Forside

UNIVERSITETET I OSLO
Det matematisk-naturvitenskapelige fakultet
Skriftlig eksamen i IN1000
2023 VÅR
Varighet: 31. mai kl. 09:00-13:00
Tillatte hjelpemidler: Alle trykte og skrevne. Ingen elektroniske.
Det er viktig at du leser denne forsiden nøye før du starter.

Faglærere vil besøke lokalet en gang i tidsrommet 10–12. Lurt å ha spørsmål til oppgavene (særlig langoppgaven om Kommuner og fylker) klar til da. Les derfor igjennom hele oppgavesettet før du starter med å svare.

Oppgavene

De første oppgavene er flervalgs-/kortsvarsoppgaver som rettes automatisk. Pass på å skrive inn nøyaktig svar med riktig type (ikke desimalpunktum eller -komma for heltall). Tekster kan skrives inn direkte (som her) eller med doble anførselstegn ("som her"), men ikke med enkle anførselstegn ('som her').

Siste oppgave er en oppgave som skal være et fullstendig program når delene legges sammen. I denne oppgaven er det viktig at besvarelsen/koden viser hvordan **klasser** og **konstruktører** lages og hvordan **objekter opprettes,** hvordan **objekter settes inn (og tas ut) av samlinger,** og hvordan **objekter refererer til hverandre.** Besvarelser som viser at kandidaten ikke behersker disse sentrale læringsmålene vil ikke kunne få bestått på eksamen. Poengene forteller altså hvor tidkrevende oppgavene er, men bestemmer ikke karakteren alene.

Dersom du mener en programmeringsoppgave er uklar, kan du gjøre egne forutsetninger og beskrive disse. Du kan også legge til egne funksjoner eller metoder ved behov, med begrunnelse og kommentarer i koden. Videre kan du bruke løsninger fra andre deloppgaver selv om du selv ikke har skrevet disse. Husk i såfall å referere til det du bruker. Du trenger ikke skrive kommentarer i kode som er direkte svar på en oppgave. Når du har skrevet en klasse i én deloppgave og i en senere skal legge til metoder i denne, kan du spare tid ved bare å skrive metodene, ikke det du har skrevet i en tidligere oppgave.

1 Ny oppgave 2e

Hvilket bilde viser tilstanden til variablene etter at følgende kode har kjørt?

```
class Hund:
    def __init__(self):
        self._eier = None
        self._sulten = True

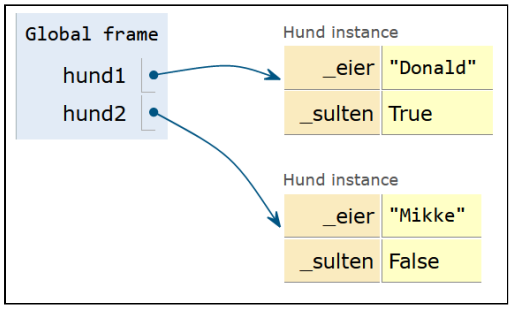
    def bor_hos(self, eier):
        self._eier = eier

    def gi_mat(self):
        self._sulten = False

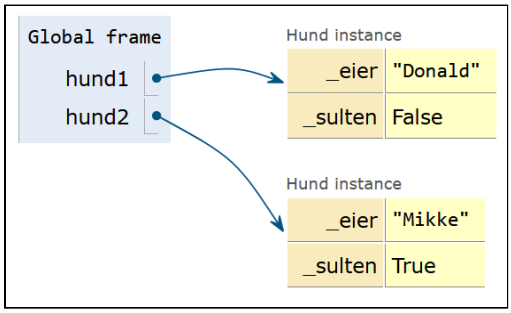
hund1 = Hund()
hund1.bor_hos("Donald")

hund2 = Hund()
hund2.gi_mat()
hund2 = hund1
hund2.bor_hos("Mikke")
```

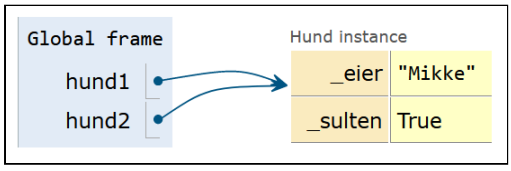
Bilde 1:



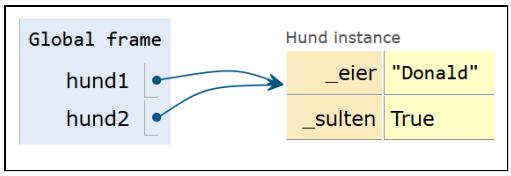
Bilde 2:



Bilde 3:



Bilde 4:



Velg ett alternativ

- Bilde 2
- Bilde 3
- Bilde 1
- Bilde 4

Maks poeng: 3

2 if

Etter at linje 4 i dette programmet har kjørt, hvilken kodelinje med **print** er den neste som kjører?

```
1   tall = 7
2
3   if tall < 10:
4       print("Tallet er mindre enn 10")
5   elif tall < 20:
6       print("Tallet er mindre enn 20")
7   elif tall < 0:
8       print("Tallet er negativt")
9   else:
10      print("Tallet er positivt")
11  print("Ferdig!")
12
```

Velg ett alternativ:

