**Informasjon**

| Oppgåve | Poeng | Oppgåvetype |
|---|---|---|
| i | | Informasjon eller ressurser |
| i | | Informasjon eller ressurser |
| i | | Informasjon eller ressurser |

**Automatisk rettet**

| Oppgåve | Poeng | Oppgåvetype |
|---|---|---|
| 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 | Fleirval |
| 1(g) | 10 | Kompilering |

**Forklaringer**

| Oppgåve | Poeng | Oppgåvetype |
|---|---|---|
| 2 | 12 | Langsvar |

**Koding**

| Oppgåve | Poeng | Oppgåvetype |
|---|---|---|
| 3(a) | 3 | Programmering |
| 3(b) | 3 | Programmering |
| 3(c) | 10 | Programmering |
| 3(d) | 14 | Programmering |

**Poeng fra semesteret**

| Oppgåve | Poeng | Oppgåvetype |
|---|---|---|
| 4 | 20 | Munnleg |

# Egenerklæring

Eg erklærer med dette at svaret eg leverer er mitt eige arbeid.
Eg har ikkje:

- samarbeidd med andre studentar
- brukt andre sitt arbeid utan at dette er oppgitt
- brukt eige tidlegare arbeid (innleveringar/eksamenssvar) utan at dette er oppgitt

Dersom eg har brukt litteratur, vil ei litteraturliste innehalde alle kjelder eg har brukt i svaret, og referansar vil vise til denne lista.

**Eg er kjend med at brot på desse reglane blir rekna som fusk og kan føre til annullert eksamen og/eller utestenging.**

Dersom du er usikker på om du kan stille deg bak erklæringa, sjå retningslinjer for bruk av kjelder i skriftlege arbeid ved Universitetet i Bergen, og ta eventuelt kontakt med studierettleiar/emneansvarleg.

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

**Merk: Ved å halde fram stadfestar eg at eg har lese erklæringa, og at svaret eg leverer under denne eksamenen er mitt eige arbeid (og berre mitt eige arbeid), i full samsvar med ovannemnde erklæring.**

# Informasjon

Eksamen består av tre delar, og gir til saman 80 poeng:

1. Automatisk retta oppgåver (**38** poeng)
2. Forklaringsoppgåver (**12** poeng)
3. Kodeoppgåver (**30** poeng)

Du bør ha byrja på del 3 når du har 2 timar att. I del 3 vil du ikkje ha høve til å køyre koden du skriv. Sensor er klar over dette, og vil ikkje trekkje poeng for småfeil som til dømes skrivefeil i funksjonsnamn frå standardbiblioteket. Du må likevel skrive koden så tydeleg og korrekt som mogleg, slik at du viser at du forstår nyansane i koden du skriv.

*Når du skriv kode, la han vere så generell/dynamisk som mogleg; unngå hardkoda verdiar så langt du klarer, og tenk at programmet skal fungere òg på annan input enn dei som er oppgitt i eksempelet.*

Tillatne hjelpemiddel på eksamen: bøker frå pensumlista og opp til 6 tosidige A4-ark med eigne notat.

### Generelle råd

- Les spørsmålet før du svarar.
- Arbeid deg *raskt* gjennom alle spørsmåla i første omgang, og kom heller tilbake til krevjande oppgåver på slutten om du får tid. Oppgåvene er sorterte etter type, ikkje etter vanskegrad.
- Hjelp sensor å hjelpe deg! Om du er usikker på tolkninga av ei oppgåve, gje ein kort kommentar om korleis du tolkar usikkerheiter i oppgåva. Om du ikkje hugsar korleis noko skal gjerast med kode, skriv ein kommentar som forklarar kva du prøver å gjere så presist som mogleg.

# Kursnotat

På denne sida finn du alle kursnotata i emnet samla i ein PDF. Dei er tilgjengelege for deg dersom du ønskjer det. Det er heilt frivillig å bruke dei. Vanlege reglar for sitering gjeld: dersom du kopierer noko frå kursnotata, er det viktig at du oppgir dei som kjelde i svaret ditt.

