UNIVERSITETET I OSLO

KANDIDAT

15591

PRØVE

IN1000 1 Introduksjon til objektorientert programmering

Emnekode	IN1000
Vurderingsform	Individuell skriftlig prøve
Starttid	02.12.2024 08:00
Sluttid	02.12.2024 12:00
Sensurfrist	--
PDF opprettet	31.08.2025 06:31

Introduksjon

Oppgave	Tittel	Oppgavetype
ℹ	Informasjon	Informasjon eller ressurser

Oppgave 1 (9 poeng)

Oppgave	Tittel	Oppgavetype
1(a)	Oppgave 1a	Fyll inn tall
1(b)	Oppgave 1b	Fyll inn tall
1(c)	Oppgave 1c	Fyll inn tall
1(d)	Oppgave 1d	Fyll inn tall
1(e)	Oppgave 1e	Fyll inn tall

Oppgave 2 (8 poeng)

Oppgave	Tittel	Oppgavetype
2(a)	Oppgave 2a	Flervalg
2(b)	Oppgave 2b	Flervalg
2(c)	Oppgave 2c	Flervalg
2(d)	Oppgave 2d	Flervalg

Oppgave 3 (23 poeng)

Oppgave	Tittel	Oppgavetype
3(a)	Oppgave 3a	Programmering
3(b)	Oppgave 3b	Programmering
3(c)	Oppgave 3c	Programmering

Oppgave 4 (50 poeng)

Oppgave	Tittel	Oppgavetype
4	Oppgave 4	Programmering

Oppgave 5 (10 poeng)

Oppgave	Tittel	Oppgavetype
5(a)	Oppgave 5a	Programmering
5(b)	Oppgave 5b	Programmering

1(a) # Oppgave 1a

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

Hva skrives ut når koden over kjøres?

24 .

Maks poeng: 1

1(b) Oppgave 1b

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

Hva skrives ut når koden over kjøres?

15 .

Maks poeng: 2

1(c) 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?

12 .

Maks poeng: 2

1(d) 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?

3 .

Maks poeng: 2

1(e) 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?

5

Maks poeng: 2

2(a) 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:

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

Maks poeng: 2

2(b)

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

Maks poeng: 2

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

Maks poeng: 2

2(d) 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:

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

Maks poeng: 2

3(a) 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

```
def jages (dyreliste):
    if "hund" in dyreliste and "katt" in dyreliste:
        return True
    elif "katt" in dyreliste and "mus" in dyreliste:
        return True
    return False

```

Maks poeng: 7

3(b) 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

```
#Er litt usikker på om jeg skjønte oppgaven korrekt, men antar at så lenge èn av
    skal True returneres, ellers skal False returneres. Det er det jeg forstår i
def utvidet_jages(dyrelise, jaging):
    for dyr in jaging:
        if dyr in dyreliste and jaging[dyr] in dyreliste:  #Om både nøkkel og ve
            return True
    return False

```

Maks poeng: 8

3(c) 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

```
1  def flertall (dyreliste):
2      ordbok = {}                          #Lager en ordbok for å holde styr på ar
3      for dyr in dyreliste:
4          if dyr not in ordbok:
5              ordbok[dyr] = 1
6          else:
7              ordbok[dyr] += 1
8
9      flest = 0
10     flest_dyr = None
11     for dyr in ordbok:
12         if ordbok[dyr] > flest:
13             flest = ordbok[dyr]
14             flest_dyr = dyr
15
16     for dyr in ordbok:              #Går gjennom ordbok igjen og sjekker om det
           hvis ja returneres "uavgjort"
17         if ordbok[dyr] == flest:
18             return "uavgjort"
19
20     return flest_dyr                #Hvis ikke "uavgjort" over returneres, da r
21
```