- Linje 11
- Linje 6
- Linje 8
- Linje 10

Maks poeng: 2

3 tilordning + uttrykk

Hva er verdien til variabelen **tall** etter at følgende kode er utført?

```
tall = 42
annet_tall = tall / 2
tall = annet_tall + annet_tall
```

Velg ett alternativ:

- 42.0
- 21.0
- 2.0
- 84.0

Maks poeng: 1

4 løkker

Hvor mange tall vil denne løkka skrive til terminalen?

```
from random import randint

terningkast = 0
while terningkast < 6:
    # tilfeldig tall fra og med 1
    # til og med 6:
    terningkast = randint(1, 6)
    print(terningkast)
print("Ferdig!")
```

Velg ett alternativ:

- Det går ikke an å vite dette på forhånd
- 0
- 1
- 6

Maks poeng: 1

5 løkker 2

Til hvilke av disse situasjonene er en for-løkke naturlig å bruke?

Velg ett eller flere alternativer

- [ ] Gjøre noe med alle elementer i en liste
- [ ] Repetere helt til bruker skriver inn riktig svar
- [ ] Gjøre noe med alle nøkler eller verdier i en ordbok
- [ ] Repetere noe 100 ganger
- [ ] Repetere helt til klokka er 10

Maks poeng: 2

6 parameteroverføring

Hva skrives ut av dette programmet?

```
def doble(tall):
    tall = tall*2
    print(tall)

tall = 5
doble(tall)
print(tall)
```

Velg ett alternativ:

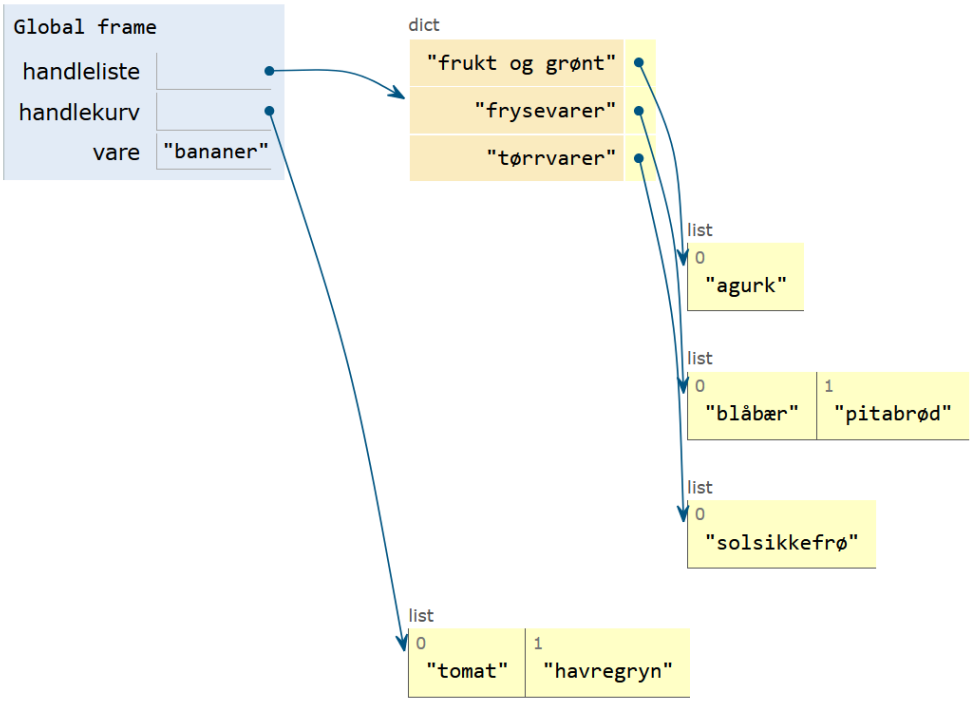
- ( ) Først 10, deretter 10
- ( ) Først 10, deretter 5
- ( ) Først 5, deretter 10
- ( ) Først 5, deretter 5

Maks poeng: 2

7 kodeforståelse samlinger

```
1   handleliste = {}
2   handleliste["frukt og grønt"] = ["agurk", "tomat", "bananer"]
3   handleliste["frysevarer"] = ["blåbær", "pitabrød"]
4   handleliste["tørrvarer"] = ["havregryn", "solsikkefrø"]
5
6   handlekurv = []
7   vare = handleliste["frukt og grønt"].pop(1)
8   handlekurv.append(vare)
9   vare = handleliste["tørrvarer"].pop(0)
10  handlekurv.append(vare)
11  vare = handleliste["frukt og grønt"].pop(1)
12  handlekurv.append(vare)
13  vare = handleliste["frysevarer"].pop(1)
14  handlekurv.append(vare)
```

Etter hvilken linje i programmet over er tilstanden til objektene som vist under?



Tilstanden er som vist rett **etter** at linje [ ] i programmet har kjørt.

Maks poeng: 3

8 referanser

Gitt følgende klassedefinisjon:

```
class Hus:
    def __init__(self, farge):
        self._farge = farge

    def farge(self):
        return self._farge
```

Hva skrives ut når følgende kode kjøres?

```
mitt_hus = Hus("blå")
ditt_hus = Hus("hvit")
ditt_hus = mitt_hus
mitt_hus = Hus("gul")

print(ditt_hus.farge())
```

Velg ett alternativ:

- blå
- hvit
- gul
- <__main__.Hus object at 0x000001870FE81A90>

Maks poeng: 3

9 input

Hvilken av disse kodelinjene gjør at variabelen **tall** inneholder et *heltall* (**int**) dersom brukeren skriver inn et tall?

