Prøveeksamen i BED-1304 Python-Lab

Markus J. Aase

Oktober 2025

Prøveeksamen

Denne **prøveeksamenen** er ment som en forberedelse, til eksamen i BED-1304/BED-1504. Denne får ikke vurdering på denne, det er ment som egenstudium.

Informasjon

Denne prøveeksamenen er ment for å teste din forståelse av kjernepensum i **BED-1304 Python-Lab**. Den er strukturert *relativt* likt som en ekte eksamen og dekker tematikk vi har gjennomgått i kurset.

Jeg anbefaler, på det sterkeste, å prøve å gjøre dette uten hjelpemidler. Det er fordi dere ikke har tilgang på hjelpemidler under eksamen. AI/Google er kjempenyttig for å løse problemer når man jobber med programmering, men det er utrolig viktig at man vet hva man gjør før man får generert kode, eller kopierer kode fra nett. Så mitt tips er - *train as you fight.*

- **Del 1:** x antall flervalgsoppgaver. Ett riktig svar per oppgave.
- **Del 2:** x antall kortsvarsoppgaver som tester kodeforståelse, pseudokode og feilsøking.
- **Del 3:** x praktiske programmeringsoppgaver hvor du skal skrive kode for å løse et problem.

OBS: Antall oppgaver av de ulike oppgavetypene er ikke nødvendigvis lik på denne *prøve-eksamen* og selve eksamen. Legg også merke til at dette ikke nødvendigvis dekker **hele** pensum. Dere står selv ansvarlig for å kunne det nødvendige fra pensum.

Lykke til!

Del 1: Flervalgsoppgaver

1. Hva skriver `print(type(12.56))` ut til konsollen?

   a) `<class 'int'>`

   b) `<class 'string'>`

   c) `<class 'str'>`

   d) `<class 'float'>`

2. Hva er en variabel i Python?

   a) `Et reservert ord`

   b) `En data-type`

   c) `En funksjon som utfører en operasjon`

   d) `En lokasjon i minne hvor data lagres`

3. Hva brukes en hashtag/emneknagg (#) til i Python?

   a) `Kommentere kode`

   b) `If-else setninger`

   c) `Definere en funksjon`

   d) `Spesifisering av tallverdier`

4. Hvordan kan du endre verdien til to variabler i Python, uten å bruke en tredje variabel?

   a) `x=y: y=x`

   b) `x, y = y, x`

   c) `temp = x; x = y; y = temp`

   d) `x + y; y = x; x = y`

5. Hvilke av de følgende variabeltypene er **ikke** mulig å endre (immutable)?

   a) `List`

   b) `Tuple`

   c) `Variabel`

   d) `Dictionary`

6. Hva er resultatet av uttrykket `[1,2,3] + [4,5]` i Python?

   a) `[1,2,3,4,5]`

   b) `[1,2,3][4,5]`

   c) `Error`

   d) `[[1,2,3], [4,5]]`

7. Hvilket uttrykk gjør at en funksjon returnerer en verdi?

   a) `print`

   b) `return`

   c) `yield`

   d) `pass`

8. Hvordan oppretter du en NumPy-array fra en Python-liste `lst`?

   a) `np.array(lst)`
   b) `np.tolist(lst)`
   c) `np.ndarray(lst)`
   d) `np.makearray(lst)`

9. Hva gjør `pandas.DataFrame.describe()`?

   a) Viser statistiske oppsummeringer (mean, std, min, max ...)
   b) Sorterer DataFrame
   c) Fjerner NaN-verdier
   d) Oppretter en ny DataFrame

10. Hvilket av følgende er korrekt syntaks for en `for`-løkke som itererer over en liste `L`?

    a) `for i in range(L):`
    b) `for i in L:`
    c) `for (i : L):`
    d) `foreach i in L:`

11. Hva returnerer `len("heia")`?

    a) 2
    b) 3
    c) 4
    d) `Error`

12. Hvordan importerer du sympy og definerer variabelen `x`?

    a) `import sympy as sp; x = sp.symbols('x')`
    b) `from sympy import symbols; x = symbols(x)`
    c) `import sympy; x = sympy.var('x')`
    d) `a & b`

13. Hvilket uttrykk sjekker om `x` er lik 5 i en if-setning?

    a) `if x = 5:`
    b) `if x == 5:`
    c) `if x === 5:`
    d) `if (x equals 5):`

14. Hva gjør `df.dropna()` i pandas?

    a) Fjerner rader med minst én NaN-verdi
    b) Erstatter NaN med 0
    c) Returnerer antall NaN
    d) Lager en kopi av DataFrame uten endringer

