Introduksjon

Oppgave	Maks poeng	Oppgavetype
i		Informasjon eller ressurser
i		Informasjon eller ressurser
i		Informasjon eller ressurser

Automatisk rettet

Oppgave	Maks poeng	Oppgavetype
1(a)	5	Paring
1(b)	3	Fyll inn tekst
1(c)	2	Nedtrekk
1(d)	2	Nedtrekk
1(e)	2	Nedtrekk
1(f)	2	Nedtrekk
1(g)	2	Flervalg
1(h)	2	Fyll inn tekst
1(i)	2	Fyll inn tekst
1(j)	2	Fyll inn tekst
1(k)	2	Fyll inn tekst
1(l)	2	Fyll inn tekst
1(m)	5	Fyll inn tekst

Forklaring

Oppgave	Maks poeng	Oppgavetype
2(a)	10	Langsvar
2(b)	10	Langsvar
2(c)	10	Langsvar

Kodeskriving

Oppgave	Maks poeng	Oppgavetype
3(a)	3	Programmering
3(b)	4	Programmering
3(c)	20	Programmering
3(d)	10	Programmering

Egenerklæring

Jeg erklærer herved at besvarelsen som jeg leverer er mitt eget arbeid.
Jeg har ikke:

- samarbeidet med andre studenter
- brukt andres arbeid uten at dette er oppgitt
- brukt eget tidligere arbeid (innleveringer/ eksamenssvar) uten at dette er oppgitt

Om jeg har benyttet litteratur, vil en litteraturliste inneholde alle kilder jeg har brukt i besvarelsen og referanser vil vise til denne listen.

Jeg er kjent med at brudd på disse bestemmelsene er å betrakte som fusk og kan føre til annullert eksamen og/eller utestengelse.

Dersom du er usikker på om du kan stille deg bak erklæringen, se retningslinjer for bruk av kilder i skriftlige arbeider ved Universitetet i Bergen , og eventuelt ta kontakt med studieveileder/emneansvarlig.

Alle eksamensbesvarelser ved UiB blir sendt til manuell og elektronisk plagiatkontroll.

Merk: Ved å fortsette bekrefter jeg at jeg har lest erklæringen og at besvarelsen jeg leverer under denne eksamenen er mitt eget arbeid (og bare mitt eget arbeid), i full overensstemmelse med ovennevnte erklæringen.

Informasjon

Eksamen består av tre deler, og gir tilsammen 100 poeng:

1. Automatisk rettede oppgaver (totalt 33 poeng)
2. Forklaringsoppgaver (totalt 30 poeng)
3. Kodeoppgaver (totalt 37 poeng)

I tredje del vil du *ikke* ha anledning til å kjøre koden du skriver. Sensor er klar over dette, og vil ikke trekke poeng for småfeil som f. eks. skrivefeil i funksjonsnavn. Du må likevel skrive koden så tydelig og korrekt som mulig, slik at du demonstrerer at du forstår nyanser i koden du skriver.

Generelle råd

- Les spørsmålet før du svarer
- Jobb deg *raskt* gjennom alle spørsmålene i første runde, og kom heller tilbake til krevende oppgaver på nytt hvis du får tid på slutten.
- Hjelp sensor å hjelpe deg! Hvis du er usikker på tolkningen av en oppgave, gi en kort kommentar om hvordan du tolker usikkerheter i oppgaven. Hvis du ikke husker presist hvordan noe skal gjøres med kode, skriv en kommentar som forklarer hva du prøver på.

