Løsningsforslag TDT4110 H25 1500

Emne: TDT4110 Informasjonsteknologi grunnkurs
Eksamensdato: mandag 8. desember 2025 (1500-1900)

Kommentar:
Et vedlegg «Nyttige funksjoner» manglet på eksamen. Dette er hensyntatt i sensuren.

1 Datatyper (5p)

Vi har koden:

u = [1.0, 2, "Feliz"]
v = "Navidad"
N = "None"

I tabellen nedenfor står en del uttrykk, hvorav noen vil kunne beregnes og få en verdi, mens andre vil gi feilmelding.

For hver rad, se på uttrykket til venstre, og kryss av hvilken datatype resultatet vil bli - eller velg ERROR dersom uttrykket vil gi feilmelding. Ingen minuspoeng for feil svar, så du BØR svare noe også der du er usikker.

	int	float	bool	string	list	ERROR
2.0 ** 2	○	✔	○	○	○	○
9 // 2	✔	○	○	○	○	○
v + N	○	○	○	✔	○	○
u - ["Feliz"]	○	○	○	○	○	✔
v[u[0]]	○	○	○	○	○	✔
N == "None"	○	○	✔	○	○	○
2 * v + N	○	○	○	✔	○	○
False * 1	✔	○	○	○	○	○
if N	○	○	○	○	○	✔

2 Eksakt (3p)

Hvilke av disse tallene kan representeres eksakt som enten heltall eller flyttall i Python?

Hvis man skriver uttrykket i venstre kolonne, så velg "Heltall" hvis tallet lagres som heltall, "Eksakt flyttall" hvis det lagres som flyttall med den eksakte verdien til uttrykket, "Ikke eksakt flyttall" hvis det lagres med en annen verdi enn det som står i uttrykket. Hvis uttrykket gir en feil, så velg ERROR.

For hvert tall, sett kryss i rett kolonne. Ingen minuspoeng for feil svar, så du BØR svare noe også der du er usikker.

	Heltall	Eksakt flyttall	Ikke eksakt flyttall	ERROR
1 / 8	○	✔	○	○
1 / 5	○	○	✔	○
2 ** 10000	✔	○	○	○
1.25e-5	○	✔	○	○
0.5 ** 10000	○	○	✔	○
10.0 ** 1000	○	○	○	✔

3 Numerikk (4p)

Bildet under viser kode med utskrift. Under kjøreresultatene står noen påstander knyttet til koden og resultatene.

Husk at matematisk så er

- $\sin(2k\pi) = 0$ for alle heltall k
- $\cos(2k\pi) = 1$ for alle heltall k
- $(\cos x)^2 = 1 - (\sin x)^2$ for alle x

```
import math
for n in range(1, 30, 4):
    a = math.sin(2**n * math.pi)
    b = math.cos(2**n * math.pi)
    print(f"{n}\t{a}\t{b}\t{1.0-a**2}")
```

✓ 0.0s

```
1       -2.4492935982947064e-16 1.0     1.0
5       -3.91886975727153e-15   1.0     1.0
9       -6.270191611634448e-14  1.0     1.0
13      -1.0032306578615117e-12 1.0     1.0
17      -1.6051690525784188e-11 1.0     1.0
21      -2.56827048412547e-10   1.0     1.0
25      -4.109232774600752e-09  1.0     1.0
29      -6.574772439361198e-08  0.9999999999999979      0.9999999999999957
```

For hver påstand, kryss av om den er korrekt eller feil.

	Feil	Korrekt
De første radene i siste kolonne blir 1.0 fordi a**2 gir underflyt.	✔	
Alle radene til siste kolonne har avrundingsfeil fordi 1.0 er mye større enn a**2.		✔
Verdiene i andre kolonne er forskjellig fra 0, og andre kolonne er lik 1, fordi vi flyttall ligger mye tettere rundt 0 enn de gjør rundt pi og 1.		✔
Verdiene i den andre kolonnen øker fordi avrundingsfeilen mellom matematisk pi og math.pi blir forstørret ved multplikasjon med 2**n		✔

4 Unntak (3p)

Følgende Python-funksjon er gitt:

```
def do_stuff2(s):
    try:
        x = float(s)
        y = (x**2 - x) / (x - 1)
    except ZeroDivisionError:
        y = 1.0
    except Exception as e:
        y = -1.0
    finally:
        print(y)
```

Under skal du fylle inn det som skrives ut når funksjonen kalles med forskjellige argumenter. Hvis funksjonen ikke skriver ut noe, så skriv "ingenting" (uten anførsetstegn).
Hvis funskjonen ikke returnerer på vanlig måte pga feil, skriv ERROR.

do_stuff("2") : _____ (2.0, 2)

do_stuff("1"): _____ (1.0, 1)

do_stuff("tull"): _____ (-1.0, -1)

5 Slicing (4p)

Vi har et numpy-array **arr** som i en pen printvisning er som følger:

```
[[11  12  13  14  15]
 [21  22  23  24  25]
 [31  32  33  34  35]
 [41  42  43  44  45]
 [51  52  53  54  55]]
```

