Informasjon

UNIVERSITETET I OSLO
Det matematisk-naturvitenskapelige fakultet
Skriftlig eksamen i IN1000
2024 HØST
Varighet: 2.12.24, 9:00 til 2.12.24, 13:00
Tillatte hjelpemidler: Alle trykte og skrevne, ingen elektroniske
Det er viktig at du leser hele forsiden før du begynner å besvare eksamensoppgavene.
Om du ønsker å zoome i oppgavesettet, hold ctrl nede og trykk + eller - på numerisk tastatur.

Faglærer vil besøke lokalet ca kl. 10-12.

Eventuelle meldinger om feil eller presisering av oppgaver vil bli varslet med bjelle-ikonet øverst til høyre.

Dersom du mener en programmeringsoppgave er uklar, kan du gjøre egne forutsetninger og beskrive disse.

Du kan veksle mellom ulike språk (bokmål, nynorsk eller engelsk) øverst på siden.

Oppgavene gir til sammen 100 poeng. Poengfordelingen sier noe om vektingen av deloppgaver i sensuren.

1 Oppgave 1a

```
tall = 10
tall = tall + 2
print(tall*2)
```

Hva skrives ut når koden over kjøres?

Maks poeng: 1

2 Oppgave 1b

```
tekst1 = '1'
tekst2 = '2'
tekst3 = '3'
tall = int(tekst1 + tekst2) + int(tekst3)
print(tall)
```

Hva skrives ut når koden over kjøres?

Maks poeng: 2

3 Oppgave 1c

```
tall = 9
if tall<10:
    tall = tall+2
    tall = tall+1
else:
    tall = tall - 10
print(tall)
```

Hva skrives ut når koden over kjøres?

Maks poeng: 2

4 Oppgave 1d

```
summen = 0
for t1 in [1,2]:
    for t2 in [3,4]:
        if t1*t2 < 4:
            summen += t1*t2
print(summen)
```

Hva skrives ut når koden over kjøres?

Maks poeng: 2

5 Oppgave 1e

```
ordbok = {1:2}
teller = 1
while not 5 in ordbok:
    ordbok[1] = ordbok[1] + teller
    ordbok[ ordbok[1] ] = teller
    teller = teller+1
print(ordbok[1] )
```

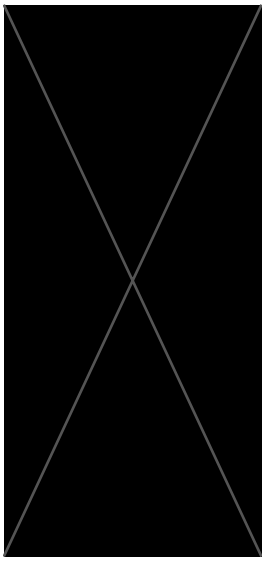
Hva skrives ut når koden over kjøres?

Maks poeng: 2

6 Oppgave 2a

```
class Person:
    def __init__(self, alder):
        self._alder1 = 0
        alder2 = alder #merk at ikke self
    def sett_alder1(self, alder):
        self._alder1 = alder
    def sett_alder3(self, alder):
        alder3 = alder #merk at ikke self
    def hent_alder1(self):
        return self._alder1
    def hent_alder2(self):
        return alder2 #merk at ikke self
    def hent_alder3(self):
        return alder3 #merk at ikke self
p1 = Person(10)
p2 = Person(20)
#Koden over er felles for oppgave 2a,2b,2c og 2d.
print( p1.hent_alder1() )
```

Hva skjer når koden over kjøres?



Velg ett alternativ:

- Tallet 10 skrives ut
- Tallet 20 skrives ut
- Tallet 15 skrives ut
- Man får en feilmelding (kodelinjene vil ikke kjøre)
- Tallet 0 skrives ut

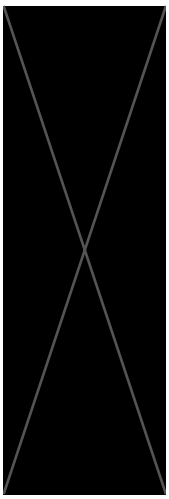
Maks poeng: 2

7 Oppgave 2b

```
class Person:
    def __init__(self, alder):
        self._alder1 = 0
        alder2 = alder #merk at ikke self
    def sett_alder1(self, alder):
        self._alder1 = alder
    def sett_alder3(self, alder):
        alder3 = alder #merk at ikke self
    def hent_alder1(self):
        return self._alder1
    def hent_alder2(self):
        return alder2 #merk at ikke self
    def hent_alder3(self):
        return alder3 #merk at ikke self
p1 = Person(10)
p2 = Person(20)
#Koden over er felles for oppgave 2a,2b,2c og 2d.
tall = p1.hent_alder1()
p3 = p1
p3.sett_alder1(15)
print(tall)
```

Hva skjer når koden over kjøres?

