UNIVERSITETET I BERGEN

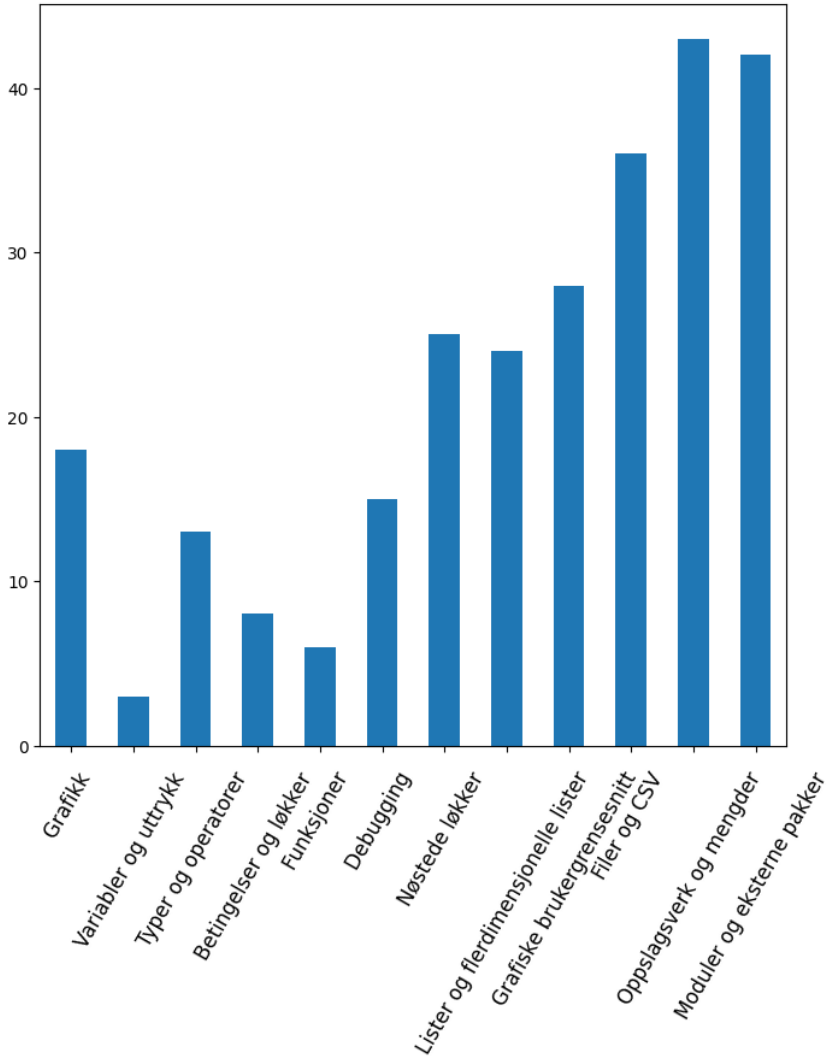
GJENNOMGANG

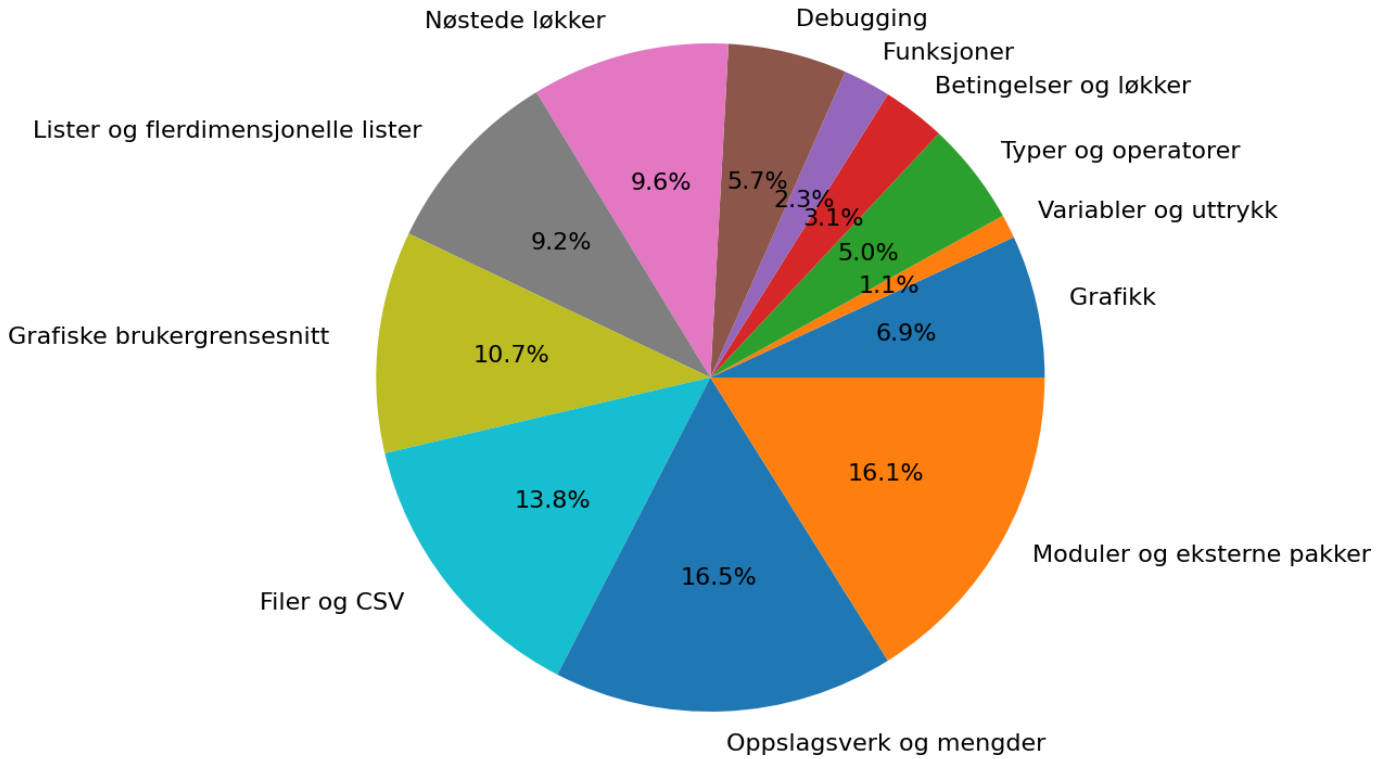
INF100

VÅR 2024

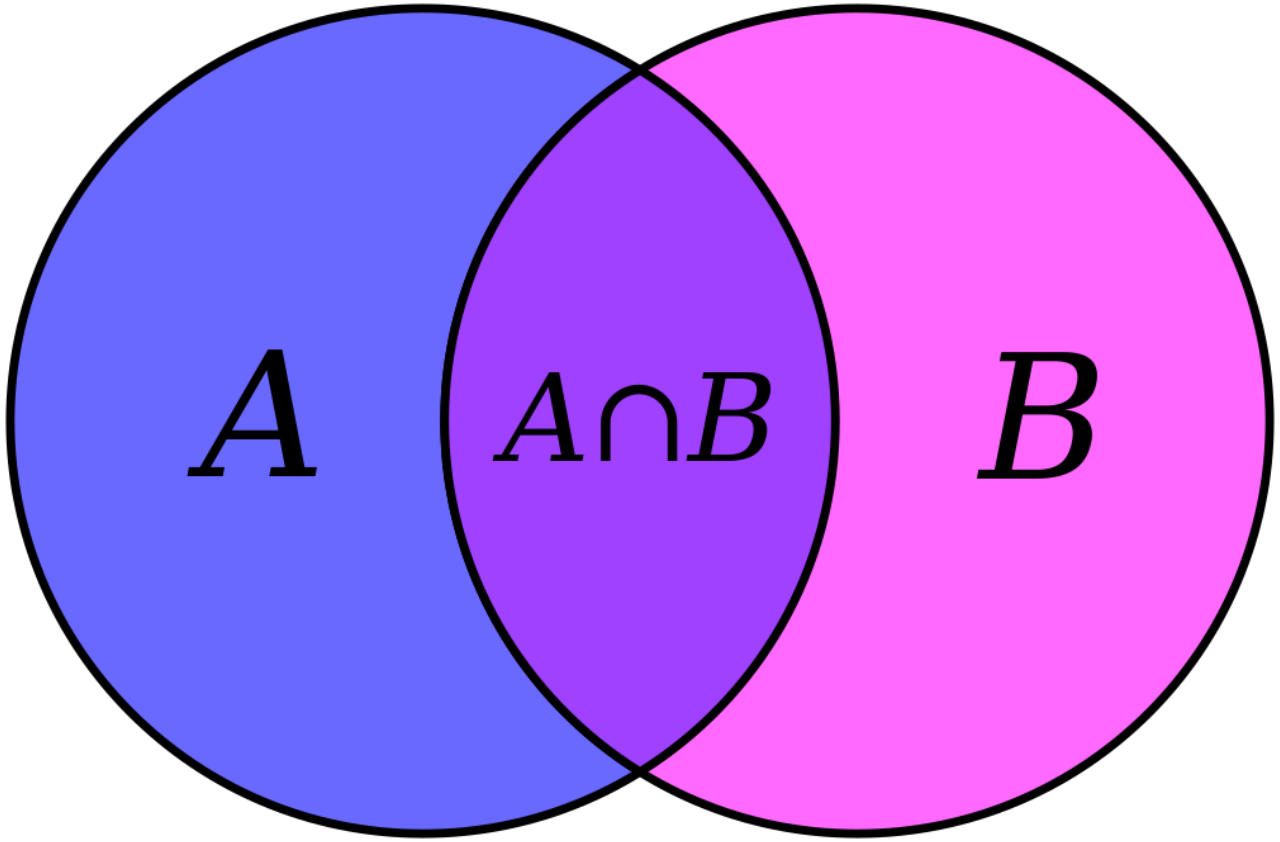
Jakob Sverre Alexandersen

Agenda


40
30
20
10
0
Grafikk
Variabler og uttrykk
Typer og operatorer
Betingelser og løkker
Funksjoner
Debugging
Nøstede løkker
Lister og flerdimensjonelle lister
Grafiske brukergrensesnitt
Filer og CSV
Oppslagsverk og mengder
Moduler og eksterne pakker