Velg ett alternativ:

- tall = input(int("Skriv et tall:"))
- tall = input(int)
- tall = input("Skriv et tall:")
- tall = int(input("Skriv et tall:"))

Maks poeng: 1

10 ordbøker

Hvilke tall skrives ut av dette programmet (hvis vi ser bort fra linjeskift)?

```
måneder = {
    1: 31,
    2: 28,
    3: 31,
    4: 30,
    5: 31,
    6: 30,
    7: 31,
    8: 31,
    9: 30,
    10: 31,
    11: 30,
    12: 31
}

for x in måneder:
    print(x)
```

Velg ett alternativ:

- 0 1 2 3...
- 1 31 2 28...
- 1 2 3 4...
- 31 28 31 30...

Maks poeng: 2

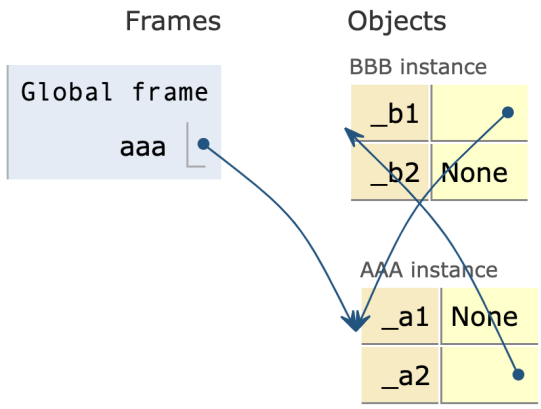
11 PythonTutor1

Gitt følgende kode:

```
class AAA:
    def __init__(self, rrr):
        self._a1 = rrr
        self._a2 = BBB(None, self)

class BBB:
    def __init__(self, a, b):
        self._b1 = b
        self._b2 = a
```

Skriv et **så kort** hovedprogram **som mulig** som gir opphav til denne datastrukturen i PythonTutor:



Skriv kode for hovedprogram her:

```
1
```

Maks poeng: 3

12 feilsøking (python tutor)

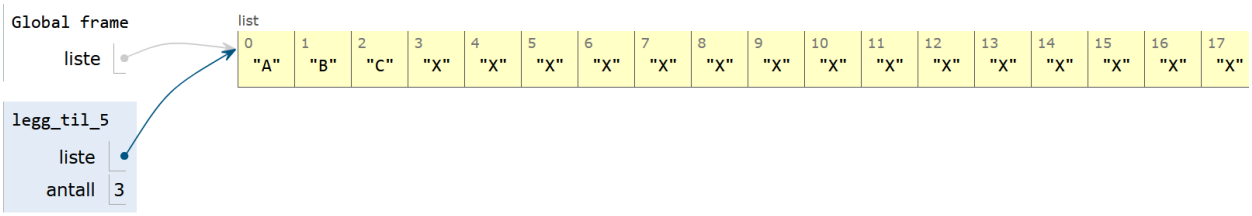
Du har skrevet en funksjon **legg_til_5** som tar inn en liste som parameter og om listen er for kort, legger den til nye elementer "X" helt til lista har 5 elementer.

Dette gjør du ved hjelp av en while-løkke med en løkkevariabel **antall** som skal holde styr på hvor mange elementer det er i lista til enhver tid.

Når du tester funksjonen med følgende testkode

```
liste = ["A", "B", "C"]
legg_til_5(liste)
```

oppdager du at dette skjer i debuggeren (hele listen er altfor lang til å vises i bildet):



Hva har skjedd her, og hvordan kan det løses?

Skriv ditt svar her

Format

Words: 0

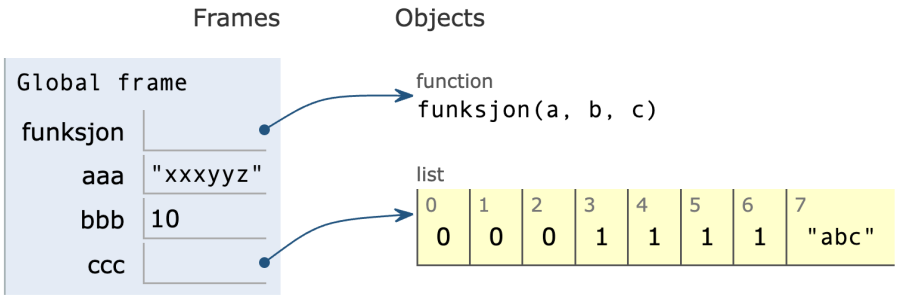
Maks poeng: 3

13 PythonTutor2

Gitt følgende funksjon:

```
def funksjon(a, b, c):
    aaa = 3*a
    bb = 2*b
    return aaa+bb+c
```

Skriv et hovedprogram med 3 linjer/kall på funksjon som gir opphav til denne tilstanden/datastrukturen i PythonTutor:



Skriv 3 tilordningssetninger her:

```
1
```

Maks poeng: 3

14 Ny oppgave 3e

Forenkle følgende kode ved hjelp av en funksjon (husk å tenke på hva som er fornuftig parameter og returverdi for funksjonen). Du må skrive både funksjonen og koden som bruker funksjonen.

