UNIVERSITETET I BERGEN

KANDIDAT

834

PRØVE

INF100 0 Innføring i programmering

Emnekode	INF100
Vurderingsform	Skriftlig eksamen
Starttid	30.11.2020 08:00
Sluttid	30.11.2020 13:00
Sensurfrist	--
PDF opprettet	02.10.2022 11:17

Eksamensinformasjon & Egenerklæring

Oppgave	Status	Poeng	Oppgavetype
i			Informasjon eller ressurser
i			Informasjon eller ressurser

Kontaktinformasjon under eksamen

Oppgave	Status	Poeng	Oppgavetype
1	Ubesvart	0/0	Muntlig

Datatyper (5 %)

Oppgave	Status	Poeng	Oppgavetype
2	Delvis riktig	4/5	Paring

Snippets (20 %)

Oppgave	Status	Poeng	Oppgavetype
3	Riktig	2/2	Nedtrekk
4	Riktig	2/2	Nedtrekk
5	Riktig	2/2	Paring
6	Riktig	2/2	Nedtrekk
7	Riktig	2/2	Nedtrekk
8	Riktig	2/2	Nedtrekk
9	Riktig	2/2	Nedtrekk
10	Riktig	2/2	Nedtrekk
11	Riktig	2/2	Nedtrekk

12	Riktig	2/2	Nedtrekk

Programmering (3 x 25%)

Oppgave	Status	Poeng	Oppgavetype
13	Besvart	13/25	Programmering
14	Besvart	14/25	Programmering
15	Besvart	19/25	Programmering

1

David Grellscheid er tilgjengelig på Discord for faglige spørsmål om eksamensinnhold.

I kanalen #exam_questions kan du klikke på konvolutt-ikonet. Det åpnes en privat "ticket"-kanal hvor du kan legge inn ditt spørsmål.

Kunngjøringer som er relevant for alle skjer i #announcements kanalen.

Alle andre kanaler blir ignorert.

Om du ikke er med i Discord ennå, kan du joine her:
https://discord.gg/uAf5VaN

--
Om dere har spørsmål relatert til eksamen generelt, praktiske problemer, problemer med innlogging eller systemet generelt ta kontakt med studieveileder i studieadministrasjonen ved Institutt for Informatikk. Det er 2 forskjellige kanaler:

Per e-post: studieveileder@ii.uib.no
i emnefeltet skriv: INF100 - eksamen
i selve e-posten skriver du studentnummeret & kandidatnummeret ditt
Beskriv kort hva problemet

Per telefon i denne rekkefølgen

1. 55 58 41 59 - Eirik R. Thorsheim
2. 55 58 41 82 - Mo Yan Yuen
3. 55 58 32 95 - Marianne K. Holmedal
4. 55 58 30 31 - Tone Stokka
5. 55 58 90 14 - Iselin T. Tjensvold

Når du tar kontakt med oss i studieadministrasjon (enten per e-post eller telefon) ber vi deg å ha følgende informasjon tilgjengelig:

kandidatnummeret ditt (3 sifre, finnes i Inspera & studentweb)
studentnummeret ditt (6 sifre, finnes på studentkortet ditt)
kontaktinfoen (telefon & e-post) dersom vi må henvise saken din videre.

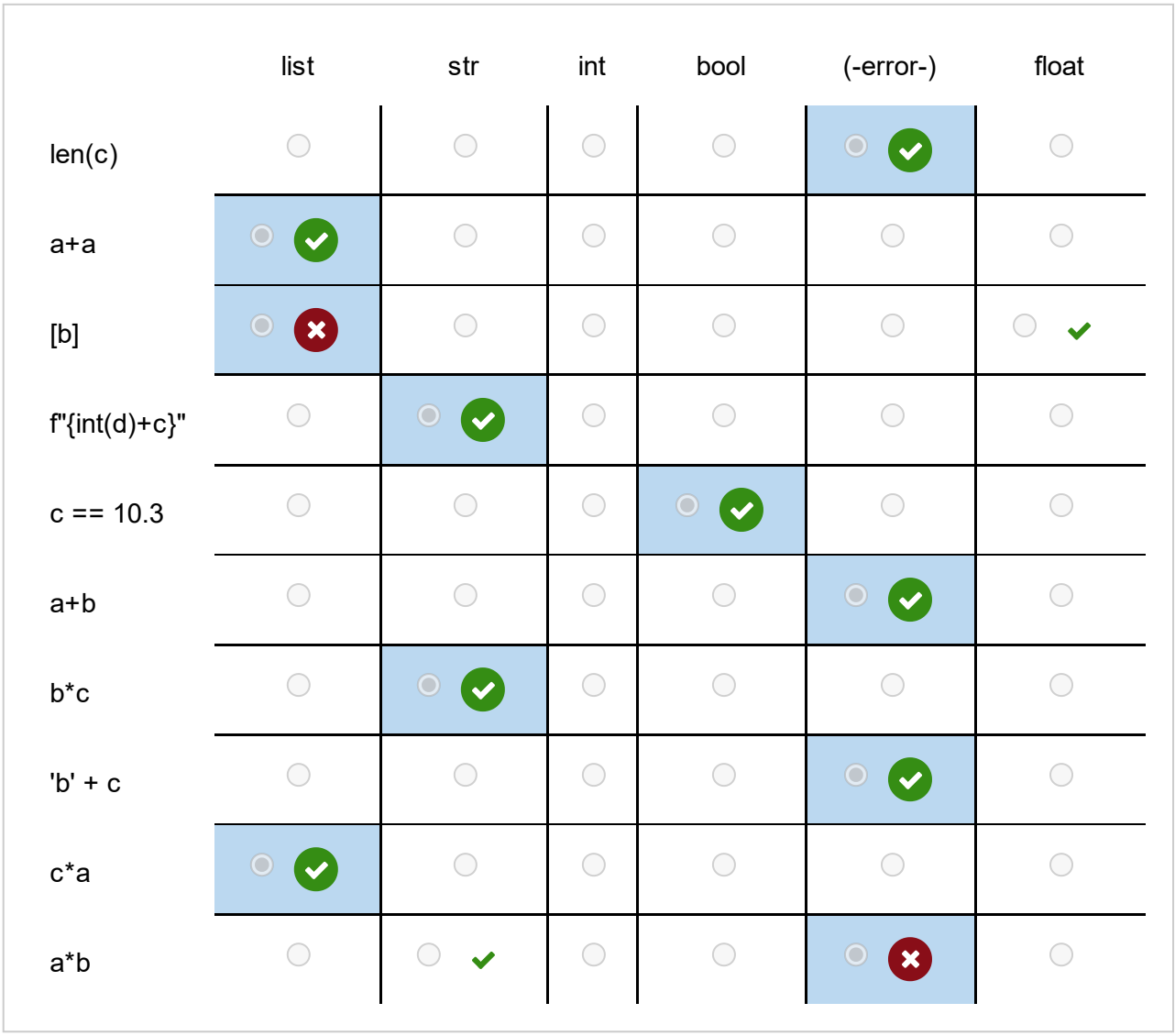
For generelle eksamensinformasjon har fakultetet laget en infoside:
https://www.uib.no/matnat/56756/eksamen-ved-det-matematisk-naturvitenskapelige-fakultet#eksamen-og-korona-nbsp-ofte-stilte-sp-rsm-l

