UiT Norges arktiske universitet

Python lab Repetisjon

BED-1304 (Python-lab), 7.5 ECTS

Markus J. Aase

markus.j.aase@uit.no, kontor 02.411

Universitetslektor i matematikk og statistikk

Handelshøgskolen, UiT

Økonomi og administrasjon og samfunnsøkonomi med datavitenskap

DAGENS PLAN

- Repetert til nå
  - Logikk/løkker
  - Pandas
  - Matplotlib
- Kort repetisjon av SymPy og funksjoner
  - Presentasjon
  - VSCode
- Oppgaveregning/dere får løsningsforslag på arbeidskrav

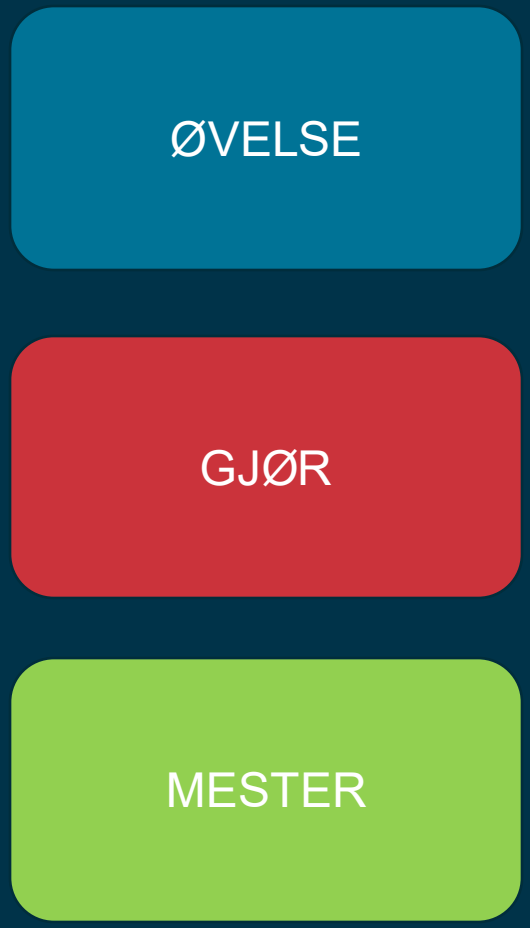
Men først, litt om eksamen!

- HJELPEMIDLER
  - Ingen hjelpemidler
- Eksamensformat
  - Multiple choice
  - Kortsvar
  - Langsvar (kodeoppgave)
- WiseFlow
  - Skjerm vil låses

Tips til eksamen

- Arbeidskrav
  - Løsningsforslag kommer
- Prøve-eksamen
  - Ligger ute i forelesningsplanen
- Forelesningsnotater/seminar-oppgaver
  - Dekker pensum
- Repetisjonsark
  - Gode for repetisjon av fagstoff, og oppgaver

- Mitt aller beste tips



Skrive kode i Wiseflow

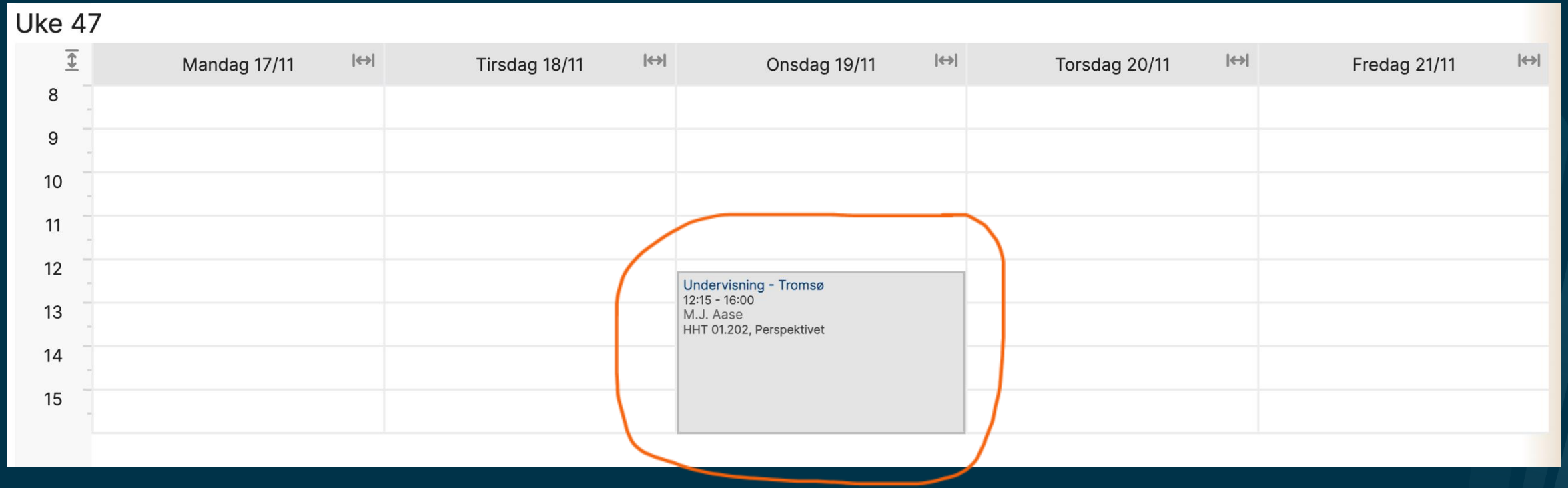
```
# Her skriver dere deres kommentarer


# Definerer en funksjon som tar inn bredde og lengde
# og beregne areal
def funk(bredde, lengde):
    areal = round(bredde * lengde, 2) # bruker round() for å runde av
    return areal
```