For hvert slice-uttrykk under, kryss av for hvilke tall som vil inngå i resultatet, eller kryss av for ERROR dersom uttrykket vil gi feilmelding.

arr[1:3, 2:]
Velg ett alternativ:

- 22, 23, 24, 32, 33, 34
- 23, 24, 33, 34
- ERROR
- 23, 24, 25, 33, 34, 35 ✔

arr[: , 2]
Velg ett alternativ

- ERROR
- 14, 24
- 14, 24, 34, 44, 54
- 13, 23, 33, 43, 53 ✔

arr[3:, 1:3]
Velg ett alternativ

- 42, 43, 52, 53 ✔
- 43, 53
- ERROR
- 43, 44, 53, 54

```
temp = arr[:,1].copy()
arr[:,1] = arr[:,2]
arr[:,2] = temp
```
Velg ett alternativ

- Kolonna 13,23,33,43,53 blir endra til 12,22,32,42,52. Kolonna 12,22,32,42,52 er uendra.
- ERROR
- Kolonnene 12,22,32,42,52 og 13,23,33,43,53 vil innbyrdes bytte innhold ✔
- Kolonna 12,22,32,42,52 blir endra til 13,23,33,43,53. Kolonna 13,23,33,43,53 er uendra.

6 Zoologi ordbok (6p)

En klubb for dyreentusiaster holder rede på hvilke eksotiske dyr hvert medlem har tatt selfies med. Disse opplysningene er lagret i en dictionary med følgende format:

```
zoo_club = {
    "Ada": {"gnu", "løve", "tiger", "elefant", "sjiraff", "sebra"},
    "Bob": {"løve", "gnu"},
    "Dan": {"panda", "gnu", "koala", "sjiraff", "tiger"},
    "Una": {"gnu", "sjiraff", "pingvin", "panda", "elefant", "tiger"},
    "Eva": {"gnu", "løve", "elefant", "sjiraff", "sebra", "wombat"},
    "Frank": {"gnu", "panda", "koala", "elefant"},
    "Grace": {"panda", "gnu"},
    "Hank": {"gnu", "panda", "koala", "sjiraff"},
    "Ivy": {"gnu", "sjiraff", "panda", "tiger"},
    "Jack": {"løve", "gnu", "elefant", "sjiraff", "pingvin", "koala"}
}
```

Som bildet viser, er hver nøkkel i dictionary navnet på et medlem, og hver verdi er et sett med dyrene som vedkommende har selfies med. Klubben ønsker nå å lage noen funksjoner som skal analysere dataene. De to funksjonene er:

felles_dyr(zoo_club, medlem_a, medlem_b) skal returnere mengda av dyr som både medlem_a og medlem_b har selfies med. Eksempel: felles_dyr("Ada", "Bob") returnerer {'gnu', 'løve'}.

eksklusive_dyr(zoo_club, medlem) skal returnere mengda av dyr som bare dette ene medlemmet og ingen andre i klubben har. Eksempel: for "Eva" returnes {"wombat"}.

Trekk kodefragmenter til rett sted slik at de to funksjonene virker som de skal. NB: Samme kodefragment kan potensielt brukes flere steder i oppgaven, og det kan også finnes fragment som ikke skal brukes i det hele tatt.

Hjelp

intersection	discard	difference	zoo_club[medlem]
add	symmetric_difference	medlem	union
zoo_club[person]			

def felles_dyr(zoo_club, medlem_a, medlem_b):

return zoo_club[medlem_a] . intersection (zoo_club[medlem_b])

def eksklusive_dyr(zoo_club, medlem):

```
mengde = set()
for person in zoo_club:
    if person != medlem :
        mengde = mengde. union ( zoo_club[person] )
return zoo_club[medlem] . difference (mengde)
```

7 Filbruk (6p)

Vi har ei tekstfil med tall over flere linjer, innad i hver linje adskilt med mellomrom. Fila har like mange tall på hver linje. Funksjonen **edit_data(filename)** får inn et filnavn og skal endre data fra denne fila ved at alle tall skal **rundes ned** til nærmeste fjerdedel. Forøvrig skal data være uendret. Reviderte data skal skrives til ei fil med navn **rev_** pluss gammelt filnavn.

Eksempel, hvis fila **tall.txt** før inneholdt:

2.29 1.69 3.0 5.25 1.9

1.75 3.07 4.5 0.88 5.53

så skal kjøring av **edit_data('tall.txt')** lage fila **rev_tall.txt** med disse tallene:

2.25 **1.5** 3.0 5.25 **1.75**

1.75 **3.0** 4.5 **0.75** **5.5**

OPPGAVE: Trekk kodelinjer til riktig sted så funksjonen virker som den skal. 3 av kodelinjene skal ikke brukes.