Koden skal være gjenbrukbar i andre programmer som sjekker input fra bruker på samme måte.

```
navn = input("Skriv inn navn:")
while len(navn) == 0:
    print()
    print("Ugylig verdi!")
    navn = input("Skriv inn navn:")

print("Velkommen, " + navn + "!")

destinasjon = input("Skriv inn hvor du skal:")
while len(destinasjon) == 0:
    print()
    print("Ugyldig verdi!")
    destinasjon = input("Skriv inn hvor du skal:")

print("God tur til, " + destinasjon + "!")
```

Skriv ditt svar her

Maks poeng: 5

15 klasse og objekt

Hva er forskjellen på en klasse og et objekt?

Skriv ditt svar her

Format

Σ

Words: 0

Maks poeng: 2

16 grensesnitt 1

Skriv en klasse **Drage** som implementerer grensesnittet vist under. Du må selv finne ut hva som er fornuftige instansvariabler å bruke. Hensikten med denne klassen er å simulere en drage som spiser folk ved å telle hvor mange folk den spiser.

Metodenavn	Parametre	Returverdier	Beskrivelse
__init__	kapasitet: int	-	Oppretter en ny Drage som ikke har spist noen folk enda, men har kapasitet til å spise et visst antall folk.
mett	-	bool	Returnerer **True** hvis Dragen har spist et antall folk større enn eller lik sin kapasitet. **False** hvis ikke.
spis	antall: int	-	Hvis Dragen er mett (se metoden over), skjer det ingenting når denne metoden kalles. Hvis den ikke er mett, spiser den det spesifiserte antall folk. Etter at Dragen har spist, kan den bli mett hvis den har spist nok til det.

Skriv ditt svar her

```
1
```

Returnerer **True** hvis dragen har spist mer enn eller lik sin kapasitet. **False** hvis ikke.
Hvis Dragen er mett (se metoden over), skjer det ingenting når denne metoden kalles.

Hvis den ikke er mett, spiser den det spesifiserte antall folk. Etter at dragen har spist, kan den bli mett hvis den har spist nok til det.

Maks poeng: 4

17 grensesnitt 2

Bruk klassen **Drage** fra forrige oppgave og lag et program som lager en liste med 5 Drage-objekter som hver har kapasitet til å spise 10 folk. Hver Drage skal spise et tilfeldig antall folk fra og med 1 til og med 3 hver gang den spiser, og programmet ditt skal da skrive ut "Drage X spiste Y folk". (X er her indeksen til Dragen i listen, og Y er det tilfeldige antall folk Dragen spiste.) I tillegg skal programmet skrive ut "Drage X er mett" når Dragen ikke klarer å spise flere folk.

Programmet skal kjøre helt til alle Dragene er mette. Det er helt OK å la Drage 0 spise seg mett først, og så la Drage 1 spise seg mett, og så videre.

(Hint: Med "from random import randint" som første linje, vil funksjonskallet "randint(min, max)" returnere et tilfeldig tall fra og med **min** til og med **max**.)

Skriv ditt svar her

```
1
```

Maks poeng: 5

Kommuner og fylker i Norge

Dette er informasjon om langoppgaven som består av 7 deloppgaver. Hele oppgaven med alle deloppgavene finnes som et pdf-dokument på neste side. (Kun norsk). pdf-en finnes også som vedlegg.