Skrive kode i Wiseflow

1. Kommenter koden
2. Skriv Python-syntaks etter beste evne
3. Småfeil i syntaks går helt fint
4. Får du ikke til å skrive noen linjer kode, prøv å forklar hvordan du ville løst oppgaven

Eksamensøving

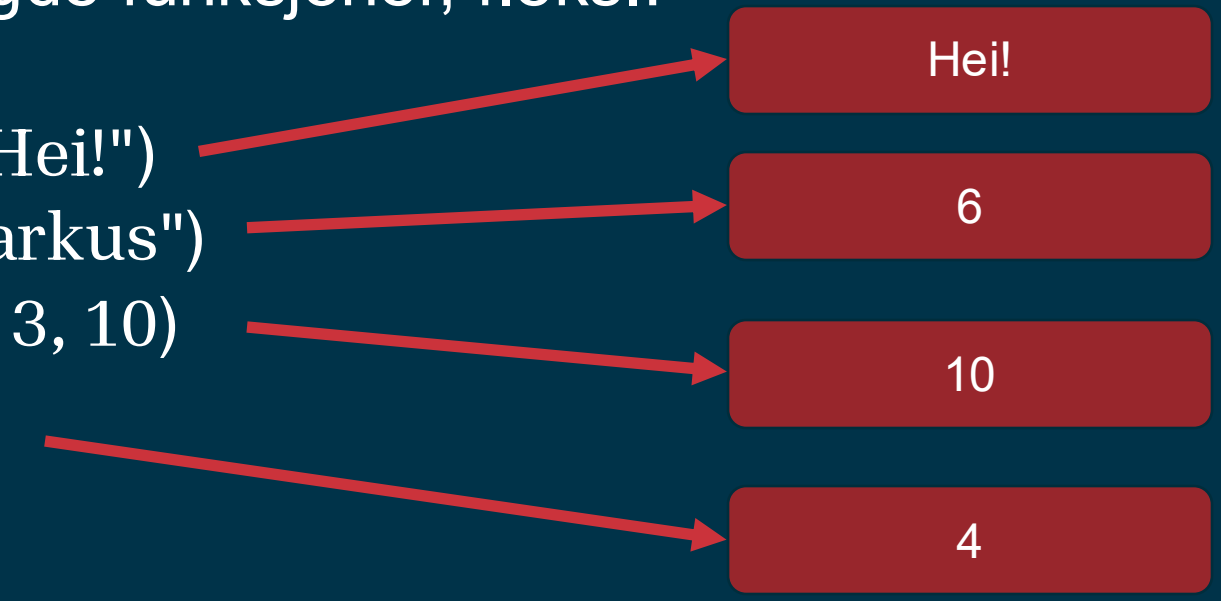
- Neste uke!



Innebygde funksjoner

- Python har mange ferdig, innebygde funksjoner, f.eks.:

- `print("Hei!")`
- `len("Markus")`
- `max(5, 3, 10)`
- `abs(-4)`



💬 Disse er alltid tilgjengelige uten å måtte importeres.