Maks poeng: 0

2

- a = []
- b = "True"
- c = 42
- d = -1.5

Velg riktig data type



Maks poeng: 5

3 Spør om 5 ord og skriv ut summen av lengdene

length = 0

for _ in range(5): ✓ (while False:, for _ in range(5):, while True:, for length in range(5):)

- text = input("Text: ") ✓ (open("Text: "), input("Text: "), read("Text: "), print("Text: "))
- length += len(text) ✓ (text += length, length = len(text), length + len(text), length += len(text))

print(f"The texts had { length } ✓ ((len{text}), { len(text) }, { length }, length) characters.")

Maks poeng: 2

4 Velg slik at alle sammenligninger er True. dict xs ser slik ut:

```
xs = {
'a' : 5,
'5' : 'hello',
'z' : 3.1415,
5 : 7
}
```

xs[5] ✓ (xs['5'], xs['a'], xs[a], xs[5]) == 7

'5' in xs.keys() ✓ (xs.items(), xs.values(), xs.keys(), xs.setdefault())

list(xs.items())[-1] ✓ (tuple(xs[-1]), list(xs.items())[-1], tuple(xs[5]), list(xs.keys())[-1])== (5,7)

len(xs['5']) ✓ (xs[5], xs[7], len(xs['5']), len(xs[5])) == xs['a']

Maks poeng: 2

5 Velg resultatet av hvert *boolsk* uttrykk.

	False	True
False and True	✓	
not (not (not False))		✓
5 < 7 or 4 > 5		✓
5 in range(5)	✓	
25 // 2 == 12		✓
[x**2 for x in range(3)] == [0,1,4,9]	✓	
18 < 20 < 21 < 27 < 25	✓	
list(range(3)) == [1,2,3]	✓	
list(zip([4],[7])) == [(4,7)]		✓
True or False		✓

Maks poeng: 2

6 Gitt to tall a og b, returner True om begge er partall, ellers False

def both_even(a,b):

- return a % 2 == 0 and b % 2 == 0 ✓ (return a % 2 and b % 2 == 0, return a % 2 == 0 and b % 2 == 0, return a and b % 2 == 0, return a % 2 == 0 or b % 2 == 0)

Maks poeng: 2

7 Skriv løkken med *while* i steden for *for*:

sum = 0
for x in xs[:3]:

- if x > 5:
  - sum += x

---

i = 0 ✓ (i = xs, i = len(xs), i = None, i = 0)

sum = 0

while i < 3: ✓ (i <= 3:, x < len(xs):, i < 3:, x < xs:)

- x = xs[i] ✓ (i = xs[i], x = xs[0], x = xs[i], x = xs[3])
- if x > 5:
  - sum += x
- i += 1 ✓ (i += 1, return x, break, x += 1)

Maks poeng: 2

8 Returner True om *input* listen er sortert fra små til stor, ellers returner False

def is_sorted(input):

- x = input[0]
- for e in input[1:]:
  - if x > e:
    - return False ✓ (return True, continue, break, return False)
  - x = e
- return True ✓ (e != x, False, e == x, True)

Maks poeng: 2

9 Velg de riktige linjene slik at outputet blir:
A
B
C

a = 450

if a > 100: ✓ (if a > 100:, if 'a' > 100:, if a < 100:, if 'a' < 100:)

- print('A')

if a > 400: ✓ (elif a > 400:, else:, if a > 400:, elif a < 400:)

