**Informasjon**

| Oppgave | Poeng | Oppgavetype |
|---|---|---|
| ℹ | | Informasjon eller ressurser |
| ℹ | | Informasjon eller ressurser |
| ℹ | | Informasjon eller ressurser |

**Automatisk rettet**

| Oppgave | Poeng | Oppgavetype |
|---|---|---|
| 1(a) | 9 | Fyll inn tekst |
| 1(b) | 3 | Fyll inn tekst |
| 1(c) | 6 | Fyll inn tekst |
| 1(d) | 3 | Fyll inn tekst |
| 1(e) | 5 | Fyll inn tekst |
| 1(f) | 2 | Flervalg |
| 1(g) | 10 | Kompilering |

**Forklaringer**

| Oppgave | Poeng | Oppgavetype |
|---|---|---|
| 2 | 12 | Langsvar |

**Koding**

| Oppgave | Poeng | Oppgavetype |
|---|---|---|
| 3(a) | 3 | Programmering |
| 3(b) | 3 | Programmering |
| 3(c) | 10 | Programmering |
| 3(d) | 14 | Programmering |

**Poeng fra semesteret**

| Oppgave | Poeng | Oppgavetype |
|---|---|---|
| 4 | 20 | Muntlig |

# 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 erklæring.**

# Informasjon

Eksamen består av tre deler, og gir tilsammen 80 poeng:

1. Automatisk rettede oppgaver (**38** poeng)
2. Forklaringsoppgaver (**12** poeng)
3. Kodeoppgaver (**30** poeng)

Du bør ha begynt del 3 når du har 2 timer igjen. I del 3 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 fra standardbiblioteket. Du må likevel skrive koden så tydelig og korrekt som mulig, slik at du demonstrerer at du forstår nyanser i koden du skriver.

*Når du skriver kode, la den være så generell/dynamisk som mulig; unngå hard-kodete verdier så langt du klarer, og tenk at programmet skal fungere også på andre input enn dem som er oppgitt i eksempelet.*

Tillatte hjelpemidler på eksamen: bøker fra litteraturlisten og opp til 6 tosidige A4 ark med egne notater.

### 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. Oppgavene er sortert etter type, ikke etter vanskelighetsgrad.
- 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 hvordan noe skal gjøres med kode, skriv en kommentar som forklarer hva du prøver på så presist som mulig.

# Kursnotater

På denne siden finner du alle kursnotatene i emnet samlet i en PDF. De er tilgjengelig for deg dersom du ønsker. 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 svaret ditt.

Du kan også finne kursnotatene under «ressurser» nede i venstre hjørne gjennom hele eksamen. Da vil notatene åpnes i en ny fane.

PS: Har sidene plutselig snudd seg sidelengs? Prøv å trykk på 'R' for å snu riktig vei igjen.

**1(a)**

```
x = 2
y = {
  x: 42,
  'y': x,
  1: 3
}
y[42] = 'x'
z = [-1, x, y, 'z']
```

Anta at kodesnutten over har blitt kjørt, og at en av setningene under er den neste setningen som utføres. Hva skrives ut? Hvis programmet krasjer, skriv kun Error.

(Husk at apostrofer og hermetegn som omgir strenger i kildekoden *ikke* er blir inkludert i utskriften.)

| | |
|---|---|
| print(**x**) | |
| print(**'x'**) | |
| print(**y[1]**) | |
| print(**y[2]**) | |
| print(**y['x']**) | |
| print(**z[0]**) | |
| print(**z[1]**) | |
| print(**z[x]['y']**) | |
| print(**y[z[x][x]]**) | |

---

Maks poeng: 9

**1(b)**

```
x = 3
y = 2 * x
x = 4
print(x + y)
```

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

---

Maks poeng: 3

**1(c)**

```
def foo(x):
    x = x + x
    return x
def bar(x):
    x = foo(x)
    return x + foo(x)
```

Anta at funksjonene over er definert, og at neste linje med kode er en av setningene under. Hva skriver programmet i så fall ut? (hvis programmet krasjer, skriv kun Error)

| | |
|---|---|
| print(foo(3)) | |
| print(bar(3)) | |
| print(foo(foo('foo'))) | |

Maks poeng: 6

**1(d)**

```
line = 'Per;Ida;Kari;Are;Christine;Hans;Une'
p = line.split(';')
r = []
for x in p:
    r.append(x[0])
s = ''.join(r)
print(s)
```