15. Hvordan beregner du den deriverte av x^2 med sympy?

a) `sp.diff(x**2, x)` gir $2x$

b) `sp.diff(x**2)` gir x

c) `sp.derive(x**2)` gir x^2

d) `sp.integrate(x**2, x)` gir $2x$

16. Hvordan velger du kolonnen `"A"` fra DataFrame `df`?

a) `df.A`

b) `df['A']`

c) Begge metodene over

d) Ingen av dem

17. Hva er riktig måte å definere en funksjon som tar argumentet `x` og returnerer `x*2`?

a) `def f(x): return x*2`

b) `function f(x)  return x*2`

c) `def f -> x*2`

d) `lambda x: x*2`

18. Hva gjør `plt.plot(x, y)` i matplotlib?

a) Lager en linjeplot av y mot x

b) Lager histogram

c) Lager scatterplot

d) Viser tekst

19. Hvilken av følgende er en korrekt måte å lese en CSV-fil i pandas?

a) `pd.read_csv('fil.csv')`

b) `pd.open('fil.csv')`

c) `pandas.load('fil.csv')`

d) `np.read('fil.csv')`

20. Hva gjør `np.exp(1)`?

a) e

b) 1

c) 0

d) Error

21. Hvilket statement lager en tom liste i Python?

a) `lst = {}`

b) `lst = []`

c) `lst = ()`

d) `lst = ''`

22. Hvordan oppretter du en tom numpy-array med 10 nuller?

a) `np.zeros(10)`

b) `np.empty(10)`

c) `np.ones(10)`

d) `np.arange(10)`

23. Hva gjør `if x % 2 == 0:`?

a) Sjekker om x er et primtall

b) Sjekker om x er lik 0

c) Sjekker om x er et partall

d) Sjekker om x er et oddetall

24. Hva er resultatet av `sorted([3,1,2])`?

a) `[3,1,2]`

b) `[1,2,3]`

c) `[2,3,1]`

d) `{1,2,3}`

25. Hva gjør `np.mean([1,2,3,4])`?

a) 2

b) 2.5

c) 3

d) Error

26. Hvordan skriver du en while-løkke som kjører så lenge `i < 10`?

a) `while i < 10:`

b) `for i in range(10):`

c) `loop (i < 10) { }`

d) `while (i < 10); end`

27. Hvilken funksjon i sympy løser et likningssystem eller en ligning?

a) `sp.solve(...)`

b) `sp.diff(...)`

c) `sp.plot(...)`

d) `sp.integrate(...)`

28. Hva er konsekvensen av å bruke `np.random.seed(42)`?

a) Gjør tilfeldighetsgeneratoren deterministisk

b) Øker tilfeldighetene

c) Nullstiller alle variabler

d) Sletter arrays

29. Hvordan konverterer du en liste `lst` til en pandas Series?

a) `pd.Series(lst)`

b) `pd.DataFrame(lst)`

c) `np.array(lst)`

d) `list(lst)`

Del 2: Kortsvar

Svar kort og presist — maks 4–6 linjer per spørsmål.

1. Forklar forskjellen på `==` og `=` i Python. Gi et kort kodeeksempel.
2. Forklar kort hva en DataFrame er i pandas og nevn to vanlige operasjoner man gjør på en DataFrame.
3. Bruk sympy for å finne kritiske punkter (derivert lik 0) for funksjonen $f(x) = x^3 - 3x^2 + 2$. Skriv de nødvendige uttrykkene (ikke utfør beregningene for hånd).
4. Hva bruker vi simulering til, og hva er fordelen med programmering for å gjennomføre det?
5. Forklar hva koden under gjør

```
    lst = list(range(1, 100))

    for i in lst:
        if i % 2 == 0:
            print(f'{i} er ...')
        elif i % 3 == 0:
            print(f'{i} er ...')
        else:
            print(f'{i} er verken ...')
```

   Svar også på hva det hadde vært naturlig å bytta ut `...` med her. Altså, hva bør stå inne i string-setningene?

Del 3: Programmeringsoppgaver (30 poeng)

Skriv fullstendig Python-kode for begge oppgaver. Kommenter kort hva hver del gjør.

Oppgave 1

En studentbedrift ønsker å evaluere hvor vellykket virksomheten har vært etter årets slutt.

Et vanlig mål på lønnsomhet er å beregne både **fortjeneste (profit)** og **avkastning på investering (Return On Investment, ROI)**.

Definisjoner:

- **Total costs:** hvor mye som er investert i virksomheten (utgifter, kostnader)
- **Total sales:** hvor mye som er tjent inn (inntekter)

Formler:

- Fortjeneste = `total_sales - total_costs`
- $\text{ROI} = \frac{\text{fortjeneste}}{\text{total\_costs}} \times 100\%$

Hvis fortjenesten er negativ, har virksomheten gått med *tap*. Hvis fortjenesten er positiv, har virksomheten gått med *overskudd*.