- print('B')

if a % 10 == 0: ✓ (elif a % 10 == 0:, if a % 10 != 0:, elif a % 10 != 0:, if a % 10 == 0:)

- print('C')

elif a < 1000: ✓ (elif a < 1000:, elif:, if a < 1000:, if a < 500:)

- print('D')

Maks poeng: 2

10 Velg slik at alle sammenligninger er True. Listen xs ser slik ut: xs = [3, "hei", False, [7]]

xs[1] ✓ (xs[-1], xs[1], xs[3], xs[2:3]) == 'hei'

'e' == xs[1][1] ✓ (xs[1][1], xs[1,1], xs[1 1], xs[1:1])

xs[-1:-3:-1] ✓ (xs[-1:-3:-1], xs[-1:-3], xs[-3:-1], xs[-3:-1:-1])== [[7],False]

len(xs[3]) ✓ (len(xs[3]), len(xs[1]), len(xs), len(xs[2])) == 1

Maks poeng: 2

11 Hvor ofte finnes *x* i *input* listen

def count(input, x):

- ct = 0
- for i in input:
  - if i == x:
    - ct += 1 ✓ (ct += 1, ct = i, break, ct = x)
- return ct ✓ (i, x, input, ct)

Maks poeng: 2

12 Les hver linje fra filen frem til vi finner 'Alice'. Skriv ut linjenummer der vi stoppet

filename = "foo.txt"
with ✓ (open, with, read, file) open(filename) ✓ (open(filename), with(filename), read(filename), filename) as f: ✓ (to f:, as f:, with f:, from f:)

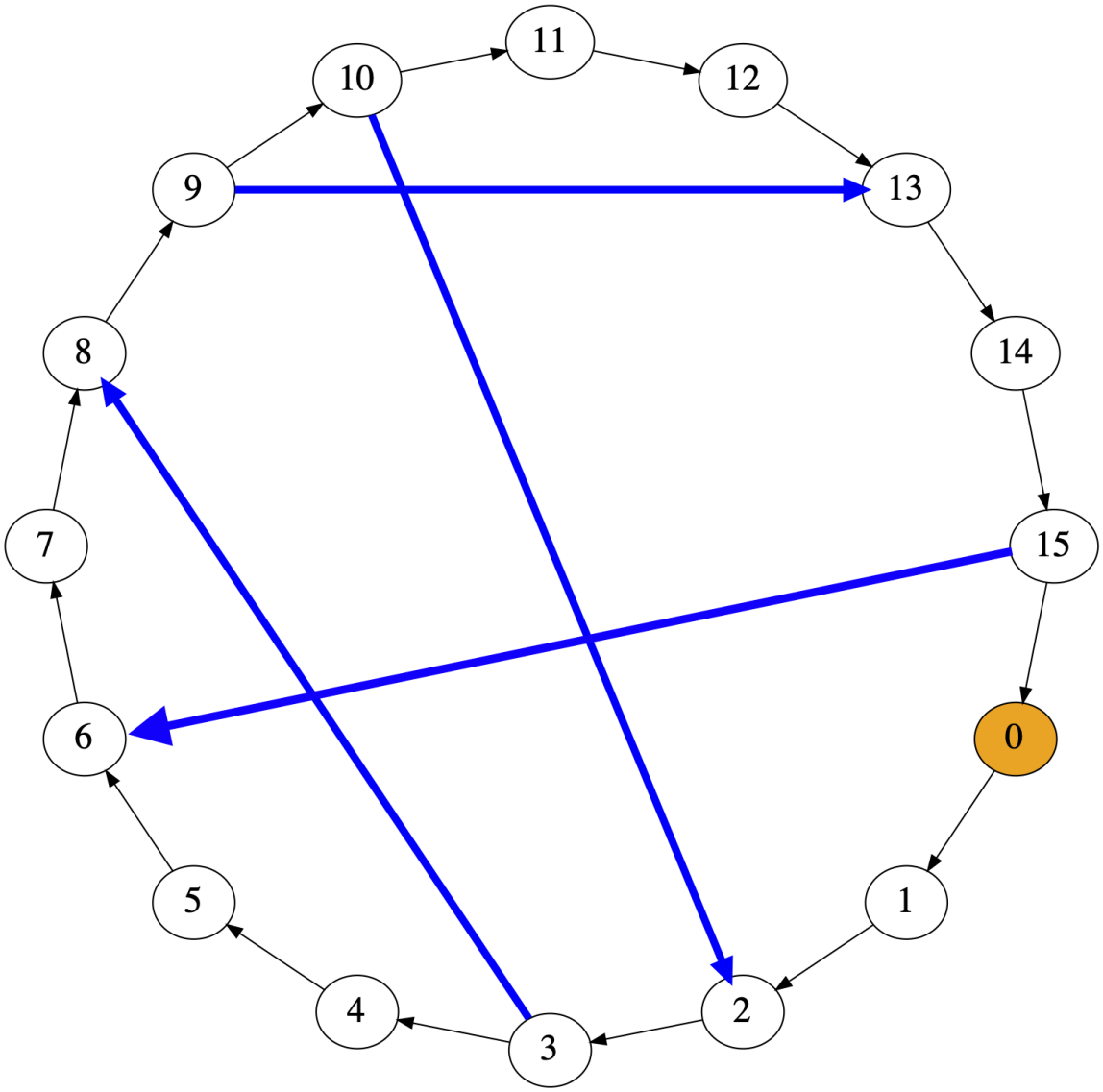
- for i, line in enumerate(f) ✓ (for i, line in enumerate(f), for i, line in zip(f), line = f.read(), line = f.readlines()):
  - if 'Alice' in Line:
    - print(f'Alice found in line {i+1}')
    - break

Maks poeng: 2

13 Oppgaven har to deler A og B med ulik vekt.

I en variasjon av spillet "Slanger og Stiger" finnes 16 felt arrangert i en sirkel, nummerert fra 0 til 15.