Du kan òg finne kursnotata under «ressursar» nede i venstre hjørne gjennom heile eksamen. Då vil notata opnast i ein ny fane.

PS: Har sidene plutseleg snudd seg sidelengs? Prøv å trykke på 'R' for å få dei rett veg 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 køyrd, og at ei av setningane under er den neste setninga som blir utført. Kva blir skrive ut? Dersom programmet krasjar, skriv berre Error.

(Hugs at apostrofar og hermeteikn som omgir strengar i kjeldekoden ikkje blir tekne med i utskrifta.)

| | |
|---|---|
| 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)
```

Kva skriv dette programmet ut? (Dersom programmet krasjar, skriv berre 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 funksjonane ovanfor er definerte, og at neste linje med kode er ei av setningane under. Kva skriv programmet då ut? (Dersom programmet krasjar, skriv berre 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)
```

Kva skriv dette programmet ut? (Dersom programmet krasjar, skriv berre 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 ovanfor har blitt definert, og at neste linje med kode er ei av setningane under. Kva skriv programmet då ut? (Dersom programmet krasjar, skriv berre 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)** Kva for eitt av uttrykka under er nøyaktig det same som

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

**Vel eitt 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 ein funksjon *find_shortest* med ein parameter *li*. Anta at *li* er ei liste med strengar. La funksjonen returnere den kortaste strengen i lista, eller *None* dersom lista er tom. Dersom det er fleire kortaste strengar, skal du returnere den av dei som kjem først i lista.

*I denne oppgåva kan du prøve koden du skriv mot nokre testar undervegs i eksamen. For å få full utteljing på oppgåva må alle testane (inkludert hemmelege testar) passere. Testane blir køyrde når du trykkjer på knappen «test kode» under kodefeltet.*

*Dersom du ikkje passerer alle testane, kan likevel sensor gi delvis utteljing etter ei manuell vurdering.*

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

| Test case # | Input | Forventa 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

*(ikkje eigentleg viktig for denne oppgåva)*

I førelesing 18. oktober dette semesteret, skreiv vi ein bitteliten språkmodell som ein kunne trene opp til å finne på nye ord som liknar på ulike språk, avhengig av kva treningsdata vi brukte.

Språkmodellen vår bestod av to algoritmar:

- Ein treningsalgoritme som las treningsdata og tel kor mange gongar ein bokstav opptrer i ulike kontekstar. Dette blir lagra som vekter i ei json-fil.
- Ein inferens-algoritme som finn på nye ord basert på ein kombinasjon av vektene i json-fila og tilfeldighet.

I vedlagte PDF ser du ein versjon av inferens-algoritmen, som er litt forbetra i forhold til den vi såg i førelesinga. Då vi i førelesinga berre såg på den relevante delen av konteksten som eitt enkelt symbol, tek denne varianten av koden også omsyn til større lengder på den relevante konteksten; først prøver den ein relevant kontekst på storleik 3, og deretter forsøker den gradvis kortare og kortare relevante kontekstar heilt til den finn ein kontekst som er kort nok til at det finst noko å velje mellom i vektene.

*PS: Hugs at ein viktig eigenskap for ein god programmerar er evna til å oversjå det uviktige.*

**Oppgåvetekst**

Betinginga i while-løkka er **context_length > 0**. Kva ville skjedd om vi snudde ulikskapen slik at det vart **context_length < 0** i staden? Forklar så detaljert som mogleg kva som ville skjedd. Kva slags feil ville oppstått? Kva klasse av feil (syntaks, køyringstid, logisk) høyrer feilen til? Kan du seie noko om kva som skil denne typen feil frå andre typer feil?

