

**Gruppenummer:** 3 A**Navn på gruppemedlem:** Elias Helland

## FORSIDE

ved besvarelse av individuell oppgave

<b>Emnekode:</b>	<b>UIA IS-114</b>
<b>Emnenavn:</b>	<i>Introduksjon til samskaping i informasjonssystemer</i>
<b>Emneansvarlig (faglærer):</b>	<i>Janis Gaillis og Niels F. Garmann-Johnsen</i>
<b>Eventuell veileder:</b>	<i>Mahsa Rasoli</i>
<b>Innleveringsfrist/tidspunkt:</b>	<i>23.10.2024</i>
<b>Antall ark inkl. denne forside:</b>	<i>31</i>
<b>Merknader/Evt. tittel på oppgaven:</b>	<i>Oblig 3</i>

Jeg bekrefter at jeg ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	Ja <input checked="" type="checkbox"/>	Nei <input type="checkbox"/>
---	--	------------------------------

*Kopiering av andres tekster eller annen bruk av andres arbeider uten kildehenvisning kan bli betraktet som fusk.*

Kan besvarelsen gjenbrukes til forskning og undervisningsformål, og i evt. publiseringer?	Ja <input checked="" type="checkbox"/>	Nei <input type="checkbox"/>
---	--	------------------------------

# Oblig 3 – IS 114

## Elias Østerhus Helland

### Oppgave 1

<b>Problemdefinisjon</b>
A. Implementer funksjonene <code>my-str-len</code> og <code>my-max</code> (DCIC kapittel 5.2) i Python. B. Implementer (både i Pyret og Python) de nødvendige strukturerte data (nye typer som <code>Student</code> og <code>Gruppe</code> , for eksempel) for å representere grupper av studenter. Skriv en funksjon, som finner ut hvilken gruppe en student hører til. Finn selv på "mock data" <sup>1</sup> . C. Implementer alle oppgavene i "Exercise"-boksene fra kapittel 9.1.1, kapittel 9.1.6.1, kapittel 9.1.8.3 og kapittel 9.2.2.
<b>Gitt (vanligvis pre-strukturerte data)</b>
For hver av deloppgavene må du selv definere relevant INPUT.
<b>Leveranser (L)</b>
L1. Skriv en relevant algoritme (skrevet med egne ord) for hver funksjon i deloppgavene A, B, og C. L2. Tegn flytdiagrammer for alle deloppgavene med symbolene, som er definert i standard ISO 5807 <sup>2</sup> . L3. Lag en liste av kontrakter for både funksjonene som du implementerer selv og funksjoner som du finner i biblioteker. L4. Lag koden og legg koden inn i din personlige (ikke offentlige) Github-repositorien og spesifiser URL-en <sup>3</sup> (lenken til den spesifikke filen). Inviter lærere og læringsassistentene, som "Collaborators" (samarbeidspartnere) i Github. L5. Ta skjermbilder, som viser utførelsen av din kode, og kommenter dem. L6. Reflekter over problemløsningsstrategier, som du brukte for å løse denne oppgaven, og evaluér resultatet i forhold til kravene (hva kunne du gjort bedre / annetledes?).
<b>Spesielle krav</b>

A)

Funksjonene my\_str\_len:

```
1 def my_str_len(s):
2     """Beregner lengden av en streng."""
3     if s == "":
4         return 0
5     else:
6         return 1 + my_str_len(s[1:]) # Legg til 1 og kall rekursivt med resten av strengen
7
8 # Teste funksjonen
9 print(my_str_len("hello")) # Forventet output: 5
10 print(my_str_len("")) # Forventet output: 0
11 print(my_str_len("abc")) # Forventet output: 3
```

L1. Algorithm for my\_str\_len:

| Starter med streng (S) som input > Sjekke om strengen er tom (Tom  
streng = returner 0 som output

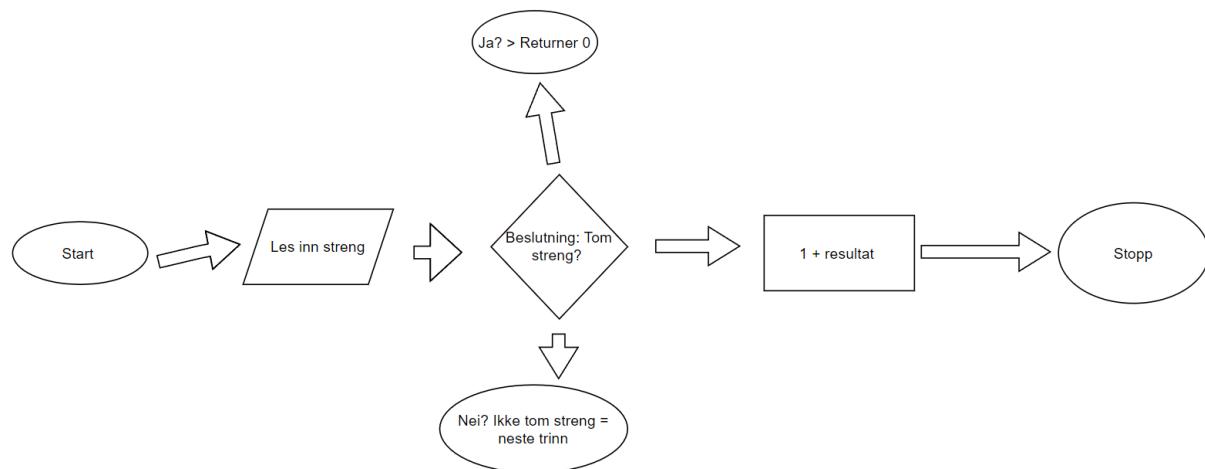


| Ikke tom streng = Fjern første tegn fra streng = slik at en arbeider med  
resten av strengen (s (1:)) > Returnering resultat (1 + lengden av resten av  
strengen)

- Eks: «Elias blir 1 + lias»

**Ferdig algoritme: Tegnene er talt opp, og resultatet blir returnert**

L2. Flytdiagram my\_str\_len



L3. Kontrakt my\_str\_len;

A1. – my\_str\_len: String > Integer

- Input: Streng "s" (kanskje tom)
- Output: Heltall som representerer lengden av strengen

L4. Done

L5.

```
1 def my_str_len(s):
2     """Beregner lengden av en streng."""
3     if s == "": # Basis-tilfellet: Tom streng
4         return 0
5     else:
6         return 1 + my_str_len(s[1:]) # Legg til 1 og kall rekursivt med resten av strengen
7
8 # Teste funksjonen
9 print(my_str_len("hello")) # Forventet output: 5
10 print(my_str_len("")) # Forventet output: 0
11 print(my_str_len("abc")) # Forventet output: 3
```

---

```
# Teste funksjonen
print(my_str_len("elias")) # Forventet output: 5
print(my_str_len("")) # Forventet output: 0
print(my_str_len("abc")) # Forventet output: 3
print(my_str_len("nicolai")) #Forventet output: 7
```

```
>>> %Run -c $EDITOR_CONTENT
      5
      0
      3
      7
>>>
```

**Her har jeg lagt inn koden i Thonny, hvor koden beregner lengden av min streng. Dette spesifiseres i hermetegn, linje 2. Dersom strengen er «» altså 0, vil returnert verdi være 0. Dersom den ikke er null, altså «else» vil returnert resultat være 1 + lengden på min streng, visualisert ved linje 6. Denne funksjonen gjør at vi kan eksemplifisere. Det gjør jeg ved «hello» eksempelet og «Elias/Nicolai eksempelet». Lengden på strengen Hello og Elias er 5, derfor er forventet output 5. Lengden på strengen ABC er 3, forventet output 3. Nicolai – lengde 7, forventet outcome 7. Dette testet jeg ved «run» funksjonen som resulterte i figuren til høyre, som samsvarer med forventet output.**

## L6.

Jeg benyttet meg av DCIC instruksjoner, og det jeg har lest på forhånd i kapitlene. Jeg prøvde meg frem, og fikk flere feilmeldinger, men ettersom Thonny og Phyton er mer forståelig en Pyret, fant jeg ut hvor feilen var. Igjen som i de andre obligene brukte jeg en prøve og feile – tilnærming. Resultatet mitt er jeg fornøyd med. Funksjonen fungerte, noe jeg mener er hovedformålet med oppgaven.