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

Maks poeng: 3

**1(e)**

```
def foo(x, y):
    if x > y:
        x -= 10
    if y < 10:
        if y < 5:
            return 0
        if x > 5:
            y = 1000
        elif y > 2:
            x += y
        x += 1
    return x + y
```

Anta at funksjonen over har blitt definert, og at neste linje med kode er en av setningene under. Hva skriver programmet i så fall ut? (hvis programmet krasjer, skriv kun Error)

| | |
|---|---|
| print(foo(2, 1)) | |
| print(foo(1, 6)) | |
| print(foo(11, 6)) | |
| print(foo(20, 6)) | |
| print(foo(20, 30)) | |

Maks poeng: 5

**1(f)** Hvilket av uttrykkene under er nøyaktig det samme som

```
a or b in c and d
```

**Velg ett alternativ:**

- ((a or b) in c) and d
- (a or (b in c)) and d
- (a or b) in (c and d)
- a or ((b in c) and d)
- a or (b in (c and d))

---

Maks poeng: 2

**1(g)** Skriv en funksjon *find_shortest* med en parameter *li*. Anta at *li* er en liste med strenger. La funksjonen returnere den korteste strengen i listen, eller *None* hvis listen er tom. Hvis det er mer enn én korteste streng, skal du returnere den av dem som kommer først i listen.

*I denne oppgaven kan du prøve koden du skriver mot noen tester underveis i eksamen. For å få full uttelling på oppgaven må alle testene (inkludert hemmelige tester) passere. Testene kjøres når du trykker på knappen «test kode» under kodefeltet.*

*Hvis du ikke passerer alle testene kan likevel sensor gi delvis uttelling etter en manuell vurdering.*

**Skriv svaret ditt her. Endringer blir lagret automatisk.**

| Test case # | Input | Forventet output |
|---|---|---|
| 1 | find_shortest(['x', 'yy', 'zzz']) | x |
| 2 | find_shortest(['xxx', 'yy', 'zzz', 'æææ']) | yy |
| 3 | find_shortest([]) | None |
| 4 | find_shortest(['ccc', 'bb', 'aa']) | bb |
| 5 | find_shortest(['abcdefg']) | abcdefg |
| 6 | find_shortest(['foo', '']) | |

```
# Skriv din kode her
def find_shortest(li):
    ...

# IKKE ENDRE PÅ LINJEN UNDER, ELLERS VIL IKKE TESTENE VIRKE
print(eval(input()))
```

Test kode

Maks poeng: 10

## 2 Bakgrunn

*(ikke egentlig viktig for denne oppgaven)*

I forelesning 18. oktober dette semesteret, skrev vi en bitteliten språkmodell man kunne trene til å finne på nye ord som lignet på ulike språk avhengig av hvilken treningsdata vi benyttet.

Språkmodellen vår bestod av to algoritmer:

- en treningsalgoritme som leste treningsdata og teller hvor mange ganger en bokstav opptrår i ulike kontekster. Dette lagres som vekter i en json-fil.
- en inferens-algoritme som finner på nye ord basert på en kombinasjon av vektene i json-filen og tilfeldighet.