- Mange funksjoner avhenger av pakker, som numpy, sympy, pandas osv.

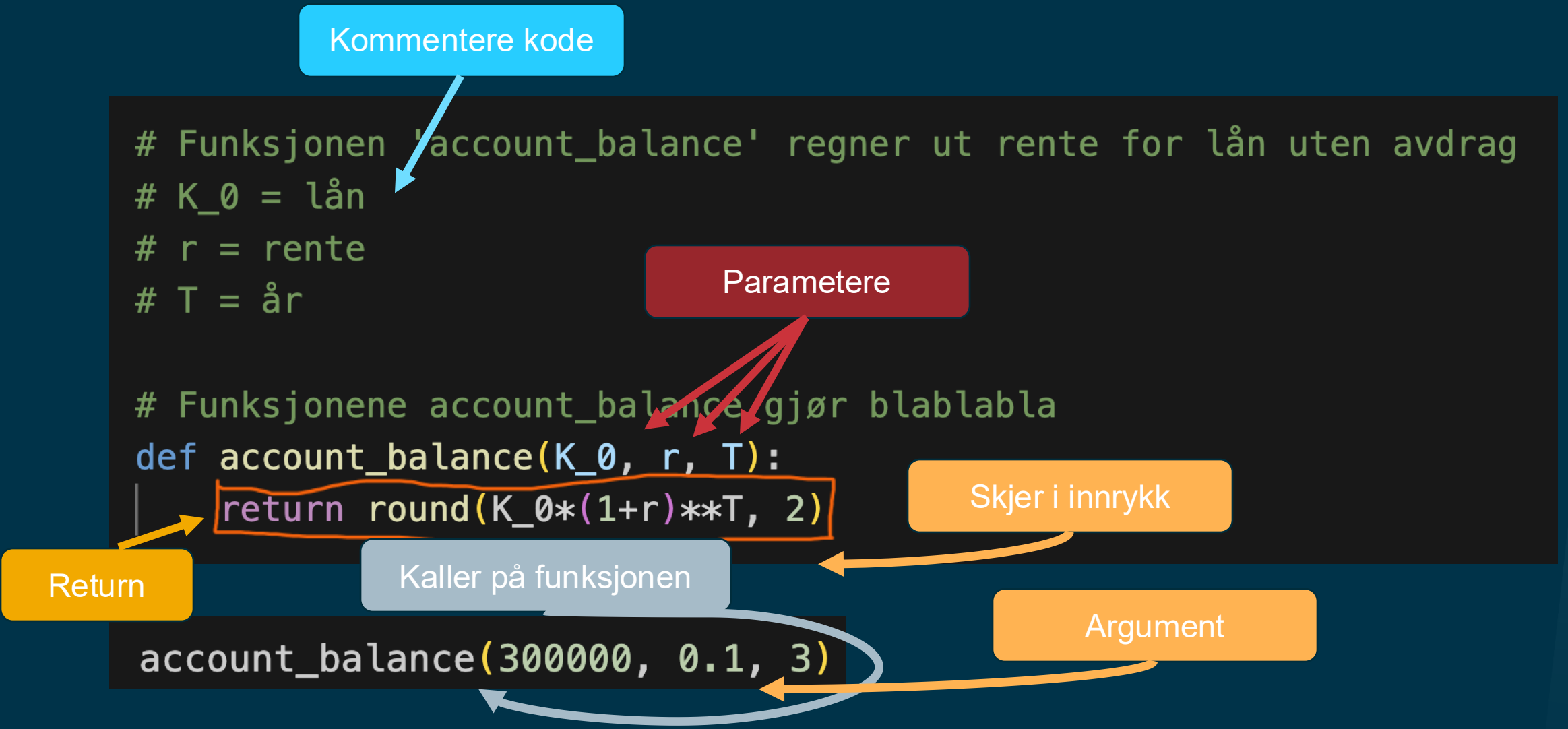
Funksjoners hensikt

- ✅ Mindre kode – du slipper gjentakelser
- ✅ Enklere å lese og forstå
- ✅ Lettere å feilsøke og vedlikeholde
- ✅ Kan gjenbrukes i flere programmer