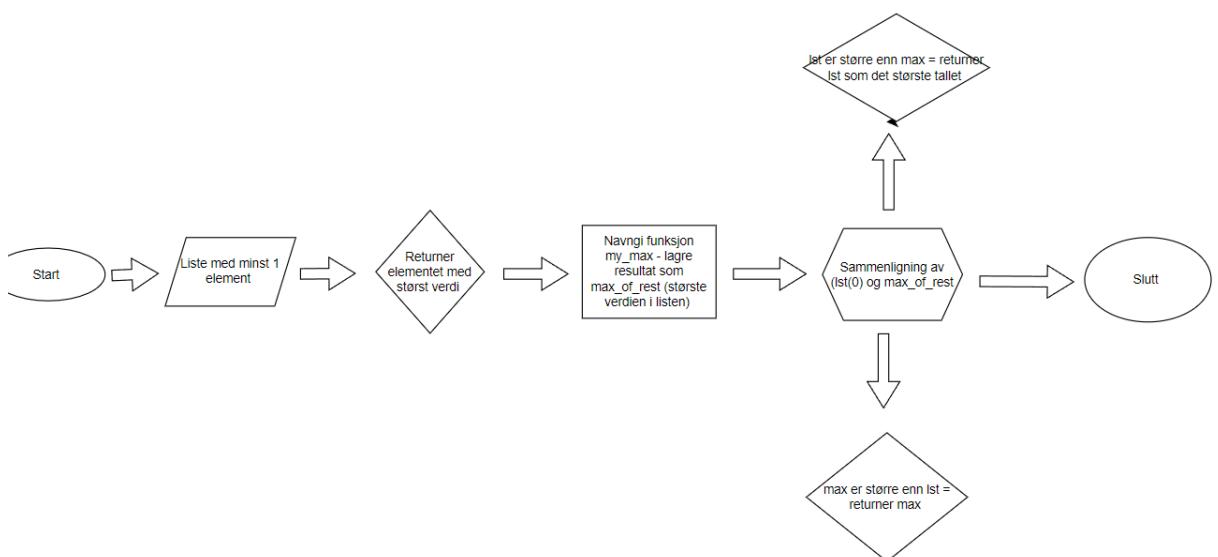
### My\_Max

#### L1.

1. Liste med minst et element → 2. Lengden på listen 1 = returner elementet i listen som størst verdi → 3. Navngi funksjon my\_max (u/ første element): my\_max (lst[1:]) – Lagre resultat som max\_of\_rest (dvs. den største verdien i resten av listen) → 4. Sammenligning av første element (lst[0]) og max\_of\_rest: Hvis lst[0] er større enn «max», returner lst[0] som det største tallet, hvis ikke motsatt → 5.

Slutt

#### L2.



### L3. «Kontrakt»

Input: My\_max → Lengde lst = 1 → Returner lst (0) #Kalles for basistilfelle

Mellomledd: max\_of\_rest = my\_max(lst( 1:)) → Hvis lst (0) > max\_of\_rest = Return; lst(0) ellers Return max\_of\_rest

Ouput: Returnering av lst(0) eller max\_of\_rest som gjenspeiler en verdi eller et tall

### L4. Done

### L5.

```
def my_max(lst):
    if len(lst) == 1: #Basis-tilfellet: En liste med en verdi eller et tall, som gjør at verdien er den største
        return lst[0]
    else:
        #Sammenligner første element med den største verdien i listen
        max_of_rest = my_max(lst[1:])
        if lst[0] > max_of_rest:
            return lst[0]
        else:
            return max_of_rest

print(my_max([1, 2, 3, 4, 5])) #Resultat 5
print(my_max([100, 200, 300, 400, 500])) #Resultat 500
print(my_max([100/50, 600/50, 700/50, 800/50])) #Resultat 16.0
print(my_max([0, 0, 0, 0, 0])) #Resultat 0 |
```

Shell ×

```
5
500
16.0
0
```

**Her har jeg igjen benyttet meg av DCIC og relevante eksempler for å visualisere koden. Definisjonen er my\_max (lst). Dersom lengden på listen er en 1, vil returnert resultat være 0. Dersom listen er lengre enn en verdi sammenligner em første element med den største verdien.**

**Dersom lst (0) er større enn max\_of\_rest: returneres lst (0). Hvis ikke returneres høyest verdi i listen av tall, altså my\_max som er illustrert i bildene ovenfor. Nederst kan en se hvordan funksjonen fungerer for sitt formål for å finne den største verdien i tallrekken.**

## L6.

Som nevnt ovenfor benyttet jeg meg av DCIC, og kapitlene lest på forhånd. For å forstå begreper på en bedre måte brukte jeg ChatGPT, for enkelhetens skyld. Jeg løste problemene, igjen, ved å prøve meg frem. På denne måten kan jeg lære av feilene mine for så å rette dem opp igjen. Det blir en del feilmeldinger på grunn av vanskeligheter med nytt språk, men jeg mener at dette er det ekstremt god læring i. Jeg vil si at resultatet er OK, er ikke helt fornøyd med hvordan funksjonen fungerer på grunn av hvordan den skal returnere lst (0) dersom den er større enn my\_max, men vil trekke frem hvordan funksjonene fungerer knyttet til å beregne høyest verdi av en liste med tall, altså my\_max. My\_max er definisjonen av funksjonen som da blir det viktigste å gjøre rede for gjennom min gjennomførte funksjon. Det jeg kunne gjort bedre er å gjerne eksemplifisere mer. Da får jeg og leseren en bedre forståelse for hvordan funksjonen fungerer, og hvordan den brukes til et konkret formål.

Funksjonen my\_max:

```
def my_max(lst):
    if len(lst) == 1: #Basis-tilfellet: En liste med en verdi eller et tall, som gjør at verdien er den største
        return lst[0]
    else:
        #Sammensligner første element med den største verdien i listen
        max_of_rest = my_max(lst[1:])
        if lst[0] > max_of_rest:
            return lst[0]
        else:
            return max_of_rest
```

B)

## L1.

Algoritme Phyton:

1. Definere en «class» for Student → (self, navn, student\_id): → 2.
- Samme for Gruppe → (self, gruppenavn, studenter): → 3. Finne gruppe for student → Hvis studentID ikke samsvarer med ettersøkt

student – return «Studenten tilhører ingen gruppe» → 4. Eksempler med utg. punkt i funksjonen → Tilgjengelige grupper → 5. «Print» finn\_gruppe\_for\_student → Output...

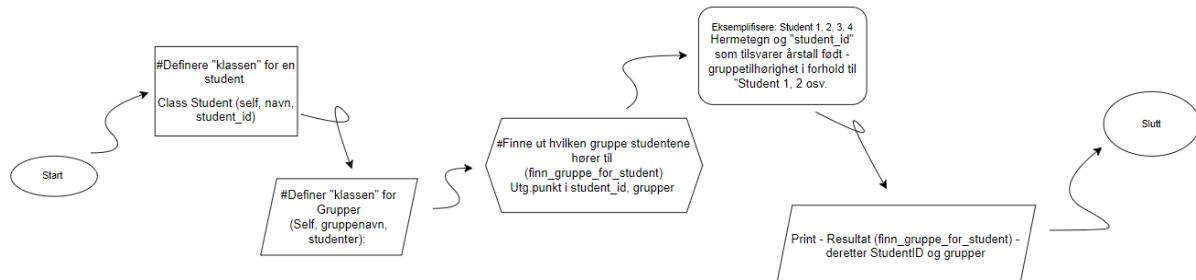
Algoritme Pyret:

1. «Opprett Studenter» - opprette student for student → 2. Students = table (row): Med navn, studentID og gruppe → Row for Row med input → 3. Fun, finne gruppe for student → 4. Test funksjonen → →→→

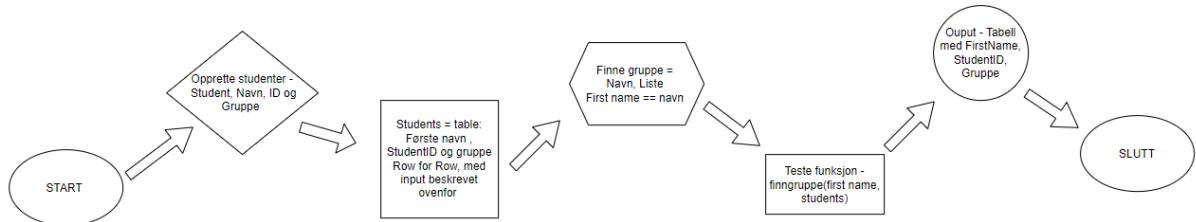
firstName	studentID	group
"Seline Helland"	"2"	"Gruppe1"

L2.