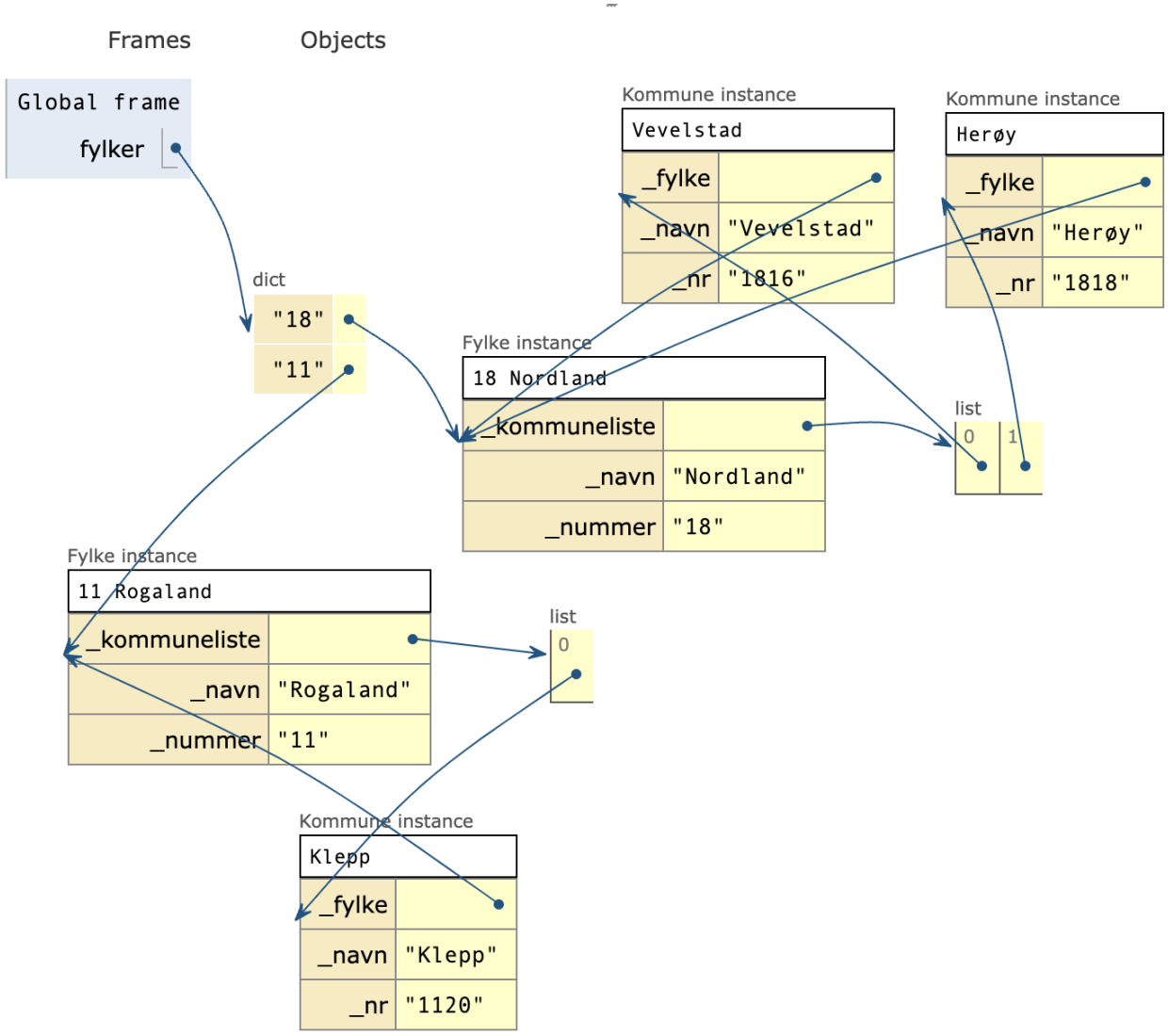
Norge er delt inn i 11 fylker. Hvert fylke er videre inndelt i kommuner. En kommune tilhører bare ett fylke. Vi gjør et søk i Wikipedia og finner denne lista (sterkt forkortet):

Nr [13]	Kommunenavn	Adm. senter	Fylke	Folketall[14]	Areal[15]	Kart	Våpen	Målform[16]
0301	Oslo	Oslo	Oslo	709 037	454,09			nøytral
1101	Eigersund	Egersund	Rogaland	15 011	432,48			bokmål
1103	Stavanger	Stavanger	Rogaland	146 011	262,52			nøytral
1106	Haugesund	Haugesund	Rogaland	37 855	72,68			bokmål
1108	Sandnes	Sandnes	Rogaland	82 548	1 040,56			nøytral
1111	Sokndal	Hauge	Rogaland	3 324	294,98			nøytral
1112	Lund	Moi	Rogaland	3 206	408,42			nøytral
1114	Bjerkreim	Vikeså	Rogaland	2 848	650,56			nøytral
								

De to første sifrene i kommunenummeret er fylkesnummer, f.eks. "03" for Oslo og "11" for Rogaland i lista over.

I de følgende oppgavene skal vi lage og bruke klasser som representerer fylker og kommuner, og lese inn informasjon om kommuner fra en tekstfil som ligner tabellen i figuren over. Det er naturlig å løse oppgavene i den rekkefølgen de står, men dette velger du selvfølgelig selv. Selv om du hopper over en oppgave (typisk en funksjon e.l.) eller har løst den feil, kan du i det videre bruke den som om du hadde løst den riktig.

Her er en datastrukturtegning laget med PythonTutor for hovedstrukturen vi skal fram til:



Her er det to fylker (Nordland og Rogaland) med henholdsvis to og én kommune. Vi ser ei global ordbok (fylker) som inneholder fylkesobjektene; fylkesnummeret er nøkkel (type str). I hvert fylke er det ei liste (_kommuneliste) med kommuner (kommuneobjekter). Kommunen har en instansvariabel (_fylke) som referer til fylket kommunen tilhører.

La oss se på Nordland; for å hente fylkesobjektet må vi slå opp i ordboka fylker med nøkkel "18" (fylkesnummeret til Nordland). Dette objektet (18 Nordland) har ei liste med kommuneobjekter (Vevelstad og Herøy). Disse to kommuneobjektene 'peker tilbake til' fylket de tilhører.

NB! *Kommunebjektene mangler noen instansvariable for å få fram datastrukturen.*

Hele oppgave 3 som pdf (bare norsk)

Ved siden ser du en PDF-versjon av hele oppgave 3 om kommuner og fylker. For hver side fremover vil du bli bedt om å besvare én av deloppgavene.

Du kan til enhver tid komme tilbake til denne siden om du ønsker å se oppgaven som helhet i PDF-format. Du kan endre på størrelsen av PDF-vinduet ved å dra i symbolet med de tre prikkene.

PDF-filen finnes også som vedlegg, se nedre marg.