1(a)

- a = "Hello, how are you!"
- b = { 3, 5, 1 }
- c = -1.4
- d = 64
- e = [42, "Bye", -3]

Velg datatype til uttrykket

	str	(-error-)	bool	list	float	int
3 in e	○	○	○	○	○	○
e*d	○	○	○	○	○	○
a+a	○	○	○	○	○	○
[b]	○	○	○	○	○	○
b+d	○	○	○	○	○	○
a*d	○	○	○	○	○	○
c*d	○	○	○	○	○	○
a[e[2]]	○	○	○	○	○	○
len(b)	○	○	○	○	○	○
f'{b}'	○	○	○	○	○	○

Maks poeng: 5

1(b)

```
a = [1, 0, -2, 2, 5, 3]
```

Gitt at koden over er kjørt.
Hva skrives ut i de følgende setningene? (hvis programmet krasjer, skriv kun 'Error')

print(a[4])	
print(a[3 - 1])	
print(a[3] - 1)	
print(a[a[-1]-1])	
print(a[a[0] - a[1]])	
print(a[a[a[0]]])	

Maks poeng: 3

1(c) Velg slik at alle sammenligningene er True. Dict'et xs ser slikt ut:

```
xs = {
    'a' : 5,
    '5' : 'hello',
    7 : 9.3781,
    5 : [7, 'Bergen', {3,5}],
}
```

Velg alternativ (xs[5], xs[5][2], xs[5][1], xs[2]) == 'Bergen'

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

Velg alternativ (len(xs[5]) * 4, xs[5][0] + xs['a'], xs[5][0] + len(xs['5']), xs[5][1] * 2) != 12

Velg alternativ (len(xs[5][2]), xs[5], xs[7], len(xs['5'])) == xs['a']

Maks poeng: 2

1(d) Velg slik at alle sammenligninger blir True. Sets xs og ys ser slikt ut:

```
xs = set("Welcome")
ys = set("Velkommen")
```

Velg alternativ (xs.append(ys), xs, xs.union(ys), xs.intersection(ys)) == set("elom")

set("Welcom") == Velg alternativ (xs.union(ys), ys.union(xs), ys, xs)

Velg alternativ (set("Velkomn"), set("Velkome"), set("WVelkcomn"), set("Welcom"))== xs.union(ys)

Velg alternativ (len(xs), len(xs.union(ys)), len(xs.intersection(ys)), len(ys)) == 7

Maks poeng: 2

1(e) Velg verdien til dette boolske uttrykket:

a	b	c	(a and b) or c
True	False	False	Velg alternativ (True, False)
True	True	False	Velg alternativ (False, True)
True	False	True	Velg alternativ (True, False)
False	False	True	Velg alternativ (True, False)

Maks poeng: 2

1(f) Skriv en funskjon som tar inn en liste med binærtall 0 og 1, og returnerer True hvis **kun** det første tallet i listen er 1. Ellers returnerer funksjonen False.

def only_first_is_one(list_of_digits):

- x = list_of_digits[0]
- if Velg alternativ (x == 0, x != 0) :
  - return Velg alternativ (True, False)
- for d in list_of_digits[1:]:
  - if Velg alternativ (d == 0, d == 1) :
    - Velg alternativ (return False, return True, continue, break)
- return True

Maks poeng: 2

1(g) Hint: bruk gjerne presedenstabellen i kursnotatene for å minne deg selv på hvilken operator som har presedens.

Hvordan plassere parenteser for å få et uttrykk *identisk* med

12 // 2 * 3

Velg ett alternativ

- 12 // (2 * 3)
- (12 // 2) * 3

Hvordan plassere parenteser for å få et uttrykk *identisk* med

x and y or z in a

Velg ett alternativ:

- x and (y or (z in a))
- x and ((y or z) in a)
- (x and y) or (z in a)
- (x and (y or z)) in a
- ((x and y) or z) in a

1 poeng gis for hvert riktig svar, 0 poeng for feil svar eller ubesvart.