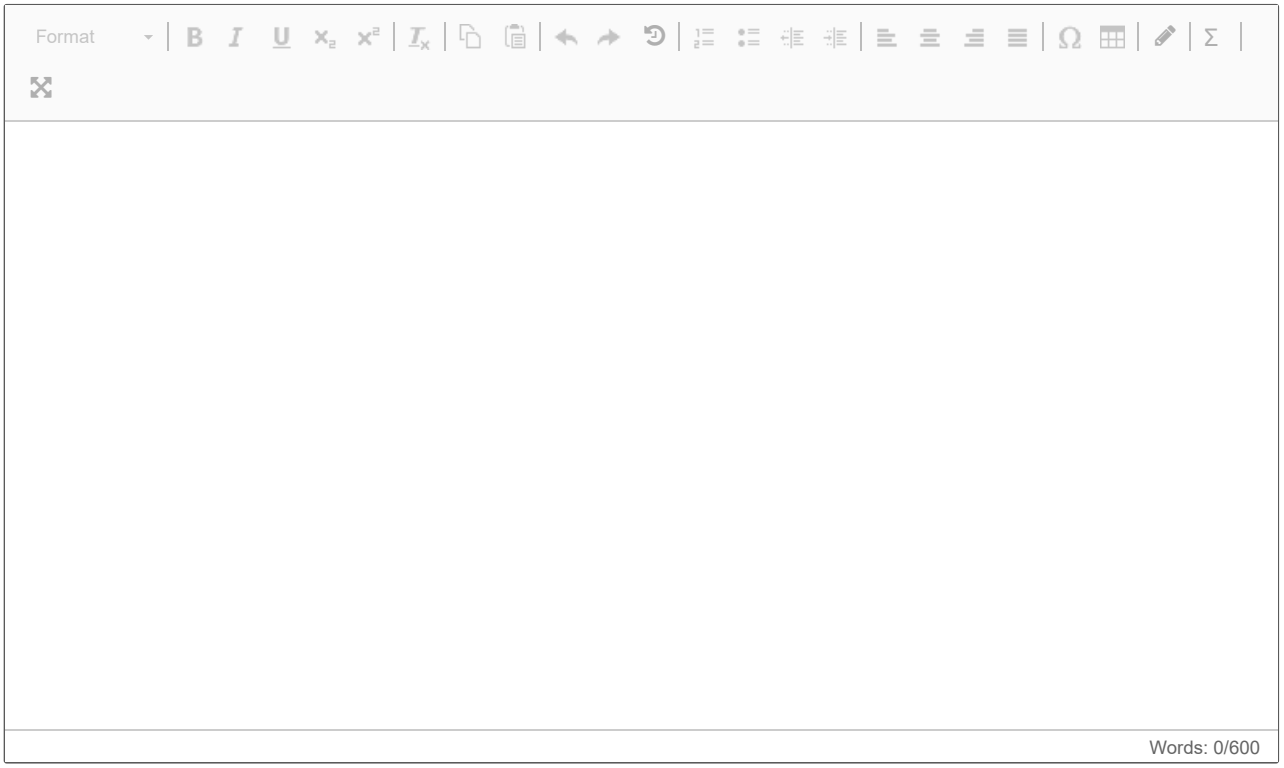
I vedlagte PDF ser du en versjon av inferens-algoritmen, som er litt forbedret i forhold til den vi så i forelesning. Da vi i forelesning kun anså den relevante delen av konteksten til å være ett enkelt symbol, tar denne varianten av koden også hensyn til større lengder på den relevante konteksten; først prøver den en relevant kontekst på størrelse 3, og deretter forsøker den gradvis kortere og kortere relevante kontekster helt til den finner en kontekst som er kort nok til at det finnes noe å velge mellom i vektene.

> *PS: husk at en viktig egenskap for en god programmerer er evnen til å overse det uvesentlige.*

**Oppgavetekst**

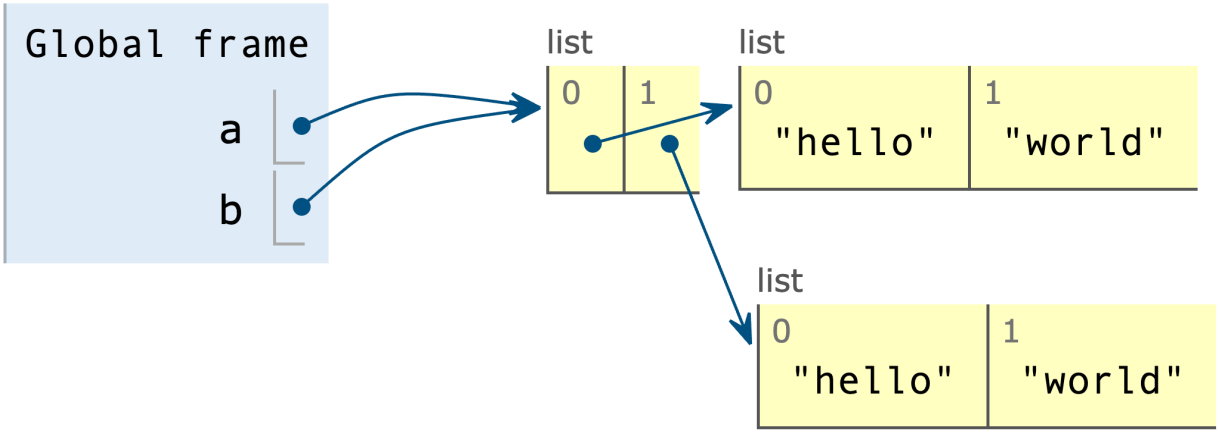
Betingelsen i while-løkken er **context_length > 0**. Hva ville skjedd om vi snudde ulikheten slik at den ble **context_length < 0** i stedet? Forklar så detaljert som mulig hva som ville skjedd. Hva slags feil ville oppstått? Hvilken klasse av feil (syntaks, kjøretid, logisk) tilhører feilen? Kan du si noe om hva som skiller denne typen feil fra andre typer feil?