18 Klassen Fylke

Et objekt av klassen Fylke skal ha informasjon om **navnet** og **nummeret** til fylket, samt **ei liste med kommuner** (objekter) som ligger i fylket. Denne lista skal være tom når et objekt av klassen opprettes. Navnet og nummeret til fylket skal være parametre til konstruktøren.

Fylke skal også ha to metoder til:

- en metode **legg_til_kommune** som legger en kommune (instans av klassen Kommune) til kommunelista
- en metode **antall_kommuner** som returnerer antallet kommuner som fylket har

Skriv klassen Fylke:

```
1
```

Maks poeng: 8

19 Klassen Kommune

Konstruktøren til klassen **Kommune** skal ha én parameter som er ei liste med 6 elementer (informasjon om kommunen), f.eks.

['1111', 'Sokndal', 'Hauge', 'Rogaland', '3 324', '294,98']

Klassen skal ha en instansvariabel **_fylke** som når objektet opprettes skal ha verdien None.

Klassen skal også ha metoder for å hente ut følgende informasjon om kommunen:

- nummer
- navn
- navn på fylket kommunen ligger i

Skriv klassen Kommune:

```
1
```

Maks poeng: 10

20 Metoder i Kommune

Utvid klassen **Kommune** med **2** metoder.

En 'magisk metode' som **returnerer en tekststreng (str)** slik at hvis variablen **k** refererer til et kommuneobjekt, skal setningen

print(**k**)

skrive ut kommunens nummer, navn og fylkets navn, f.eks.

1112 Lund i Rogaland

Klassen Kommune skal også ha en metode **sett_fylkesref** som tar (en referanse til) et fylkesobjekt som parameter og som setter instansvariabelen **_fylke** i kommuneobjektet til å referere til fylkesobjektet som kommunen hører til.

Skriv metodene her:

```
1
```

Maks poeng: 5

21 Infometoder i Fylke

I hovedprogrammet antar vi at variablene **rogaland**, **agder** og **innlandet** refererer til fylkesobjektene for henholdsvis Rogaland, Agder og Innlandet.

Du skal utvide klassen Fylke med **3 metoder**, slik at følgende setninger:

```
print(rogaland)
print(agder.kortinfo())
print(innlandet.kortinfo())
print(innlandet.info())
```

gir følgende resultat:

```
11 Rogaland
42 Agder har 25 kommuner.
34 Innlandet har 46 kommuner.
Innlandet(34): Kongsvinger, Hamar, Lillehammer, Gjøvik, Ringsaker, Løten, Stan
ge, Nord-Odal, Sør-Odal, Eidskog, Grue, Åsnes, Våler, Elverum, Trysil, Åmot, S
tor-Elvdal, Rendalen, Engerdal, Tolga, Tynset, Alvdal, Folldal, Os, Dovre, Les
ja, Skjåk, Lom, Vågå, Nord-Fron, Sel, Sør-Fron, Ringebu, Øyer, Gausdal, Østre
Toten, Vestre Toten, Gran, Søndre Land, Nordre Land, Sør-Aurdal, Etnedal, Nord
-Aurdal, Vestre Slidre, Øystre Slidre, Vang,
```

11, 42 og 34 er fylkesnummeret til henholdsvis Rogaland, Agder og Innlandet.
Skriv de 3 metodene her:

```
1
```

Maks poeng: 7

22 Lese inn kommuneinformasjon

Skriv en funksjon **les_kommuneinfo** som en del av hoveprogrammet. Funksjonen tar et filnavn (str) som parameter. Denne filen er på samme format som filen kommuneinfo.txt. Funksjonen returnerer **ei liste av lister** med informasjon fra tekstfilen. Her er de første linjene i filen kommuneinfo.txt:

```
0301    Oslo    Oslo    Oslo    709 037 454,09
1101    Eigersund   Egersund    Rogaland    15 011  432,48
1103    Stavanger   Stavanger   Rogaland    146 011 262,52
1106    Haugesund   Haugesund   Rogaland    37 855  72,68
1108    Sandnes Sandnes Rogaland    82 548  1 040,56
1111    Sokndal Hauge   Rogaland    3 324   294,98
1112    Lund    Moi Rogaland    3 206   408,42
1114    Bjerkreim   Vikeså  Rogaland    2 848   650,56
```

Hver linje er på formatet:

"1101\tEigersund\tEgersund\tRogaland\t15 011\t432,48"

ei tekstlinje med 6 "ord" adskilt med "\t" for hver kommune. Vi bruker dette spesialtegnet til å dele opp linja i 6 "biter".

Lista (ei liste av lister med 6 tekststrenger) som returneres skal ha dette formatet (liste nr 2, 1101 Eigersund, svarer til tekststrengen ovenfor):

```
[  ['0301', 'Oslo', 'Oslo', 'Oslo', '709 037', '454,09'],
   ['1101', 'Eigersund', 'Egersund', 'Rogaland', '15 011', '432,48'],
   ['1103', 'Stavanger', 'Stavanger', 'Rogaland', '146 011', '262,52'],
   ['1827', 'Dønna', 'Solfjellsjøen', 'Nordland', '1 391', '192,55'],
   ['1828', 'Nesna', 'Nesna', 'Nordland', '1 783', '183,19'],
   ['1832', 'Hemnes', 'Korgen', 'Nordland', '4 459', '1 589,51'],
   ....
]
```

Rekkefølgen spiller ingen rolle, men alle kommuner fra innfila skal være med.