Nøstede løkker
Debugging
Funksjoner
Betingelser og løkker
Lister og flerdimensjonelle lister
Typer og operatorer
Variabler og uttrykk
Grafikk
Grafiske brukergrensesnitt
Moduler og eksterne pakker
Filer og CSV
Oppslagsverk og mengder
9.6%
5.7%
2.3%
3.1%
5.0%
1.1%
6.9%
9.2%
10.7%
16.1%
13.8%
16.5%

Mengder (sets) 👥



Mengder (sets)

- Ingen duplikater!
- Rask(ere enn lister)
- Uordnet rekkefølge

```python
def count_overlap_fast(path1, path2):
    with open(path1, 'r') as f:
        path1 = f.read().splitlines()
    with open(path2, 'r') as f:
        path2 = f.read().splitlines()

    return len(set(path1) & set(path2))

```

Ca. 390 ms

```python
def count_overlap_slow(path1, path2):
    with open(path1, 'r') as f:
        path1 = f.read().splitlines()
    with open(path2, 'r') as f:
        path2 = f.read().splitlines()

    count = 0
    for line in path1:
        if line in path2:
            count += 1
    return count
```

Aner ikke, jeg avbrøyt koden etter den hadde kjørt i 45 sek

Forts.

```
a = {1, 2, 3}
b = {3, 4, 5}

print(a & b)                        #printer {3}
print(a.intersection(b))            #printer {3}
print(a | b)                        #printer {1, 2, 3, 4, 5}
print(a.union(b))                   #printer {1, 2, 3, 4, 5}
print(a ^ b)                        #printer {1, 2, 4, 5}
print(a.symmetric_difference(b))    #printer {1, 2, 4, 5}
print(a - b)                        #printer {1, 2}
print(a.difference(b))              #printer {1, 2}
print(b - a)                        #printer {4, 5}

```

Oppslagsverk 📖

- Keys
- Values
- Items

```python
my_dict = {'key': 'value', 43: 'foo'}
print(my_dict['key'])           #printer 'value'
print(my_dict.keys())           #printer ['key', 43]
print(my_dict.values())         #printer ['value', 'foo']
```

```python
country_map = {
    'Oslo': 'Østlandet',
    'Bergen': 'Vestlandet',
    'Drammen': 'Østlandet',
    'Stavanger': 'Vestlandet',
    'Kristiansand': 'Sørlandet',
}

for city, location in country_map.items():
    print(f'{city} is in {location}')

#output:
# Oslo is in Østlandet
# Bergen is in Vestlandet
# Drammen is in Østlandet
# Stavanger is in Vestlandet
# Kristiansand is in Sørlandet

```

Forts.

- Ingen duplikate keys i dict()!
  - Duplikate values er lov☺
- Dict i en dict?

```
1  double_dict = {'dette': {'er': {'en': {'nested' : 'dictionary!'}}}}
2  print(double_dict['dette']['er'])       #printer '{'en': {'nested': 'dictionary!'}}'
```

  - Har dere sett noe som minner om dette?

Moduler 🧩

- CSV
- Json / Requests
- Matplotlib
- UiB graphics
- Time / Datetime
- Math

Requests / Json 🌐

```python
import json
import requests

url = 'https://api.met.no/weatherapi/locationforecast/2.0/compact?lat=69.6489&lon=18.95508'
headers = {'User-Agent': 'inf100.ii.uib.no jaale0655'}
response = requests.get(url, headers=headers)
data = response.json()

print(data.keys())
print(data['properties'].keys())
print(data['properties']['timeseries'][0])      #klokken 06L → 07L
print(data['properties']['timeseries'][1])      #klokken 07L → 08L
print(data['properties']['timeseries'][2])      #klokken 08L → 09L
print(data['properties']['timeseries'][3])      #klokken 09L → 10L

```

```
{
    "type": "Feature",
    "geometry": {
        "type": "Point",
        "coordinates": [
            18.9551,
            69.6489,
            7
        ]
    },
    "properties": {
        "meta": {
            "updated_at": "2024-04-10T05:47:44Z",
            "units": {
                "air_pressure_at_sea_level": "hPa",
                "air_temperature": "celsius",
                "cloud_area_fraction": "%",
                "precipitation_amount": "mm",
                "relative_humidity": "%",
                "wind_from_direction": "degrees",
                "wind_speed": "m/s"
            }
        }}} ...
```

