Prøveeksamen i BED-1304 Python-Lab

Lagd av: Markus J. Aase

Høst 2025

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.

- **Del 1:** 40 flervalgsoppgaver. Ett riktig svar per oppgave.
- **Del 2:** 5 kortsvarsoppgaver som tester kodeforståelse, pseudokode og feilsøking.
- **Del 3:** 2 praktiske programmeringsoppgaver hvor du skal skrive kode for å løse et problem.

Lykke til!

Legg merke til at dette ikke nødvendigvis dekker **hele** pensum. Dere står selv ansvarlig for å kunne det nødvendige fra pensum.

Del 1: Flervalgsoppgaver (40 poeng)

1. Hva skriver ‘print(type("123"))‘ ut til konsollen?

   a) `<class 'int'>`

   b) `<class 'string'>`

   c) `<class 'str'>`

   d) `<class 'float'>`

2. Hvilken operator brukes for eksponentiering (opphøyd i) i Python?

   a) `^`

   b) `**`

   c) `exp()`

   d) `pow`

3. Hva er resultatet av regnestykket ‘7 // 2‘?

   a) `3.5`

   b) `3`

   c) `1`

   d) `Error`

4. Hvordan definerer man en funksjon korrekt i Python?

   a) `function min_funk():`

   b) `def min_funk():`

   c) `define min_funk():`

   d) `function = min_funk():`

5. Hva er formålet med ‘return‘-kommandoen i en funksjon?

   a) Å skrive ut en verdi til skjermen.

   b) Å avslutte funksjonen og eventuelt sende en verdi tilbake.

   c) Å definere en ny variabel.

   d) Å importere en pakke.

6. Hvilket bibliotek (pakke) er mest sentralt for numeriske beregninger og håndtering av lister/arrays?

   a) `matplotlib`

   b) `pandas`

   c) `sympy`

   d) `numpy`

7. Hva returnerer ‘numpy.arange(1, 5)‘?

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

   b) `[0, 1, 2, 3, 4]`

   c) `[1, 2, 3, 4]`

d) `[1, 5]`

8. Hvis du har en numpy-array ‘a = np.array([10, 40, 20, 30])‘, hva vil ‘np.argsort(a)‘ returnere?

   a) `[10, 20, 30, 40]`
   b) `[40, 30, 20, 10]`
   c) `[0, 2, 3, 1]`
   d) `[1, 3, 2, 0]`

9. Hva er den primære datastrukturen i Pandas-biblioteket?

   a) Array
   b) List
   c) Dictionary
   d) DataFrame

10. Hvordan leser du typisk inn en CSV-fil til en Pandas DataFrame?

    a) `pd.load_csv('fil.csv')`
    b) `pd.read_csv('fil.csv')`
    c) `pd.open('fil.csv')`
    d) `pd.DataFrame.from_csv('fil.csv')`

11. Gitt en DataFrame ‘df‘, hvordan velger du kun kolonnen med navnet ’BNP’?

    a) `df.get('BNP')`
    b) `df.column['BNP']`
    c) `df['BNP']`
    d) `df(kolonne='BNP')`

12. Hvilken Pandas-metode brukes for å beregne gjennomsnittet av en kolonne?

    a) `.average()`
    b) `.mean()`
    c) `.median()`
    d) `.std()`

13. Hvilket nøkkelord starter en betingelsessetning (conditional statement)?

    a) `check`
    b) `when`
    c) `if`
    d) `for`
    e) `while`

14. Hva gjør en ‘for‘-løkke?

    a) Kjører kode så lenge en betingelse er sann.
    b) Definerer en funksjon.
    c) Itererer (går gjennom) en sekvens av elementer.

d) Stopper programmet.

15. Hvilket bibliotek importeres vanligvis som ‘plt‘?

    a) `pandas`
    b) `matplotlib.pyplot`
    c) `numpy`
    d) `sympy`

16. Hvordan gir du en tittel til et ‘matplotlib‘-plot?

    a) `plt.header("Min tittel")`
    b) `plt.add_title("Min tittel")`
    c) `plt.label("Min tittel")`
    d) `plt.title("Min tittel")`

17. Hva brukes ‘sympy‘ primært til?

    a) Datavisualisering
    b) Symbolsk matematikk
    c) Filbehandling
    d) Maskinlæring

18. Hvordan definerer du ‘x‘ og ‘y‘ som symbolske variabler i ‘sympy‘?

    a) `x, y = sp.var('x, y')`
    b) `x, y = sp.make('x', 'y')`
    c) `x, y = sp.symbols('x y')`
    d) `x, y = symbols.sp('x y')`

19. Hva er hensikten med en ‘while‘-løkke?

    a) Å iterere gjennom en liste.
    b) Å kjøre en kodeblokk gjentatte ganger så lenge en betingelse er sann.
    c) Å kjøre en kodeblokk et bestemt antall ganger.
    d) Å håndtere feil i koden.