Velg ett alternativ:



- Det skrives ut 15 til terminalen
- Det skrives ut 0 til terminalen
- Man får en feilmelding (kodelinjene vil ikke kjøre)
- Det skrives ut 10 til terminalen
- Det skrives ut 20 til terminalen

Maks poeng: 2

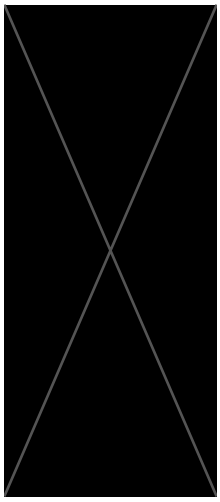
8 Oppgave 2c

```
class Person:
    def __init__(self, alder):
        self._alder1 = 0
        alder2 = alder #merk at ikke self
    def sett_alder1(self, alder):
        self._alder1 = alder
    def sett_alder3(self, alder):
        alder3 = alder #merk at ikke self
    def hent_alder1(self):
        return self._alder1
    def hent_alder2(self):
        return alder2 #merk at ikke self
    def hent_alder3(self):
        return alder3 #merk at ikke self
p1 = Person(10)
p2 = Person(20)
#Koden over er felles for oppgave 2a,2b,2c og 2d.
print( p1.hent_alder2() )
```

Hva skjer når koden over kjøres?

Velg ett alternativ:

- Det skrives ut 20 til terminalen
- Det skrives ut 0 til terminalen
- Man får en feilmelding (kodelinjene vil ikke kjøre)
- Det skrives ut 15 til terminalen
- Det skrives ut 10 til terminalen



Maks poeng: 2

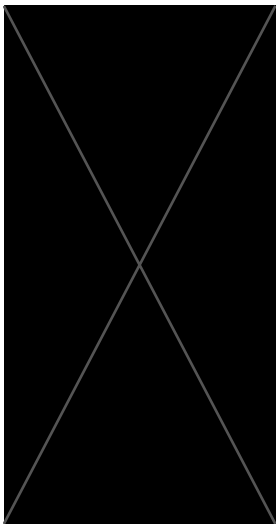
9 Oppgave 2d

```
class Person:
    def __init__(self, alder):
        self._alder1 = 0
        alder2 = alder #merk at ikke self
    def sett_alder1(self, alder):
        self._alder1 = alder
    def sett_alder3(self, alder):
        alder3 = alder #merk at ikke self
    def hent_alder1(self):
        return self._alder1
    def hent_alder2(self):
        return alder2 #merk at ikke self
    def hent_alder3(self):
        return alder3 #merk at ikke self
p1 = Person(10)
p2 = Person(20)
#Koden over er felles for oppgave 2a,2b,2c og 2d.
p1.sett_alder3(15)
p2.hent_alder3()
```

Hva skjer når koden over kjøres?

Velg ett alternativ:

- Det skrives ut 15 til terminalen
- Det skrives ut 0 til terminalen
- Det skrives ut 10 til terminalen
- Man får en feilmelding (kodelinjene vil ikke kjøre)
- Det skrives ut 20 til terminalen



Maks poeng: 2

10 Oppgave 3a

Skriv funksjonen ***jages*** som har en parameter *dyreliste* som er en liste med dyr, hvor hvert dyr er en av strengene ‘mus’, ‘katt’ eller ‘hund’. Hvert dyr kan være flere ganger i lista.

Hvis noen av dyrene i lista vil jage noen av de andre dyrene skal funksjonen returnere True, ellers False. Vi antar at en hund alltid vil jage en katt, og at en katt alltid vil jage en mus, men at dyrene ellers ignorerer hverandre.

Kallet *jages(['mus','katt'])* skal altså returnere *True* fordi katten vil jage musa,
mens kallet *jages(['mus','mus','hund'])* skal returnere *False.*

Skriv ditt svar her

Maks poeng: 7

11 Oppgave 3b

Skriv en funksjon ***utvidet_jages*** som ligner på funksjonen i oppgave 3a, bortsett fra at man her ønsker å være fleksibel på hvilke dyr som finnes og på hvilke dyr som jager hvilke.