Maks poeng: 8

4 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

```
#Oppgave 4a)
class Abonnent:
    def __init__ (self, navn, preferanser):
        self._navn = navn
        self._preferanser = preferanser
        self._påbegynte_serier = {}             #Antar at den ved opprettelse er
            oppgitt i oppgaven. I tillegg, skal det skrives en metode senere for
            , altså se_en_episode()-metoden

    def hent_preferanser (self):
        return self._preferanser

    def sjekk_om_sett (self, serienavn):
        if serienavn in self._påbegynte_serier:
            return True
        return False

    def se_en_episode (self, serienavn):
        if serienavn in self._påbegynte_serier:
            self._påbegynte_serier[serienavn] += 1
        else:
            self._påbegynte_serier[serienavn] = 1

#Oppgave 4b)
class Serie:
    def __init__ (self, navn):
        self._serienavn = navn
        self._episoder = {}                     #Tom i starten, da man skal lese
        self._tager = {}
        self._les_fra_fil()

#Oppgave 4c)
    def _les_fra_fil (self):
        fil = open(self._serienavn + ".txt", "r")
        episodenr = 0

        for linje in fil:
            if linje != "":                 #Dersom linjen ikke er en tom linje,
                episodenr += 1
                tager = linje.strip().split()
                self._episoder[episodenr] = tager

        for episode_tager in self._episoder.values():
            for tag in episode_tager:               #Går gjennom tagene i tag-li
                if tag not in self._tager:          #Legger til i ordboka self._
                    episoder med tag-en er verdien
                    self._tager[tag] = 1
                else:
                    self._tager[tag] += 1

        fil.close()

    def hent_serietags (self):
        return self._tager

    def hent_serienavn (self):          #Lagd metoden selv for å bruke i oppgave
        return self._serienavn

#Oppgave 4f)
    def beregn_match (self, preferanser):
```

```
        antall_liker_episoder = 0
        antall_misliker_episoder = 0

        for tag in self._tager:
            if preferanser[tag] == 1:
                antall_liker_episoder += self._tager[tag]                #Øker a
                    verdien i self._tager - ordboka
            elif preferanser[tag] == -1:
                antall_misliker_episoder += self._tager[tag]

        if antall_liker_episoder > antall_misliker_episoder:             #Antar a
            misliker, da det ikke står spesifisert direkte i oppgaven hvilke hel
            return 1
        elif antall_misliker_episoder > antall_liker_episoder:
            return -1

        return 0                                #Dersom serien ikke har noen av tage

#Oppgave 4d)
class Tjeneste:
    def __init__ (self, liste_serienavn):
        self._serier = {}
        for navn in liste_serienavn:
            self._serier[navn] = Serie(navn)

        self._tager = []
        for serie in self._serier.values():
            tager = serie.hent_serietags()
            for tag in tager:
                if tag not in self._tager: #Passer på at kun unike tags blir lag
                    siden flere serier kan ha samme tager
                    self._tager.append(tag)

        self._abonnenter = {}                  #Antar at den starter tom, da man
            legge til abonnenter

#Oppgave 4e)
    def ny_abonnent (self):
        navn = input("Skriv inn et navn:")
        preferanser = {}

        for tag in self._tager:
            preferanse_verdi = int(input(f"Hva synes du om følgende tag: {tag}:
            while preferanse_verdi not in [-1, 0, 1]:  #Hvis ikke -1,0 eller 1 b
                nytt
                preferanse_verdi = int(input(f"Hva synes du om følgende tag: {ta
            preferanser[tag] = preferanse_verdi

        nytt_abonnent = Abonnent(navn, preferanser)
        self._abonnenter[navn] = nytt_abonnent

#Oppgave 4g)
    def foreslå_serier (self, abonnentnavn):
        abonnent = self._abonnenter[abonnentnavn]           #Henter abonennt-obj
        preferanser = abonnent.hent_preferanser()           #Henter preferanser

        foreslåtte_serier = []

        for serienavn in self._serier:
            if not abonnent.sjekk_om_sett(serienavn) and self._serier[serienavn]
                foreslåtte_serier.append(self._serier[serienavn])

        if len(foreslåtte_serier) > 0:
            print("Foreslåtte serier basert på dine preferanser: ")
            for serie in foreslåtte_serier:
                print(serie.hent_serienavn())    #Lagde en egen metode for å hen
                    innkapslingen
        else:
            print("Ingen serier å foreslå")
```

Maks poeng: 50

5(a) 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

```
1  def felles (tall_lister):
2      ordbok = {}          #Bruker en ordbok for å telle antall forekomster av et
3
4      for liste in tall_lister:
5          for tall in liste:
6              if tall not in ordbok:
7                  orbok[tall] = 1
8              else:
9                  orbok[tall] += 1
10
11     liste = []              #Lager en tom liste for å holde styr på tallene som
12
13     for tall in ordbok:
14         if ordbok[tall] >= 2:
15             liste.append(tall)
16
17     return liste
18
```

Maks poeng: 5

5(b) 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

```
1  def adskilt (tall_lister):
2      ordbok = {}
3
4      teller = 0
5
6      for liste in tall_lister:
7          minst = min(liste)
8          storst = max(liste)
9          ordbok[teller] = [minst, storst]: #Lagrer da indeksen som nøkkel og ver
               og største tallet i den indre listen
10         teller += 1
11
12     for i in range(len(ordbok)-1):
13         if ordbok[i][1] > ordbok[i+1][1] and ordbok[i][0] > ordbok[i+1][0] and
               ordbok[i][0] > ordbok[i+1][1]: #Sjekker for alle kombinasjoner, der
14             return True
15
16     return False                #Ellers False
17
```

Maks poeng: 5