- Tenk: én oppgave = én funksjon!

```python
skattefri_grense = 15000 # De første 10.000,- er skattefri
skattesats = 0.25        # Skattesats, etter tjente 10.000,-, er 22%

# Definerer skattefunksjonen
def skatte_funksjon(inntekt, skattesats, skattefri_grense):

    # Hvis inntekt er mindre eller lik skattefri grense
    # returner at skatten er lik 0.
    if inntekt <= skattefri_grense:
        return 0
    else:
        skattepliktig_beløp = inntekt - skattefri_grense
        skatt = skattepliktig_beløp * skattesats
        return skatt
```

Funksjoner


Kommentere kode
Funksjonen 'account_balance' regner ut rente for lån uten avdrag
K_0 = lån
r = rente
T = år
Parametere
# Funksjonene account_balance gjør blablabla
def account_balance(K_0, r, T):
return round(K_0*(1+r)**T, 2)
Skjer i innrykk
Return
Kaller på funksjonen
Argument
account_balance(300000, 0.1, 3)

Egendefinerte funksjoner

- Du lager dine egne med nøkkelordet `def`:

```
def hilsen(navn):
    print(f"Hei, {navn}!")
```

- Kall funksjonen slik:

```
hilsen("Ola")
```

→ Skriver ut: Hei, Ola!

Parametere og argumenter

- Parametere = variabler i funksjonen.
- Argumenter = verdiene du sender inn.

```
def areal(rektangel_bredde, rektangel_høyde):
    return rektangel_bredde * rektangel_høyde

print(areal(4, 2))
```

→ Returnerer 8

Return-verdier 

- `return` sender verdien tilbake til programmet.

```
def dobbel(x):
    return x * 2

resultat = dobbel(5)
print(resultat)  # 10
```

- 🧠 Uten `return` får du bare utskrift, ikke verdi.

Standardverdier 

- Funksjoner kan ha standardverdier:

```
def hilsen(navn="verden"):
    print(f"Heisann, {navn}!")
```

- ➡️ `hilsen()` gir *Heisann, verden!*
- ➡️ `hilsen("Ola")` gir *Heisann, Ola!*

Feil å se opp for ⚠️

- ⛔ Glemme parenteser når du kaller funksjonen
- ⛔ Mangle `return` hvis du trenger verdien
- ⛔ Bruke `print` istedenfor `return` (eller motsatt)

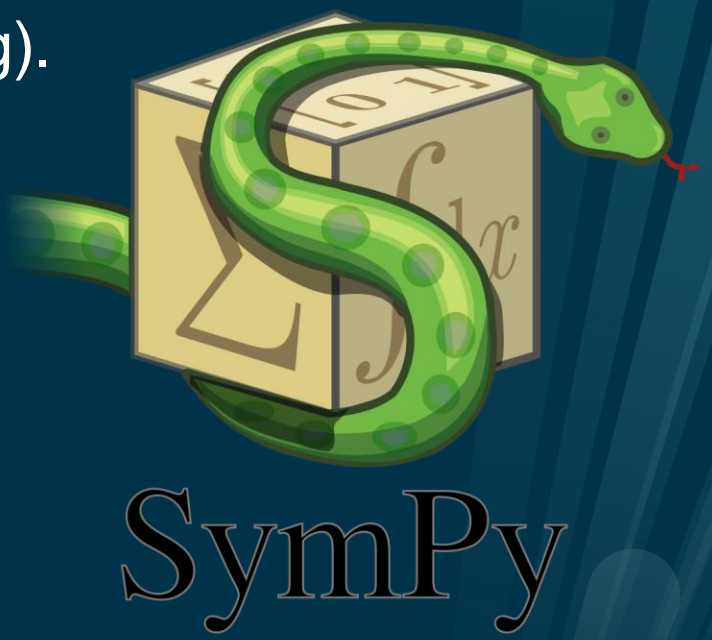
- 💡 Tips: Test hver funksjon for seg!

Oppsummering ✨

- 🎯 Funksjoner gjør koden din enklere og mer oversiktlig
- 🎯 Du bruker `def` for å lage dem
- 🎯 *Parametere* er variabler i funksjonen
- 🎯 Du kan sende inn *argument* og få returverdier
- 🎯 Skille mellom `print()` og `return`
- 🎯 Bruk funksjoner for gjenbruk og struktur!

SymPy – symbolsk matematikk

- Et Python-bibliotek for **symbolsk matematikk**.
- Hvorfor? Lar oss regne analytisk (med symboler) i stedet for bare numerisk (med tall).
- **Nyttig i økonomi for å:**
  - Løse optimeringsproblemer (f.eks. profittmaksimering).
  - Finne likevekter (tilbud = etterspørsel).
  - Derivere og integrere funksjoner.