Hjelp

np.loadtxt(filename, data)	data[i, j] -= (data[i,j] % 0.25)
if row[j] % 0.25 != 0:	np.savetxt(f'rev_{filename}', data)
for j in range(len(row)):	filename = np.savetxt(data)
data = np.loadtxt(filename)	for row in data:
row[j] -= (row[j] % 0.25)	

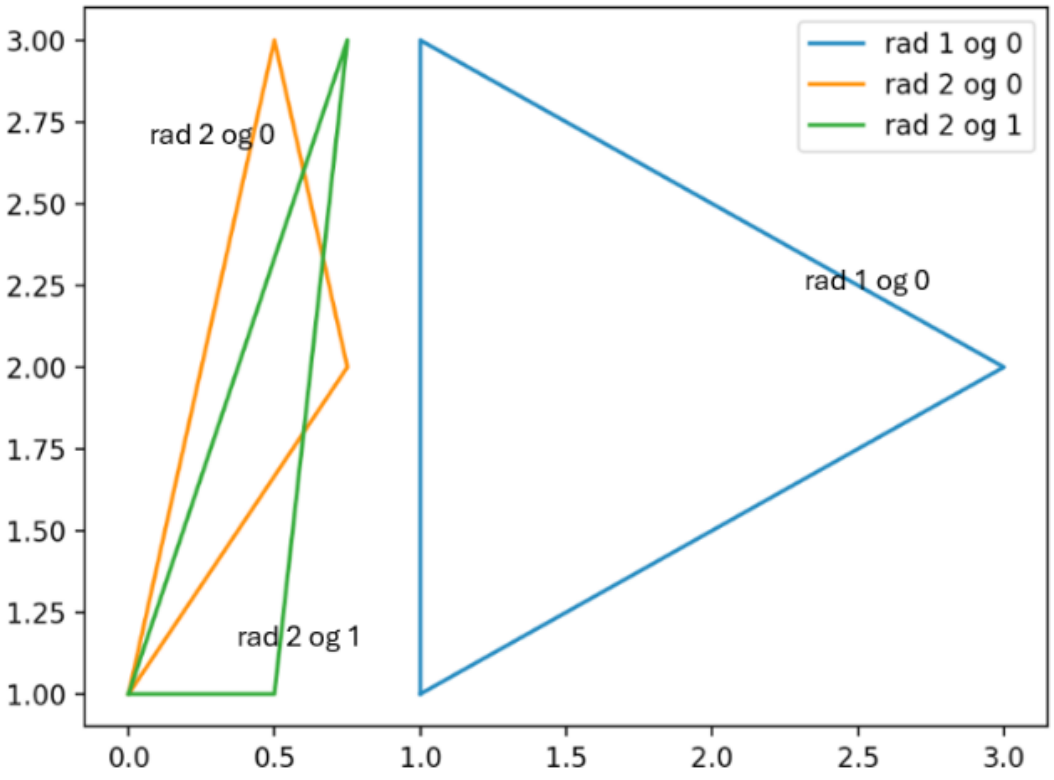
```
import numpy as np
def edit_data(filename):
    data = np.loadtxt(filename)
    for row in data:
        for j in range(len(row)):
            if row[j] % 0.25 != 0:
                row[j] -= (row[j] % 0.25)
    np.savetxt(f'rev_{filename}', da
```

8 Plotte fra fil (7p)

Generelt hint i denne oppgaven: Antall rader i et 2d-array **a** kan finnes på flere ulike måter. En måte er **len(a) -** hvor len(a[0]) eller hvilken som helst annen indeks innenfor arrayet vil være antall kolonner. En annen måte er **a.shape[0]** - hvor a.shape[1] tilsvarende vil være antall kolonner fordi a.shape generelt er et tuppel hvor første element er antall rader, andre kolonner (og evt. flere tall i tuppelet hvis det er et array med mer enn 2 dimensjoner). Dvs. hvis et array har 3 rader, 4 kolonner vil a.shape være (3, 4), len(a) være 3, og len(a[0]) være 4.

Funksjonen **plot_triangles(filename)** får inn navnet på ei fil, og skal plotte trekanter basert på tall fra fila. Det kan antas at alle rader på fila har nøyaktig fire tall, og at kolonnene lengst til venstre og høyre er identiske, slik at hvert plott alltid kommer tilbake til startpunktet og lager en lukket trekant. Koordinater for hver trekant fås ved å kombinere to ulike rader med tall, men alltid med den øverste av radene som y-verdier og nederste som x-verdier.

Eksempel: hvis innholdet på fila er:
1.0 3.0 2.0 1.0
1.0 1.0 3.0 1.0
0.0 0.5 0.75 0.0
så gir rad 1 og 0 i tabellen den blå trekanten til høyre i bildet med koordinater (1.0, 1.0), (1.0, 3.0), (3.0, 2.0); og kombinasjon av rad 2 og 0 og rad 2 og 1 gir de to andre trekantene.