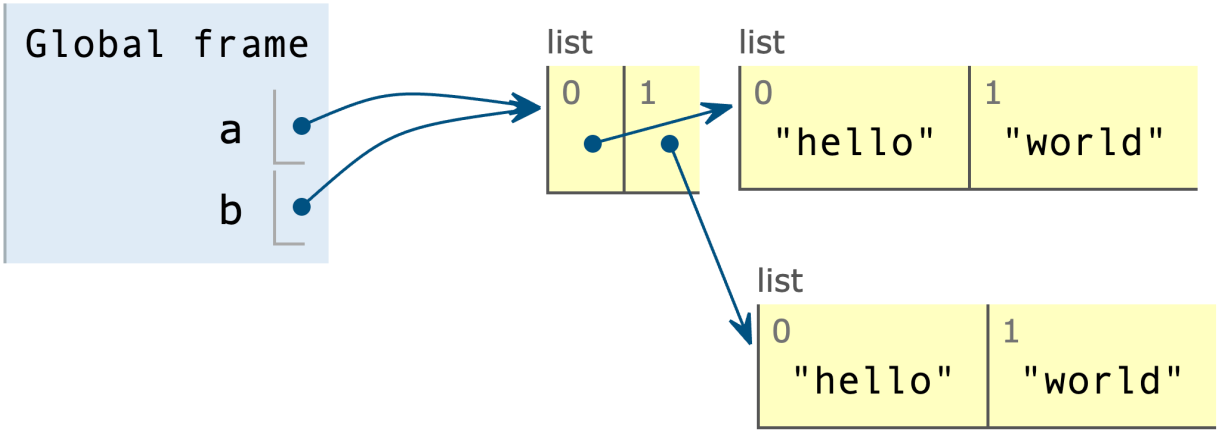
**Skriv svaret ditt her**

Format

Words: 0/600

Maks poeng: 12

**3(a)**



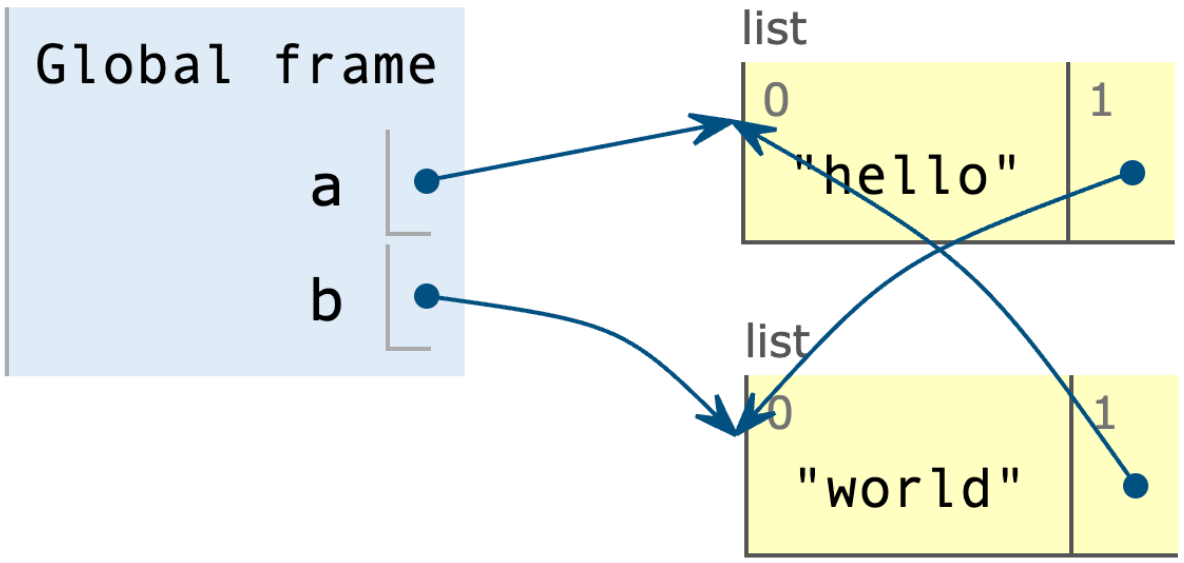

Skriv ein kodesnutt slik at minnet får den tilstanden som er vist ovanfor.