Syntaks

```python
import sympy as sp
```

importere pakken

```python
# Definerer L, a, w, p, K som symboler
L, a, w, p, K = sp.symbols("L a w p K")
```

Definere symbol

```python
# Produksjonsfunksjon
def f(L, a):
    return 60 * L**a

# Profittfunksjon
def profit(L, a, w, p, K):
    return p * f(L, a) - w * L - K
```

Definerer funksjoner med disse symbolene

Symboler og uttrykk i SymPy

Definer symboler

```
import sympy as sp
x, y = sp.symbols('x y')
```

$$x$$
$$y$$

Faktoriser uttrykk

```
expr = x**2 + 2*x + 1
fact = sp.factor(expr)
fact
```

$$(x+1)^2$$

Ekspandere uttrykk

```
expand = sp.expand(fact)
expand
```

$$x^2 + 2x + 1$$

Definere funksjoner og løse dem

Definer uttrykk

```
x = sp.symbols('x')
eq1 = sp.Eq(x + 2, 5)
eq1
```

$x + 2 = 5$

Løs uttrykk

```
sol = sp.solve(eq1, x)
sol
```

$[3]$

Løs andregradsuttrykk

```
eq2 = sp.Eq(x**2, 9)
sol2 = sp.solve(eq2, x)
```

$[-3, 3]$

Derivasjon og substitusjon

Deriver uttrykk

```
expr = x**2 + 2*x + 1
dx_expr = sp.diff(expr, x)
dx_expr
```

$2x + 2$

Substitusjon

```
# Erstatter x med 2
dx_expr.subs(x, 2)
```

6

Substitusjon

```
# Erstatter x med 3
dx_expr.subs(x, 3)
```

8

Praktisk eksempel

Oppgave

1. Definer funksjonen $f(x) = -x^2 + 4x + 1$
2. Deriver funksjonen, altså finn $f´(x)$
3. Løs $f´(x) = 0$ ved hjelp av $sp.Eq()$
4. Bruk **matplotlib** for å vise toppunktet