OPPGAVE: Trekk kodefragment til riktig plass så programmet virker som det skal.
3 av kodelinjene skal ikke brukes.

plt.show()	np.loadtxt(filename, a)	a = np.savetxt(filename)
import matplotlib.pyplot as plt	1, a.shape[0]	plt.subplots()
a.shape[1]	a[i], a[j]	a = np.loadtxt(filename)
i		

```
import numpy as np
import matplotlib.pyplot as ✔
def plot_triangles(filename):
    fig, ax = plt.subplots() ✔
    a = np.loadtxt(filename) ✔
    for i in range( 1, a.shape[0] ✔ ):
        for j in range( i ✔ ):
            ax.plot( a[i], a[j] ✔ )

    plt.show() ✔
```

9 Collatz (10p)

Collatz-sekvensen med start i det positive heltallet n får vi ved å finnes neste tall ved å

- dele på 2 hvis tallet er delelig med 2
- gange med 3 og legge til 1, hvis tallet ikke er delelig med 2

Vi fortsetter sekvensen ved å gjenta operasjonen på forrige tall.

Eksempel:
Hvis vi starter med n = 5, får vi følgende sekvens:
5,
5*3 + 1 = 16,
16 / 2 = 8,
8 / 2 =4,
4 /2 = 2,
2 / 2 = 1,
1 * 3 + 1 = 4,
Hvis sekvensen kommer til 1, så vil den fortsette i en løkke: 4, 2, 1, 4, 2, 1, ...

Vi definerer Collatz-tallet til n til å være antall steg i sekvensen før vi får verdien 1. For n = 5 så er Collatz-tallet lik 5, mens Collatz-tallet til 1 er lik 0.

Funksjonen **collatz(limit)** skal returnere en liste av lengde limit + 1, hvor elementet med indeks n er Collatz-tallet til n.

Trekk kodelinjer til riktig plass så funksjonen virker som den skal:

Hjelp

length = 0	length += 1	if k % 2 == 0
for n in range(2, limit + 1)	n	sequence_lengths = [None, 0]
k = 3 * k + 1	length + sequence_lengths[k]	while k >= n
k //= 2		

```
def collatz(limit):
    sequence_lengths = [None,
    for n in range(2, limit + 1) :
        length = 0
        k = n
        while k >= n :
            length += 1
            if k % 2 == 0 :
                k //= 2
            else :
                k = 3 * k + 1

        total_length = length + sequence_lengths[
        sequence_lengths.append(total_length)

    return sequence_lengths
```

10 Analyser 2d liste-data (8p)

Funksjonen **oppsummer(data)** skal motta data i form av ei liste av lister med tall. Den skal returnere ei liste av tupler. Hvert tuppel skal inneholde tre tall:

- det største tallet i tilsvarende indre liste,
- det minste tallet i tilsvarende indre liste, og
- tallet som er nærmest null i tilsvarende indre liste.

Eksempel: Hvis data som kommer inn, er:

[[3, -2, -4, 5], [3, 0, -8], [-1, 9]]

skal returnert liste av tupler være

[(5, -4, -2), (3, -8, 0), (9, -1, -1)]

F.eks. blir det første tuppelet i lista (5, -4, -2) fordi den første indre lista i data har største tall 5, minste tall -4, og det som er nærmest 0 er -2.

Velg riktig alternativ i hvert felt slik at funksjonen virker som den skal.

```
def oppsummer(data):
    svar = [ ]

    for rad in [Velg alternativ] (range(data), range(len(data)), data):

        stor = liten = nullest = [Velg alternativ] (rad[0], data[0], data[i][0])

        for tall in [Velg alternativ] (rad, range(1, len(data[i])), range(1, len(rad)):

            if [Velg alternativ] (tall[j], data[i][j], tall) > stor:

                stor = [Velg alternativ] (tall, data[i][j], rad[j])

            [Velg alternativ] (elif tall < liten:, elif data[i][j] < liten:, elif rad[j] < liten:)

                liten = [Velg alternativ] (rad[j], data[i][j], tall)

            [Velg alternativ] (if abs(rad[j]) < abs(nullest):, if abs(tall) < abs(nullest):, if abs(data[i][j]) <
abs(nullest):)
                nullest = tall
    svar.append((stor, liten, nullest))
    return svar
```

11 Mask (6p)

Funksjonen **mask(s1, s2**) skal finne alle steder hvor strengen s1 forekommer i s2.
Første forekomst markeres med 1'ere, andre forekomst markeres med 2-ere, osv.
i listen som returneres. Der hvor s1 ikke matcher, skal det stå 0.

Eksempler:

- **mask("a", "banan")** returnerer listen [0,1,0,2,0], fordi den første forekomsten av "a" forekommer ved indeks 1, og den andre ved indeks 3.
- **mask("an", "banana")** returnerer listen [0,1,1,2,2,0]
- **mask("ana", "banana")** returnerer [0,1,1,1,0,0]. Den tar bare med første "ana" i "banana" fordi den andre forekomsten overlapper med den første forekomsten.
- **mask("c", "banana")** returnerer [0,0,0,0,0,0], siden "c" ikke forekommer i "banana".