**Skriv svaret ditt her**

```
1
```

Maks poeng: 3

**3(b)**




Skriv ein kodesnutt slik at minnet får den tilstanden som er vist ovanfor. Hint: Du må mutere minst ei av listene etter at ho blir oppretta første gong.

**Skriv svaret ditt her**

```
1
```

Maks poeng: 3

**3(c) Bakgrunn**

På iskremfabrikken er det ein iskremmaskin. Maskina har moglegheit for å angi kva for smak iskremen som blir produsert skal ha. Maskina skriv ned alt som skjer i ei loggfil.

- Logg-filen er ei JSON-fil, og objektet fila skildrar er eit oppslagsverk med ein nøkkel "log" som viser til ei liste med hendingar.
- Ei hending er representert som eit oppslagsverk, og har alltid nøklane "timestamp" og "event_type"
- Dersom "event_type" er strengen "set_flavour", finst det òg ein nøkkel "value" i same hending der verdien er ein streng som skildrar smaken maskina frå no av er innstilt på.

Maskina vil aldri lage ein iskrem utan at det er valt ein smak først. Du kan gå ut frå at tidspunkta for hendingane i loggen alltid er i stigande rekkjefølgje. Sjå døme på ei slik logg-fil *machine_log.json* i PDF-vedlegget til oppgåva.

**Oppgåvetekst**

Skriv ein funksjon *production_summary* med ein parameter *path*. Anta at *path* er eit filnamn som peikar på ei loggfil frå iskremmaskina.

La funksjonen returnere eit oppslagsverk der dei ulike smakane som finst i loggen er nøklane, medan dei tilhøyrande verdiane er talet på iskremer av den gitte typen som vart produserte. Det er berre hendingar av typen "ice_cream_created" som skal teljast opp for kvar av dei ulike smakane.

For eksempel, dersom *log* viser til fila i dømet, skal funksjonen returnere eit oppslagsverk

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

**Skriv svaret ditt her**

```
1
```

Maks poeng: 10

**3(d) Bakgrunn**

Vareteljing ved iskremfabrikken finn som vanleg stad 24. november, så akkurat denne dagen skjer det ingenting anna ved fabrikken. Og no som vi endeleg er ferdige med å telje, har tida kome for å ta ein liten fot i bakken for å sjå om tala stemmer. Vi skal samstemme informasjon frå fleire kjelder:

- ***machine_log.json*** er ei fil som blir produsert av iskremmaskina. I denne fila finn vi informasjon om alle hendingar som skjer i iskremmaskina mellom vareteljinga i fjor og i år. Loggfila inkluderer ei hending for kvar gong ein boks med iskrem kjem ut på enden av samlebandet. Denne fila inneheld altså informasjon om kva som kjem inn på iskrem-lageret.
- ***packing_orders.json*** er ei fil som inneheld oversikt over alle pakkeordrar. Det er éi pakkeordre for kvar lastebil som forlèt lageret, og kvar pakkeordre inneheld informasjon om kor mange boksar iskrem av kvar type som blir med i lastebilen.
- ***inventory_2024.csv*** og ***inventory_2025.csv*** er semikolon-separerte CSV-filer som inneheld talet på iskremboksar av kvar type for vareteljingane i høvesvis i fjor og i år.

Du kan sjå eksempel på korleis filene ser ut i vedlagde PDF. Gjer ditt beste for å forstå strukturen i korleis filene er bygde opp.

**Oppgåvetekst**

Skriv eit program som reknar ut (og skriv ut) kor mange iskremer av kvar type som på mystisk vis har forsvunne i løpet av året.