—

Programmeringsoppgave:
Skriv et Python-program som:

1. Tar inn følgende parametere i en funksjon:
   - Total amount spent / invested (total costs)
   - Total amount raised / earned (total sales)
2. Beregner og skriver ut fortjenesten.
3. Skriver ut en passende melding om bedriften har gått med `profit` eller `loss`.
4. Beregner og skriver ut ROI i prosent.

TIPS: Start med å skrive en pseudokode!

Oppgave 2

I økonomiske applikasjoner er det viktig å beskytte sensitiv informasjon, som for eksempel kredittkortnummer. Dette finner vi i BankID, kort, nettbetaling og så videre. En vanlig teknikk er å vise kun de siste fire sifrene av kortnummeret, mens resten erstattes med stjerner (*).

Eksempel:

- Input: 4925569932906874
- Output: **********6874

Programmeringsoppgave:
Skriv en Python-funksjon `mask_credit_card(card_number)` som:

1. Tar inn et kredittkortnummer som parameter (heltall eller streng).

2. Konverterer kortnummeret til en streng.

3. Erstatter alle sifre unntatt de siste fire med stjerner (`*`).

4. Returnerer det maskerte kortnummeret som en streng.

Tips: - Bruk funksjonen `len()` for å finne lengden på kortnummeret. - Bruk slicing for å hente ut de siste fire sifrene.

—

Eksempel på funksjonskall og resultat:

```
mask_credit_card(4925569932906874)
# Output: "************6874"

mask_credit_card("9876543210987654")
# Output: "************7654"
```

Løsningsforslag

Del 1: Flervalgsoppgaver – riktige svar

1. d) `<class 'float'>`
2. d) En lokasjon i minne hvor data lagres
3. a) Kommentere kode
4. b) `x, y = y, x`
5. b) `Tuple`
6. a) `[1,2,3,4,5]`
7. b) `return`
8. a) `np.array(lst)`
9. a) Viser statistiske oppsummeringer (mean, std, min, max ...)
10. b) `for i in L:`
11. c) 4
12. a) `import sympy as sp; x = sp.symbols('x')`
13. b) `if x == 5:`
14. a) Fjerner rader med minst én NaN-verdi
15. a) `sp.diff(x**2, x)` gir $2x$
16. c) Begge metodene over
17. a) `def f(x): return x*2`
18. a) Lager en linjeplot av y mot x
19. a) `pd.read_csv('fil.csv')`
20. a) `e`
21. b) `lst = []`
22. a) `np.zeros(10)`
23. c) Sjekker om x er et partall
24. b) `[1,2,3]`
25. b) 2.5
26. a) `while i < 10:`
27. a) `sp.solve(...)`
28. a) Gjør tilfeldighetsgeneratoren deterministisk
29. a) `pd.Series(lst)`

Del 2: Kortsvar

1. **Forskjellen mellom = og ==:**
   = brukes til tildeling av en verdi til en variabel, mens == tester om to verdier er like. Eksempel:

```
x = 5       # tildeling
if x == 5:  # sammenligning
    print("x er lik 5")
```

2. **Hva er en DataFrame:**
   En `DataFrame` er en todimensjonal datastruktur i pandas med rader og kolonner, lik et regneark. Vanlige operasjoner inkluderer:

   - Filtrering: `df[df['A'] > 0]` - gjør at vi får en ny dataframe som inneholder kun rader hvor $A > 0$.
     Grundigere forklart (for deres læring i øvingen):

```
import pandas as pd

data = {'A': [-1, 0, 2, 5],
        'B': [10, 20, 30, 40]}
df = pd.DataFrame(data)

df[df['A'] > 0]
```

     Resultat:

```
   A   B
2  2  30
3  5  40
```

     Hva skjer:
     - `df['A'] > 0` lager en boolsk maske som er `True` for rader der verdien i kolonne `A` er større enn 0.
     - `df[ ... ]` bruker masken til å velge kun de radene som oppfyller betingelsen.
     - Resultatet er en ny DataFrame som kun inneholder rader med `A > 0`.
   - Aggregering: `df.groupby('A').mean()` - hvor vi kan gruppere dataen i grupper basert på kolonne A. `.mean()` finner gj.snitt av hver numeriske kolonne, i hver gruppe.
     Grundigere forklart (for deres læring i øvingen):

```
import pandas as pd

data = {'A': ['X', 'Y', 'X', 'Y', 'Z'],
        'B': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)

df.groupby('A').mean()
```

     Resultat:

```
      B
A
X  20.0
Y  30.0
Z  50.0
```

Hva skjer:

- `groupby('A')` deler DataFrame inn i grupper basert på kolonne `A`.
- `.mean()` beregner gjennomsnittet for hver numerisk kolonne i hver gruppe.
- Eksempel: Gruppe X har B-verdier 10 og 30 → gjennomsnitt 20.