20. Hvordan konverterer du en streng ‘s = "50"‘ til et heltall?

    a) `convert.int(s)`
    b) `s.to_integer()`
    c) `int(s)`
    d) `integer(s)`

21. Hva er den korrekte formelen for sluttverdi (K_T) med kontinuerlig forrentning?

    a) $K_T = K_0(1 + r/n)^{nT}$
    b) $K_T = K_0(1 + r)^T$
    c) $K_T = K_0 e^{rT}$
    d) $K_T = K_0(1 + rT)$

22. Gitt en DataFrame ‘df‘, hvordan filtrerer du for å få rader der kolonnen ’År’ er større enn 2000?

    a) `df.filter(df['År'] > 2000)`
    b) `df[df['År'] > 2000]`
    c) `df.loc['År' > 2000]`
    d) `df.where('År' > 2000)`

23. Hva er resultatet av ‘(True and False) or True‘?

    a) `True`
    b) `False`
    c) `Error`
    d) `None`

24. Hvilken funksjon i ‘matplotlib‘ lager et stolpediagram?

    a) `plt.plot()`
    b) `plt.hist()`
    c) `plt.bar()`
    d) `plt.scatter()`

25. Hvilken ‘sympy‘-funksjon brukes til å derivere et uttrykk?

    a) `sp.solve()`
    b) `sp.integrate()`
    c) `sp.limit()`
    d) `sp.diff()`

26. Hvilken funksjon fra ‘numpy.random‘ vil du bruke for å simulere ett enkelt terningkast (1-6)?

    a) `np.random.rand(1, 6)`
    b) `np.random.randint(1, 7)`
    c) `np.random.choice(6)`
    d) `np.random.random(6)`

27. Hva er forskjellen på ‘=‘ og ‘==‘ i Python?

    a) Ingen forskjell, de kan brukes om hverandre.
    b) ‘=‘ er for sammenligning, ‘==‘ er for tilordning.
    c) ‘=‘ er for tilordning, ‘==‘ er for sammenligning.
    d) ‘==‘ er for lister, ‘=‘ er for tall.

28. Hva returnerer en funksjon som ikke har en ‘return‘-setning?

    a) `0`
    b) (en tom streng)
    c) `False`
    d) `None`

29. Hvordan får du tak i det siste elementet i en liste som heter ‘min_liste‘?

    a) `min_liste.last()`
    b) `min_liste[len(min_liste)]`
    c) `min_liste[-1]`
    d) `min_liste[end]`

30. Hvilken metode brukes for å legge til et element på slutten av en liste?

    a) `.add()`
    b) `.push()`
    c) `.insert()`
    d) `.append()`

31. Hva gjør ‘df.describe()‘ på en Pandas DataFrame?

    a) Viser de første 5 radene.
    b) Plotter dataene.
    c) Viser deskriptiv statistikk for numeriske kolonner.
    d) Beskriver datatypene i hver kolonne.

32. Hva betyr ‘elif‘?

    a) En forkortelse for element i for-løkke".
    b) En obligatorisk del av en ‘if‘-setning.
    c) En forkortelse for else if", og tester en ny betingelse.
    d) En måte å avslutte en løkke på.

33. Hvilken funksjon i ‘sympy‘ løser ligninger?

    a) `sp.diff()`
    b) `sp.solve()`
    c) `sp.simplify()`
    d) `sp.eq()`

34. Hva er *Loven om store tall"* i kontekst av simulering?

    a) Sannsynligheten for en hendelse er alltid 50/50.
    b) Gjennomsnittet av resultatene fra mange forsøk vil nærme seg den forventede verdien.
    c) Jo flere simuleringer, jo mer tilfeldig blir resultatet.
    d) Man må bruke store tall for at simuleringer skal fungere.