For eksempel, dersom filene er som i det viste dømet, skal programmet skrive ut

```
Forsvunne iskrem
vanilje: 3
pistasj: 5
```

*Forklaring på kvifor det manglar tre vaniljeis: I fjor hadde vi 10. I løpet av året vart det produsert 3, så til saman 13. Det vart sendt ut 8+1=9 is til kundar. Då er det forventa at det skal vere 13-9=4 vaniljeis på lageret, men vi fann berre 1 i årets vareteljing. Så då må 3 ha forsvunne.*

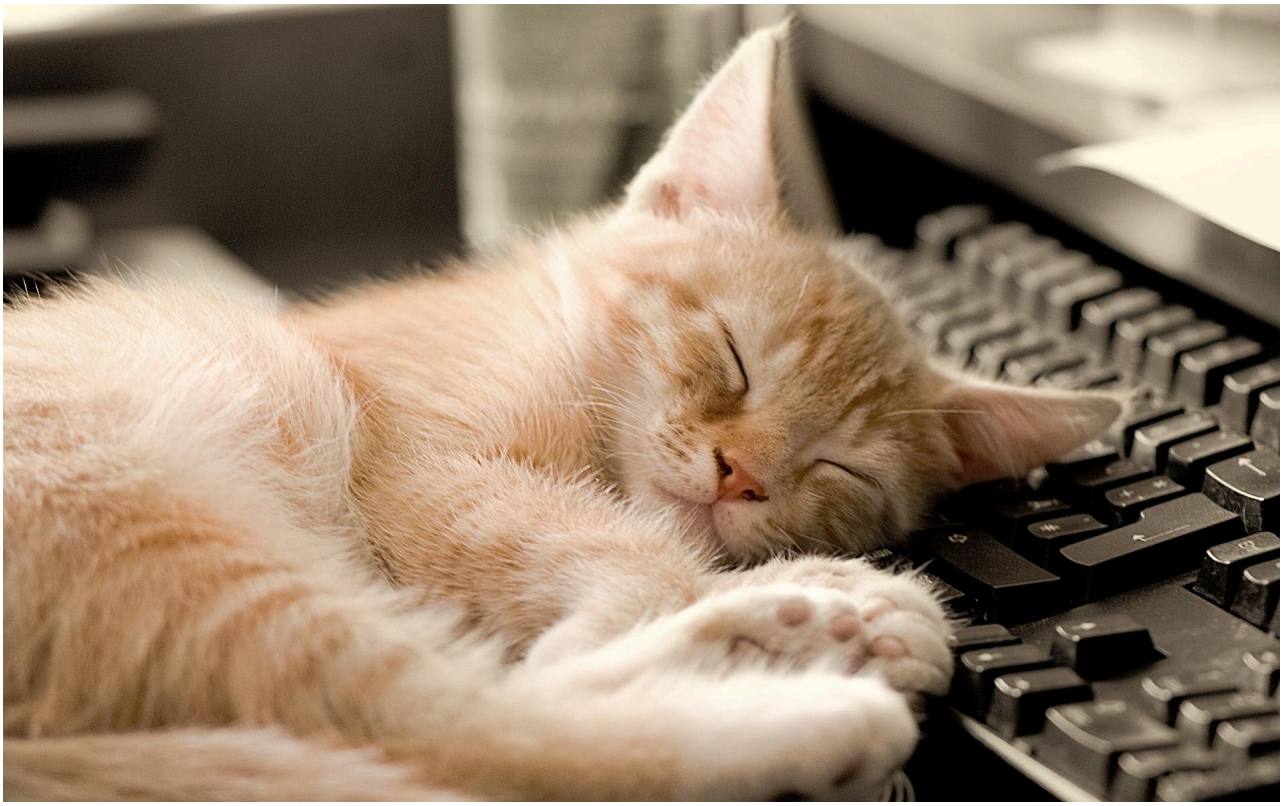
Du kan bruke ein antatt fungerande versjon av funksjonen du skreiv i førre oppgåve fritt utan å skrive ho på nytt her.

**Skriv svaret ditt her**

```
1
```

Maks poeng: 14

**4** I denne oppgåva blir det lagt inn poeng frå arbeidet ditt med labar du har gjort gjennom semesteret.



CC BY-NC 4.0 kot-spit-klaviatura by number86/goodfon.com

Lukke til med resten av eksamenssesongen, og tusen takk for eit 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()
```