Praktisk eksempel

Oppgave

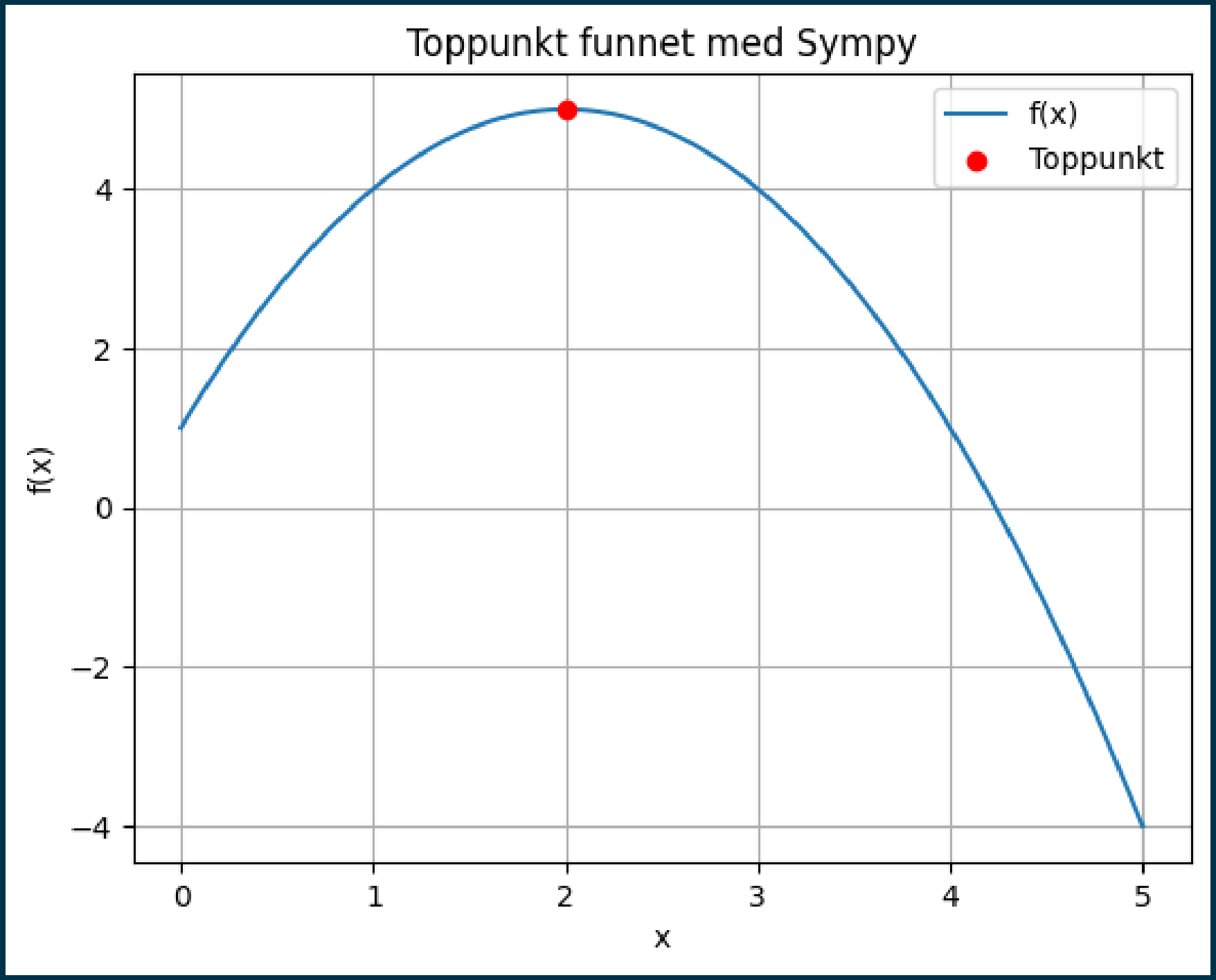
1. Definer funksjonen $f(x) = -x^2 + 4x + 1$
2. Deriver funksjonen, altså finn $f'(x)$
3. Løs $f'(x) = 0$ ved hjelp av $sp.Eq()$
4. Bruk **matplotlib** for å vise toppunktet

```
import sympy as sp
import matplotlib.pyplot as plt
import numpy as np

# Definerer symbol og funksjon
x = sp.symbols('x')
f = -x**2 + 4*x + 1

# Deriver vha sp.diff()
f_deriv = sp.diff(f, x)

# Finn kritisk punkt: der f´(x)=0
critical_p = sp.solve(sp.Eq(f_deriv, 0), x)

# Finn x og y verdiene
x_crit = critical_p[0] # siden det er en liste
y_crit = f.subs(x, x_crit)

print('Toppunkt:, (x_crit, y_crit)))
```

