Egenerklæring

Jeg erklærer herved at bidraget jeg sender inn, er mitt eget arbeid.

Jeg har ikke

- samarbeidet med andre studenter
- brukt andres arbeid uten å oppgi dette
- brukt egne tidligere arbeider (innleveringer/eksamensbesvarelser) uten å oppgi dette

Hvis jeg har brukt litteratur, skal en litteraturliste inneholde alle kildene jeg har brukt i oppgaven, og referanser skal henvise til denne listen.

Jeg er klar over at brudd på disse bestemmelsene regnes som fusk og kan føre til annullering av eksamen og/eller utestenging.

Dersom du er usikker på om du kan stå inne for denne erklæringen, se retningslinjer for kildebruk i skriftlige arbeider ved Universitetet i Bergen, og ta kontakt med din studieveileder/emnekoordinator.

Alle eksamensbesvarelser ved UiB sendes til manuell og elektronisk plagiatkontroll.

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

Informasjon om eksamen

Eksamen består av tre deler, med til sammen 80 poeng:

Type	Automatisk graderte spørsmål	Forklaringsspørsmål	Kodingsspørsmål
Poeng	46	16	18

- Del 1 består av testspørsmål i tilfeldig rekkefølge. Hvis du har problemer, kan du hoppe over et spørsmål og gå tilbake til det senere. For å unngå å glemme spørsmål du har hoppet over, anbefaler vi at du bruker flaggingsalternativet i øvre høyre hjørne.
- Du bør ha startet del 2 og 3 når du har to timer igjen. I del 3 vil du ikke ha mulighet til å kjøre koden du skriver. Eksaminator er klar over dette og vil ikke trekke poeng for mindre feil, for eksempel skrivefeil i funksjonsnavn fra standardbiblioteket. Du må imidlertid skrive koden så tydelig og korrekt som mulig for å vise at du forstår nyansene i koden du skriver.
- Tillatte hjelpemidler til eksamen: alle skriftlige og trykte hjelpemidler.

Generelle råd

- Les spørsmålet før du svarer
- Arbeid deg raskt gjennom alle spørsmålene i første runde, og kom tilbake til utfordrende spørsmål igjen hvis du har tid på slutten.
- Hjelp sensoren med å hjelpe deg! Hvis du er usikker på tolkningen av et spørsmål, kan du skrive en kort kommentar om hvordan du tolker usikkerhetsmomentene i spørsmålet. Hvis du ikke husker nøyaktig hvordan du skal gjøre noe med kode, kan du skrive en kommentar som forklarer hva du prøver å gjøre.

Materiell

På denne siden finner du alle kursnotatene i kurset samlet i en PDF-fil. De er tilgjengelige for deg hvis du ønsker det. Det er helt frivillig å bruke dem. Vanlige regler for sitering gjelder: Hvis du kopierer noe fra kursnotatene, er det viktig at du oppgir dem som kilde i besvarelsen din.

Du kan også finne kursnotatene nederst i venstre hjørne under hele eksamen. Da åpnes notatene i en ny fane.

PS: Har sidene plutselig snudd sidelengs? Prøv å trykke på «R» for å snu dem riktig vei igjen.

1(a)

```
def check_outage(voltage_data, limit):
    for district in voltage_data[1:]:
        if district[1] < limit:
            district[2] = 'Outage'
voltage_data = [
    ['district', 'is_voltage', 'status'],
    ['Bønes', 120, 'OK'],
    ['Varden', 0, 'OK'],
    ['Sædalen', 350, 'OK'],
    ['Fantoft', 0, 'OK'],
    ['Åsane', 700, 'OK'],
]

new_voltage_status = check_outage(voltage_data, 10)
```

William jobber i et team som overvåker **strømforsyningen** i ulike bydeler i Bergen. Hver bydel sjekkes for spenningsnivåer, og hvis spenningen faller under en kritisk grense, markeres bydelen som strømløs.