Funksjonen utvidet_jages har to parametre: en parameter *dyreliste* som er en liste av strenger, og en parameter *jaging* som er en ordbok hvor hver nøkkel er et dyr (en streng) som jager dyret (streng) i den tilhørende innholdsverdien. Denne funksjonen kan håndtere alle slags dyr.

Kallet *utvidet_jages(['ulv','mus','sau'], {'ulv':'sau'})* skal altså returnere *True* fordi vi har definert at ulv jager sau og begge disse dyrene finnes i dyrelista.
Kallet *utvidet_jages(['ulv','mus','hund'], {'ulv':'sau', 'katt':'mus'})* skal derimot returnere *False*.

Skriv ditt svar her

Maks poeng: 8

12 Oppgave 3c

Skriv en funksjon ***flertall*** som tar inn en liste *dyreliste* med navn på ulike dyr (en liste av strenger), og som returnere navnet på dyret det finnes flest av i lista (hvilken streng som finnes flest ganger i lista). Dersom det er likt mellom to - altså at de to dyrene det finnes flest av er til stede like mange ganger i lista - så skal funksjonen returnere "uavgjort".

Kallet *flertall(['ulv','sau','ulv'])* skal altså returnere *'ulv',* mens kallet *flertall(['ulv','sau','sau','ulv'])* skal returnere *'uavgjort'.*

Skriv ditt svar her

Maks poeng: 8

13 Oppgave 4

Oppgaveteksten til oppgave 4 ligger i vedlagte PDF dokument. Du kan legge til egne metoder i løsningen din ved behov, kommenter i så fall disse kort. Om du står fast på en deloppgave, fortsett på de som kommer etter. Du kan bruke metoder beskrevet i oppgaveteksten selv om du ikke selv skriver dem.

Skriv ditt svar her

Maks poeng: 50

14 Oppgave 5a

Skriv en funksjon ***felles*** som har en parameter *tall_lister* som tar inn en nøstet liste - en liste av lister av heltall. Funksjonen skal returnere en liste med alle heltall som finnes to eller flere ganger i den nøstede lista. Det er uten betydning om et slikt heltall finnes i flere ulike lister i den nøstede lista eller om det finnes flere ganger i den samme lista. Det er også uten betydning hvilken rekkefølge de felles tallene returneres i.

Kallet *felles(*[[1,5,1], [3,4], [2,3]]*)* skal altså returnere enten *[1,3]* eller [3,1] siden tallene 1 og 3 finnes to steder i den nøstede lista (og siden rekkefølgen på tallene som returneres ikke er av betydning).

Skriv ditt svar her

Maks poeng: 5

15 Oppgave 5b

Skriv en funksjon ***adskilt*** som har en parameter ***tall_lister*** som tar inn en nøstet liste - en (ikke-tom) liste av (ikke-tomme) lister av tall. Funksjonen skal returnere True dersom det i den nøstede listen finnes to lister hvor det er slik at hvert av tallene i den ene lista er ekte større enn alle tallene i den andre lista.

Kallet *adskilt([[1,10], [6,8], [2,3]])* skal altså returnere True fordi hvert tall i den andre lista er større enn alle tall i den tredje.
Kallet *adskilt([[1,10], [6,8], [2,3,7]])* skal returnere False.

Skriv ditt svar her

Maks poeng: 5

Question 13
Attached

Oppgave 4. Strømmetjeneste (50 poeng)

I denne oppgaven skal du lage en første versjon av en enkel *strømmetjeneste*. Strømmetjenesten tilbyr *abonnentene* en rekke ulike *serier*. Hver serie har én eller flere episoder.

For å gjøre det lettere for abonnenter å finne frem til serier de liker, bruker tjenesten tag-er. En *tag* sier noe om innholdet i en episode. Eksempler på tag-er er «komedie», «dokumentar» og «drama».

Hver episode kan ha flere tag-er. Til sammen beskriver tag-ene på alle episoder i serien hva slags innhold serien har, og hvor mye det er av ulike typer innhold.

Når det opprettes en ny abonnent, må abonnenten oppgi sine *preferanser.* For hver tag som finnes spør tjenesten om abonnenten liker serier med denne typen innhold, er nøytrale til innholdet, eller misliker innholdet.

Tjenesten kan dermed foreslå serier som passer for en abonnent ved å beregne en *match* mellom abonnentens preferanser og tag-ene på de ulike seriene.

Husk at du kan bruke klasser og metoder fra en tidligere deloppgave selv om du ikke har besvart den. Bruk navn på klasser og metoder som gitt i oppgaveteksten (navn i **bold**).