Flytdiagram Python;



Flytdiagram Pyret;



L3.

Kontrakt Python:

**START**

**Input:**

- Definere student (navn, studentID)
- Definere grupper, (gruppenavn, studenter)
- Finne gruppe for student = finn\_gruppe\_for\_student
- Dersom StudentID ikke samsvarer med identitet i noen grupper → «Studenten tilhører ingen gruppe»

**Output:**

- Eksempler med student 1, 2, 3, 4 (navn og studentID som er studentens fødselsår)
- Print (finn\_gruppe\_for\_student(studentID, grupper))
- Grupperom basert på StudentID som resultat

**SLUTT**

Kontrakt Pyret:

**Input:**

- Start
- Navn → String som representerer studentens navn.
- Liste → Liste som skal lage en slags tabell basert på informasjon om studenter, inkludert navn, studentID og gruppe.

**Output:**

- Returnerer en enkel tabell som illustrerer studentens grupperom → dersom navnet matcher med grupperom
- Dersom input og output ikke matcher → vil det returneres en tom liste
- Slutt

L4.

Done

## L5.

Python:

```
1 # Definere "klassen" for en student
2 class Student:
3     def __init__(self, navn, student_id):
4         self.navn = navn
5         self.student_id = student_id
6
7 # Definerer "klassen" for Grupper - for å funksjonalisere forholdet mellom student og gruppe
8 class Gruppe:
9     def __init__(self, gruppenavn, studenter):
10        self.gruppenavn = gruppenavn
11        self.studenter = studenter
12
13 #Funksjonen som finner ut hvilken gruppe en student hører til basert på student_id og gruppenr.
14 def finn_gruppe_for_student(student_id, grupper):
15     for gruppe in grupper:
16         for student in gruppe.studenter:
17             if student.student_id == student_id:
18                 return gruppe.gruppenavn
19     return "Studenten tilhører ingen gruppe"
20
21
22 #Eksempler som tar utgangspunkt i funksjonen
23 student1 = Student("Kristian", 2004)
24 student2 = Student("Nicolai", 2001)
25 student3 = Student("Endi", 2005)
26 student4 = Student("Brage", 2003)
27
28 gruppe1 = Gruppe("Gruppe Alfa", [student1, student2])
29 gruppe2 = Gruppe("Gruppe Beta", [student3, student4])
30
31 #Tilgjengelige grupper
32 grupper = [gruppe1, gruppe2]
33
34 print(finn_gruppe_for_student(2001, grupper)) #Output: Gruppe Alfa
35 print(finn_gruppe_for_student(2003, grupper)) #Output: Gruppe Beta
36 print(finn_gruppe_for_student(1945, grupper)) #Output: Studenten tilhører ingen gruppe
```

Shell <

```
>>> %Run -c $EDITOR_CONTENT
Gruppe Alfa
Gruppe Beta
Studenten tilhører ingen gruppe
```

**Her visualiseres det en funksjon som skal vise hvilken gruppe studentene hører til. Dette gjøres gjennom forskjellig input og Mock Data for simpelhetens skyld. Funksjonens struktur er delt inn i flere deler. Først på en definere de ulike klassene eller inputs. Det vil si student og grupper, før en påfølgende funksjon som skal finne gruppe for studenten. Innledningsvis visualiseres en funksjon for Student. Denne inneholder «self» som referer til det aktuelle objektet som er instansen i klassen, navn og studentID. Deretter definerer vi eller funksjonaliserer vi «gruppe». Den har samme struktur som foregående funksjon, men input er (self, gruppenavn, studenter). For å finne riktig grupperom for studenten er vi avhengig av en trede funksjon: «finn\_gruppe\_for\_student(student\_ID, grupper)». Denne funksjonen matcher studentens ID mot grupper, og søker etter match. Dersom inputs- ene ikke matcher vil det returneres «Studenten tilhører ingen gruppe». Deretter visualiseres det eksempler med «Student 1, 2, 3, 4, navn og ID som i dette tilfelle er fødselsår. Videre**

**vil vi få faktisk output som gjøres gjennom Print funksjonen. Da får vi #forventet ouput, og faktisk output.**

Pyret:

```
1 | use context starter2024
2 | #Opprett studenter
3 | #student1 = Student "(Elias Helland, 1, gruppe1)
4 | #student2 = Student "(Seline Helland, 2, gruppe1)
5 | #student3 = Student "(Brage Olsen, 3, gruppe2)
6 | #student4 = Student "(Trym Jansen, 4, gruppe2)
7 | #student5 = Student "(Nicolai Østerhus, 5, gruppe1)
8 |
9 * students = table:
10   firstName, studentID, group
11   row: "Elias Helland", "1", "Gruppe1"
12   row: "Seline Helland", "2", "Gruppe1"
13   row: "Brage Olsen", "3", "Gruppe2"
14   row: "Trym Jansen", "4", "Gruppe2"
15   row: "Nicolai Østerhus", "5", "Gruppe1"
16 end
17
18 * fun finngruppe(navn, liste):
19   liste. filter(lambda row: row ["firstName"] == navn
20   end)
21 end
22
23 #Test funksjonen
24
25 finngruppe("Seline Helland", students)
```

firstName	studentID	group
"Seline Helland"	"2"	"Gruppe1"
»»»		

**Denne koden er enklere. Her oppretter vi studenter en og en. Her skal funksjonen inneholde navn, ID og gruppe. For enkelhetens skyld har jeg implementert en tabell-funksjon. Det vil si at vi skriver inn students = table med firstName, studentID og gruppe. Deretter skal en finne navnet i listen hvor «firstName» er navnet. Deretter for å teste funksjonen kan en skrive inn «finn gruppe («Navn», students), som fører til en tabell som i bildet ovenfor.**

L6.

Min problemløsningsstrategi var her å benytte meg av info fra DCIC, men ut fra det fikk jeg en ganske lang kode i Thonny. Selv om denne fungerte, basert på lesestoffet og delvis hjelp av ChatGPT (til begreper), ønsket jeg å begrense koden i Pyret. Det utførte jeg ved å redusere mock data, og simplifisere instansene i koden. Derfor er jeg mer fornøyd med Pyret koden min, kontra Phyton koden. Det er verdt å nevne at begge fungerte til sitt formål som er gunstig. Det jeg kunne gjort bedre er å sette meg bedre inn i hvordan en kan korte ned koden for å ikke få såpass omfattende data, og omfattende mengder input som kreves.

C)

#### L1. – 9.1.1 – Algoritme:

- 1. «Definere vekt måne, basert på jordens vekt» → 2. Returnere jordens vekt \* 1/6 → 3. Print: vekt måne (verdi – jordens vekt)) – Resultat  $100 * 1/6 = 16.666 =$  Månenes vekt

#### 9.1.6.1 – Algoritme:

1.

- 1. «Definere hva string representerer = numbers → 2. Returnere = «neg» hvis nummeret er under 0, «pos» hvis tallet er over 0, og «zero» hvis tallet er 0 → 3. Eksempel: en rekke med nummer, 2 negative, 2 positive og 1 zero → 4. Resultat = pos, neg, zero, pos neg

2.