- json.loads(argument) for å konvertere fra json til dict

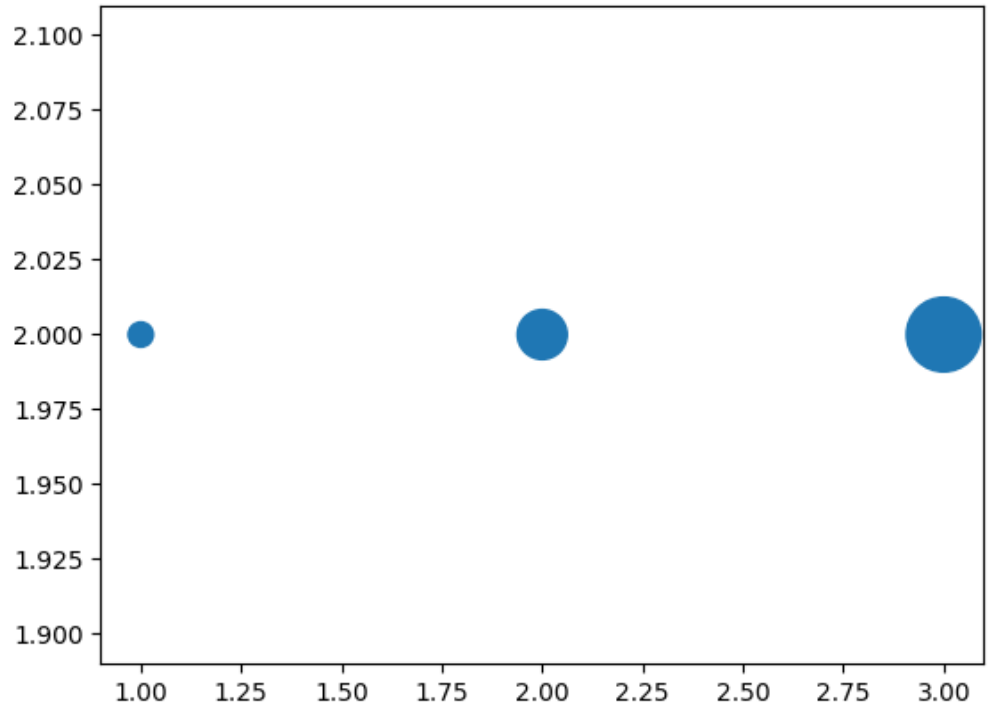
```python
weather = data['properties']['timeseries'][3]
data_dict = weather['data']['instant']['details']

pressure = data_dict['air_pressure_at_sea_level']
temperature = data_dict['air_temperature']
cloud_cover = data_dict['cloud_area_fraction']
humidity = data_dict['relative_humidity']
wind_directon = data_dict['wind_from_direction']
wind_speed = data_dict['wind_speed']

print(f'Temperature: {temperature}°C')
print(f'Pressure: {pressure} hPa')
print(f'Cloud cover: {cloud_cover}%')
print(f'Humidity: {humidity}%')
print(f'Wind direction: {wind_directon}°')
print(f'Wind speed: {wind_speed} m/s')
```

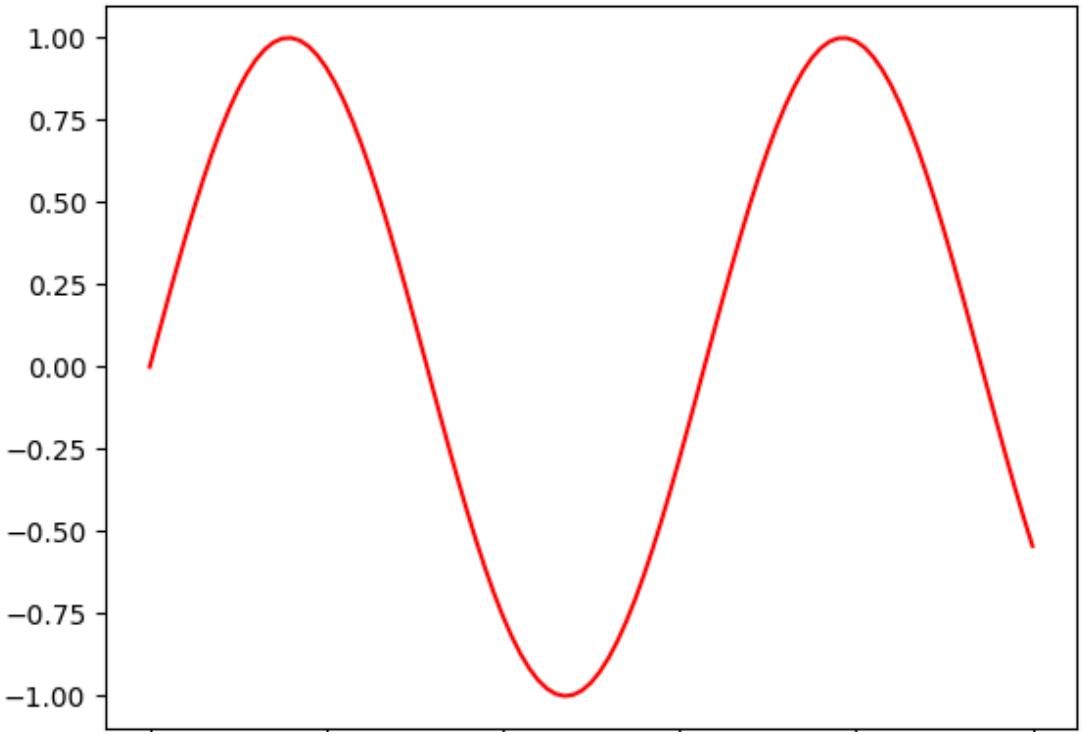
```
Temperature: 6.0°C
Pressure: 994.2 hPa Cloud
cover: 100.0% Humidity:
68.1%
Wind direction: 37.3°
Wind speed: 2.1 m/s
```