Maks poeng: 2

1(h)

```
b = 3
a = a * b
b = a + b
a -= 1
print(b - a)
```

Hva skriver dette programmet ut? (hvis programmet krasjer, skriv kun 'Error')

Maks poeng: 2

1(i)

```
def foo(x):
    x += 2
    return x
x = 10
y = foo(x)
print(x + y)
```

Hva skriver dette programmet ut? (hvis programmet krasjer, skriv kun 'Error')

Maks poeng: 2

1(j)

```
def woz(s):
    s += 'buzz'
    print(s, end=' ')
s = 'fizz'
r = woz(s)
print(f'{s} {r}')
```

Hva skriver dette programmet ut? (hvis programmet krasjer, skriv kun 'Error')

Maks poeng: 2

1(k)

```
def qiz(x, a):
    for e in a:
        if x % 2 == 0:
            x += e
    return x
q = [2, 4, 5, 6]
print(qiz(0, q))
```

Hva skriver dette programmet ut? (hvis programmet krasjer, skriv kun 'Error')

Maks poeng: 2

1(l)

```
def charlie(p, q):
    r = p - q
    s = delta(r, p)
    s = delta(s, q)
    return s
def delta(t, u):
    v = t + u
    return v - 1
print(charlie(5, 2))
```

Hva skriver dette programmet ut? (hvis programmet krasjer, skriv kun 'Error')

Maks poeng: 2

1(m)

```
def foxtrot(x):
    if x >= 10:
        return 10
    else:
        x += 10
    if x > 10:
        if x % 2 == 0:
            x += 1
        elif x >= 15:
            x -= 1
    else:
        return 42

    return x - 10
```

Gitt at funksjonen over er definert.
Hva skrives ut i de følgende setningene? (hvis programmet krasjer, skriv kun 'Error')

print(foxtrot(0))	
print(foxtrot(2))	
print(foxtrot(3))	
print(foxtrot(5))	
print(foxtrot(foxtrot(2))	

Maks poeng: 5

2(a)

```
def count_a(s):
    for c in range(len(s)):
        count = 0
        if c == "a":
            count += 1
        return count
```

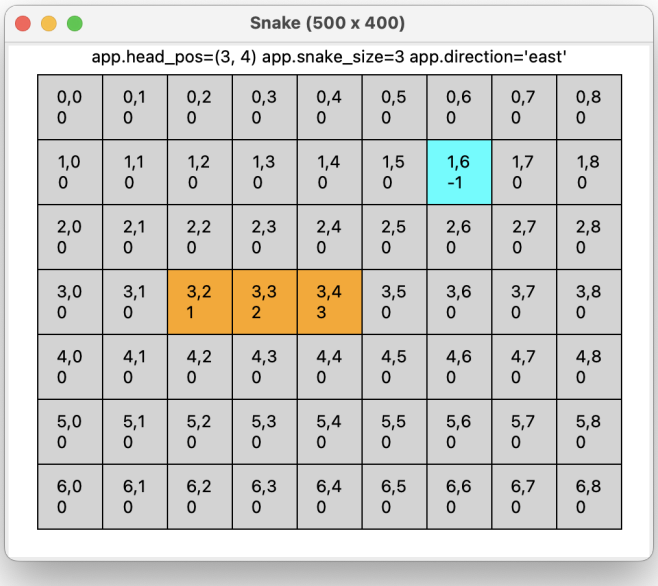
Koden over skal telle hvor mange ganger tegnet "a" opptrer i en streng **s**, men gir feil svar. Forklar hva som er feil, og hva man kan gjør for å fikse funksjonen.

Ca to-tre avsnitt, helst ikke mer enn 200 ord.

Skriv ditt svar her

Maks poeng: 10

2(b)



- Forklar forskjellen på en destruktiv og en ikke-destruktiv funksjon.
- Vis til eksempler på begge deler i det vedlagte løsningsforslaget til lab6