1. Definere lengde på string = 5 → 2. Returnere ord som har 5 tegn, eller 5 bokstaver, returnere true hvis det er noen ord som har 5 bokstaver → 3. Eksempel: 5 ord hvor 2 ord har 5 bokstaver → 4. Returnere True fordi det er 2 ord med 5 bokstaver

3. 1. Definere partall mellom 10 og 20 → 2. Returnere num for num «In» numbers → 3. En rekke med nummer m/ to nummer som er partall mellom 10 og 20 → 4. Print = Resultat

### 9.1.8.3 – Algoritme

1 og 2.

1. Definere alle «z\_wordz» i en liste → 2. Produsere en liste med ord fra input som innehar bokstaven Z → 3. Returnere liste med ord som har Z i seg → 4. Print = all\_z\_words(zoom, zoo, flaske, tv) = zoom og zoo

### 9.2.2 – Algoritme

1. Oversikt over rom og hvor mange plasser det er i rommet → 3. Seter i rom David → 3. Print seter i rom David → 4. Rom David har 40 plasser

2.

1. Oversikt over rom og hvor mange plasser det er i rommet → 2. Room Elias += 10 = Elias får 10 ekstra plasser → 3. Rom Elias har 25 seter

3.

1. Oversikt over rom og hvor mange plasser det er i rommet → 2. Store rooms = for å finne rom med plass til over 50 studenter → 3. Print rom som har plass til 50 eller flere studenter = Room Alfa, Room Beta og Room Catrine

L2. Flytdiagrammer – Bruker Napkin. AI – Godkjent av Janis Gailis

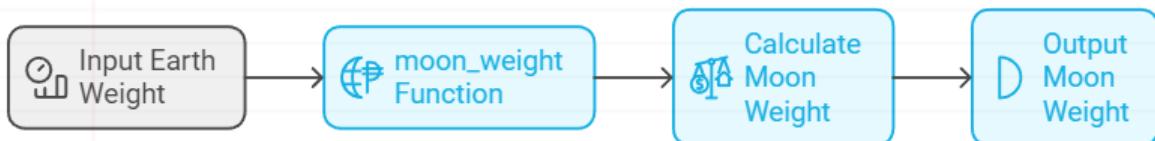
### 9.1.1

#### def moon\_weight(earth\_weight):

"Compute weight on moon from weight on earth"

```
return earth_weight * [1/6]
```

```
print(moon_weight(100)) #Forventet output 16.66
```

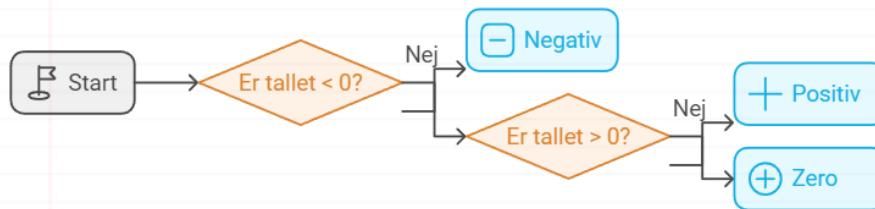


### 9.1.6.1

1.

#### Flowchart

```
def sign_to_string(numbers):
    return ["neg" if num < 0 else "pos" if num > 0 else "zero" for num in numbers]
#Eksempel på nummer
numbers = [100, -50, 0, 350, -400]
result = sign_to_string(numbers)
print(result) # Resultat [pos, neg, zero, pos, neg]
```



2.

#### Flowchart

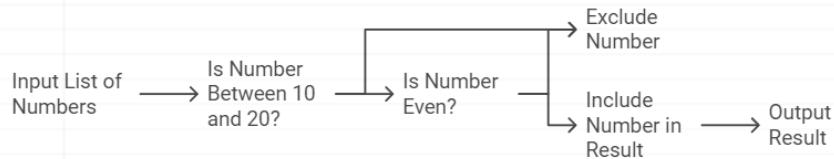
```
def any_string_length_5(strings):
    return any(len(s) == 5 for s in strings)
#Eksempel på strings med 5 verdier
strings = ["Telefon", "Epler", "Datamaskin", "Gryte"]
result = any_string_length_5(strings)
print(result)
```



3.

## Flowchart

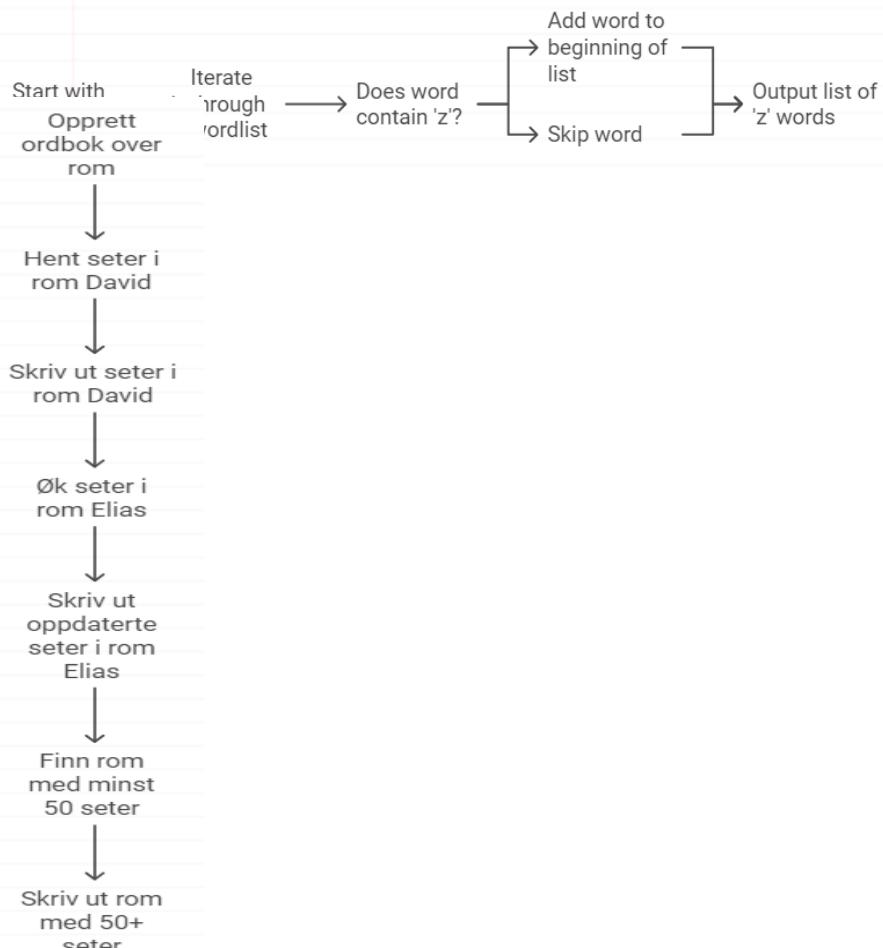
```
def even_numbers_10_to_20(numbers: list) -> list:
    return [num for num in numbers if 10 <= num <= 20 and num % 2 == 0]
#Eksempel
numbers = [1, 7, 14, 16, 22]
result = even_numbers_10_to_20(numbers)
print(result) # Output: 14 og 16
```



### 9.1.8.3

## Flowchart

```
def all_z_words(wordlist : list) -> list:
    """Produce list of words from the input that contain 'z'."""
    zlist = [] # start with an empty list
    for wd in wordlist:
        if 'z' in wd:
            zlist = [wd] + zlist # add wd to the beginning of zlist
    return zlist
#Teste funksjonen:
print(all_z_words(['zoom', 'zoo', 'flaske', 'tv']))
```



### 9.2.2

## L3. Kontrakter

### 9.1.1

# Kontrakt:

# def - Moon\_weight

#Beregner vekt på månen basert på jordvekten

#Input – return: earth\_weight \* 1/6

#Output: tilsvarende vekt på månen (1/6 av jordvekten)

### 9.1.6.1

#### 1. Kontrakt

#Sign\_to string: (list of int) → list of str

#Input: En liste med heltall (int) som kan være positive negative eller null

#Output: En liste med strenger (str), hvor:

- «neg» representerer negative tall
- «pos» representerer positive tall
- «zero» representerer tallet null

#### 2.

# any\_string\_length\_5: (list of str) → bool

# Input: En liste med strenger (list of str)

#Output: En boolsk verdi (True eller False)

#Formålet: Funksjonen returnerer True hvis minst en av strenge i listen består av 5 bokstaver

#### 3.

# even\_numbers\_10\_to\_20: (list of int) → list of int – None

# Input: En liste med heltall (list of int). – liste

#Output: En liste med heltall (list of int). – liste

#Formålet: Returnerer en liste med partallene i tallrekken

### 9.1.8.3

#all\_z\_words: (list of str) → list of str

#Input: En liste med ord (list of str).