Han har skrevet funksjonen `check_outage()` som tar inn en liste med spenningsdata og en grenseverdi. Hjelp William med å teste hva funksjonen gjør ved *å forutsi outputen for følgende fire uttrykk*:

Velg riktig datatype til uttrykkene:

	str	ERROR	list	int	NoneType	dict
print(type(voltage_data[3][3]))	○	○	○	○	○	○
print(type(new_voltage_status))	○	○	○	○	○	○
print(type(voltage_data[2][2]))	○	○	○	○	○	○
print(type(voltage_data[1][1]))	○	○	○	○	○	○

Maks poeng: 2

1(b)

```
bags = 42
brand = "Norwegian Rain"
price_drop = -15.75
display = [20, "Tote bag", 3]

a = bags + bags
b = display + [1]
c = bags * brand
d = bool(bags - price_drop)
```

Lisa er designpraktikant ved en anerkjent **motebutikk** som spesialiserer seg på yttertøy. Hun hjelper teamet med å bruke Python til å holde oversikt over produkter fra deres vårkolleksjon – en blanding av statement-plagg og dristige accessories:

Anta at kodesnutten over er kjørt. *Hva er datatypen til hver av variablene? (Hvis programmet krasjer på linjen, skriv "Error").*

Velg riktig datatype for variablene:

	int	float	str	bool	list	- ERROR -
a	○	○	○	○	○	○
b	○	○	○	○	○	○
c	○	○	○	○	○	○
d	○	○	○	○	○	○

Maks poeng: 2

1(c)

```
planks = 96
logo = "WoodyWorld"
clients = {102, 104, 107}
discount = -4.5
table = [120, "Skrivebord", 4]
```

Et digitalt system brukes til å holde oversikt over **verkstedets varelager**, kundeordre og produksjonsspesifikasjoner i en stor byggevarebutikk. Den forrige utvikleren har rotet til koden og blandet sammen flere uttrykk. Henrik har fått i oppgave å rydde opp. Hjelp ham med å *finne riktig datatype til hvert uttrykk.*

Velg riktig datatype til uttrykkene:

	-- ERROR --	list	float	str	int	bool
clients[2]	○	○	○	○	○	○
[clients]	○	○	○	○	○	○
table[1][table[2]]	○	○	○	○	○	○
"bord" in table	○	○	○	○	○	○
len(clients)	○	○	○	○	○	○
planks // planks	○	○	○	○	○	○

Maks poeng: 3

1(d)

```
def medicine_reminder(days):

    for day in range (1, days + 1):

        if day % 3 == 0:
            print(f'{day} drink your medicine!')

        else:
            print(day)
```

Ida er praktikant ved et legesenter, der hun jobber med å automatisere påminnelser basert på pasientenes **helsedata**. For eksempel har hun skrevet funksjonen nedenfor:

Hva er den siste dagen i februar 2025 (har 28 dager) hvor denne funksjonen skriver ut "husk å ta medisinen!"? Skriv et heltall nedenfor (hvis programmet krasjer, skriv bare Error).

Svar: [ ] .

Maks poeng: 2

1(e)

```
def motion_range_improve(R0):
    R = R0 + 10
    return R

first_throw = 20
second_throw = motion_range_improve(first_throw)
print(first_throw + second_throw)
```

Kristian øver på å modellere **prosjektilbevegelse**. For å teste det, kastet han en rugbyball i gata og noterte data. Etterpå dro han hjem og skrev dette programmet:

Hva skriver programmet ut? (Hvis programmet krasjer, skriv kun "Error").

Svar:

Maks poeng: 2

1(f) Marius setter opp et system for **filnavigasjon på PC-en** sin som en ekte proff. Han må lage en funksjon som tar filnavnet som argument, sjekker om filen slutter med `.txt`, `.png`, eller andre filendinger, og returnerer en tekstmelding om filtypen.

Hjelp Marius med å gjøre ferdig funksjonen ved å fylle inn manglende deler:

Fullfør funksjonen nedenfor

Test case #	Input	Forventet output
1	get_file_format("IMG3906.png")	This is an image
2	get_file_format("script.py")	Unknown or executable file
3	get_file_format("README.txt")	This is a text file

```
def get_file_format(filename):
    ...
# DO NOT EDIT THE LINE BELOW, OTHERWISE THE AUTOTEST WILL NOT WORK
print(eval(input()))
```

Test kode

Maks poeng: 6

1(g) Inspirert av Emily Dickinson (som er kjent for sine korte og rytmiske diktlinjer), jobber Maja med et verktøy som filtrerer **poesilinjer**. Funksjonen skal beholde kun de linjene som er kortere enn 32 tegn – perfekt for gamle tweet-lignende fragmenter.

Men deler av koden har blitt smusset til av blekk. Nå trenger Maja *å fylle inn manglende Python-syntaks for at programmet skal fungere som det skal*.