Skriv ditt svar her

Maks poeng: 10

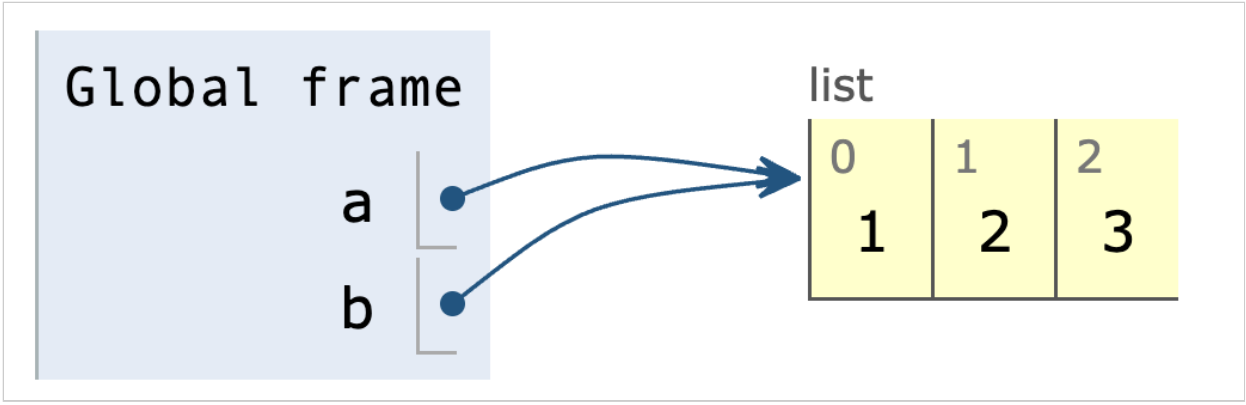
2(c) Lille Tidemann (10) skal snart ha eksamen i INF100, men skjønner ikke helt hva et oppslagsverk (dict) er. Skriv en innføring som hjelper ham. Bruk forklaringer med ord sammen med illustrerende eksempler på kode. Inkluder ditt eget eksempel på en realistisk situasjon der det er lurt å bruke et oppslagsverk for å løse oppgaven.

Vi forventer ca 250 ord.

Skriv ditt svar her

Maks poeng: 10

3(a)

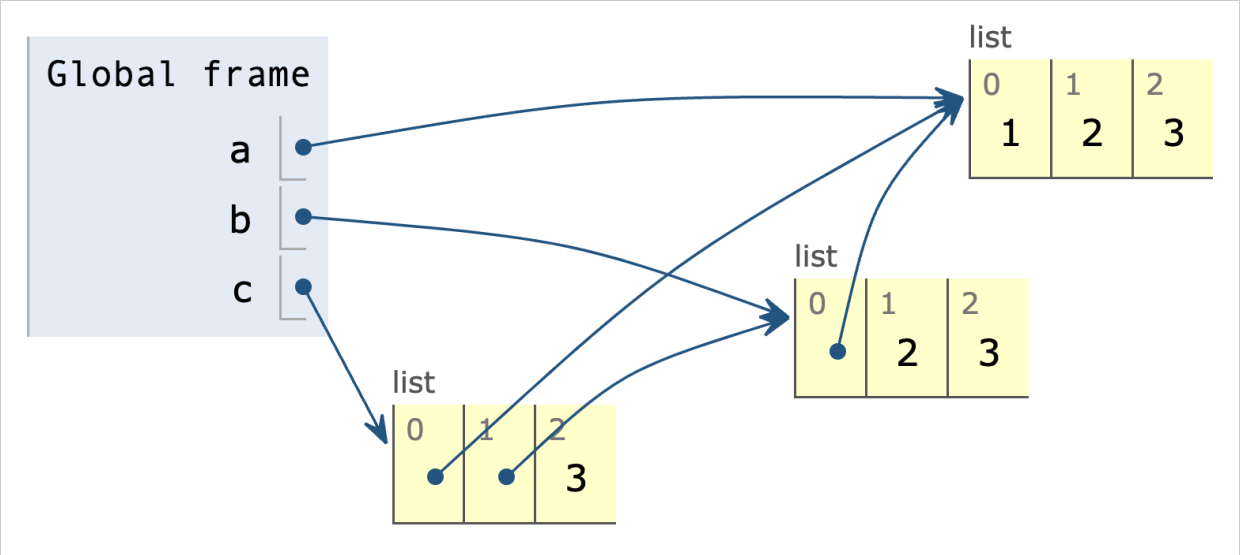


Opprett to variabler a, og b som refererer til den samme listen, slik at minnets tilstand blir som vist på bildet over (bilde er hentet fra https://pythontutor.com/)