**Skriv svaret ditt her. Endringer blir lagret automatisk.**




Maks poeng: 12

**3(a)**



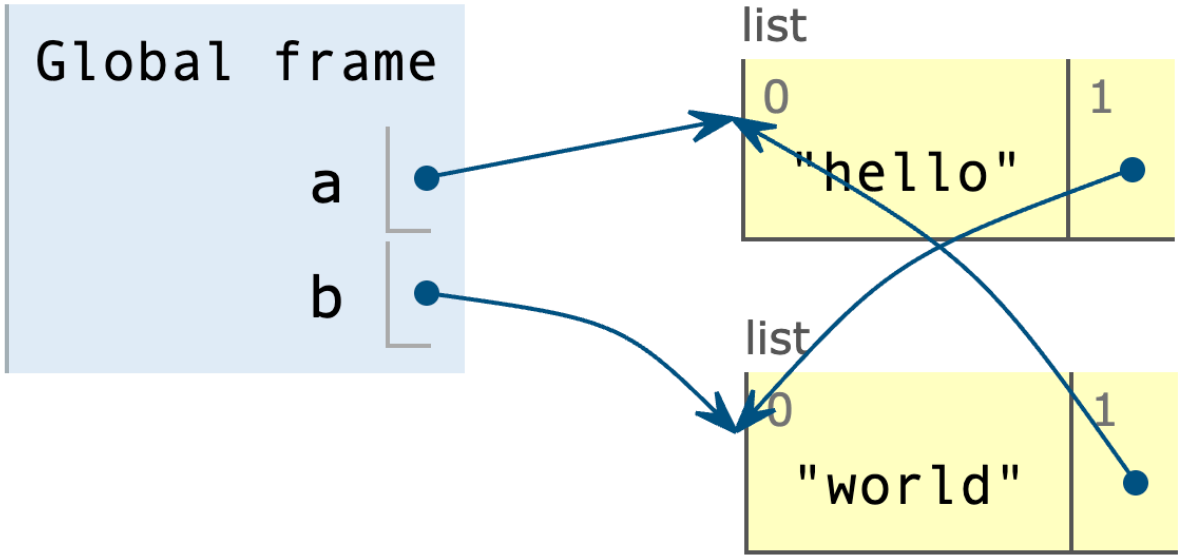

Skriv en kodesnutt slik at minnets tilstand blir som vist over.

**Skriv svaret ditt her. Endringer blir lagret automatisk.**

```
1
```

Maks poeng: 3

**3(b)**




Skriv en kodesnutt slik at minnets tilstand blir som vist over. Hint: du er nødt til å mutere minst en av listene etter at den blir opprettet første gang.

**Skriv svaret ditt her. Endringer blir lagret automatisk.**

```
1
```

Maks poeng: 3

**3(c) Bakgrunn**

På iskremfabrikken er det en iskremmaskin. Maskinen har mulighet for å angi hvilken smak iskremen som blir produsert skal ha. Maskinen skriver ned alt som skjer i en logg-fil.

- Logg-filen er en JSON-fil, og objektet filen beskriver er et oppslagsverk med en nøkkel "log" som viser til en liste med hendelser.
- En hendelse er representert som et oppslagsverk, og har alltid nøklene "timestamp" og "event_type".
- Dersom "event_type" er strengen "set_flavour", finnes det også en nøkkel "value" i samme hendelse hvor den tilhørende verdien er en streng som beskriver smaken maskinen fra nå av er innstilt på.

Maskinen vil aldri lage en iskrem uten at det er valgt en smak først. Du kan anta at tidspunktene for hendelsene i loggen alltid er i stigende rekkefølge. Se et eksempel på en slik logg-fil *machine_log.json* i PDF-vedlegget til oppgaven.

**Oppgavetekst**

Skriv en funksjon *production_summary* med en parameter *path*. Anta at *path* er et filnavn som peker på en logg-fil fra iskremmaskinen.