#Output: En liste med ord (list of str).

#Formål: Funksjonen returnerer ny liste som inneholder ord som har Z i seg

### 9.2.2

#### 1. Antall seter i et spesifikt rom

#Input: Streng som representerer romnavn (f.eks. «Room David»)

#Ouput: Tall (int) som representerer antall seter, eller feilmelding dersom rommet ikke finnes

#### 2. Øke kapasitet med 10 seter

#Input: Streng som representerer romnavnet (f.eks. «Room Elias»).

#Output: Oppdatert antall seter i rommet etter å ha lagt til 10 seter.

#### 3. Finne rom med plass til minst 50 studenter

#Input: Dict – med romnavn som «nøkler» og seter som verdier

#Output: Liste (list) med romnavnene som har minst 50 seter

### L4. Done

## L5.

### 9.1.1



```
1 def moon_weight(earth_weight):
2     "Compute weight on moon from weight on earth"
3
4     return earth_weight * (1/6)
5
6 print(moon_weight(100)) #Forventet output |16.66
```

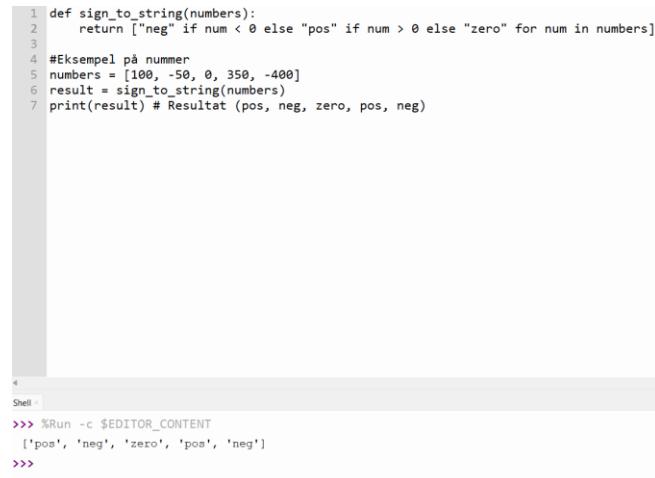
Shell >

```
>>> %Run -c $EDITOR_CONTENT
16.666666666666664
```

En relativt simple kode som beregner månens vekt basert på formelen av jordens vekt. Månens vekt er en sjette del av jordens vekt, som tilsvarer formel og funksjon «return» `earth_weight * (1/6)`. Derfor må en oppgi jorden vekt  $(100) * 1/6 = 16.66$  som var forventet output.

### 9.1.6.1

#### 1.



```
1 def sign_to_string(numbers):
2     return ["neg" if num < 0 else "pos" if num > 0 else "zero" for num in numbers]
3
4 #Eksempel på nummer
5 numbers = [100, -50, 0, 350, -400]
6 result = sign_to_string(numbers)
7 print(result) # Resultat (pos, neg, zero, pos, neg)
```

Shell >

```
>>> %Run -c $EDITOR_CONTENT
['pos', 'neg', 'zero', 'pos', 'neg']
>>>
```

Definere string som er nummer i denne situasjonen. Funksjonen skal returner «neg» dersom tallet er negativt, «pos» dersom tallet er positivt, og «zero». Eksemplifisere med nummer, tallrekke på fem tall som

representerer verdier knyttet til neg, pos og zero. Deretter printe resultatet hvor forventet output er det faktiske outputet som tilsvarer: pos, neg, zero, pos, neg.

## 2.

```
1 def any_string_length_5(strings):
2     return any(len(s) == 5 for s in strings)
3
4 #Eksempel på strings med 5 verdier
5 strings = ["Telefon", "Epler", "Datamaskin", "Gryte"]
6 result = any_string_length_5(strings)
7 print(result)
```

Shell < />

```
>>> %Run -c $EDITOR_CONTENT
```

```
True
```

Her defineres også en streng, hvor strengen skal være 5 verdier, eller bokstaver lang. Det skal returneres true dersom det er en streng som er tilsvarende lengde, altså 5. Deretter eksemplifiserer vi med en rekke med ord der noen har 5 verdier og noen ikke. Resultatet skal altså tilsvare ordet True, ettersom ordet Epler og Gryte har 5 verdier eller 5 bokstaver. Igjen blir forventet output det samme som det faktiske outputet.

## 3.

```
1 def even_numbers_10_to_20(numbers: list) -> list:
2     return [num for num in numbers if 10 <= num <= 20 and num % 2 == 0]
3
4 #Eksempel
5 numbers = [1, 7, 14, 16, 22]
6 result = even_numbers_10_to_20(numbers)
7 print(result) # Output: 14 og 16
```

Shell < />

```
>>> %Run -c $EDITOR_CONTENT
```

```
[14, 16]
```

Her skal funksjonen finne tallene som er partall mellom 10 og 20 i en gitt tallrekke. Først må en definere even\_numbers\_10\_to\_20 → list. Skal returnere alle partallene i tallrekken bestående av 1, 7, 14, 16 og 22. Igjen

eksemplifiserer vi med den overnevnte tallrekken før vi skrive inne definisjonen: even\_numbers\_10\_to\_20. Forventet output 14 og 16. Faktisk output 14 og 16 .

### 9.1.8.3

#### 1.

```

1 def all_z_words(wordlist : list) -> list:
2     """Produce list of words form the input that contain 'z'."""
3     zlist = [] #start with an empty list
4     for wd in wordlist:
5         if 'z' in wd:
6             zlist = [wd] + zlist # add wd to the beginning of zlist
7     return zlist
8 #Teste funksjonen:
9 print(all_z_words(['zoom', 'zoo', 'flaske', 'tv']))
10

```

Shell <br>>> %Run 'Z list.py'  
['zoo', 'zoom']

Her skal funksjonen finne ordene med en ‘Z’ i seg. Def all\_z\_words. Skal produsere en liste med alle ordene som inneholder z. If ‘z’ in wd = dersom z er en del av ordet, skal man returnere ‘zlist’. Deretter Print all\_z\_words. Har en rekke av ord (zoom, zoo, flaske og tv). Forventet output er zoom og zoo. Faktisk output er zoom og zoo. Funksjonen fungerer til sitt formål.

#### 2.

```

1 def all_z_words_filter(wordlist: list) -> list:
2     """Produce list of words form the input that contain 'z'."""
3     return list(filter(lambda wd: 'z' in wd.lower(), wordlist))
4
5 def test_all_z_words_filter():
6     assert all_z_words_filter(['zoom', 'zoo', 'flaske', 'tv']) == ["zoom", "zoo"]
7
8 #Teste funksjonen:
9 print(all_z_words_filter(['zoom', 'zoo', 'flaske', 'tv']))
10

```

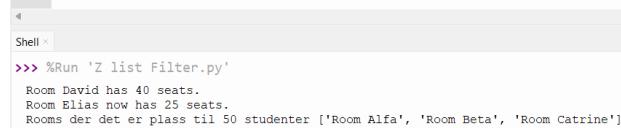
Shell <br>>> %Run 'Z list REAL.py'  
['zoom', 'zoo']

Forskjellen mellom de to er hvordan man utfører koden. Manuell vs. Automatisk. Den første versjonen går gjennom hvert ord selv, hvor ordene legges til manuelt. I den andre versjonen automatiseres det. Filter tar seg av

alt automatisk, og koden blir simplere og mer forståelig, samtidig som rekkefølgen fra den første koden beholdes.