35. Hva blir resultatet av koden under?

```
min_liste = [10, 20, 30]
min_liste[1] = 5
print(min_liste)
```

    a) `[10, 20, 30]`
    b) `[5, 20, 30]`
    c) `[10, 5, 30]`

d) `Error`

36. Hva blir datatypen til variabelen ‘resultat‘ i koden ‘resultat = 100 / 4‘?

    a) `int`
    b) `float`
    c) `str`
    d) `numpy.array`

37. For å finne det året et land hadde høyest BNP per capita i en DataFrame ‘df‘, hva kan du gjøre?

    a) `df['BNP_per_capita'].max()`
    b) `df.loc[df['BNP_per_capita'].idxmax()]`
    c) `df.groupby('År')['BNP_per_capita'].max()`
    d) `df.sort_values('BNP_per_capita').first()`

38. Du har en ‘while‘-løkke: ‘x = 0; while x < 5: print(x)‘. Hva mangler for at løkken ikke skal kjøre evig?

    a) En ‘if‘-setning.
    b) En ‘break‘-kommando.
    c) En ‘return‘-setning.
    d) En oppdatering av ‘x‘ (f.eks. ‘x = x + 1‘).

39. Hvilken kodebit lager et enkelt linjediagram av ‘y‘ mot ‘x‘?

    a) `plt.plot(y, x)`
    b) `plt.scatter(x, y)`
    c) `plt.plot(x, y)`
    d) `plt.bar(x, y)`

40. Hva vil ‘sp.diff(3*x**2 + 2*x, x)‘ gi i Sympy?

    a) `x**3 + x**2`
    b) `6*x + 2`
    c) `3*x + 2`
    d) `6*x`

Del 2: Kortsvar (30 poeng)

1. **Kodeforståelse:** Hva blir den endelige verdien til variabelen 'resultat' etter at denne koden er kjørt? Forklar kort hvorfor.

```
profitt_marginer = [0.1, -0.05, 0.2, 0.05, -0.1]
resultat = 1
for margin in profitt_marginer:
    if margin > 0:
        resultat = resultat * (1 + margin)
```

2. **Pseudokode:** En bedrift ønsker å beregne den totale lønnskostnaden for en måned. Skriv pseudokode for en algoritme som gjør følgende:

   - Tar en liste med timelønner for alle ansatte.
   - Ganger hver timelønn med et fast antall timer (150 timer).
   - Summerer alle de individuelle lønningene for å finne den totale kostnaden.
   - Hvis totalkostnaden overstiger 1 000 000 kr, skal den skrive ut *Budsjett overskredet*. Ellers, skriv ut den totale kostnaden.

3. **Kodeforklaring:** Forklar hva koden under gjør, linje for linje.

```
import pandas as pd
data = {'Land': ['Norge', 'Sverige', 'Norge', 'Sverige'],
        'Year': [2020, 2020, 2021, 2021],
        'Vekst': [1.1, 1.3, 1.2, 1.4]}
df = pd.DataFrame(data)
avg_vekst = df.groupby('Land')['Vekst'].mean()
print(avg_vekst)
```

4. **Feilsøking:** Finn og fiks feilen(e) i Python-funksjonen under. Forklar hva som var galt.

```
def beregn_naaverdi(sluttverdi, rente, aar)
naaverdi = sluttverdi / (1 + rente)**aar
    return naaverdi
```

5. **Anvendelse av Sympy:** Du har en kostnadsfunksjon for en bedrift gitt ved $C(Q) = 100 + 10Q + 0.5Q^2$, hvor Q er produsert kvantum. Hvordan vil du bruke 'sympy' til å finne bedriftens marginalkostnad, $MC(Q)$? Beskriv stegene.

Del 3: Programmeringsoppgaver (30 poeng)

1. **Funksjon for nedbetaling av lån**

   Du skal skrive en Python-funksjon, 'beregn_nedbetalingstid()', som beregner hvor mange måneder det tar å nedbetale et lån.

   Funksjonen skal:

   - Ta inn tre argumenter: 'laanebelop' (startbeløp), 'aarlig_rente' (f.eks. 0.05 for 5%), og 'maanedsbelop' (fast månedlig innbetaling).
   - Bruke en 'while'-løkke for å simulere nedbetalingen måned for måned.
   - For hver måned skal renten legges til det gjenværende beløpet FØR det månedlige beløpet trekkes fra. Husk å konvertere årlig rente til månedlig rente.
   - Funksjonen skal returnere antall måneder det tok å nedbetale lånet.
   - Kommenter koden nøye.

   Hint: Månedlig rente = 'aarlig_rente / 12'.