Spillere begynner på 0. Hver omgang kaster de én terning (1-6) og går videre et tilsvarende antall steg. Om spilleren avslutter på feltene 3, 9, 10, eller 15 (der det finnes slanger / stiger), **må** spilleren gå videre/tilbake til 8, 13, 2, eller 6.



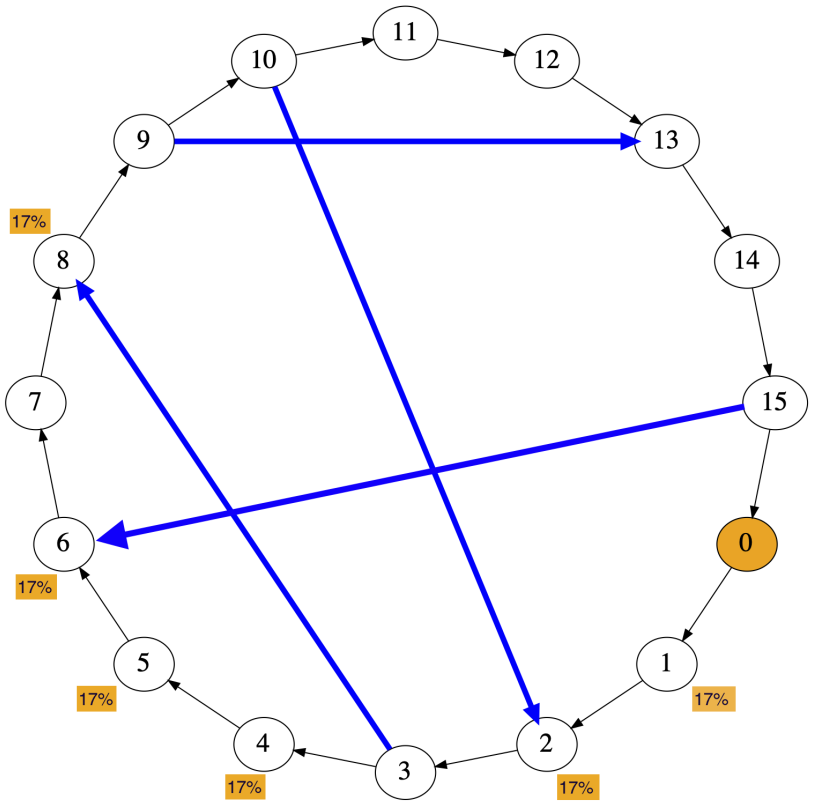
Alle spillere får 20 omganger, og vinneren er den som krysset 0 som oftest. (Slangen fra 15 til 6 teller **ikke** som en kryssing). Vi skal simulere mange tusen spillere, og se på hvilke felt de kommer å stå på etter 20 omganger.

Et eksempel: Først står alle spillere (100%) på felt 0, og ingen på de andre feltene. Vi skal printe det på en linje som 16 prosentverdiene (en for hver felt):

100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Etter den første omgangen finnes omtrent 1 av 6 spillere (17%) på hver av feltene 1,2,4,5,6,8.

Ingen står på 3 siden de slippet videre til 8.



Den situasjonen skriver vi ut slikt som prosentverdiene:
0 17 17 0 17 17 17 0 17 0 0 0 0 0 0 0

Etter to omganger har vi:
0 0 14 0 6 8 11 14 17 0 0 8 6 14 3 0

Oppgaven
(Del A - 14 poeng)
Skriv en funksjon **def simulate(rounds)** som simulerer 100000 spillere. **rounds** er et heltall som angir antallet omganger som skal simuleres. Funksjonen skal **print**'e hvor spillere befinner seg etter **rounds** omganger (du skal runde til hele prosent):

simulate(0) -> 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
simulate(1) -> 0 17 17 0 17 17 17 0 17 0 0 0 0 0 0 0
simulate(2) -> 0 0 14 0 6 8 11 14 17 0 0 8 6 14 3 0
simulate(20) -> 5 6 13 0 5 5 11 7 14 0 0 6 6 13 7 0

(Tallene kan avvike litt i din simulasjon)

(Del B - 6 poeng)
Tilpass funksjonen slikt at vi kan også printe ut hvor mange av spillere klarte å gå rundt flere ganger:

simulate(20) skal printe ut

```
5   6  13   0   5   5  11   7  14   0   0   6   6  13   7   0
   148 players managed  0 rounds (  0 %)
  2432 players managed  1 rounds (  2 %)
 13784 players managed  2 rounds ( 14 %)
 32791 players managed  3 rounds ( 33 %)
 34315 players managed  4 rounds ( 34 %)
 14330 players managed  5 rounds ( 14 %)
  2103 players managed  6 rounds (  2 %)
    96 players managed  7 rounds (  0 %)
     1 players managed  8 rounds (  0 %)
```

(Tallene kan avvike litt i din simulasjon)

Tips

Bruk random.randint(1,6) eller lignende for terningen.
Et dict() er nyttig for slanger / stiger: board[3] = 8; board[10] = 2; ...