La funksjonen returnere et oppslagsverk der de ulike smakene som finnes i loggen er nøkler, mens tilhørende verdier er antallet iskrem av den gitte typen som ble produsert. Det er kun hendelser av typen "ice_cream_created" som skal bli talt opp for hver av de ulike smakene.

For eksempel, hvis *log* viser til filen i eksempelet skal funksjonen returnere et oppslagsverk

```
{
    "vanilje": 3,
    "sjokolade": 1
}
```

**Skriv svaret ditt her. Endringer blir lagret automatisk.**

```
1
```

Maks poeng: 10

**3(d) Bakgrunn**

Varetelling ved iskremfabrikken finner som vanlig sted 24. november, så akkurat denne dagen foregår det ingenting annet ved fabrikken. Og nå som vi endelig er ferdige med å telle, har tiden kommet for å ta en aldri så liten fot i bakken for å sjekke om tallene stemmer overens. Vi skal samstemme informasjon fra flere kilder:

- ***machine_log.json*** er en fil som produseres av iskremmaskinen. I denne filen finner vi informasjon om alle hendelser som skjer i iskremmaskinen mellom varetelling i fjor og i år. Logg-filen inkluderer en hendelse for hver gang en boks med iskrem kommer seilende ut på enden av samlebåndet. Denne filen inneholder altså informasjon om hva som kommer *inn* på iskrem-lageret.
- ***packing_orders.json*** er en fil som inneholder oversikt over alle pakke-ordrer. Det er én pakke-ordre for hver lastebil som *forlater* lageret, og hver pakke-ordre inneholder informasjon om hvor mange bokser iskrem av hver type som blir med i lastebilen.
- ***inventory_2024.csv*** og ***inventory_2025.csv*** er semikolon-separerte CSV-filer som inneholder antall iskrem-bokser av hver type for varetellingene i henholdsvis i fjor og i år.

Du kan se eksempler på hvordan filene ser ut i vedlagte PDF. Gjør ditt beste for å forstå strukturen i hvordan filene er bygget opp.

**Oppgavetekst**

Skriv et program som regner ut (og skriver ut) hvor mange iskrem av hver type som på mystisk vis har forsvunnet i løpet av året.

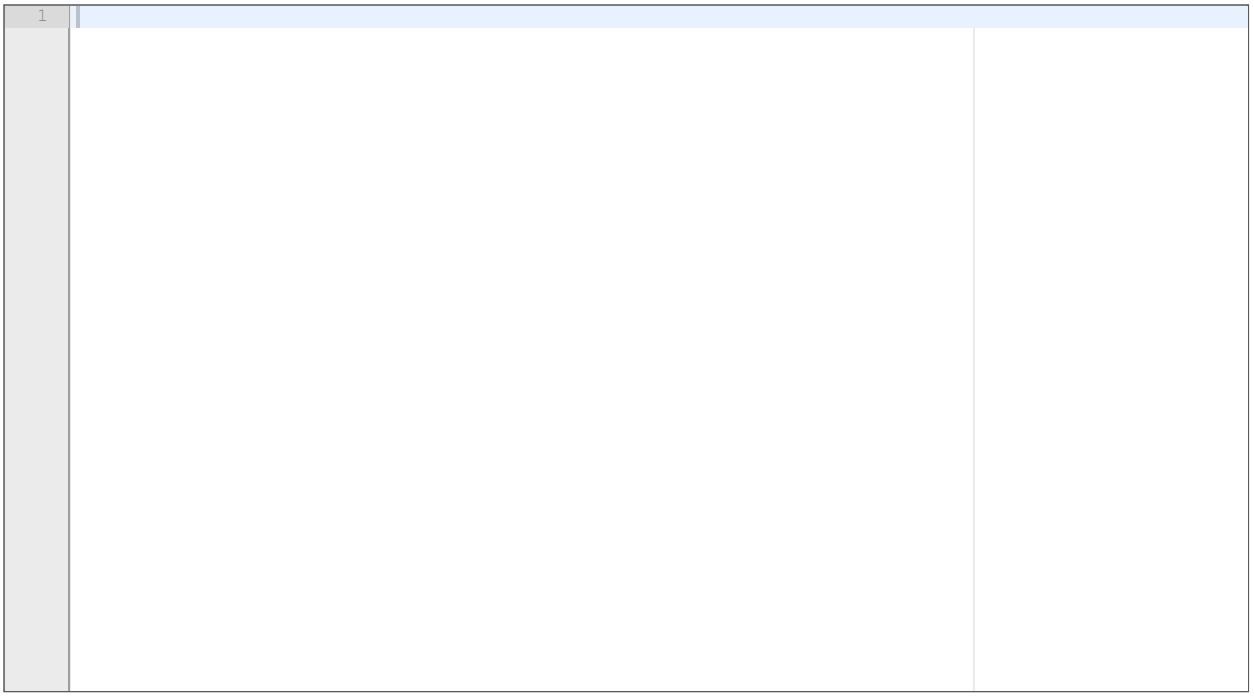
For eksempel, hvis filene er som i det viste eksempelet, skal programmet skrive ut

```
Forsvunnede iskrem
vanilje: 3
pistasj: 5
```