Skriv ditt svar her

Maks poeng: 3

3(b)



Opprett tre variabler a, b og c, slik at minnets tilstand blir som vist på bildet over (bilde er hentet fra https://pythontutor.com/)

Skriv ditt svar her

Maks poeng: 4

3(c)

Skriv ditt svar her

Maks poeng: 20

3(d) Anta at **a** er en liste med heltall (int). Vi sier at en sammenhengende subsekvens av etterfølgende tall i listen **a** utgjør en *serie* dersom alle tallene er positive, eller alle tallene er negative. Tallet 0 er hverken positivt eller negativt, og kan ikke inngå i noen serie.

Eksempel:
a = [5, 0, 1, 3, 6, 4, 0, 0, 3, 3, 1, -7, -8, -2]

De «maksimale» seriene (de seriene som ikke kan utvides videre) i listen **a** over er:

- 5
- 1, 3, 6, 4
- 3, 3, 1
- -7, -8, -2

Skriv en funksjon **longest_series_start** med en parameter **a**. La funksjonen returnere *indeksen* hvor den lengste serien i **a** begynner. I eksempelet over skal altså returverdien bli **2**, siden serien (1, 3, 6, 4) begynner på indeks 2 og dette er den lengste serien i listen.
Hvis det er flere enn én slik lengste serie, skal den tidligste av dem returneres. Hvis det ikke er noen serier i a, skal verdien **-1** returneres.

Skriv ditt svar her

Maks poeng: 10

Question 15
Attached

```
from uib_inf100_graphics import *
import random

################
### Initialisering av modellen
################

def app_started(app):
    app.timer_delay = 100 # milliseconds
    app.debug_mode = True
    init(app)

def init(app):
    # Gjør klart for et nytt spill
    app.board = new_default_board()
    app.snake_size = get_max_value_in_2dlist(app.board)
    app.head_pos = position_of_value_in_2dlist(app.board, app.snake_size)
    app.direction = "east"
    app.state = "active"

def new_default_board():
    return [
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0,-1, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 2, 3, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0],
    ]

### Hjelpefunksjoner for å initalisere modellen

def get_max_value_in_2dlist(a):
    max_value = 0
    for row in a:
        for value in row:
            max_value = max(max_value, value)
    return max_value

def position_of_value_in_2dlist(list2d, value):
    for row in range(len(list2d)):
        for col in range(len(list2d[row])):
            if list2d[row][col] == value:
                return (row, col)


################
### Kontrollere
################

def timer_fired(app):
    if app.debug_mode:
        return
    if app.state == "active":
        move_snake(app)

def key_pressed(app, event):
    if event.key == "d":
        app.debug_mode = not app.debug_mode
    if app.state == "gameover":
        return
```

```
    if event.key == "Space": move_snake(app)
    elif event.key == "Up": app.direction = "north"
    elif event.key == "Down": app.direction = "south"
    elif event.key == "Left": app.direction = "west"
    elif event.key == "Right": app.direction = "east"


### Hjelpefunksjoner for kontrollere

def move_snake(app):
    # Finn neste posisjon for hodet
    next_row, next_col = get_next_head_position(
        old_head_row=app.head_pos[0],
        old_head_col=app.head_pos[1],
        direction=app.direction
    )
    # Sjekk om neste posisjon er lovlig/sjekk for game over
    if not is_legal_move(next_row, next_col, app.board):
        app.state = "gameover"
        return
    # Flytt slangekroppen og spis epler
    if app.board[next_row][next_col] == -1:
        # Spiser eple
        app.snake_size += 1
        add_apple_at_random_location(app.board)
    else:
        # Trenger ikke å trekke fra 1 når slangen spiser eple
        subtract_one_from_all_positives(app.board)
    # Flytt hodet
    app.head_pos = (next_row, next_col)
    app.board[next_row][next_col] = app.snake_size

def get_next_head_position(old_head_row, old_head_col, direction):
    if direction == "east": return (old_head_row, old_head_col + 1)
    elif direction == "west": return (old_head_row, old_head_col - 1)
    elif direction == "north": return (old_head_row - 1, old_head_col)
    elif direction == "south": return (old_head_row + 1, old_head_col)

def is_legal_move(row, col, board):
    if not 0 <= row < len(board):
        return False
    if not 0 <= col < len(board[row]):
        return False
    return board[row][col] <= 0

def add_apple_at_random_location(board):
    available_positions = []
    for row in range(len(board)):
        for col in range(len(board[row])):
            if board[row][col] == 0:
                available_positions.append((row, col))
    row, col = random.choice(available_positions)
    board[row][col] = -1

def subtract_one_from_all_positives(list2d):
    for row in list2d:
        for i in range(len(row)):
            row[i] = row[i] - 1 if row[i] > 0 else row[i]


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