Skriv ditt svar her

```python
import random

def simulate(rounds):

    brett = {0 : 10_000, 1 : 0, 2 : 0, 3 : 0, 4 : 0, 5 : 0, 6 : 0, 7 : 0, 8 : 0, 9
        14 : 0, 15 : 0}

    # Hvor mange runder som skal gjøres
    for runde in range(rounds):

        #Temp brett:
        tempBrett = {0 : 0, 1 : 0, 2 : 0, 3 : 0, 4 : 0, 5 : 0, 6 : 0, 7 : 0, 8 : 0,
            , 14 : 0, 15 : 0}

        # Ser hvor spillere er på brettet
        for plassering in brett:


            # For hver spiller i plasseringen
            for spillere in range(brett[plassering]):

                terning = random.randint(1,6)
                flytt_til = plassering + terning

                # Land på 3 flytt til 8
                if flytt_til == 3:
                    tempBrett[8] += 1

                # Land på 9 flytt til 13
                elif flytt_til == 9:
                    tempBrett[13] += 1

                # Land på 10 flytt til 2
                elif flytt_til == 10:
                    tempBrett[2] += 1

                # Land på 15 flytt til 6
                elif flytt_til == 15:
```

```
                elif flytt_til > 15:
                    tempBrett[6] += 1

                # Ellers bare flytt
                else:
                    if flytt_til > 15:
                        tempBrett[flytt_til - 15] += 1
                    else:
                        tempBrett[flytt_til] += 1




        brett.update(tempBrett)


    txt = ""
    for i in brett.values():
        prosent = i/10_000*100
        txt = txt + str(int(prosent)) + " "

    return print(txt)
```

Maks poeng: 25

14 Fila https://folk.uib.no/dgr061/INF100/NO_ADM12.csv er eit CSV-fil som inneheld ei oversikt over alle norske fylke og kommunar (adaptert frå http://www.geonames.org/ CC-BY-3.0).

Kolonne 1 ("name") viser namnet til fylke eller kommune

Kolonne 5 ("feature code") viser "ADM1" for fylke, og "ADM2" for kommunar.

Kolonne 7 ("admin1 code") viser to sifrer (01-20) og kan brukast til å finna ut kva fylke eit gitt kommune høyrer til.

Oppgåver

Programmet ditt skal gjera det følgjande:

(a) Lesa data frå fila inn i passande datastrukturar. Det er lurt å skilja mellom fylke og kommunar allereie her. Du kan bruka vanlig filhåndtering eller csv-biblioteket

(b) skriva ut dei fem største og dei fem minste kommuner (sortert etter kolonne 9 ("population"))

(c) definera ein funksjon **def print_fylke(num)** der num er ein streng frå "01" til "20" . Funksjonen skal så **printa** ut namnet til fylke og ei alfabetisk liste til alle kommunane i fylket med talet på innbyggarar. Pass på fin formatering her: namna til venstre, tala til høgre.

print_fylke("01") skal printa:

```
==========================
01 Akershus fylke
==========================
Asker                     53756
Aurskog-Høland            14158
Bærum                    109700
Eidsvoll                  20321
Enebakk                   10153
Fet                       10139
Frogn                     14435
Gjerdrum                   5567
Hurdal                     2621
Lørenskog                 32300
Nannestad                 10800
Nes                       18629
Nesodden                  17129
Nittedal                  20555
Oppegård                  24612
Rælingen                  15345
Skedsmo                   46668
Ski                       27699
Sørum                     14942
Ullensaker                28138
Vestby                    14095
Ås                        15863
```

(d) Lag ein løkke der du bruker **input("Search word [q to quit]? ")** for å spørja brukaren om eit delvis fylkenamn fleire gonger fram til brukaren svarer "q".
Kvar gong, sjekk om det finst eit fylke som passar. Viss ja, skal du kalla **print_fylke**-funksjonen. Elles skal du skriva ei melding til brukaren og dei prøver igjen.

Ei dømekøyring:

```
Search word (q to quit)? Hord
==========================
07 Hordaland Fylke
==========================
Askøy                    24432
Austevoll                4417
Austrheim                2576
...
Search word (q to quit)? Foo
No matching fylke found. Try again
Search word (q to quit)? q
Bye!
```

Skriv ditt svar her

```
import csv

def hilo():

    # Åpner først for å finne gjennomsittet slik at vi har et utgangspunkt for lis
    with open("NO_ADM12.csv", newline='', encoding='iso-8859-1') as f:
        doc = csv.reader(f, delimiter=';')

        totalt = 0
        k = 0
        for snitt in doc:
            try:
                totalt += int(snitt[9])
                k += 1
            except ValueError:
                pass


        gjennomsnitt = int(totalt / k)


        Høyest = {1:gjennomsnitt, 2:gjennomsnitt, 3:gjennomsnitt, 4:gjennomsnitt,
        Lavest = {1:gjennomsnitt, 2:gjennomsnitt, 3:gjennomsnitt, 4:gjennomsnitt,

    # Så åpner vi filer for og finne de høyeste verdiene
    with open("NO_ADM12.csv", newline='', encoding='iso-8859-1') as f:
        doc = csv.reader(f, delimiter=';')

        for befolkning in doc:
            try:
                populasjon = int(befolkning[9])
            except ValueError:
                continue
            # Hvis det er over gjennomsnittet
            if populasjon >= gjennomsnitt:

                for k, v in Høyest.items():