Implementer mask ved å velge riktig alternativ i nedtrekksfeltene i koden under. Ingen minuspoeng for feil valg.

```
def mask(s1, s2):

    result = [Velg alternativ] ([0] * len(s2), [], [] * len(s2), [0] * len(s1))

    count = [Velg alternativ] (0, -1, 1)

    pos = [Velg alternativ] (s2.find(s1), 1, 0, s2.find(s1, pos))

    [Velg alternativ] (for count in range(len(s2)):, while pos < len(s2):, while True:, while pos != -1:)

        [Velg alternativ] (result.append(count), result[pos : pos + len(s1)] = [count] * len(s1), result += [count] * len(s1))
        count += 1

        [Velg alternativ] (pos = s2.find(s1), pos = s2.find(s1, pos + 1), pos = s2.find(s1, pos + len(s1) + 1))

    return result
```

12 Numpy 2-dim array (10p)

Funksjonen **mindiffpath(A, starting_row)** tar inn to parametre:

- **A** er en 2-dimensjonal numpy array bestående av flyttall
- **starting_row** er en gyldig indeks til en rad i **A**

Funksjonen skal starte med elementet i første kolonne og raden svarende til **starting_row**.

Den skal så finne det elementet i neste kolonne som enten ligger en rad over, i samme rad, eller raden under, som er nærmest. Dette elementet blir utgangspunktet for å finne elementet i neste kolonne etter det igjen.

Når den har gått gjennom alle kolonnene, skal den returnere disse verdiene som en 1-dimensjonal numpy array.

Eksempel:

Med A er array'en med verdiene under

[[4 6 4 8 6]
[5 3 6 1 1]
[2 7 8 5 1]
[7 6 1 2 0]
[2 7 5 3 9]]

Så skal path(A, 2) returnere [[2, 3, 4, 1, 1]:

- Starter med verdien 2 i rad 3.
- Vi sammenligner 2 med verdiene 3, 7 og 6 i kolonne to. Den nærmeste i verdi er 3 (i raden over).
- Vi sammenligner 3 med 4, 6 og 8, hvor den næremeste i verdi er 4.
- 4 er i øverste rad, så vi sammenligner med 8 og 1, det nærmeste er 1
- 1 sammenlignet med 6, 1 og 1. Vi velger den første verdien 1.

OPPGAVE: Trekk kodelinjer til riktig sted så funksjonen virker som den skal. 4 av kodelinjene skal ikke brukes. Ingen minuspoeng for feil svar, så du BØR svare noe også der du er usikker.

result[0] = A[row, 0]	rows, cols = A.shape
if min_diff == -1 or diff < min_diff	abs(A[i, col] - reference_value)
for i in range(rows)	for i in range(max(0, row - 1), min(rows, row + 2))
np.zeros(cols, dtype=A.dtype)	[ ]
A[row, col-1]	if diff < min_diff
for col in range(cols)	for col in range(1, cols)
min_diff = diff	result[col] = (A[row, col])

```
def path(A, row):
    rows, cols = A.shape
    result = np.zeros(cols, dtype=A.dtype)
    result[0] = A[row, 0]

    for col in range(1, cols) :
        min_diff = -1
        reference_value = A[row, col-1]

        for i in range(max(0, row - 1), min(rows, row + 2)) :

            diff = abs(A[i, col] - reference_value)

            if min_diff == -1 or diff < min_diff :
                min_diff = diff
                row = i
        result[col] = (A[row, col])

    return result
```

13 Droneløp (6p)

I et drone-løp vurderes hver runde etter **tidsbruk** og **manøverkvalitet**.
Du skal lage funksjonen beregnScore() som returnerer totalpoeng.

Inputparametere:

- **tid** – tiden (i sekunder) dronen brukte
- **referanseTid** – forventet tid for banen
- **dommerPoeng** – liste med *fem* dommerpoeng fra 0 til 10, f.eks. [7, 8, 6, 9, 8]

Regler:

1. Tidspoeng

- Dronen får 50 poeng hvis tiden er nøyaktig lik referanseTid.
- For hvert sekund under legges det til 0.8 poeng.
- For hvert sekund over trekkes det fra 0.8 poeng.

2. Manøverkvalitet

- Fjern høyeste og laveste dommerpoeng.
- Summer de tre midterste poengene.