### 9.2.2

```
1 # 1. Kartleggelse av rom som har et spesifikt antall seter
2 rooms = {
3     "Room Alfa": 70,
4     "Room Beta": 60,
5     "Room Catrine": 85,
6     "Room David": 40,
7     "Room Elias": 15
8 }
9
10 # Seter i spesifikt rom
11 seats_in_room_David = rooms.get("Room David", "Room not found")
12 print(f"Room David has {seats_in_room_David} seats.")
13
14 # 2. Øke kapasitet med 10 ekstra seter
15 rooms["Room Elias"] += 10
16
17 print(f"Room Elias now has {rooms['Room Elias']} seats.")
18
19 # 3. Finne rommene som har plass til minst 50 studenter
20 store_rooms = [room for room, seats in rooms.items() if seats >= 50]
21 print("Rooms der det er plass til 50 studenter", store_rooms)
```

Formålet her er å finne seter i et spesifikt rom, øke kapasitet på et rom +10 og Finne rommene som har plass til minst 50 studenter. Først og fremst definere romnavn og hvor mange seter som er i hvert rom. For å finne seter i et spesifikt rom (Seats\_in\_room\_X) – deretter print for å få resultat = hvor mange seter i rom David = 40. Oppgave nr to er å øke kapasiteten med 10 ekstra seter . Ta et spesifikt rom f.eks. «Room Elias» += 10 – deretter print som fører til at rom Elias har 10 mer seter =25. Oppgave tre skal finne ut hvilke rom som har plass til mer enn 50 studenter. «Store rooms» if seats  $\geq 50$ . Print, og resultatet er en liste over hvilke rom som har plass til mer enn 50 studenter.

## L6.

Denne oppgaven var omfattende, med et høyt antall oppgaver som var krevende nettopp på grunn av mengden. Jeg brukte igjen de samme strategiene for å løse oppgavene nemlig å prøve meg frem. Igjen brukte jeg DCIC, naturligvis med hjelp av ChatGPT. Dette gjorde at jeg fikk en dyp forståelse av hvordan jeg skal skrive funksjoner på en ryddig måte, og hvordan jeg skulle gjennomføre L oppgavene på en så ryddig måte som overhodet mulig. Resultatet ble forholdsvis bra i forhold til kravene ettersom funksjonene fungerte, og output samsvarte med forventningene. Det jeg kunne gjort bedre her er å ta bedre «vare» på alle filene i Thonny. Det

vil si mer struktur både når jeg skrev koden, og når jeg prøvde å finne relevant lesestoff for å løse oppgavene. Rett og slett prøve å ha mer struktur rundt arbeidet mitt for å finne lettere frem, og effektivisere oppgaveprosessen betraktelig.

## Oppgave 2.

A)

Den kommunen som har hatt den høyeste prosenten av barn i ett- og toårsalderen i barnehagen i 2023 er Modalen (4629) kommune ifølge funksjonen min i Phyton/ Thonny. Prosentpoeng = 142.9 %

B)

Den kommunen med den laveste prosenten av barn i ett- og toårsalderen i Barnehagen 2023 er Etnedal og Fedje kommune (henholdsvis 3450, og 4633) ifølge funksjonen min i Phyton/Thonny. Prosentpoeng = 55,6 %.

C)

Den kommunen med høyest gjennomsnittlig prosent i perioden 2015 til 2023, angående prosent av barn i ett- og toårsalderen i barnehage er igjen Modalen kommune med en gjennomsnittlig prosent på 133,7 % i perioden 2015-2019.

D)

Kommunen med den laveste gjennomsnittlige prosenten i perioden 2015-2023, angående prosent av barn i 1-2 års alderen i barnehage er Fosnes kommune med 49,25% i perioden 2018-2019.

E) Samme som C

F)

Den gjennomsnittlige prosenten for alle kommuner fra et spesifikt år = 2017 er 81,3%.

- G) Gjort – ligger på github
- H) Gjort – ligger på github

L1.

#### A - (df\_filtered)

→#Set koden: høyeste verdi til maksimumsverdi i df\_filtered (2023) – unngå float (inf) på grunn av at det representerer uendelig høy verdi, noe som kan føre til at de «tomme kolonnene» blir tatt med i betraktning

→ Filtrere: df\_filtered for kommune hvor df\_filtered (2023) er lik høyeste\_verdi (samsvarer 2023 med kommunen med høyest prosent)

→ Return: kommune/ kommuner med høyeste prosenten i 2023.

#### B – (df\_filtered)

→laveste\_verdi til minimumsverdi i df\_filtered(2023), igjen unngå float(int)

→filtrere: df\_filtered for kommune, hvor df\_filtered(2023) er lik den laveste verdien i excel arket

→return kommune/kommuner med laveste prosent i 2023

#### C – (df\_filtered, headers):

→Beregn gjennomsnitt fra 2015 til 2023, erstatte float(inf) med NA, for å hindre programmet i å ta med kolonnene uten data

→høyeste\_gjennomsnitt til maksimum av df\_filtered (Avarage 2015 – 2023)

→Filtrere; df\_filtered for kommune hvor Average 2015-2023 er lik høyeste gjennomsnitt

→Return kommuner med høyeste gjennomsnittlige prosent

#### D (df\_filtered)

→Laveste\_gjennomsnitt til minimum av df\_filtered (Avarage 2015 – 2023)

→Filtrere; df\_filtered for kommune hvor Average 2015-2023 er lik laveste gjennomsnitt

→Return kommuner med laveste gjennomsnittlige prosent

#### F (df\_filtered, headers)

#Legg in år som Input fra bruker

#If år finnes i dataen then →

- Beregn gjennomsnitt for df\_filtered (år), erstatte float(inf) med NA
- Returnere: gjennomsnittlig prosent for året

Else

Return feilmelding: «Året finnes ikke i dataen».

Slutt

G

Start

Spør bruker om å velge en kommune

Filtrer df\_filtered for valgt\_kommune →

- Hvis data finnes:
- Melt data til langt format
- Generer diagram med Altair (år og prosent, tilsvarende x og y akse)
- Lagre diagram som «xxxxx»
- Print «Diagram lagret»

Ellers: →

- Print: «Data ikke tilgjengelig

SLUTT

H

START

Filtrer df\_filtered for kommuner som har data fra alle årene.

Hvis minst 10 kommuner finnes:

- Beregn gjennomsnittlig prosent for hver enkelt kommune
- Sorter og velge de 10 høyeste
- Melt data til langt format

- Genererer diagram med Altair (kommune vs. Gjennomsnittlig prosent)
- Lagre diagram som «XXXX»
- Print «Diagram lagret»

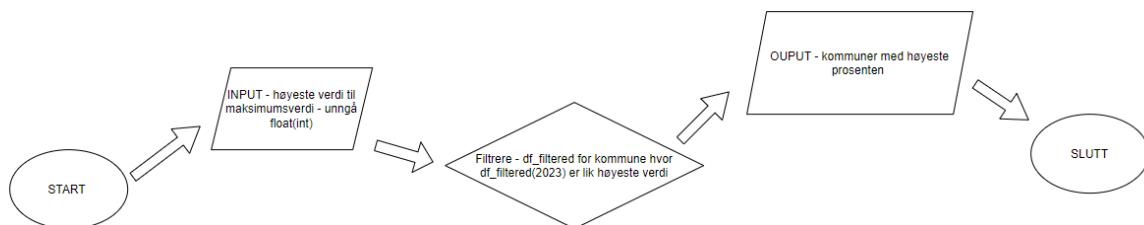
Else:

- Print «Ikke nok kommuner for å generere data»

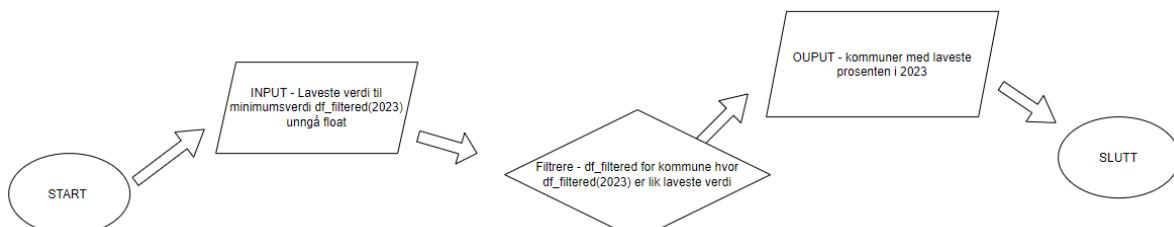
SLUTT

L2.