```

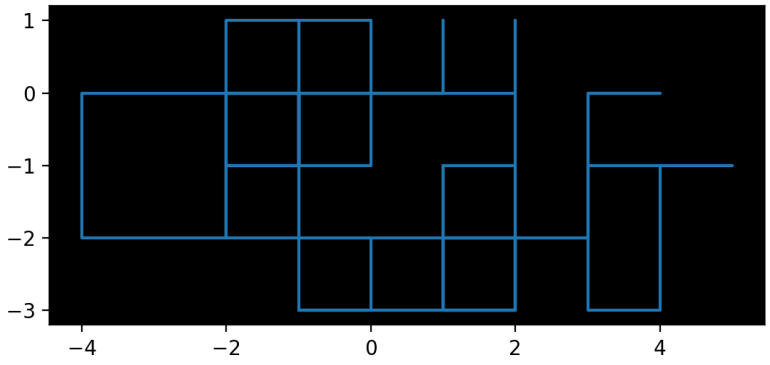
```
                    # Hvis den har en høyere verdi en de som er på topp listen
                    if populasjon > v:
                        Høyest[k] = populasjon
                        break

    # så de laveste verdiene
    with open("NO_ADM12.csv", newline='', encoding='iso-8859-1') as f:
        doc = csv.reader(f, delimiter=';')
        for folk in doc:
            try:
                populasjon = int(folk[9])
            except ValueError:
                continue

            # Hvis det er under gjennomsnittet
            if populasjon <= gjennomsnitt:

                for k, v in Lavest.items():

                    if populasjon < v:
                        Lavest[k] = populasjon
                        break


    return print(f"Top 5: {Høyest}. De minste 5: {Lavest}")



def print_fylke(num):

    Kommuner = {}

    with open("NO_ADM12.csv", newline='', encoding='iso-8859-1') as f:
        doc = csv.reader(f, delimiter=';')

        #Finner fylket
        for code in doc:

            if code[5] == "ADM2" and code[7] == str(num):
                Kommuner.setdefault(code[1], code[9])

            if code[5] == "ADM1" and code[7] == str(num):
                fylke = code[1]

    print("--------------------------------")
    print(f"{num} {fylke}")
    print("--------------------------------")
    for k, v in Kommuner.items():
        print(f"{k:16} {v}")



def loop():
    while True:
        inn = input("Serch word [q to quit]?")

        if inn == "q":
            break

        with open("NO_ADM12.csv", newline='', encoding='iso-8859-1') as f:
            doc = csv.reader(f, delimiter=';')
            for word in doc:
                if word[5] == "ADM1":
                    result = word[1].find(inn)
                    if result == 0:
                        print_fylke(word[7])
                        break
                    elif result == -1:
```

```
107                 continue
108
109             print("No matching fylke found. Try again")
```

Maks poeng: 25

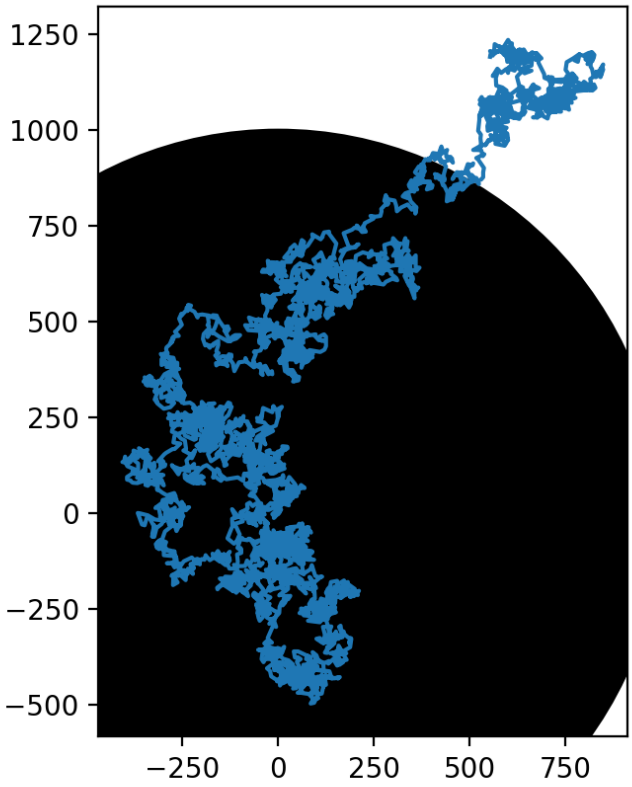
15 Last ned filen https://folk.uib.no/dgr061/INF100/walk.py . Du skal tilpasse filen for å løse de følgende oppgavene.

walk.py simulerer en "Random walk" (tilfeldig tur) i to dimensjoner. Vi begynner på (0,0) og tar 100 tilfeldige skritt opp, ned, til venstre eller til høyre. Turen er lagret i numpy array **steps**. Du trenger ikke å endre den delen der vi genererer **steps**.
Hver linje i **steps** inneholder den nåværende x- og y-posisjon, slik at vi enkelt kan plotte turen:



Oppgaver (lim inn den endelige koden som fullfører alle deler)