Match riktig alternativ med de tomme områdene i koden.

```
poem_lines = [
    "“Hope” is the thing with feathers",
    "That perches in the soul",
    "And sings the tune without the words",
    "And never stops - at all"
]

def select_lines(poem_lines):
    selected = []
          line       poem_lines:
              len(line) < 32:
            selected.append(line)
          selected

print(select_lines(poem_lines))
```

Alternatives:

if	return
for	in

Maks poeng: 2

1(h) Kunstgalleriet inneholder sitater fra **kjente malere**. I anledning jubileet til Frida Kahlo – en sentral skikkelse innen primitivisme og surrealisme – har Aurora fått i oppdrag å hente ut sitatene hennes, som er blandet med sitater av modernistiske Georgia O'Keeffe. Hun har skrevet et skript for å hente ut bestemte linjer fra en liste med sitater.

```
gallery_notes = '''
O'Keeffe: I had to create an equivalent for what I felt about what I was
looking at–not copy it.
Kahlo: I paint flowers so they will not die.
O'Keeffe: I'm not a joiner and I'm not a precisionist or anything else.
Kahlo: Fall in love with yourself, with life and then with whoever you want.
O'Keeffe: My painting is what I have to give back to the world for what the
world gives to me.
'''

quotes_by_line = gallery_notes._____()
kahlo_quotes = [quotes_by_line[1], quotes_by_line[3]]
print(kahlo_quotes)

# Output:
# ['Kahlo: I paint flowers so they will not die.',
# 'Kahlo: Fall in love with yourself, with life and then with whoever you
want.']
```

Skriv nedenfor hvilken metode som mangler for at Aurora skal få kjørt koden.

Svar: gallery_notes.[ ]()

Maks poeng: 2

1(i) Martha **strikker** en koselig strikkegenser som gave til søskenbarnet sitt. Hun vet at det trengs 50 000 sting, og nå hun velger mellom ulike typer garn - `Wool`, `Cotton`, og luksuriøs `Cashmere` - og vil vite hvor mange nøster (på engelsk kalt "clews") med garn hun trenger. Hver garntype gir forskjellig antall sting per nøste (7000, 9000 og 12000).

Hjelp Martha med å regne ut hvor mange nøster hun trenger til en hel genser. *Skriv en funksjon som tar garntypen som argument og returnerer antall nøster som trengs.*

Fullfør funksjonen nedenfor

Test case #	Input	Forventet output
1	is_enough_yarn("Wool")	8
2	is_enough_yarn("Cashmere")	5
3	is_enough_yarn("Cotton")	6

```
def is_enough_yarn(yarn_type):
    # Estimated total stitches needed for one sweater
    stitches_per_sweater = 50000

    # Estimated stitches per clew, based on yarn_type (from description)
    ...
        ...
    ...
        ...

    # Calculate number of clews needed
    clews_needed = ...
    return clews_needed
# DO NOT EDIT THE LINE BELOW, OTHERWISE THE AUTOTEST WILL NOT WORK
print(eval(input()))
```

Test kode

Maks poeng: 6

1(j) **Barbie** vurderer om hun skal dra på strandtur. Beslutningen hennes avhenger av flere ting: om Sasha er ledig, om Gloria blir med, om solnedgangen er fin, og om det finnes gode snacks. Hvis alt det slår feil, vurderer hun å dra med Ken – men bare hvis det ikke er sent på kvelden (fordi kvelder er for jentekvelder).

Dette er logikken bak valget hennes:

```
sasha or gloria and sunset <= snack or ken
```

Velg det uttrykket nedenfor som er logisk ekvivalent, og som gjenspeiler hvordan Python ville evaluert det:

Velg ett alternativ:

- (((sasha or gloria) and sunset) <= snack) or ken
- ((sasha) or (gloria)) and sunset <= (snack or ken
- sasha or (gloria and ((sunset <= snack) or ken))
- (sasha or (gloria and (sunset <= snack))) or ken
- sasha or (gloria and (sunset <= (snack or ken)))
- ((sasha or gloria) and (sunset <= snack)) or ken

Maks poeng: 2

1(k)

```
def spa_schedule_switch(spa_services):

    for service in spa_services:
        new_status = 0
        if spa_services[service] == 0:
            new_status = 1
        spa_services[service] = new_status

spa_masters = {
    'Manicurist': 2,
    'Waxing Specialist': 0,
    'Eyebrow Technician': 1,
    'Skincare Specialist': 1,
    'Massage Therapist': 0,
}

weekend_services = spa_schedule_switch(spa_masters)
```