3. Total score = tidspoeng + manøver

Eksempel:

tid = 48
referanseTid = 50
dommerPoeng = [7, 8, 6, 9, 8]
print(beregnScore(tid, referanseTid, dommerPoeng))
74.6

Løsningsforslag

```python
def beregnScore(tid, referanseTid, dommerPoeng):
    """
    Beregner total score for en runde i drone-løp.

    Parametere:
        tid (float): tiden dronen brukte
        referanseTid (float): forventet tid for banen
        dommerPoeng (list): liste med 5 dommerpoeng (0–10)

    Returnerer:
        float: totalpoeng for runden
    """

    if len(dommerPoeng) != 5:
        raise ValueError("dommerPoeng må inneholde nøyaktig 5 poeng.")

    # 1. Beregn tidspoeng
    differanse = tid - referanseTid

    if differanse < 0:
        tidspoeng = 50 + abs(differanse) * 0.8
    else:
        tidspoeng = 50 - differanse * 0.8

    # 2. Beregn manøverpoeng
    sortert = sorted(dommerPoeng)
    midterste = sortert[1:-1]   # fjern høyeste og laveste
    manover = sum(midterste)

    # 3. Total score
    total = tidspoeng + manover

    return total
```

14 Naturreservat (6p)

Et naturreservat samler inn data om dyreobservasjoner i ulike områder gjennom dagen. Det registreres hvilken dyreart som er observert, antall dyr, hvilket område observasjonen er gjort i og hvor langt inn i reservatet observasjonen ble gjort.

Hver observasjon lagres som en ordbok (**dictionary**) med følgende struktur:

```
{
  "art": "Hjort",
  "antall": 4,
  "område": "Nord",
  "distanse": 120
}
```

Du får følgende datasett å jobbe med:

```
observasjoner = [
  {"art": "Hjort", "antall": 4, "område": "Nord", "distanse": 120},
  {"art": "Rev", "antall": 1, "område": "Sør", "distanse": 30},
  {"art": "Elg", "antall": 2, "område": "Øst", "distanse": 200},
  {"art": "Hjort", "antall": 3, "område": "Nord", "distanse": 150},
  {"art": "Ugle", "antall": 1, "område": "Vest", "distanse": 80}
]
```

Lag funksjonen **analyserArt(data, artNavn)** som gjør følgende:

1. Henter alle observasjoner av den gitte arten
2. Beregner:
    - totalt antall dyr
    - gjennomsnittlig distanse inn i reservatet
    - hvor mange observasjoner det er
3. Returnerer resultatet som en ny dictionary.

Eksempel:

```
print(analyserArt(observasjoner, "Hjort"))
{
  'art': 'Hjort',
  'antall_totalt': 7,
  'gjennomsnittlig_distanse': 135.0,
  'observasjoner': 2
}
```

Løsningsforslag

```python
observasjoner = [
    {"art": "Hjort", "antall": 4, "område": "Nord", "distanse": 120},
    {"art": "Rev", "antall": 1, "område": "Sør", "distanse": 30},
    {"art": "Elg", "antall": 2, "område": "Øst", "distanse": 200},
    {"art": "Hjort", "antall": 3, "område": "Nord", "distanse": 150},
    {"art": "Ugle", "antall": 1, "område": "Vest", "distanse": 80}
]

def analyserArt(data, artNavn):
    """
    Analyserer alle observasjoner av en gitt art.
    Returnerer en dictionary med totalt antall dyr, gjennomsnittlig distanse,
    og antall observasjoner.
    """
    # Hent alle observasjoner av arten
    observasjoner = [obs for obs in data if obs["art"] == artNavn]

    if not observasjoner:
        return None  # eller raise ValueError("Ingen observasjoner av denne arten")

    # Beregn totalt antall dyr
    totalt_antall = sum(obs["antall"] for obs in observasjoner)

    # Beregn gjennomsnittlig distanse
    gjennomsnitt_distanse = sum(obs["distanse"] for obs in observasjoner) / len(observasjoner)

    # Antall observasjoner
    antall_observasjoner = len(observasjoner)

    # Returner resultatet som en dictionary
    return {
        "art": artNavn,
        "antall_totalt": totalt_antall,
        "gjennomsnittlig_distanse": gjennomsnitt_distanse,
        "observasjoner": antall_observasjoner
    }

print(analyserArt(observasjoner, "Hjort"))
```

15 Temperatur (6p)

Lag et Python-program som fungerer som en fullverdig temperaturkonverterer.

Programmet skal:

1. Be brukeren skrive inn hvilken konverteringstype de ønsker:
    - "C2F" - Celsius til Fahrenheit
    - "F2C" - Fahrenheit til Celsius
    - "C2K" – Celsius til Kelvin
    - "K2C" – Celsius til Kelvin
2. Be brukeren skrive inn en temperaturverdi som skal konverteres.
3. Utføre konverteringen og vise resultatet med to desimaler.
4. Håndtere følgende feil:
    - Ved ikke-numerisk temperatur skal programmet skrive ut "Ugyldig temperatur!"
    - Ved ugyldig konverteringstype skal programmet skrive ut "Ugyldig konverteringstype!"
    - Ved negativ temperatur for Kelvin-konvertering skal programmet skrive ut "Kelvin kan ikke være negativ!"
5. La brukeren gjøre flere konverteringer i samme program, helt til de skriver "avslutt".