- Håndtering av manglende verdier: `df.dropna()` eller `df.fillna(0)` **Grundigere forklart (for deres læring i øvingen):**

```
import pandas as pd
import numpy as np

data = {'A': [1, np.nan, 3],
        'B': [4, 5, np.nan]}
df = pd.DataFrame(data)

# Fjerner rader med minst én NaN-verdi
df_drop = df.dropna()

# Erstatter NaN med 0
df_fill = df.fillna(0)
```

Resultat:

```
df_drop:
     A    B
0  1.0  4.0

df_fill:
     A    B
0  1.0  4.0
1  0.0  5.0
2  3.0  0.0
```

Hva skjer:

- `df.dropna()` fjerner alle rader som inneholder minst én `NaN`-verdi.
- `df.fillna(0)` erstatter alle `NaN`-verdier med 0 (eller en annen verdi du spesifiserer).
- Dette brukes for å rense eller fylle manglende data før videre analyse.

3. **Sympy – kritiske punkter for $f(x) = x^3 - 3x^2 + 2$:**

```
import sympy as sp
x = sp.symbols('x')
f = x**3 - 3*x**2 + 2
df = sp.diff(f, x)
kritiske_punkter = sp.solve(sp.Eq(df, 0), x)
```

4. **Forklaring av koden:**
Koden lager en liste med heltallene fra 1 til 99 (ved hjelp av `range(1, 100)`). Deretter går den gjennom hvert tall i listen med en `for`-løkke, og tester:

- Om tallet er delelig med 2 (`i % 2 == 0`) — i så fall skriver den ut at tallet er et **partall**.
- Hvis ikke, men tallet er delelig med 3 (`i % 3 == 0`) — da skriver den ut at tallet er **delelig med 3**.

- Ellers skriver den at tallet er verken delelig med 2 eller 3.

En naturlig utfylling av ... i utskriftssetningene er derfor:

```
lst = list(range(1, 100))

for i in lst:
    if i % 2 == 0:
        print(f'{i} er et partall')
    elif i % 3 == 0:
        print(f'{i} er delelig med 3')
    else:
        print(f'{i} er verken delelig med 2 eller 3')
```

Del 3: Programmeringsoppgaver

Løsningsforslag oppgave 1: Beregn fortjeneste og avkastning (ROI)

En mulig løsning kan implementeres som en funksjon i Python som tar inn kostnader og salg som parametere, beregner fortjeneste og ROI, og skriver ut resultatene.

```
# Funksjon som beregner fortjeneste og ROI (Return on Investment)

def evaluate_business(total_costs, total_sales):
    # Beregn fortjeneste
    profit = total_sales - total_costs
    print(f"Profit: Euro {profit:.2f}")

    # Sjekk om bedriften har gått med overskudd, tap eller i null
    if profit > 0:
        print("The business made a profit.")
    elif profit < 0:
        print("The business made a loss.")
    else:
        print("The business broke even (no profit or loss).")

    # Beregn ROI (Return on Investment)
    roi = (profit / total_costs) * 100
    print(f"Return on Investment (ROI): {roi:.2f}%")

# Eksempel på bruk:
evaluate_business(5000, 6000)
```

Forklaring av løsningen:

- Funksjonen `evaluate_business()` tar inn to tallverdier: totale kostnader og totale salg.
- Den beregner fortjenesten som forskjellen mellom salg og kostnader.
- Den vurderer om resultatet er positivt (overskudd), negativt (tap) eller null (balanse).
- Til slutt beregnes ROI som en prosentandel av kostnadene, og verdiene skrives ut med to desimaler.

Eksempel på utskrift:

```
Profit: €1000.00
The business made a profit.
Return on Investment (ROI): 20.00%
```

Løsningsforslag oppgave 2

Her (som alle programmeringsoppgaver) er det flere fremgangsmåter. Vi har valgt å:

- Lage en funksjon som tar inn `card_number`.
- `card_number` konverteres til string.
- Bruker *slicing*, altså at vi bruker indeksering for å returnere det vi ønsker. Som er, mange stjerner "***"(så mange som kortnummeret inneholder minus 4), pluss de fire siste sifferene.

```python
def mask_credit_card(card_number):
    # Konverter kortnummeret til streng
    card = str(card_number)
    # Lag en streng med stjerner og legg til de siste 4 sifrene
    return "*" * (len(card) - 4) + card[-4:]

# Eksempelbruk
print(mask_credit_card(1234567812345678))  # ************5678
print(mask_credit_card("9876543210987654"))  # ************7654
```