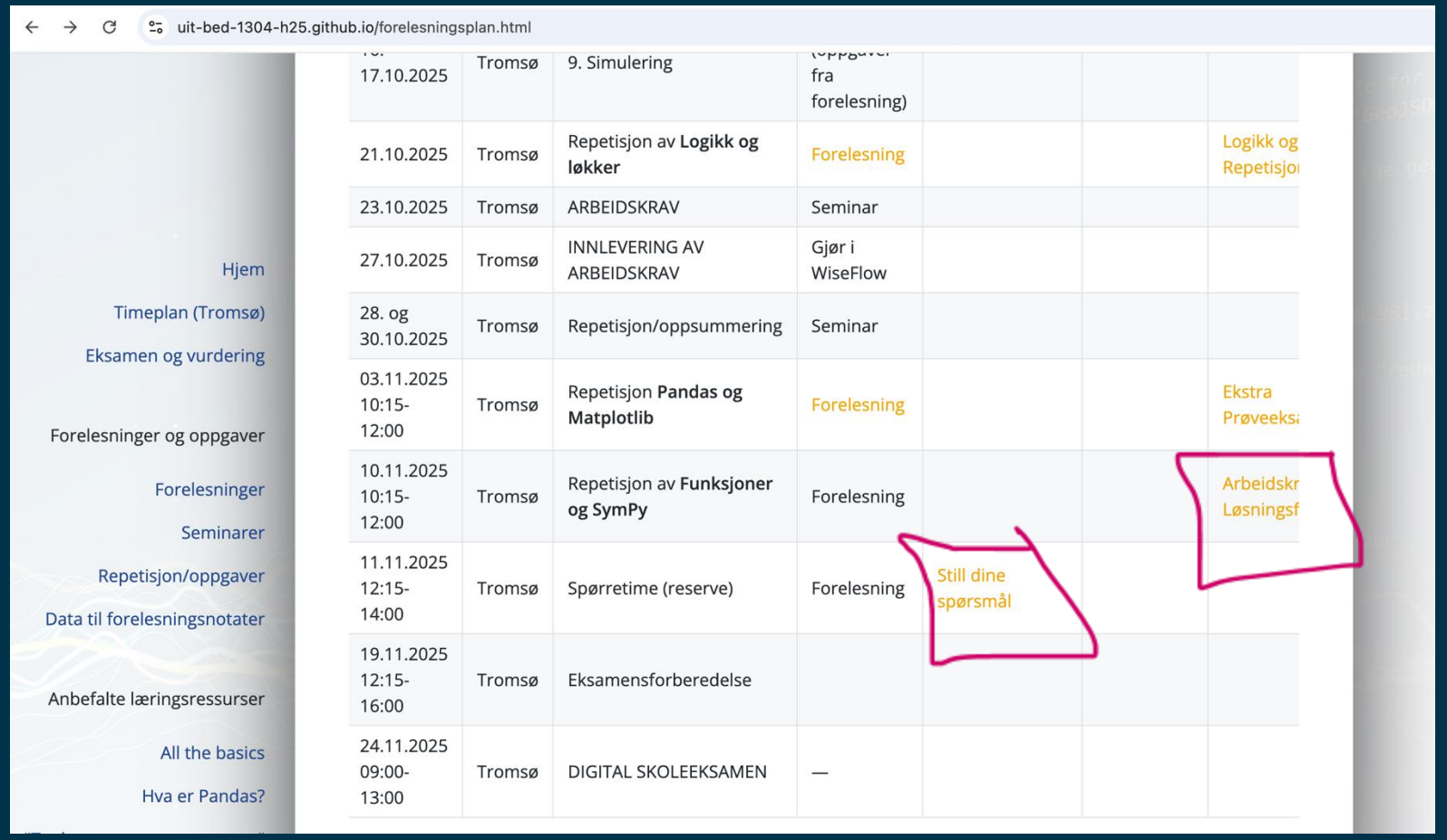
Oppgave

1. Definer funksjonen $f(x) = -x^2 + 4x + 1$
2. Deriver funksjonen, altså finn $f'(x)$
3. Løs $f'(x) = 0$ ved hjelp av $sp.Eq()$
4. Bruk **matplotlib** for å vise toppunktet

```
X = np.linspace(0, 5, 200)
Y = [f.subs(x, val) for val in X]

# Plotter grafen
plt.plot(X, Y, label='f(x)')
plt.scatter([x_crit, y_crit], zorder=5, label='Toppunkt')
plt.title('Toppunkt funnet vha Sympy)
plt.xlabel('x)
plt.ylabel('f(x)')
plt.legend()
plt.grid(True)
plt.show()
```


Toppunkt funnet med Sympy
f(x)
Toppunkt
4
2
0
−2
−4
f(x)
0
1
2
3
4
5
x

Spørsmål?


uit-bed-1304-h25.github.io/forelesningsplan.html
Hjem
Timeplan (Tromsø)
Eksamen og vurdering
Forelesninger og oppgaver
Forelesninger
Seminarer
Repetisjon/oppgaver
Data til forelesningsnotater
Anbefalte læringsressurser
All the basics
Hva er Pandas?
17.10.2025 Tromsø 9. Simulering fra forelesning)
21.10.2025 Tromsø Repetisjon av Logikk og løkker Forelesning Logikk og Repetisjo
23.10.2025 Tromsø ARBEIDSKRAV Seminar
27.10.2025 Tromsø INNLEVERING AV ARBEIDSKRAV Gjør i WiseFlow
28. og 30.10.2025 Tromsø Repetisjon/oppsummering Seminar
03.11.2025 10:15-12:00 Tromsø Repetisjon Pandas og Matplotlib Forelesning Ekstra Prøveeks
10.11.2025 10:15-12:00 Tromsø Repetisjon av Funksjoner og SymPy Forelesning Arbeidskr Løsningsf
11.11.2025 12:15-14:00 Tromsø Spørretime (reserve) Forelesning Still dine spørsmål
19.11.2025 12:15-16:00 Tromsø Eksamensforberedelse
24.11.2025 09:00-13:00 Tromsø DIGITAL SKOLEEKSAMEN —

Oppgaveregning

- Repetisjonsark/oppgaver
- Løsningsforslag på prøve-eksamen