Tips til implementering:

- Bruk try / except for å håndtere feil.
- Husk at alle input fra input() er tekst, så du må konvertere til int eller float.
- Omregningsformler:
    - $F = C \times 9/5 + 32$
    - $C = (F - 32) \times 5/9$
    - $K = C + 273{,}15$
    - $C = K - 273{,}15$

Eksempel på kjøring:

Velg konverteringstype (C2F, F2C, C2K, K2C) eller 'avslutt': C2F
Skriv inn temperaturen: 100
100.00 °C er 212.00 °F
Velg konverteringstype (C2F, F2C, C2K, K2C) eller 'avslutt': K2C
Skriv inn temperaturen: -5
Kelvin kan ikke være negativ!
Velg konverteringstype (C2F, F2C, C2K, K2C) eller 'avslutt': avslutt
Program avsluttet.

Løsningsforslag

```python
def temperaturkonverterer():
    while True:
        konv_type = input("Velg konverteringstype (C2F, F2C, C2K, K2C) eller 'avslutt': ").strip()

        if konv_type.lower() == "avslutt":
            print("Program avsluttet.")
            break

        if konv_type not in ["C2F", "F2C", "C2K", "K2C"]:
            print("Ugyldig konverteringstype!")
            continue

        temp_input = input("Skriv inn temperaturen: ").strip()

        try:
            temp = float(temp_input)
        except ValueError:
            print("Ugyldig temperatur!")
            continue

        # Konvertering
        if konv_type == "C2F":
            resultat = temp * 9/5 + 32
            print(f"{temp:.2f} °C er {resultat:.2f} °F")
        elif konv_type == "F2C":
            resultat = (temp - 32) * 5/9
            print(f"{temp:.2f} °F er {resultat:.2f} °C")
        elif konv_type == "C2K":
            resultat = temp + 273.15
            if resultat < 0:
                print("Kelvin kan ikke være negativ!")
            else:
                print(f"{temp:.2f} °C er {resultat:.2f} K")
        elif konv_type == "K2C":
            if temp < 0:
                print("Kelvin kan ikke være negativ!")
            else:
                resultat = temp - 273.15
                print(f"{temp:.2f} K er {resultat:.2f} °C")

# Kjør programmet
temperaturkonverterer()
```

16 Museum (6p)

Du jobber som dataanalytiker på et museum som ønsker å analysere besøkstallene sine gjennom en uke. Antall besøkende per dag er registrert i et datasett. Du skal lage et Python-program som analyserer besøkstallene ved hjelp av **NumPy**.

Oppgave:

1. Bruk følgende datasett som input:
   besøk = np.array([120, 150, 180, 140, 130, 200, 170])
2. Programmet skal beregne og vise følgende, og avrunde alle verdier til 2 desimaler:
   - Totalt antall besøkende for uken
   - Gjennomsnittlig antall besøkende per dag
   - Dag med høyest besøk
   - Dag med lavest besøk
   - Standardavvik
   - Et nytt array som viser forskjellen mellom hver dags besøk og gjennomsnittet
3. Lag en funksjon **analyser_besøk(besøk)** som tar et NumPy-array **besøk** som input og returnerer alle resultatene som en dictionary.

Eksempel:

```
print(analyser_besøk(besøk))
{
   'Totalt besøk': 1090,
   'Gjennomsnitt': 155.71,
   'Høyest besøk': 200,
   'Lavest besøk': 120,
   'Standardavvik': 27.29,
   'Avvik fra gjennomsnitt': array([-35.71, -5.71, 24.29, -15.71, -25.71, 44.29, 14.29])
}
```

Løsningsforslag

Kommentar: Begrepet standardavvik var ukjent for studentene og er ikke vurdert i sensuren.

```python
import numpy as np

def analyser_besøk(besøk):
    """
    Tar et NumPy-array med antall besøk per dag og returnerer en dictionary med statistikk.
    Alle verdier avrundes til 2 desimaler.
    """
    totalt_besøk = np.sum(besøk)
    gjennomsnitt = np.mean(besøk)
    høyest = np.max(besøk)
    lavest = np.min(besøk)
    std_avvik = np.std(besøk)
    avvik_fra_gjennomsnitt = besøk - gjennomsnitt

    resultater = {
        'Totalt besøk': round(totalt_besøk, 2),
        'Gjennomsnitt': round(gjennomsnitt, 2),
        'Høyest besøk': round(høyest, 2),
        'Lavest besøk': round(lavest, 2),
        'Standardavvik': round(std_avvik, 2),
        'Avvik fra gjennomsnitt': np.round(avvik_fra_gjennomsnitt, 2)
    }
    return resultater

# Datasett
besøk = np.array([120, 150, 180, 140, 130, 200, 170])

# Kjør funksjonen og skriv ut resultatene
resultater = analyser_besøk(besøk)
print(resultater)
```