Skriv funksjonen her:

```
1
```

Maks poeng: 5

23 Opprette datastrukturen med fylker og kommuner

Viktig: I denne oppgaven må du kommentere hvor koden du skriver ligger. Det må være tydelig hva som ligger i hovedprogrammet og hva som ligger i klassene Kommune og Fylke.

Lag en metode **fylkesnr** i klassen Kommune som **returnerer en tekststreng (str)** med de to første sifferne (tegnene) i kommunenummeret. (Hvis du ikke får dette til, kan du likevel bruke denne metoden som om den eksisterte.)

Skriv (en del av) et hovedprogram som oppretter kommuner (instanser av Kommune) og fylker (instanser av Fylke) på grunnlag av kallet **les_kommuneinfo("kommuneinfo.txt")**

Kommunene skal legges inn i ei liste. Lista skal hete **kommuner**.

Fylkene skal legges inn i ei ordbok. Ordboka skal hete **fylker**. **Fylkesnummer** (fra metoden nevnt tidligere i oppgaven) skal være nøkkel i ordboka, mens innholdsverdien skal være **objekter av Fylke**. Det vil si at hver gang vi leser en kommune fra fil, skal vi sjekke om fylket til kommunen eksisterer i ordboken, og opprette et nytt Fylke-objekt hvis fylket ikke eksisterer.

Du skal også opprette koblingene/referansene mellom fylkes- og kommuneobjektene:

I et **fylkesobjekt** skal (instansvariabelen for) kommunelista inneholde alle kommuner (kommuneobjekter) i fylket.

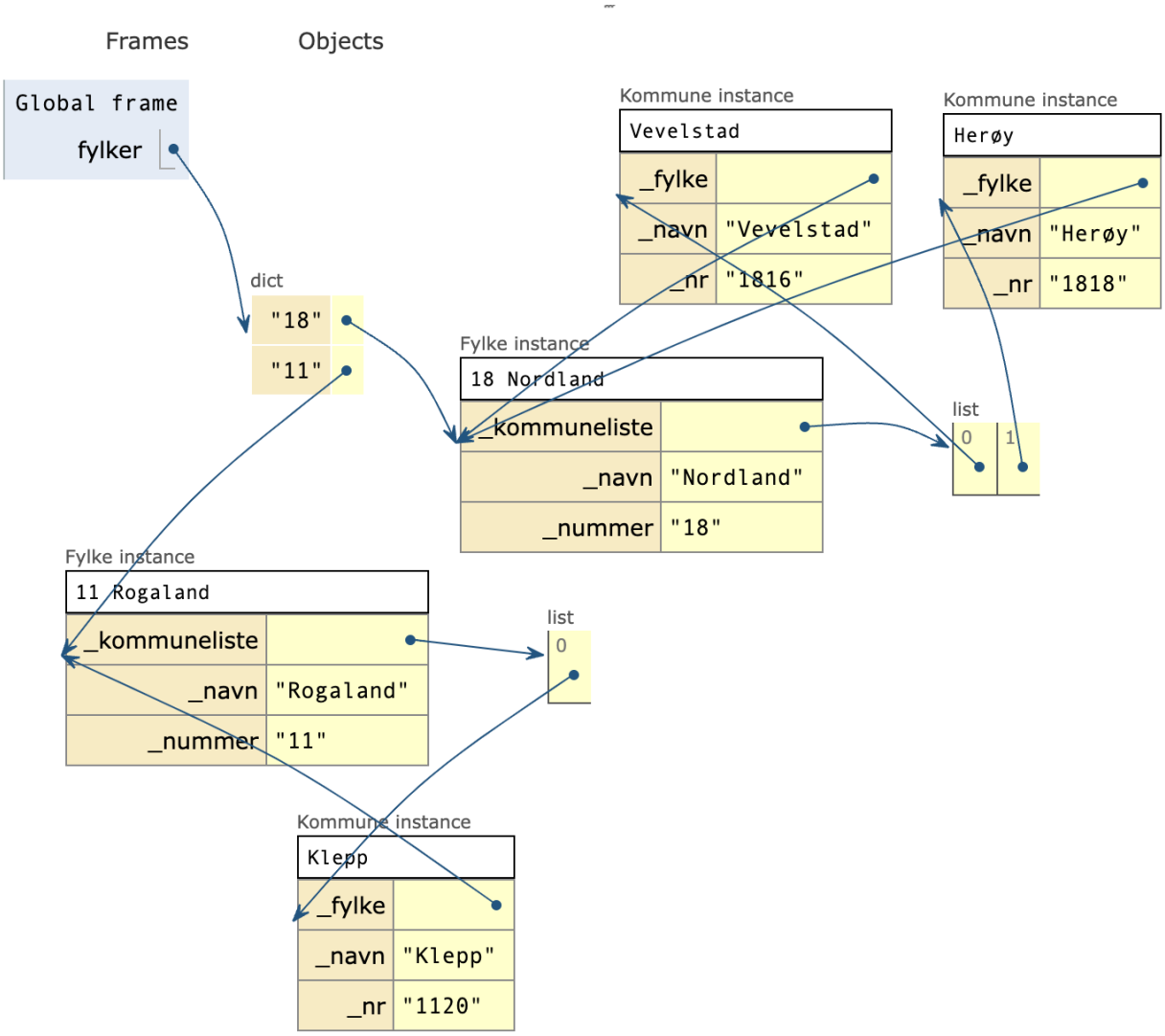
I et **kommuneobjekt**, skal instansvariabelen for fylke referere til det fylkesobjektet som kommunen hører til.