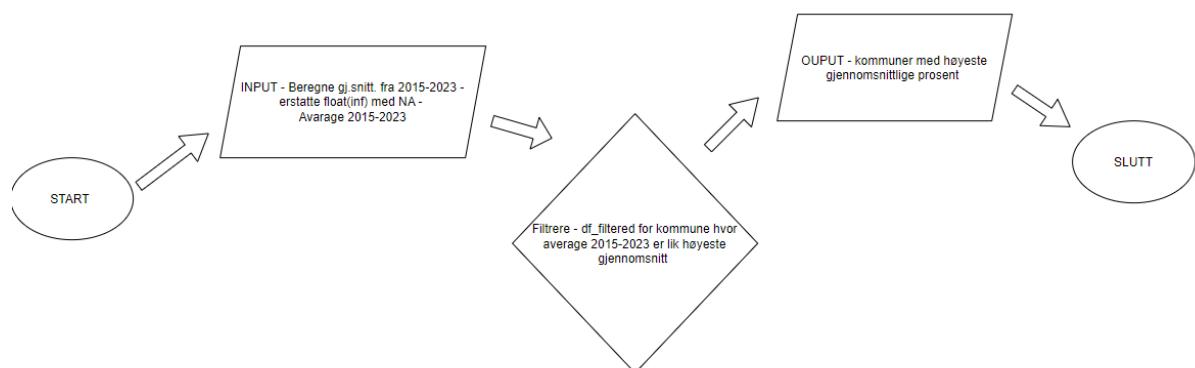
A -



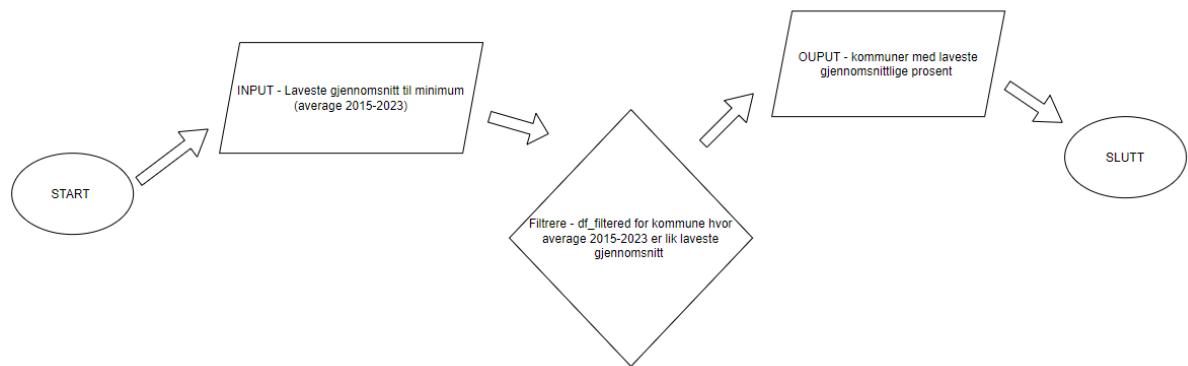
B -



C -

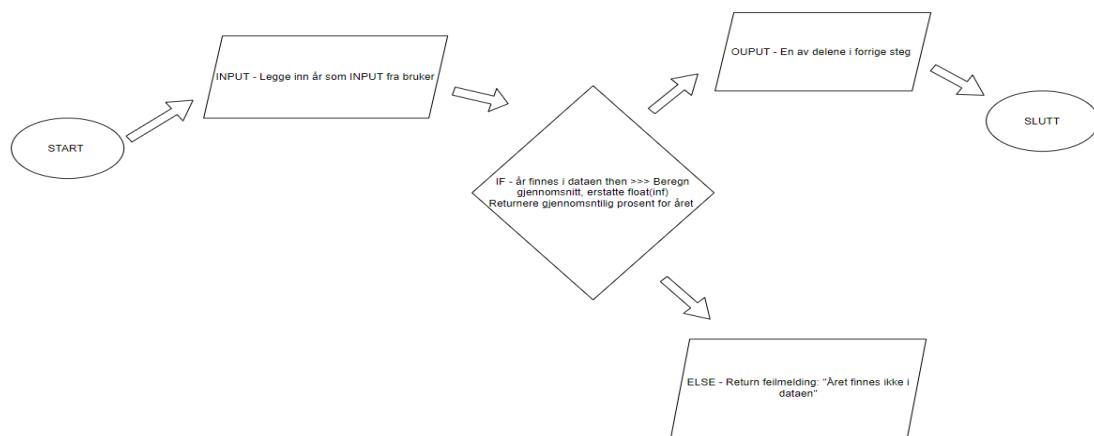


D -

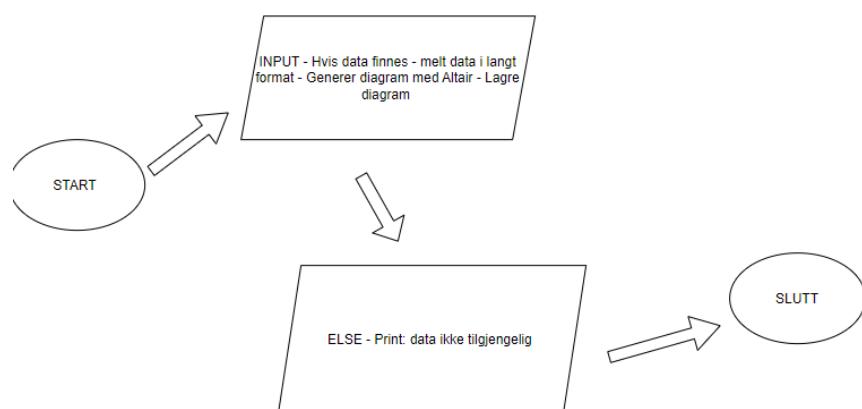


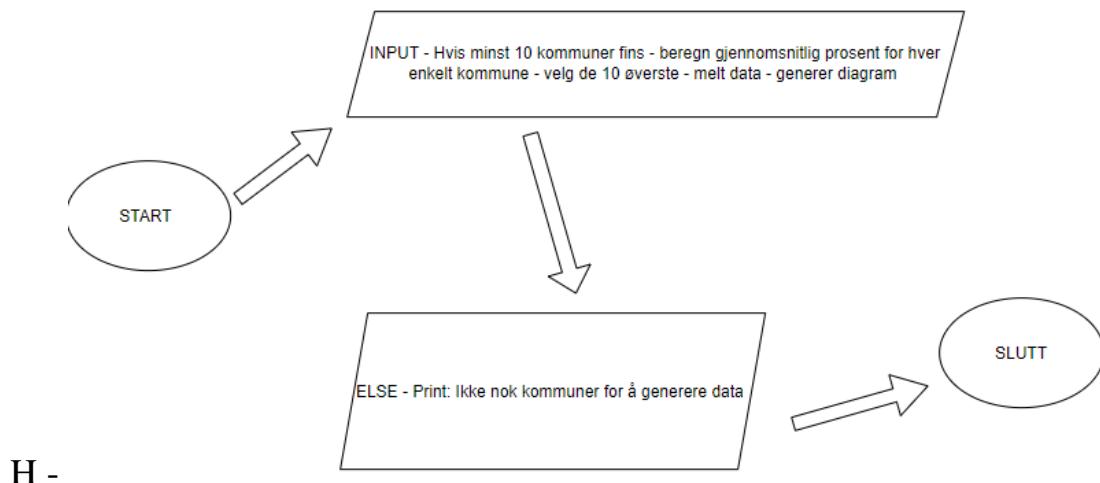
E – Samme som C

F –



G –





L3.