*Forklaring på hvorfor det mangler tre vanilje-is: i fjor hadde vi 10. I løpet av året ble det produsert 3, så til sammen 13. Det ble sendt ut 8+1=9 is til kunder. Da er det forventet at det er 13-9=4 vanilje-is på lageret; men vi fant jo bare 1 i årets varetelling. Så da må 3 ha forsvunnet.*

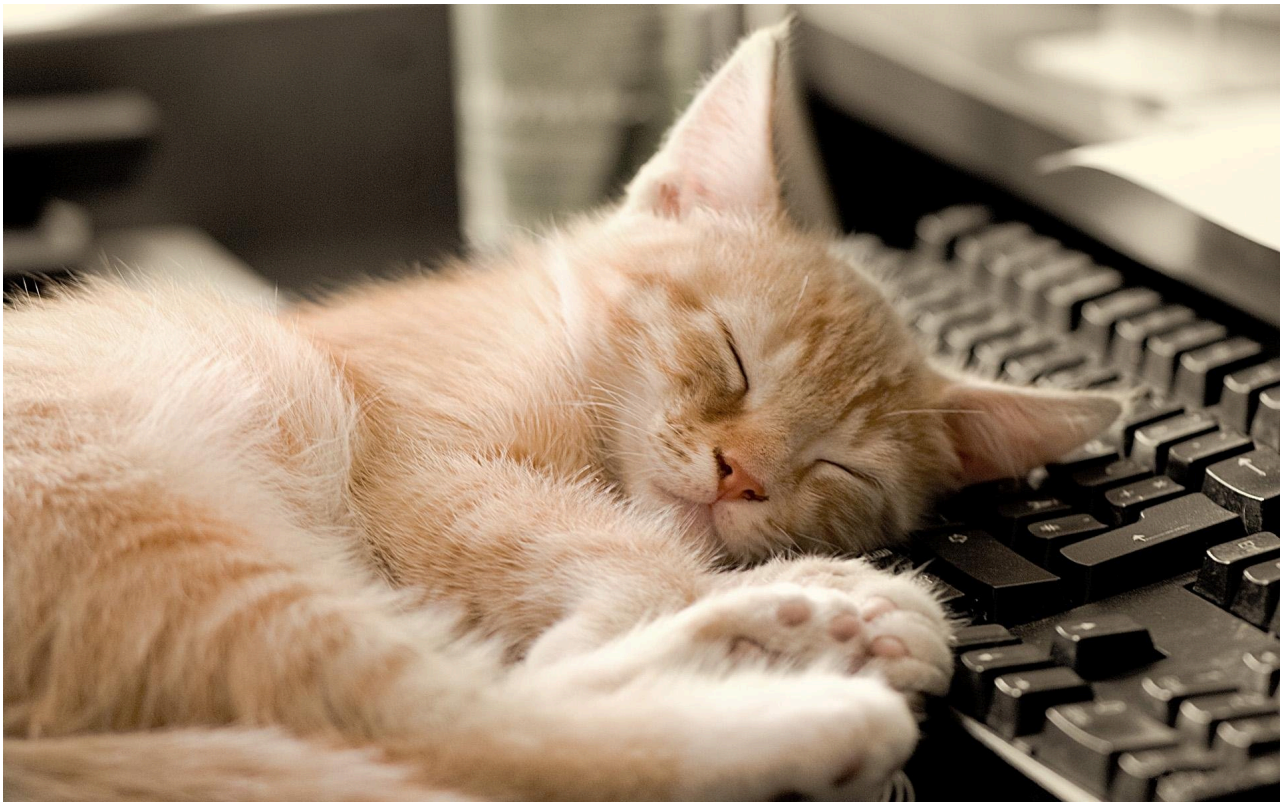
Du kan benytte en antatt fungerende versjon av funksjonen du skrev i forrige oppgave fritt uten å skrive den på nytt her.