17 Vannføring (6p)

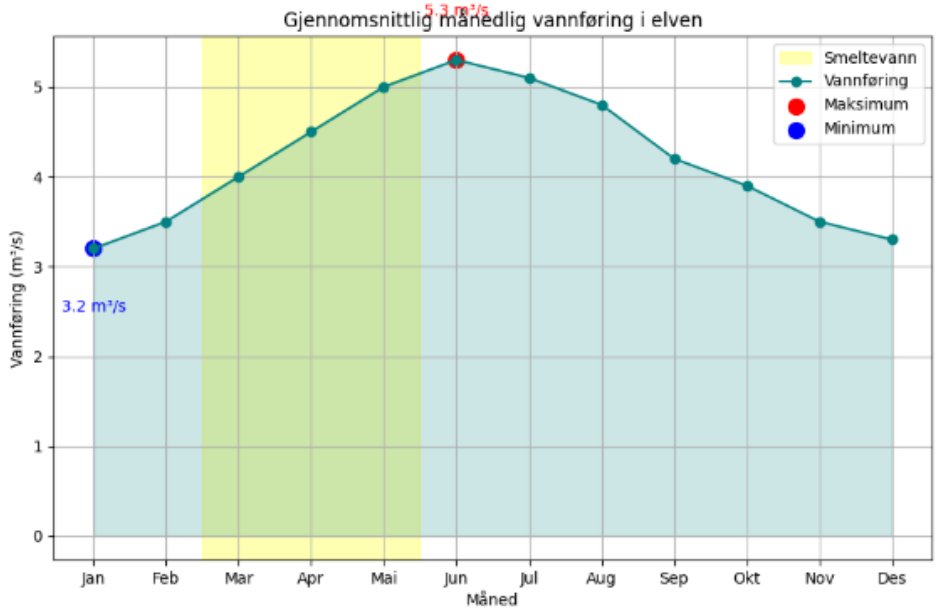
Du jobber som dataanalytiker for en kommune som overvåker vannstanden i en elv gjennom året. Målingene er gjennomsnittlig vannstand i meter per måned. Du skal lage en visuell fremstilling for å analysere variasjonene i vannstanden gjennom året.

Bruk følgende datasett:
måned = ["Jan", "Feb", "Mar", "Apr", "Mai", "Jun", "Jul", "Aug", "Sep", "Okt", "Nov", "Des"]
vannstand = [3.2, 3.5, 4.0, 4.5, 5.0, 5.3, 5.1, 4.8, 4.2, 3.9, 3.5, 3.3]

Lag en funksjon **plot_vannføring(måned, vannføring)** som tar måned- og vannførings-listene som input og lager et diagram.

Ønsket utforming:

- Plot vannføringen som linjediagram med måneder på x-aksen og vannføring (m³/s) på y-aksen.
- Legg til en tittel, x- og y-akse-labels, og grid.
- Marker **maksimum og minimum vannføring** med røde og blå prikker, og vis verdiene på grafen.
- Legg til fylling under kurven for å gjøre grafen mer visuelt informativ.
- Fremhev spesielt perioden med **smeltevann (mars-mai)**, med gul farge på området under kurven.



Løsningsforslag

```python
import matplotlib.pyplot as plt
import numpy as np

def plot_vannføring(måned, vannføring):
    """
    Plotter gjennomsnittlig vannføring gjennom året.
    Maksimum og minimum markeres, grafen får tittel, labels og grid.
    Smeltevannsperioden (mars-mai) fremheves med lys bakgrunn.
    """
    vannføring = np.array(vannføring)

    # Finn indeksene for maksimum og minimum vannføring
    maks_idx = np.argmax(vannføring)
    min_idx = np.argmin(vannføring)

    plt.figure(figsize=(10,6))

    # Fremhev smeltevannsperioden (mars=2, mai=4)
    plt.axvspan(2-0.5, 4+0.5, color='yellow', alpha=0.3, label='Smeltevann')

    # Linjediagram
    plt.plot(måned, vannføring, marker='o', color='teal', label='Vannføring')

    # Marker maksimum og minimum
    plt.scatter(måned[maks_idx], vannføring[maks_idx], color='red', s=100, label='Maksimum')
    plt.text(måned[maks_idx], vannføring[maks_idx]+0.5, f'{vannføring[maks_idx]:.1f} m³/s', color='red', ha='center')

    plt.scatter(måned[min_idx], vannføring[min_idx], color='blue', s=100, label='Minimum')
    plt.text(måned[min_idx], vannføring[min_idx]-0.7, f'{vannføring[min_idx]:.1f} m³/s', color='blue', ha='center')

    # Fyll under kurven
    plt.fill_between(måned, vannføring, color='teal', alpha=0.2)

    # Tittel, labels og grid
    plt.title('Gjennomsnittlig månedlig vannføring i elven')
    plt.xlabel('Måned')
    plt.ylabel('Vannføring (m³/s)')
    plt.grid(True)
    plt.legend()

    plt.show()

# Datasett
måned = ["Jan", "Feb", "Mar", "Apr", "Mai", "Jun", "Jul", "Aug", "Sep", "Okt", "Nov", "Des"]
vannføring = [3.2, 3.5, 4.0, 4.5, 5.0, 5.3, 5.1, 4.8, 4.2, 3.9, 3.5, 3.3]

# Kjør funksjonen
plot_vannføring(måned, vannføring)
```