#### A. Kontrakt

- **Funksjon: find\_highest\_2023(df)**
- **Input → df (Dataframe): Inneholder relevant data fra kommunene i periode fra 2015 til 2023**
- **Output → En «dataframe» med info knyttet til kommunene med høyest prosentandel barn i ett- og toårsalderen i barnehagen i årgangen 2023.**

#### B. Kontrakt

- **Funksjon: find\_lowest\_2023(df)**
- **Input → df(DataFrame): Inneholder data fra kommunene mellom 2015 og 2023.**
- **Output → Dataframe med info angående kommunene med laveste prosent barn i ett- og to-årsalderen i barnehagen i 2023**

#### C. Kontrakt

- **Funksjon: find\_highest\_avg\_2015\_2023(df)**
- **Input → df(DataFrame): inneholder kommunedata i en periode fra 2015 til 2023**
- **Output: DataFrame med kommunene som har hatt den høyeste gjennomsnittlige prosenten i perioden 2015- 2023.**  
Resultat i run boksen nedenfor funksjon.

#### D. Kontrakt

- **Funksjon: find\_lowest\_avg\_2015\_2023(df)**

- **Input → df(DataFrame):** Inneholder data fra hver enkelt kommune i perioden 2015 til 2023
- **Output → Dataframe med informasjon angående hvem som har hatt den laveste gjennomsnittlige prosenten fra 2015 til 2023**

#### **E. Samme som C ...**

#### **F. Kontrakt:**

- **Funksjon: find\_average\_for\_year(df, year)**
- **Input → df(DataFrame):** Inneholder kommunedata fra 2015 til 2023 & **year(string):** Året som skal analyseres – brukeren av funksjonen plotter inn et gitt år f.eks. 2017 som visualisert i koden
- **Output → En «float» med gjennomsnittlig prosent for alle kommuner det spesifikke året, for eksempel 2017 (må være et år mellom 2015 og 2023, ellers feilmelding = Året finnes ikke)**

#### **G. Kontrakt**

- **Funksjon: generate\_chart\_for\_municipality(df, kommune)**
- **Input → df(DataFrame):** Kommunedata fra 2015 til 2023 & **«Kommune»(string):** Navnet på kommunen som skal visualiseres
- **Output → Genererer og lagrer et diagram som viser prosentandel barn i ett- to års alderen for kommunen over årene. De spesifikke kommunene.**

#### **H. Kontrakt**

- **Funksjon: generate\_top10\_chart(df)**
- **Input → df(DataFrame):** Inneholder kommunedata fra 2015 til 2023
- **Output → Diagram som visualiserer gjennomsnittlig prosent barn for de 10 kommune med høyest andel barn i barnehagen**

## **L4. Done**

## L5.

```
import gspread as gs
import pandas as pd
import altair as alt

gc = gs.api_key("AIzaSyBKrUqkbU3gsEujXEh8N3uQTN7fM0Dpg3I")
sh = gc.open_by_key('1RYN0i4Zx_UETVuYacgaGfnFcv4l9zd9toQTTdkQkj7g')
wsh = sh.worksheet("barnehage1-2aar")
all_values = wsh.get_all_values()

for i, row in enumerate(all_values):
    if '2015' in row:
        header_row_index = i
        break

136 # Oppgave A: Hvilken kommune har hatt den høyeste prosenten av barn i ett- og to-årsalderen i barnehagen i 2023?
137 highest_2023 = df_filtered[df_filtered['2023'] == df_filtered['2023'][df_filtered['2023'] != float('inf')].max()]
138 print("\nTask A: Kommune med høyeste prosent i 2023:")
139 print(highest_2023[['Kommune', '2023']])
140
141 # Oppgave B: Hvilken kommune har hatt den laveste prosenten av barn i ett- og to-årsalderen i barnehagen i 2023?
142 lowest_2023 = df_filtered[df_filtered['2023'] == df_filtered['2023'][df_filtered['2023'] != float('inf')].min()]
143 print("\nTask B: Kommune med laveste prosent i 2023:")
144 print(lowest_2023[['Kommune', '2023']])
145
146 # Oppgave C: Hvilken kommune har den høyeste gjennomsnittlige prosenten av barn i ett- og to-årsalderen fra 2015 til 2023?
147 df_filtered['Average_2015_2023'] = df_filtered[headers].replace(float('inf'), pd.NA).mean(axis=1).round(1)
148 highest_avg = df_filtered[df_filtered['Average_2015_2023'] == df_filtered['Average_2015_2023'].max()]
149 print("\nTask C: Kommune med høyeste gjennomsnittlige prosent fra 2015 til 2023:")
150 print(highest_avg[['Kommune', 'Average_2015_2023']])
151
152 # Oppgave D: Hvilken kommune har den laveste gjennomsnittlige prosent av barn i ett- og to-årsalderen fra 2015 til 2023?
153 lowest_avg = df_filtered[df_filtered['Average_2015_2023'] == df_filtered['Average_2015_2023'].min()]
154 print("\nTask D: Kommune med laveste gjennomsnittlige prosent fra 2015 til 2023:")
155 print(lowest_avg[['Kommune', 'Average_2015_2023']])
156
157 # Oppgave F: Hva er den gjennomsnittlige prosenten for alle kommuner fra et spesifikt år mellom 2015 og 2023?
158 year = input("\nOppgi et år mellom 2015 og 2023 for å finne gjennomsnittet for alle kommuner: ")
159 if year in headers:
160     average_for_year = df_filtered[year].replace(float('inf'), pd.NA).mean().round(1)
161     print(f"Gjennomsnittlig prosent for alle kommuner i {year}: {average_for_year}%")
162 else:
163     print(f"Feil: Året {year} finnes ikke i dataene. Vennligst velg et år mellom 2015 og 2023.")
```

## L6.

Denne oppgaven var krevende. Det var store mengder kode, hvor DCIC ikke var til stor hjelp, nettopp på grunn av vanskelighetsgraden. Jeg benyttet med av kunnskapen jeg har rundt grunnleggende programmering, samt ChatGPT for å utføre koden. Det tok flere forsøk og utallige timer å gjennomføre oppgaven på grunn av dens omfang og nivå. Måten jeg løste problemene på, var å prøve med egen kunnskap i første omgang for så å benytte meg av AI for tips til å unngå feilmeldinger og rettelse av koder. Feilmeldingene i starten av mine forsøk på koden var mange, men klarte etter hvert å snevre den mer og mer inn, til den slutt fungerte. En slik tilnærming til oppgaven vil jeg påstå er den mest fornuftige i denne situasjonen. DCIC kan ofte være litt avansert hvertfall dersom man henger bak på når det gjelder å lese alle kapitlene til hver uke. Derfor vil bruk av AI til å avdekke små feil med tegn osv. være hensiktsmessig for å effektivisere prosessen til en vesentlig grad. Resultatet jeg fikk var greit, grafene mine var ikke det beste, ettersom dataene ikke var så synlige, men fikk det ikke til å stemme uansett om jeg endret målet på hvor stor dataframen osv. skulle være. Så grafene var OK, men selve funksjonen fungerte, som igjen er det viktigste. Til neste gang vil jeg prøve å være enda mer selvstendig, å sette seg enda mer inn i kodingen på forhånd for så å prøve å kode på egen hånd. Ved å benytte seg av denne fremgangsmåten vil en oppnå mestringssfølelse i større grad. I tillegg vil selvstendig arbeid, uten bruk av AI, medføre en bredere forståelse for programmeringens helhet og dybde. Alt i alt er jeg fornøyd med resultatet selv om jeg brukt enormt mye tid på det. Fornøyd.

Tusen Takk for fine tilbakemeldinger på forrige oblig 😊

Referanseliste:

ChatGPT →

OpenAI. (2024). *ChatGPT* (23.oktober versjon) <https://chatgpt.com/c/67177509-5034-8013-bb34-4248d90551d6>

DCIC →

Pyret.org. (2023). <https://dcic-world.org/2024-09-03/dictionaries.html>

Flytdiagram →

NapkinAI. (2024). *App.diagrams.net*.

<https://app.diagrams.net/#G1wlhdO91GXu6dq6t68exUORzVcuyhuKuN%7B%22pageId%22%3A%22xeZj-vV2CHF6SzETBt2k%22%7D>