```
### Visning
################

def redraw_all(app, canvas):
    # debug-info
    if (app.debug_mode):
        s = f'{app.head_pos=} {app.snake_size=} {app.direction=} {app.state=}'
        canvas.create_text(app.width/2, 3, text=s, anchor="n")
    # brette
    if (app.state == "active"):
        draw_board(canvas, 25, 25, app.width-25, app.height-25,
                   board=app.board, debug_mode=app.debug_mode)
    # game over
    if (app.state == "gameover"):
        canvas.create_text(app.width/2, app.height/2,
                           text="Game Over", font="Arial 26", fill="black")

### Hjelpefunksjoner for visningen

def get_color(value):
    # < 0: eple, 0: tom rute, > 0: slange
    if   value < 0:  return "cyan"
    elif value == 0: return "lightgray"
    else:            return "orange"

def draw_board(canvas, x1, y1, x2, y2, board, debug_mode):
    rows = len(board)
    cols = len(board[0])

    cell_width = (x2 - x1) / cols
    cell_height = (y2 - y1) / rows

    for row in range(rows):
        for col in range(cols):
            # Tegne rektangelet
            cell_x1 = x1 + col * cell_width
            cell_y1 = y1 + row * cell_height
            cell_x2 = cell_x1 + cell_width
            cell_y2 = cell_y1 + cell_height
            color = get_color(board[row][col])
            canvas.create_rectangle(cell_x1, cell_y1, cell_x2, cell_y2,
                                    fill=color)
            # Tegne teksten
            if debug_mode:
                text = f"{row},{col}\n{board[row][col]}"
                cell_mid_x = (cell_x1 + cell_x2) / 2
                cell_mid_y = (cell_y1 + cell_y2) / 2
                canvas.create_text(cell_mid_x, cell_mid_y, text=text)


################
### Kjør programmet
################

if __name__ == '__main__':
    run_app(width=500, height=400, title="Snake")
```

Question 19
Attached

Nyttårsløpet 2017

I denne oppgaven skal du lage fem funksjoner som del av et rapporteringssystem fra et idrettsstevne «Nyttårsløpet 2017» som finner sted 31. desember 2017.

Funksjonene skal operere på en liste av deltakere. Hver deltaker i listen er igjen representert som en liste av 5 elementer: en streng som inneholder fornavn, en streng som inneholder etternavn, en bokstav som representerer kjønn ('K' for kvinne og 'M' for mann), et heltall som er fødselsåret, og en streng som representerer tiden vedkommende klarte å gjennomføre en halvmaraton på. ('1:59:20' betyr 1 time, 59 minutter og 20 sekunder.) Et eksempel på en slik liste av lister er som følger:

```
data = [
    ['Kari', 'Hansen', 'K', 1969, '1:59:20'],
    ['Eli', 'Nansen', 'K', 1975, '1:49:46'],
    ['Karl', 'Jansen', 'M', 1985, '1:35:40'],
    ['Erik', 'Karlsen', 'M', 1970, '1:40:48'],
    ['Anne', 'Jensen', 'K', 1964, '2:03:09'],
    ['Kurt', 'Johnsen', 'M', 1987, '1:32:43'],
    ['May', 'Berntsen', 'K', 1989, '1:36:54'],
    ['Jan', 'Thorsen', 'M', 1990, '1:45:24'],
    ['Hans', 'Monsen', 'M', 1998, '1:25:05'],
]
```

Som del av rapporteringssystemet skal du lage følgende fem Python-funksjoner.

- group
- search
- winners
- time
- save_file

Funksjonen *time* er en hjelpefunksjon til *winners*.

group

Parametre:

- `data` en liste av lister som representerer resultatene fra et idrettsstevne som beskrevet over.
- `gender` en streng som enten er `'K'` eller `'M'`.
- `age_lower` en int som representerer nedre aldersgrense (inklusiv)
- `age_upper` en int som representerer øvre aldersgrense (eksklusiv)

Returverdi: *ingen*

Sideeffekter:

- Alle deltakere i `data` som har kjønn `gender` og som har alder mellom `age_lower` og `age_upper` på arrangementsdagent skal skrives ut på formatet `fornavn etternavn (alder) tid` på hver sin line.

Eksempelutskrift for et kall til `group(data, 'K', 25, 50)` gitt `data` som beskrevet over:

```
Kari Hansen (48) 1:59:20
Eli Nansen (42) 1:49:46
May Berntsen (28) 1:36:54
```

search

Parametre:

- `data` en liste av lister som representerer resultatene fra et idrettsstevne som beskrevet over.
- `word` en streng som skal søkes etter.

Returverdi:

- en liste som representerer deltakerne fra `data` hvor `word` finnes i enten fornavnet eller etternavnet til deltakeren. Formatet på resultatlisten skal være akkurat den samme som for `data`, altså en liste av lister.

Sideeffekter: *ingen*

Eksempel på tester:

```
# Test 1
actual = search(data, 'Hans')
expected = [
    ['Kari', 'Hansen', 'K', 1969, '1:59:20'],
    ['Hans', 'Monsen', 'M', 1998, '1:25:05'],
]
assert expected == actual

# Test 2
actual = search(data, 'M')
expected = [
    ['May', 'Berntsen', 'K', 1989, '1:36:54'],
    ['Hans', 'Monsen', 'M', 1998, '1:25:05'],
]
assert expected == actual
```

winners

Parametre:

- `data` en liste av lister som representerer resultatene fra et idrettsstevne som beskrevet over.

Returverdi: *ingen*

Sideeffekter:

- Den raskeste mannen og den raskeste kvinnen skal skrives ut på hver sin linje på formatet vist i eksempelet under.

Eksempelutskrift for et kall til `winners(data)` gitt data som i eksempelet over:

```
The fastest woman is May Berntsen with time 1:36:54
The fastest man is Hans Monsen with time 1:25:05
```

time

Parametre:

- `s` en streng på formatet `H:MM:SS` som representerer en tid hvor *H* er antall timer, *MM* er antall minutter og *SS* er antall sekunder.

Returverdi:

- En int for antall sekunder som tilsvarer tiden `s`.

Sideeffekter: *ingen*

Dersom `s` ikke er på riktig format skal funksjonen krasje.

Eksempel på test:

```
actual = time("1:59:20")
expected = 7160
assert expected == actual
```

save_file

Parametre:

- `data` en liste av lister som representerer resultatene fra et idrettsstevne som beskrevet over.
- `filename` en streng som representerer filnavnet som resultatene skal lagres i.

Returverdi: *ingen*

Sideeffekter:

- Resultatene i `data` skal lagres i filen `filename` i et CSV-format hvor verdier separeres med semikolon. Første linje skal inneholde egnede overskrifter. De neste radene i filen skal inneholde fornavn, etternavn, kjønn, fødselsår og tid for hver av deltakerne.