Matplotlib 📈

- Brukes for å blant annet plotte div. ting

```python
import matplotlib.pyplot as plt

x = [1, 2, 3]
y = [2, 2, 2]
size = [10**2, 20**2, 30**2]

plt.scatter(x, y, s=size)
plt.show()
```



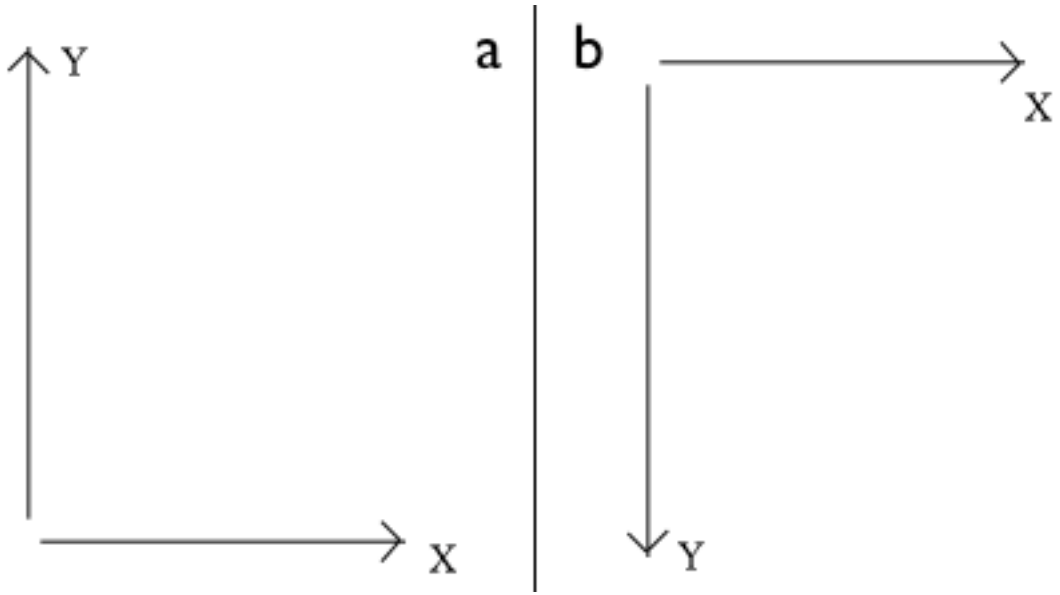



```python
from math import sin

xs = [n / 10 for n in range(101)]
ys = [sin(x) for x in xs]

plt.plot(xs, ys, 'r')
plt.show()
```