2. **Dataanalyse med Pandas**

   Anta at du har en Pandas DataFrame kalt 'salgsdata' som inneholder salgsinformasjon.

```
# Du kan bruke denne koden for å lage DataFrame for testing
import pandas as pd
data = {
    'Dato': ['2024-05-01', '2024-05-01', '2024-05-02', '2024-05-03',
    '2024-05-03'],
    'Produktkategori': ['Elektronikk', 'Matvarer', 'Elektronikk', 'Klær', '
    Matvarer'],
    'Inntekter': [5000, 1500, 8000, 3000, 2000],
    'Kostnader': [3500, 1200, 6000, 1800, 1600]
}
salgsdata = pd.DataFrame(data)

```

   Skriv Python-kode som utfører følgende oppgaver:

   a) Lag en ny kolonne i 'salgsdata' som heter ''Profitt'', som er differansen mellom ''Inntekter'' og ''Kostnader''.

   b) Beregn den totale profitten for hver ''Produktkategori''.

   c) Finn og skriv ut navnet på produktkategorien med den høyeste totale profitten.

Løsningsforslag

Del 1: Flervalgsoppgaver

1. c) `<class 'str'>`
2. b) `**`
3. b) `3` (heltallsdivisjon)
4. b) `def min_funk():`
5. b) Å avslutte funksjonen og eventuelt sende en verdi tilbake.
6. d) `numpy`
7. c) `[1, 2, 3, 4]`
8. c) `[0, 2, 3, 1]` (indeksene som sorterer arrayen)
9. d) DataFrame
10. b) `pd.read_csv('fil.csv')`
11. c) `df['BNP']`
12. b) `.mean()`
13. c) `if`
14. c) Itererer (går gjennom) en sekvens av elementer.
15. b) `matplotlib.pyplot`
16. d) `plt.title("Min tittel")`
17. b) Symbolsk matematikk
18. c) `x, y = sp.symbols('x y')`
19. b) Å kjøre en kodeblokk gjentatte ganger så lenge en betingelse er sann.
20. c) `int(s)`
21. c) $K_T = K_0 e^{rT}$
22. b) `df[df['År'] > 2000]`
23. a) `True`
24. c) `plt.bar()`
25. d) `sp.diff()`
26. b) `np.random.randint(1, 7)`
27. c) '=' er for tilordning, '==' er for sammenligning.
28. d) `None`
29. c) `min_liste[-1]`

30. d) `.append()`

31. c) Viser deskriptiv statistikk for numeriske kolonner.

32. c) En forkortelse for "**else if**", og tester en ny betingelse.

33. b) `sp.solve()`

34. b) Gjennomsnittet av resultatene fra mange forsøk vil nærme seg den forventede verdien.

35. c) `[10, 5, 30]`

36. b) `float` (standard divisjon i Python 3 gir alltid float)

37. b) `df.loc[df['BNP_per_capita'].idxmax()]`

38. d) En oppdatering av 'x' (f.eks. 'x = x + 1').

39. c) `plt.plot(x, y)`

40. b) `6*x + 2`

Del 2: Kortsvar

1. **Svar:** Den endelige verdien blir `1.38`. **Forklaring:** Løkken går gjennom listen `profitt_marginer`. Den multipliserer kun 'resultat' med '(1 + margin)' for positive marginer. Regnestykket blir: '1 * (1 + 0.1) * (1 + 0.2) * (1 + 0.05)', som er '1 * 1.1 * 1.2 * 1.05 = 1.386'. Å forklare dette er nok svar. **1.4 er nære nok for en eksamen.** Korrekt svar er 1.386.

2. **Svar (Pseudokode):**

```
START
  SET timelonner = [liste med lønner]
  SET total_kostnad = 0

  FOR hver lonn I timelonner:
      SET individuell_lonn = lonn * 150
      ADD individuell_lonn TIL total_kostnad
  ENDFOR

  IF total_kostnad > 1000000:
      PRINT "Budsjett overskredet"
  ELSE:
      PRINT "Total kostnad er:", total_kostnad
  ENDIF
END
```

3. **Svar (Kodeforklaring):**

   - `import pandas as pd`: Importerer pandas-biblioteket og gir det aliaset 'pd'.
   - `data = {...}`: Oppretter en Python dictionary som inneholder data for land, år og vekst.
   - `df = pd.DataFrame(data)`: Konverterer dictionaryen til en Pandas DataFrame kalt 'df'.