4a) 6 poeng

Skriv klassen **Abonnent** med konstruktør. Konstruktøren har parametere for abonnentnavn (streng) og preferanser (ordbok beskrevet senere i oppgaven). Instansvariabler:

- abonnentnavn (streng)
- preferanser (ordboken som er parameter til konstruktøren)
- påbegynte serier (ordbok der serienavn er nøkkel, siste episodenummer abonnenten har sett er verdi)

Skriv også metodene:

- **hent_preferanser**. Returnerer ordboken med abonnentens preferanser.
- **sjekk_om_sett**. Metoden har en parameter serienavn (streng). Den returnerer True om abonnenten har sett en eller flere episoder av serien, ellers False.
- **se_en_episode** Metoden har en parameter serienavn, og legger til en ny serie eller oppdaterer episodenummer for serien hvis den allerede er påbegynt.

4b) 4 poeng

Skriv klassen **Serie** med konstruktør. Konstruktøren har parameter serienavn. Den kaller på hjelpemetoden **_les_fra_fil** som leser episoder og tag-er fra fil (se neste deloppgave). Instansvariabler:

- serienavn
- episoder i serien (ordbok med episodenummer som nøkkel, liste med tags som verdi)
- tag-er i serien (ordbok der tag er nøkkel, antall episoder med tag-en er verdi)

4c) 10 poeng

Utvid klassen Serie med metodene:

- **_les_fra_fil**. Metoden har ingen parametere (annet enn self) eller returverdi. Den leser en fil med samme navn som serien etterfulgt av ".txt". Hver linje i denne filen inneholder tag-er for én episode, atskilt av mellomrom. På linje 1 ligger tag-er for episode 1, på linje 2 for episode 2, osv. Metoden legger inn hver episode med tag-er og oppdaterer hvilke tag-er som finnes for serien.
- **hent_serietags**. Metoden returnerer alle tag-er tilknyttet serien som en liste.

4d) 10 poeng

Skriv klassen **Tjeneste** med konstruktør. Konstruktøren har en parameter for en liste av serienavn. Instansvariabler:

- seriene tjenesten tilbyr (ordbok der serienavn er nøkkel)
- tag-er i bruk i tjenesten (liste av strenger)
- abonnenter (ordbok der abonnentnavn er nøkkel, referanse til abonnent er verdi)

Konstruktøren oppretter et **Serie**-objekt for hvert av serienavnene og lagrer disse i ordboken med serier. Tag-ene i serien legges til i tjenesten dersom de ikke allerede er registrert.

4e) 10 poeng

Skriv også følgende metode i klassen Tjeneste:

- **ny_abonnent**. Metoden har ingen parametere eller returverdi. Den ber om og leser abonnentens navn, og preferanser for hver enkelt tag i tjenesten fra terminal. Preferansene lagres i en ordbok der tag er nøkkel, verdi er -1 (*misliker*) eller 1 (*liker*). Om bruker gir annen preferanse for en tag enn -1, 0 eller 1 skal metoden be om ny verdi inntil bruker gir lovlig input. Tag-er abonnenten er nøytral til (0) tas ikke med i preferanser-ordboken. Ny abonnent med navn og preferanser opprettes og legges til tjenesten.

4f) 5 poeng

Du skal nå utvide klassene **Serie** og **Tjeneste** med metoder som gjør det mulig å foreslå serier for en abonnent basert på abonnentens preferanser.

Utvid klassen **Serie** med metoden:

- **beregn_match**. Metoden har en parameter for en abonnents preferanser (samme som i Abonnent-klassen) og returnerer et heltall som angir hvor godt denne serien matcher disse. Hvis serien ikke har noen av tag-ene i preferansene, er match = 0. Hvis det er flest tag-er abonnenten *liker* skal match være et positivt tall, hvis det er flest tag-er abonnenten *misliker*, skal match være et negativt tall. Ta hensyn til hvor mange episoder tag-ene forekommer i.

4g) 5 poeng

Utvid klassen **Tjeneste** med følgende metode:

- **foreslå_serier**. Metoden har en parameter som angir abonnentnavn, og beregner match mellom hver serie og abonnentens preferanser ved hjelp av metoden beregn_match. Metoden foreslår kun serier abonnenten ikke har sett før, og som har match > 0. Dersom ingen serier tilfredsstiller disse kravene skriver metoden ut beskjed om dette, ellers skriver den ut navnene på foreslåtte serier.