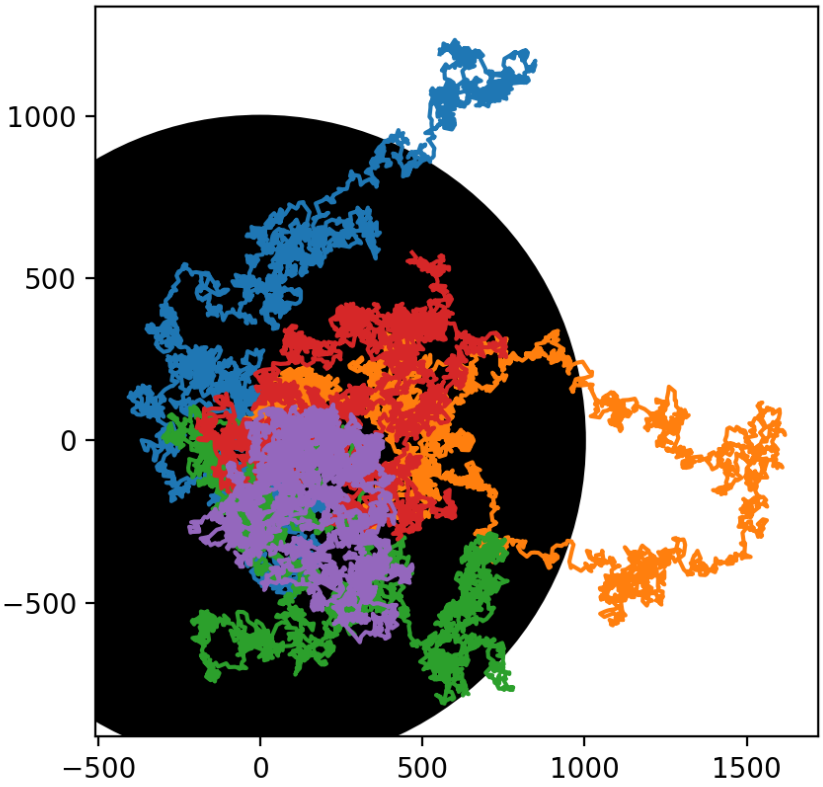
(a) Tegn 1000000 (en miljon) skritt i steden for 100



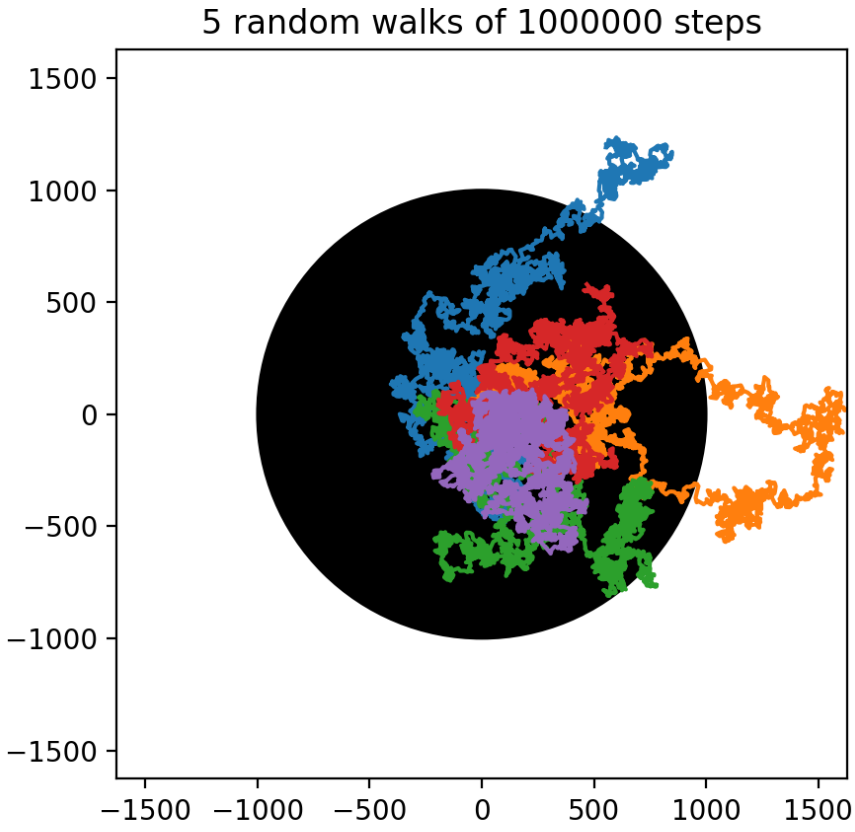
(b) Lag ein ny int-variabel **repeats** som angir hvor mange ganger vi skal simulere hele turen