**Skriv svaret ditt her. Endringer blir lagret automatisk.**




Maks poeng: 14

**4** I denne oppgaven blir det lagt inn poeng fra arbeidet ditt med laber du har gjort gjennom semesteret.





Lykke til med resten av eksamenssesongen, og tusen takk for et fint semester!

---

Maks poeng: 20

**Question 11**
Attached

*machine_log.json*


```
{
  "log": [
    {
      "timestamp": "2024-11-25T08:00:00",
      "event_type": "machine_started"
    },
    {
      "timestamp": "2024-11-25T08:00:01",
      "event_type": "set_flavour",
      "value": "vanilje"
    },
    { "timestamp": "2024-11-25T08:00:02",
      "event_type": "ice_cream_created"
    },
    {
      "timestamp": "2024-11-25T08:00:03",
      "event_type": "ice_cream_created"
    },
    {
      "timestamp": "2024-11-25T09:00:04",
      "event_type": "set_flavour",
      "value": "sjokolade"
    },
    {
      "timestamp": "2024-11-25T09:00:05",
      "event_type": "ice_cream_created"
    },
    {
      "timestamp": "2024-11-25T10:00:06",
      "event_type": "set_flavour",
      "value": "vanilje"
    },
    {
      "timestamp": "2024-11-25T10:00:07",
      "event_type": "ice_cream_created"
    }
  ]
}
```

**Question 12**
Attached

*machine_log.json*

```
{
  "log": [
    {
      "timestamp": "2024-11-25T08:00:00",
      "event_type": "machine_started"
    },
    {
      "timestamp": "2024-11-25T08:00:01",
      "event_type": "set_flavour",
      "value": "vanilje"
    },
    { "timestamp": "2024-11-25T08:00:02",
      "event_type": "ice_cream_created"
    },
    {
      "timestamp": "2024-11-25T08:00:03",
      "event_type": "ice_cream_created"
    },
    {
      "timestamp": "2024-11-25T09:00:04",
      "event_type": "set_flavour",
      "value": "sjokolade"
    },
    {
      "timestamp": "2024-11-25T09:00:05",
      "event_type": "ice_cream_created"
    },
    {
      "timestamp": "2024-11-25T10:00:06",
      "event_type": "set_flavour",
      "value": "vanilje"
    },
    {
      "timestamp": "2024-11-25T10:00:07",
      "event_type": "ice_cream_created"
    }
  ]
}
```

*packing_orders.json*

```
{
  "orders": [
    {
      "date": "2024-12-01",
      "recepient": {
        "name": "Bellevuebakken",
        "street": "Nystuveien 11",
        "zip": 5019,
        "town": "Bergen"
      },
      "items": [
        { "flavour": "vanilje", "quantity": 8 },
        { "flavour": "sjokolade", "quantity": 10 },
        { "flavour": "pistasj", "quantity": 10 }
      ]
    },
    {
      "date": "2025-01-14",
      "recepient": {
        "name": "Colonialen",
        "street": "Strandgaten 18",
        "zip": 5013,
        "town": "Bergen"
      },
      "items": [
        { "flavour": "sjokolade", "quantity": 1 },
        { "flavour": "vanilje", "quantity": 1 }
      ]
    }
  ]
}
```

*inventory_2024.csv*

```
name;count
vanilje;10
sjokolade;100
pistasj;20
```

*inventory_2025.csv*

```
name;count
vanilje;1
sjokolade;90
pistasj;5
```

**Question 8**
Attached

```
import json
import random
from pathlib import Path


def main():
    weights = json.loads(Path('weights.json').read_text(encoding='utf-8'))

    res = ''
    for _ in range(12):
        res += get_token(res, weights)
    print(res)


def get_token(all_context, weights):
    context_length = 3
    while context_length > 0:
        relevant_context = all_context[-context_length:]
        if relevant_context in weights:
            vector = weights[relevant_context]
            if len(vector) > 2:
                break
        context_length -= 1

    letters = list(vector.keys())
    rel_weights = list(vector.values())

    random_letter = random.choices(letters, weights=rel_weights, k=1)[0]
    return random_letter


if __name__ == '__main__':
    main()
```