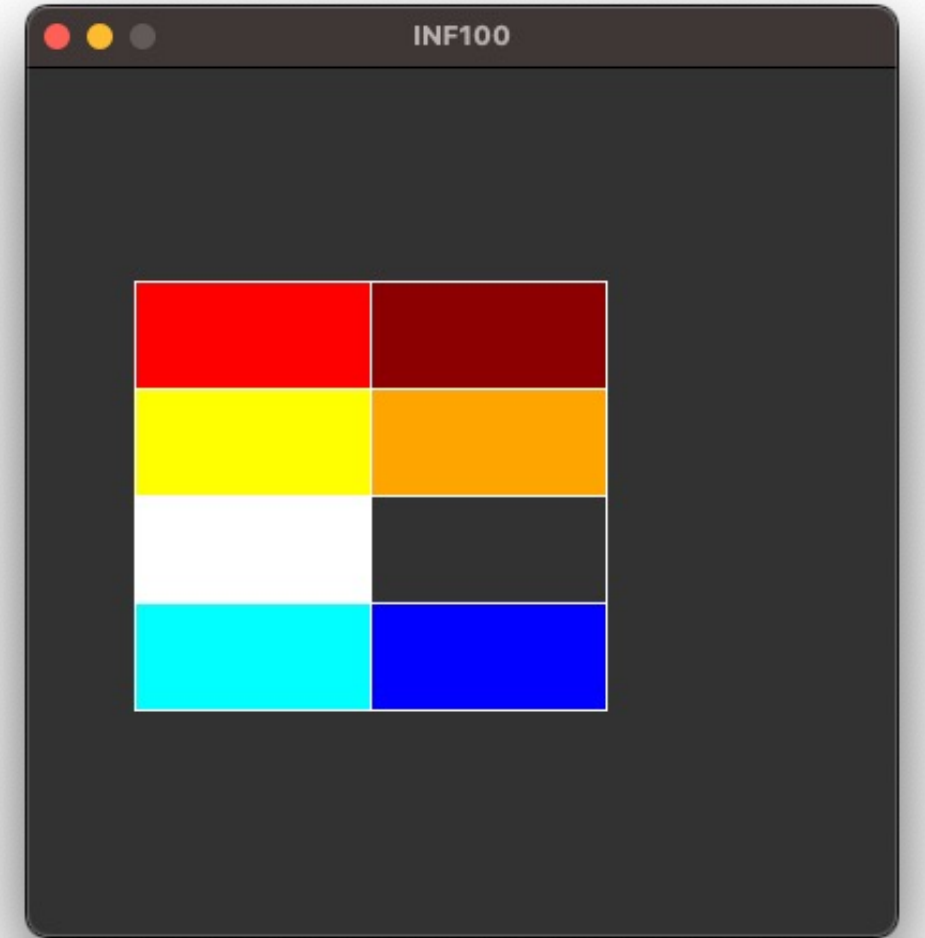
Graphics og nested for-loops ✍️

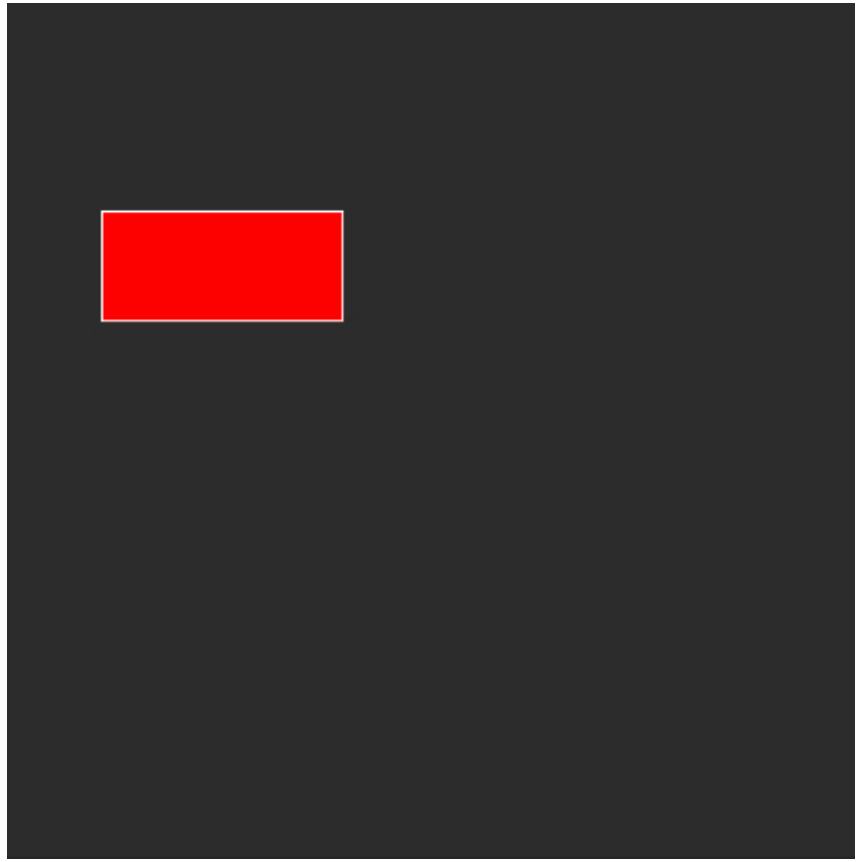
- Forenklet versjon av tkinter

- Invertert y-akse



```python
from uib_inf100_graphics.simple import canvas, display

def draw_grid(canvas, x1, y1, x2, y2, colors):

    rows = len(colors)
    cols = len(colors[0])

    cell_w = (x2 - x1) / rows
    cell_h = (y2 - y1) / cols

    for row in range(rows):
        for col in range(cols):
            cell_l = x1 + col * cell_w
            cell_t = y1 + row * cell_h
            cell_b = cell_t + cell_h
            cell_r = cell_l + cell_w

            color = colors[row][col]

            canvas.create_rectangle(cell_l, cell_t, cell_r, cell_b, fill=color)
```


INF100



Math / Datetime 

```python
import math
print(math.pi)                  # 3.141592653589793
print(math.ceil(math.pi))       # 4
```

```python
from datetime import date, timedelta, datetime

today = date.today()
print(f'I dag er {today}')                              # Dagens dato er 2024-04-11

ten_days_ago = (today - timedelta(days = 10))
print(f'For ti dager siden var datoen {ten_days_ago}')  # For ti dager siden var datoen 2024-04-01
```

