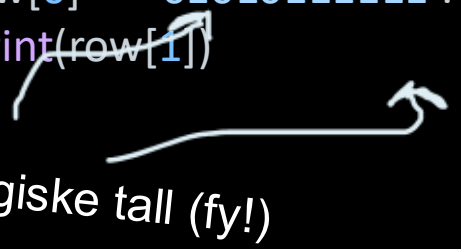
- I Python: tabell-data representert som 2D-liste

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

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

```
table_2d_list = [
    ['personnummer', 'medisin', 'startdato'],
    ['01010111111', ‘paracet', '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])
```



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

```
personnummer,medisin,startdato
01010111111,paracet,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': ‘paracet', 'startdato': '2022-08-22'},
    {'personnummer ': '22020222222', 'medisin': 'fluor', 'startdato': '2022-01-11'},
    {'personnummer ': '01010111111', 'medisin': 'tran', 'startdato': '2022-06-01'}
]
```

Demo

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

```
table_dict_list = [
    {'personnummer': '01010111111', 'medisin’: ‘paracet', '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,paracet,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,paracet,2022-08-22',
    '22020222222,fluor,2022-01-11',
    '01010111111,tran,2022-06-01',
]
```

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

```
data_lines = [
    '01010111111,paracet,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,paracet,2022-08-22'
```

```
cells = ['01010111111', ‘paracet', '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 = ‘paracet'

row_dict['medisin'] = ‘paracet'

- 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': ‘paracet', '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(number_of_cols):
        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

```
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)

Skrive 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)
```