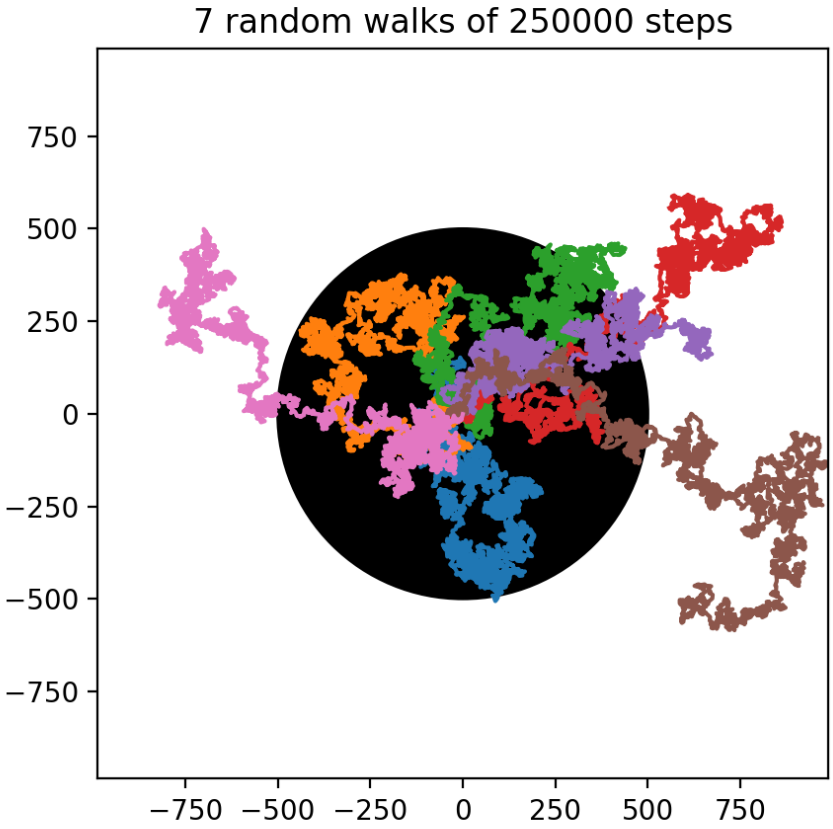
Skriv en for-løkke som repeterer simulasjonen frem til plt.plot() **repeats** ganger:

Eksempel for repeats=5, med 1M skritt:

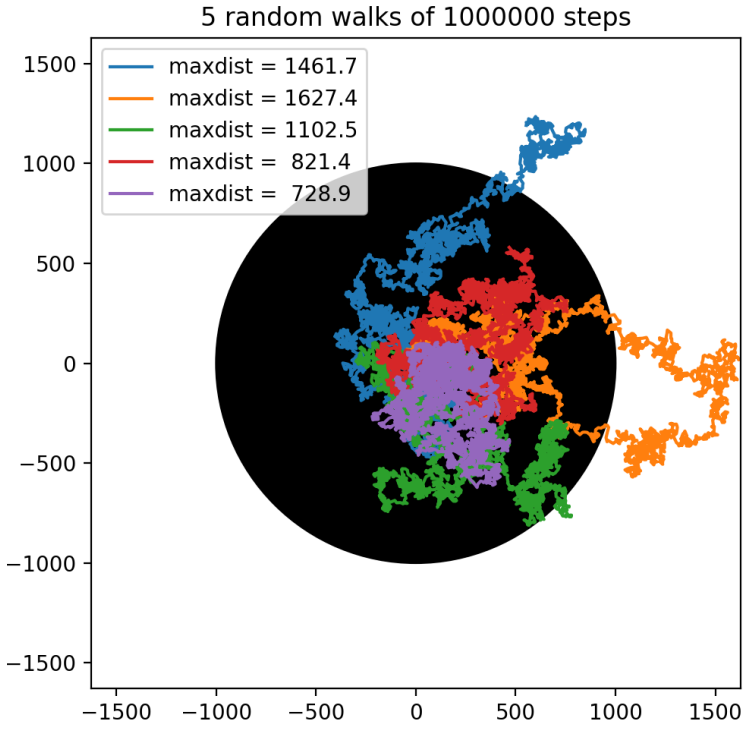


(c) Legg inn plot-overskriften og endre aksene slik at (0,0) er i midten og alle turer er helt synlig. Overskriften og akse-grensene skal tilpasses når vi endrer **N_steps** og/eller **repeats**:





(d) For hver tur, finn maksimal avstand fra sentrum (0,0). Avstand blir regnet ut som $\boldsymbol{d = \sqrt{x^2 + y^2}}$. (Tips: du kan jobbe med hele numpy-arrayene xs og ys for å få avstandene ds, og så ta maximum)
Legg inn maksimal avstand i en "legend":



Skriv ditt svar her

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(12)

N_steps = 1_000_000
repeats = 5
expected_R = np.sqrt(N_steps)

for i in range(repeats):
    ###################################
    #     generate one random walk    #
    ###################################
    # a list of 4 directions 0,1,2,3
    dirs = np.random.randint(0, 4, N_steps)
    # a 2D list of steps, empty for now
    steps = np.empty((N_steps,2))
    # fill the list of steps according to direction
    steps[dirs == 0] = [0,1]  # 0 - right
    steps[dirs == 1] = [0,-1] # 1 - left
    steps[dirs == 2] = [1,0]  # 2 - up
    steps[dirs == 3] = [-1,0] # 3 - down
    ###################################
    # use cumsum to sum up the individual steps to get current position
    steps = steps.cumsum(axis=0)
    ###################################
    print('Final position:',steps[-1])
    sqrt = int(np.sqrt(steps[-1][0]**2 + steps[-1][1]**2))

    ###################################
    # draw only a selection of points, max 5000, to save memory
    skip = N_steps//5000 + 1
    xs = steps[::skip,0]
    ys = steps[::skip,1]
    plt.plot(xs,ys, label=sqrt)
```

```
    ###################################

###################################
# add a circle with expected distance
circle = plt.Circle((0,0), radius=expected_R, color='k')
plt.gcf().gca().add_artist(circle)
# equal axis size
plt.gcf().gca().set_aspect('equal')
###################################


plt.title(f"{repeats} random walks of {N_steps} steps")

# Hvor langt det er i fra der de startet til hvor de sluttet
plt.legend()
plt.show()
```

Maks poeng: 25