Nora administrerer bookingsystemet ved sitt **spa-behandlingssenter**. I ukedagene er noen behandlere tilgjengelige og andre ikke. I helgene tilbyr spaet et annet sett med behandlinger – både for å gi personalet hvile og for å tilby kundene et bredere utvalg.
Her er koden hun bruker:

Velg det som vil bli printet etter at disse uttrykkene blir kjørt:

	2	0	- ERROR -	1	None
print(spa_masters['Manicurist'])	○	○	○	○	○
print(weekend_services)	○	○	○	○	○
print(spa_masters['Skincare Specialist'])	○	○	○	○	○
print(spa_masters['Massage Therapist'])	○	○	○	○	○
print(weekend_services['Manicurist'])	○	○	○	○	○
print(spa_masters['Manicurist']['Skincare Specialist'])	○	○	○	○	○

Maks poeng: 3

1(l)

```
pump_A = 5
pump_B = 2
pump_A = pump_A + pump_B
pump_B = pump_A * pump_B
pump_A -= 1
print(pump_B - pump_A)
```

Bensinstasjonen har to pumper: `pump_A` og `pump_B`. Noah har oppdaget at programmet som skal beregne hvor mye bensin som er igjen i pumpen, har begynt å oppføre seg merkelig og gjør fullstendig ulogiske operasjoner. Han bestemte seg for å teste det med `print`-setninger:

Hva skriver dette programmet ut? (Hvis programmet krasjer, skriv kun "Error"). Fyll inn svaret ditt

[ ] her.

Maks poeng: 2

1(m) **Verdens helseorganisasjon (WHO)** bruker et enkelt system for å merke personer som kan trenge videre oppfølging i en vaksinasjonskampanje. Hver person vurderes ut fra:

- Om personen nylig har vært i kontakt med en syk person (variabelen `contact`)
- Om personen har symptomer på sykdom (`symptoms`)
- Om personen er vaksinert (`vaccinated`)

En person blir merket for oppfølging basert på to kriterier:

1. En person som har hatt kontakt med syk og har symptomer.
2. En uvaksinert person som enten har vært i kontakt med syk, eller har symptomer.

Skriv begge disse betingelsene som logiske uttrykk i Python ved å bruke de tre variablene:

`flag_1 =` [ ]

`flag_2 =` [ ]

Maks poeng: 2

1(n)

```
brann_squad = {
    "forwards": [
        {"name": "Niklas Castro",
        "career": ["Vålerenga", "Kongsvinger", "Aalesunds", "Brann"],
        "physical": {
            "age": 29,
            "foot": "right",
            "height": 173
            }
        },
        {"name": "Aune Selland Heggebø",
         "career": ["Brann"],
         "physical": {
            "age": 23,
            "foot": "left",
            "height": 185
            }
        }
    ]
}
```

Olav bestemte seg for å lage en database over **fotballspillere i SK Brann** som del av sitt prosjekt i informatikk. Han valgte å lagre spillerne som oppslagsverk (dictionaries). Under ser du de to første observasjonene han la inn:

Nå ønsker han å teste om strukturen hans fungerer riktig.

Svar på spørsmålene nedenfor (hvis programmet krasjer på linjen, skriv "Error"):

Print Niklas Castros alder (integer):

- print(brann_squad[0][2][0])
- print(brann_squad["Niklas Castro"]["age"])
- print(brann_squad["Niklas Castro"]["physical"]["age"])
- print(brann_squad["forwards"]["Niklas Castro"]["age"])
- print(brann_squad["forwards"][0]["physical"][0])
- print(brann_squad["forwards"][0]["physical"]["age"])

Print Aune Heggebøs første klubb (string):

- print(brann_squad[1][0][0])
- print(brann_squad["forwards"][0]["career"][0])
- print(brann_squad["forwards"][1]["career"][0])
- print(brann_squad["forwards"][2]["career"][0])
- print(brann_squad["forwards"][2]["career"][1])
- print(brann_squad["forwards"]["Aune Selland Heggebø"]["career"][0])

Maks poeng: 2

1(o) Emil bestemte seg for å prøve seg i den spennende sporten **frisbeegolf**, men han er usikker på hvilke disker han bør velge til hver bane. Han fant en algoritme på internett, men noen funksjoner var utydelige. Din oppgave er *å fylle inn de manglende delene i funksjonen med riktige nøkkelord eller operatorer*, slik at Emil kan ta riktige valg:

Match riktig alternativ med de tomme områdene i koden.

```
    pick_disc(distance, wind, angle):

  if distance > 10^.
      if wind < 10       angle > 50:
          disc = "driver"
      else:
          disc = "midrange"

        distance > 50:
      disc = "midrange"

  else:
      disc = "putter"

        disc
```

Alternatives:

elif	def
return	and

Maks poeng: 2

1(p) Jacob spiller et fantasy **brettspill** med vennene sine. Spillet har en spennende ny mekanikk: hver spiller får en bonus på slutten av spillet basert på poengsummen i siste runde. For å regne ut bonusen, skrev han denne funksjonen:

```
def calculate_bonus(last_round_score):

    bonus = 0

    while last_round_score > 0:
        d = last_round_score % 10
        bonus += d
        last_round_score //= 10

    return bonus
```

Hva returnerer funksjonen hvis Jacob fikk 49 poeng i siste runde? (Hvis programmet krasjer, skriv kun "Error").

Svar:

Maks poeng: 2

1(q) Ingrid er senioranalytiker i et helsereguleringsfirma. Før en **vaksine distribueres** globalt, må hun forsikre seg om at to betingelser er oppfylt:

- Hovedlandet i forskningen må ha godkjent vaksinen.
- Ingen av de samarbeidende landene kan ha et kritisk utbrudd akkurat nå.

```
def is_vaccine_distributable(status_by_country):

    lead_approval = status_by_country[0]

    if lead_approval == 0:

        # no approval from lead research country
        return False

    for outbreak in status_by_country[1:]:

        if outbreak == 1:
            # active outbreak in a partner country
            print(status_by_country.index(outbreak), "fails the test")
            return False

    # all good :)
    return True

is_vaccine_distributable([1, 0, 0, 1, 0])
```

Hun har skrevet funksjonen `is_vaccine_distributable()` for å sjekke disse betingelsene.
Men dataene hun får er rotete. I input-listen:

- Første verdi viser godkjenningsstatus for hovedlandet
- De neste verdiene viser utbruddsstatus i samarbeidslandene

Hva returnerer funksjonen til Ingrid med inputen [1, 0, 0, 1, 0]?

Fyll inn svaret ditt her: [ ]

Maks poeng: 2

1(r)

```
games_2004 = 'half-life 2, doom 3, grand theft auto: san andreas, metal gear
solid 3: snake eater, world of warcraft, counter-strike: source, fable, halo
2'

big_games = games_2004._____()
print(big_games.split(', '))

# Output:
# ["HALF-LIFE 2", "DOOM 3", "GRAND THEFT AUTO: SAN ANDREAS",.. ]
```

Thomas er en lidenskapelig gamer. Han fant en liste over **videospill** som kom ut i 2004, og som i ettertid regnes som noen av de mest innflytelsesrike spillene gjennom tidene. Men navnene er skrevet med små bokstaver, noe som ikke ser særlig presentabelt ut. Thomas' oppgave er å skrive en metode som gjør alle spillnavn til store bokstaver.
Fyll inn den manglende delen i koden over.

Svar: games_2004.[ ] ()

Maks poeng: 2

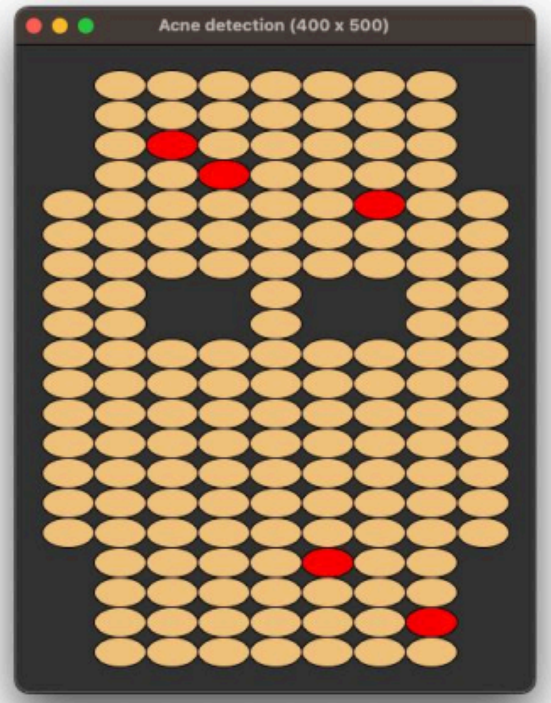
2(a)