- `avg_vekst = df.groupby('Land')['Vekst'].mean()`: Grupperer DataFrame-en etter unike verdier i 'Land'-kolonnen, velger 'Vekst'-kolonnen for hver gruppe, og beregner gjennomsnittet. Resultatet lagres i 'avg_vekst'.
- `print(avg_vekst)`: Skriver ut resultatet, som vil være en Pandas Series med gjennomsnittlig vekst for Norge og Sverige.

4. **Svar (Feilsøking): Feil:**

   1. Mangler kolon (':') på slutten av 'def'-linjen.
   2. Linjen 'naaverdi = ...' er ikke rykket inn (indented).

   Riktig kode:

```python
def beregn_naaverdi(sluttverdi, rente, aar):
    naaverdi = sluttverdi / (1 + rente)**aar
    return naaverdi
```

5. **Svar (Anvendelse av Sympy):** Marginalkostnaden er den deriverte av kostnadsfunksjonen med hensyn på kvantum, $MC(Q) = C'(Q)$.

   1. **Importer Sympy:** `import sympy as sp`
   2. **Definer symbol:** Definer Q som en symbolsk variabel: `Q = sp.symbols('Q')`
   3. **Definer funksjonen:** Skriv inn kostnadsfunksjonen: `C = 100 + 10*Q + 0.5*Q**2`
   4. **Deriver:** Bruk 'sp.diff()' for å derivere uttrykket med hensyn på Q: `MC = sp.diff(C, Q)`
   5. **Vis resultatet:** `print(MC)` vil da vise marginalkostnadsfunksjonen.

Del 3: Programmeringsoppgaver

1. **Løsning (Funksjon for nedbetaling av lån):**

```python
def beregn_nedbetalingstid(laanebelop, aarlig_rente, maanedsbelop):
    """
    Beregner hvor mange måneder det tar å nedbetale et lån.

    Parametere:
    laanebelop (float): Startbeløp på lånet (i kroner).
    aarlig_rente (float): Årlig rente (f.eks. 0.05 for 5%).
    maanedsbelop (float): Fast månedlig innbetaling (i kroner).

    Returnerer:
    int: Antall måneder det tok å nedbetale lånet.

    Merk:
    Renten legges til lånebeløpet hver måned før innbetalingen trekkes fra.
    Hvis månedlig beløp er for lavt (dvs. lavere enn rentekostnaden),
    vil løkken aldri stoppe.
    """
    maanedlig_rente = aarlig_rente / 12
    maaneder = 0

    while laanebelop > 0:
        # Legg til renter
        laanebelop = laanebelop * (1 + maanedlig_rente)
        # Trekk fra månedlig betaling
        laanebelop -= maanedsbelop
        maaneder += 1

```

```
        # Hvis lånet vokser i stedet for å krympe
        if maaneder > 10**6:  # stopper uendelig løkke
            raise ValueError("Månedlig beløp er for lavt til å nedbetale lånet.")

    return maaneder


# Eksempelbruk, 3.000.000,- i lån, 5,3% årlig rente, 17.000,- i nedbetaling i måneden
print(beregn_nedbetalingstid(laanebelop=3000000, aarlig_rente=0.053, maanedsbelop=17000))
```

2. Løsning (Dataanalyse med Pandas):

```
import pandas as pd

# Oppsett av DataFrame (som gitt i oppgaven)
data = {
    'Dato': ['2024-05-01', '2024-05-01', '2024-05-02', '2024-05-03', '2024-05-03'],
    'Produktkategori': ['Elektronikk', 'Matvarer', 'Elektronikk', 'Klær', 'Matvarer'],
    'Inntekter': [5000, 1500, 8000, 3000, 2000],
    'Kostnader': [3500, 1200, 6000, 1800, 1600]
}
salgsdata = pd.DataFrame(data)

# a) Lag en ny kolonne for profitt
salgsdata['Profitt'] = salgsdata['Inntekter'] - salgsdata['Kostnader']

# b) Beregn total profitt per produktkategori
total_profitt_per_kategori = salgsdata.groupby('Produktkategori')['Profitt'].sum()

# c) Finn kategorien med hoyest profitt
hoyest_profitt_kategori = total_profitt_per_kategori.idxmax()

# Skriv ut resultatene
print("Total profitt per kategori:")
print(total_profitt_per_kategori)
print("-" * 30)
print(f"Kategorien med hoyest profitt er: {hoyest_profitt_kategori}")
```