Filer 📂

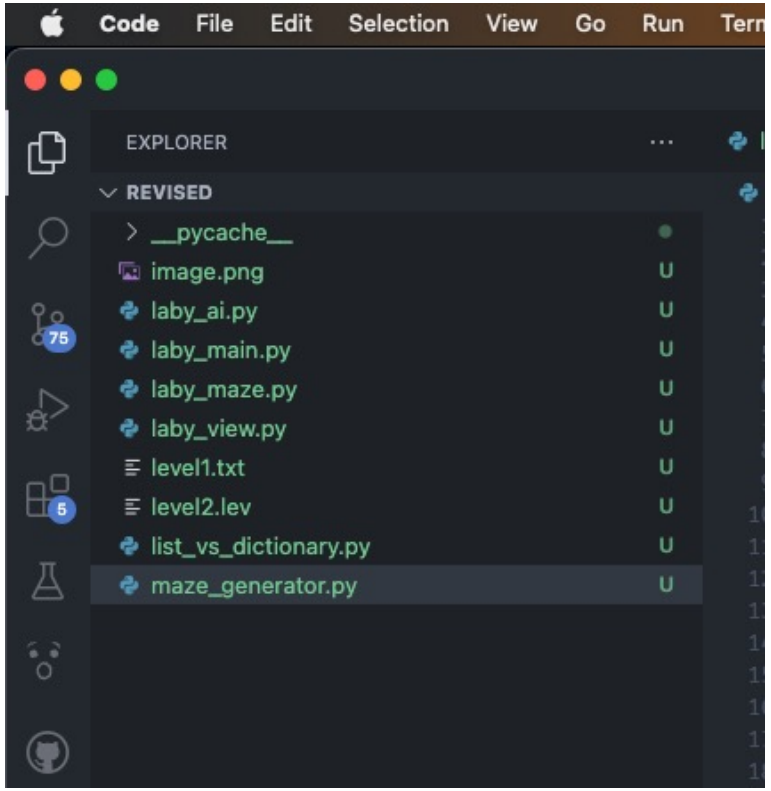
- Lese og skrive
  - readlines()

```python
def opener(path, destination):
    with open(path, 'r', encoding='utf-8') as f:
        file = f.read().splitlines()

    res = []

    for line in file:
        if 'C' in line:
            res.append(line)

    with open(destination, 'w', encoding='utf-8') as w:
        for i in res:
            w.write(i)

    return None

opener('sample1.txt', 'dest.txt')
```

```
Før (sample1.txt):
————————————
CGAA
TATT
CTTT
GAGG
————————————

Etter (dest.txt):
————————————
CGAA
CTTT

————————————
```

Viktig å tenke på🧠

- Åpne hele mappen i VSCode, ikke bare filen
  - Cmd + O (mac), ctrl + O (windows)

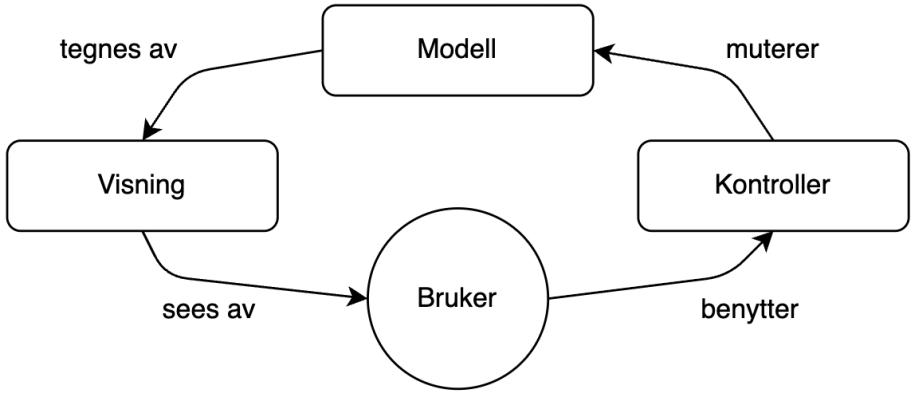



```python
import os
print(f'Current working directory: {os.getcwd()}')
```

```
Current working directory: /Users/jakobalexandersen/inf100/q-learning/revised
jakobalexandersen@Jakobs-Laptop revised %
```

Grafiske brukergrensesnitt 🕹️

- MVC-prinsippet
  - Model-View-Controller

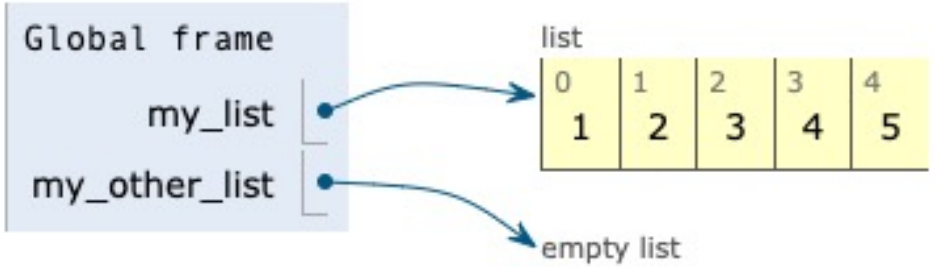


- app_started(app):
  - Modellen som sender ting til visningen
- redraw_all(app, canvas):
  - (View) Tegner selve applikasjonen
- key_pressed(app, event):
  - Kontrolleren; tar input fra bruker og sender videre til modellen