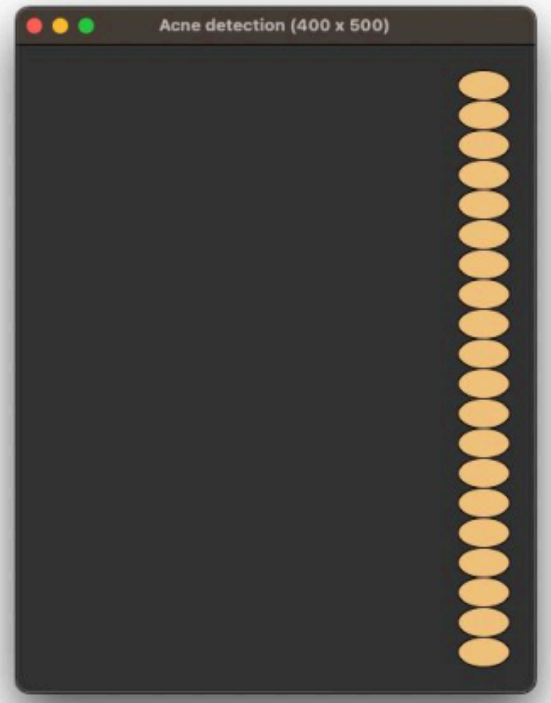
```
from uib_inf100_graphics.event_app import run_app

def app_started(app):
    app.rows = 20
    app.cols = 9
    app.margin = 20
    # acne locations (col, row)
    app.acne_data = [(2, 2), (3, 3), (6, 4), (5, 16), (7, 18)]

def get_cell_bounds(app, row, col):
    grid_width = app.width - 2 * app.margin
    grid_height = app.height - 2 * app.margin
    cell_width = grid_width / app.cols
    cell_height = grid_height / app.rows
    x0 = app.margin + col * cell_width
    y0 = app.margin + row * cell_height
    x1 = x0 + cell_width
    y1 = y0 + cell_height
    return (x0, y0, x1, y1)

def is_invalid_location(app, row, col):
    is_along_side = col == 0 or col == app.cols - 1
    is_near_top = row < 4
    is_near_bottom = row > app.rows - 5
    is_in_left_eye = col in [2, 3] and row in [7, 8]
    is_in_right_eye = col in [5, 6] and row in [7, 8]
    return is_along_side and (is_near_top or is_near_bottom) \
        or is_in_left_eye \
        or is_in_right_eye

def redraw_all(app, canvas):
    for row in range(app.rows):
        for col in range(app.cols):
            if is_invalid_location(app, row, col):
                continue

        (x0, y0, x1, y1) = get_cell_bounds(app, row, col)
        color = "red" if (col, row) in app.acne_data else "#f1c27d"
        canvas.create_oval(x0, y0, x1, y1, fill=color, outline="black")

run_app(width=400, height=500, title="Acne detection")
```

Sofie jobber med en **hudpleie**-startup-app som bruker en enkel ansiktsskanner for å visualisere **akne**. Visualiseringen bygges opp som et rutenett av “porer” (sirkler), hvor akne vises med rødfarge. Appen bruker `uib_inf100_graphics graphics`-biblioteket for grafikk.

Dessverre har Sofie gjort en feil i koden, som gjør at visualiseringen kun tegner den siste kolonnen av ansiktet. Du ser ønsket resultat til venstre og det faktiske resultatet til høyre. Koden vises under bildene.

Din oppgave er å hjelpe Sofie med å løse feilen: Skriv et detaljert svar hvor du forklarer hvor i koden problemet oppstod, hvorfor det skjer, og hvordan hun kan unngå slike feil i fremtidig utvikling.





Skriv ditt svar her

Format

Words: 0

Maks poeng: 9

2(b)

```
while stamina < exhaustion:
    still_fighting = TRUE
    print("The crowd goes wild! 💪")
```

Jax bygger en **wrestling**-liga-simulator i Python. Hver kamp har et «utmattelsesnivå» – når en bryter når dette nivået, taper de kampen. For å simulere dette bruker Jax en løkke som hele tiden sjekker bryterens utholdenhet, og legger til tretthet over tid. Du kan se et forenklet eksempel på løkken over.