Jf. datastrukturdiagrammet fra PythonTutor gjengitt nedenfor.

Alt dette kan gjøres i ei løkke, men det er også mulig å gjøre det i flere omganger. Du må gjerne lage egne hjelpemetoder i Fylke og Kommune for å få til dette. Husk i såfall å kommentere hvor metoden er definert og hva den gjør.

Skriv ditt svar her

```
1
```



NB I denne enkle tegninga av datastrukturen er ikke lista med kommuner med.

Maks poeng: 12

24 Testprogram

Skriv resten av hovedprogrammet som gir/skriver ut svar på følgende spørsmål:

- Hvilket/hvilke fylker har flest kommuner?
- Hvilke kommuner (kommunenummer, -navn og fylkesnavn) har likt kommunenavn?

Skriv ditt svar her

```
1
```

Maks poeng: 8

Ekstraoppgaver

Dette er to helt frivillige ekstraoppgaver for dem som er helt ferdige med de øvrige oppgavene. Det er *utfordrende og tidkrevende oppgaver som gir lite uttelling*, så vent med å løse disse oppgavene til du er helt fornøyd med de andre svarene dine.

Det er mulig å få beste resultat på eksamen uten å gjøre disse oppgavene. Det betyr at *hvis disse oppgavene er ubesvart kan det ikke trekke ned*. Hvis de er løst kan de veie opp for mindre feil gjort andre steder og påvirke helhetsvurderingen positivt.

Vi forventer ikke at du skal få tid til å gjøre disse oppgavene, så bare slapp av om du ikke rekker å gjøre dem eller ikke får dem til.

25 Slå sammen to eller flere kommuner

Mange mener at Norge har for mange kommuner. Derfor skal du lage funksjonalitet i programmet for å slå kommuner sammen.

Skriv en prosedyre som tar ei liste med kommuner som parameter. Etter at prosedyren er kalt, skal datastrukturen oppdateres slik at kommunene i lista som er argument til kallet slås sammen til én kommune. For enkelhets skyld skal den nye kommunen ha samme navn, adm.senter og fylkestilhørighet som den første kommunen i argumentlista.

Du kan også (for enkelhets skyld) anta at kommunene ligger i samme fylke, men hvis du vil ha full uttelling på oppgaven, må prosedyren ta høyde for at kommunene som skal slås sammen kan ligge i forskjellige fylker. Husk å oppdatere datastrukturen med fylker og kommuner.

Du må gjerne lage nye (hjelpe)metoder i Kommune og Fylke for at prosedyren skal bli enklere å programmere og forstå.

Du skal ikke oppdatere informasjon om folketall og areal, men ta gjerne vare på informasjonen (i den sammenslåtte kommunen) slik at dette kan oppdateres seinere.

Skriv ditt svar her

```
1
```

Maks poeng: 5

26 Splitte opp fylker

Fylkestinget i Viken vedtok 23. februar å søke regjeringa om oppløsning av fylket. De tre nye fylkene Akershus, Buskerud og Østfold kan følgelig gjenopprettes fra 1. januar 2024, etter vedtak i Stortinget.

For å gjennomføre denne oppsplittinga av Viken får du tilgang til ei ordbok med navn nyeFylker av lengde 3. De tre nøkkelverdiene er navnene (str) til de nye kommunene. Innholdsverdien er ei liste av alle kommunene i det nye fylket. Hvert element i denne lista igjen består av to verdier (str), nytt kommunenummer og kommunenavn. De to første tegnene (sifrene) er som før nummeret til det nye fylket. Her er en (sterkt forkortet) kopi av ordboka:

```
nyeFylker = {
                "Buskerud": [  ["0602","Drammen"] , ["0605","Ringerike"], ...  ] ,
                "Akershus": [  ["0219","Bærum"], ["0220","Asker"] , ... ] ,
                 "Østfold" : [  ["0124","Askim"], ["0136", "Rygge"], ...  ]
}
```

Ingen av kommunene i ordboka har samme navn. Skriv en prosedyre som tar ei ordbok (med samme format som nyeFylker ovenfor) som parameter og som gjør følgende med datastrukturen for alle (nye) fylker og kommuner i ordboka:

- lager nye fylkesobjekter
- oppdaterer kommunenummer
- oppdaterer fylkesreferanse
- fjerner kommunen fra kommunelista i det gamle fylket

Når prosedyren har gjort dette for alle kommunene i ordboka, tas en siste sjekk om at oppsplittingen har vært vellykket:

- det gamle fylket skal nå ha 0 kommuner og skal fjernes fra datastrukturen
- antall nye fylker skal være lik lengden av ordboka

Skriv prosedyren så generell at den kan brukes til nye argumenter (på samme format) for andre oppsplittinger (som Troms- og Finnmark, f.eks).

Skriv ditt svar her

```
1
```

Maks poeng: 8