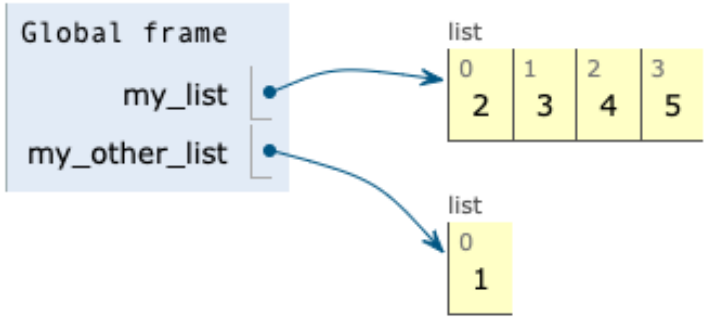
```python
from uib_inf100_graphics.event_app import run_app

def app_started(app):
    # Modellen
    app.counter = 0


def key_pressed(app, event):
    if event.key == 'Space':
        app.counter += 1

def redraw_all(app, canvas):
    # Visningen
    canvas.create_text(app.width // 2, app.height // 2,
                       text=f'Du har trykket space {app.counter} ganger',
                       font = 'Arial 12 bold'
                       )

run_app(width=300, height=100)
```

Lister 📋

```python
my_list = [1, 2, 3, 4, 5]
my_other_list = []

for i in my_list:
    my_other_list.append(i)

print(my_other_list)
```




```python
my_list = [1, 2, 3, 4, 5]
my_other_list = []

my_other_list.append(my_list.pop(0))
```



List comprehension / listeinklusjon 📋😎

```python
#tung måte
a = []
for i in range(11):
    a.append(i)                          # a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

#lett måte
b = [x for x in range(11)]               # b = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

c = [x for x in range(11) if x % 2 == 0]    #c = [0, 2, 4, 6, 8, 10]
```

Div liste-funksjoner

- .split(*element)
  - ‘Hva skjer?’.split(‘ ‘) → [‘Hva’, ‘skjer?’]
- .join(*liste)
  - ‘ ‘.join([‘Hva’, ‘skjer?’]) → ‘Hva skjer?’
- .extend(*liste)
  - [1, 2, 3].extend([4, 5, 6]) → [1, 2, 3, 4, 5, 6]
- .insert(*indeks, *element)
  - [1, 2, 3].insert(1, 1.5) → [1, 1.5, 2, 3]
- .sort(*order)
  - [2, 1, 3].sort() → [1, 2, 3]
  - Reverse = False by default

Men hva med .sorted()???????
.sort() muterer en liste
.sorted() lager en ny liste
b = a.sorted()

Flerdimensjonelle lister 📋📋

- 'En liste av lister'

```python
my_list = [[1, 2, 3, 4],
           [5, 6, 7, 8],
           [1, 1, 1, 1]]

for i in my_list:
    print(i)
```

```
[1, 2, 3, 4]
[5, 6, 7, 8]
[1, 1, 1, 1]
```

```python
for item in my_list:
    for i in item:
        print(i)
```

```
1
2
3
4
5
6
7
8
1
1
```

Slicing (og litt indeksering) 

```
1  a = [1, 2, 3, 4, 5]
2  a[1:3]                    # [2, 3]
3  a[-1]                     # 5
4  a[1:]                     # [2, 3, 4, 5]
5  a[:3]                     # [1, 2, 3]
```

- Det samme gjelder for strings

Strings 

- .upper() / .lower()
  - ‘MatPlOtLiB eR vanSkeLiG’.lower() → ‘matplotlib er vanskelig’
  - ‘MatPlOtLiB eR vanSkeLiG’.upper() + ‘!!!’ → ‘MATPLOTLIB ER VANSKELIG!!!’
- .strip()
  - ‘\n Jeg elsker mellomrom. \n.strip() → ‘Jeg elsker mellomrom.’
- .replace(*replace, *with)
  - ‘Jeg elsker mellomrom.’.replace(‘mellomrom’, ‘Python’) → ‘Jeg elsker Python.’

Funksjoner ↪️

- Gjenbrukbare
- Pleier å returnere noe
  - Om ikke → returnerer None

```python
def is_even(num: int) -> bool:
    return num % 2 == 0

print(is_even(4))
```

- Parameter
- Argument

Feilhåndtering (try/except) ❌

- Brukes når man kan forvente visse feil

```python
def divider(a: int, b: int) -> float:
    try:
        return a / b
    except ZeroDivisionError:
        return 'Deling på null er tull!'
```

- Hindrer programmet i å stoppe
- Tips: kjør programmet med feilen uten try/except og se hvilken error du får

Tips til eksamen 🎓

- Øv MASSE
- Gjør alle lab-oppgavene
- Se på tidligere eksamener
  - Denne er viktig