Jax forventet at denne løkken til slutt skulle stoppe, men det gjør den ikke. I stedet fortsetter kampen for alltid, og til slutt krasjer Python. *Hva kan ha gått galt i simuleringen? Hva bør Jax huske på når han lager denne typen logikk?*

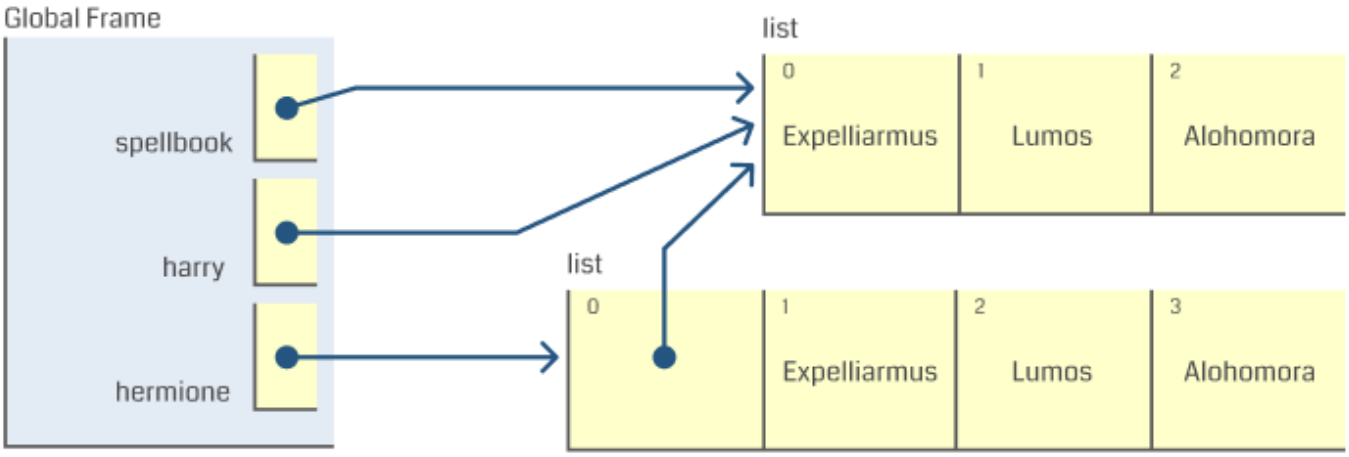
Skriv ditt svar her

Format

Words: 0

Maks poeng: 7

3(a) **Hermione** og **Harry Potter** lærer seg nye trylleformler fra en gammel tryllebok: **`Expelliarmus`**, **`Lumos`**, og **`Alohomora`**. Som en del av læringen skriver de ned de lærte formlene i sine egne bøker. Strukturen på bøkene og formlene vises i illustrasjonen under.



Skriv en kodebit som korrekt definerer disse variablene og minnetilstanden. Du får ikke trekk dersom du velger å definere flere variabler enn nødvendig.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 6

3(b)

```
# example data
norge_champions = ['Bodø/Glimt', 'Bodø/Glimt', 'Molde', 'Bodø/Glimt',
'Bodø/Glimt', 'Molde', 'Rosenborg', 'Rosenborg', 'Rosenborg', 'Rosenborg',
'Molde', 'Strømsgodset', 'Molde', 'Molde', 'Rosenborg', 'Rosenborg',
'Stabæk', 'Brann', 'Rosenborg', 'Vålerenga', 'Rosenborg', 'Rosenborg',
'Rosenborg', 'Rosenborg']

# example calls:
champions_dict = count_champions(norge_champions) # {'Bodø/Glimt': 4, 'Molde':
5, 'Rosenborg': 11...
ranked_champions_dict = sort_champions(champions_dict) # {'Rosenborg': 11,
'Molde': 5, 'Bodø/Glimt': 4...
```

Sindre ønsker å analysere **norsk fotball eliteseriehistorie** som en ekte sportsanalytiker. Han har samlet en liste over klubbene som har vunnet Eliteserien i det 21. århundre. Nå vil han lage to funksjoner som gir ham innsikt:

1. Hvor mange ganger hver klubb har vunnet: Funksjonen tar inn en liste med Eliteserie-vinnere og returnerer en ordbok (dictionary) med antall titler for hver klubb.
2. En rangering av klubbene etter antall seiere, fra flest til færrest: Funksjonen tar ordboken fra første funksjon og returnerer en ny ordbok sortert etter verdiene i synkende rekkefølge.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 12