Eksamen i IN1000 høsten 2020

Digital hjemmeeksamen torsdag 3. desember 2020 kl. 09.00–13.00.

Du finner oppdatert informasjon om årets IN1000 eksamen på semestersiden til IN1000, inkludert kontaktinformasjon til faglærere og svar som kan være nyttige for flere på spørsmål om eksamen.

Alle hjelpemidler er tillatt (lærebok, nettressurser, notater osv.)

Under eksamen er det *ikke tillatt* å samarbeide eller kommunisere med andre om oppgavene.

Man kan bli trukket ut til samtale for å kontrollere eierskap til besvarelsen sin. Samtalen har ikke innvirkning på sensuren/karakteren, men kan lede til at instituttet oppretter fuskesak. Les mer om hva som regnes som fusk på UiOs nettsider. For øvrig gjelder informasjonen på nettsiden om eksamensavvikling ved MN høsten 2020.

Dette oppgavesettet inneholder oppgaver som gir totalt 100 poeng.

Oppgaver innen hver seksjon trekkes tilfeldig for hver kandidat. Rekkefølgen på oppgaver i seksjon 1-3 trekkes tilfeldig (det er altså ikke nødvendigvis stigende vanskelighetsgrad innen seksjonen).

Les oppgaven grundig - spesielt i oppgave 1 og 2 (som blir rettet automatisk) kan det variere om du skal svare med en hel programsetning eller bare det som mangler.

Brukerstøtte for hjemmeeksamen

1(a) Kva for uttrykk manglar som argument til print i line 5 for at "Erna Solberg" skal skrivast ut av denne koden? Uttrykket skal innehalde variabelen **navn**. Fyll ut i svarboksen.

```
statsminister = False
navn = "Er"
if statsminister:
    navn = navn + " Solberg"
print ( )
```

Maks poeng: 1

1(b)

```
1  a = randint(0, 100) # gir a en tilfeldig heltallsverdi mellom 0 og 100
2  b = randint(0, 100)
3  c = randint(0, 100)
4
5  while a != b or a > c:
6      b = randint(0, 100)
7      c = randint(0, 100)
8
9  if a <= c and    ?    :  # erstatt ?-tegnet med et logisk uttrykk
10     print("OK")
11 else:
12     print("Nei, her gikk du i else-greina!")
13
14
```

Korleis kan du fullføre testen i line 9 med eit logisk uttrykk (i steden for **?**) som gjer at denne koden alltid skriv ut "OK"? Uttrykket skal inkludere variablene **a** og **b**.

Maks poeng: 1

1(c) Hva er det *største* mulige heitallet som kan tilordnes variabelen **tall** i linje 1 slik at variabelen **tekst** får verdien "**baa**" ?

```
1  tall =     # hvilket heltall skal stå her?
2  t1 = "a"
3  if tall < 8:
4      t1 = t1 + "a"
5  else:
6      t1 = t1 + "b"
7
8  tekst = "b" + t1
9
10 # skriv det største heltallet som gir tekst verdien "baa"
11
```

Skriv bare tallet :

Maks poeng: 1

1(d) Skriv et logisk uttrykk i svarfeltet, slik at utskriften alltid blir "**ja**". Bruk operatoren **>** og heltallene **3** og **4.**

```
if  False or [ ] :
    print("ja")
else:
    print("nei")
```

Maks poeng: 1

1(e)

```
a=1
while a<10:
    a = 3*a
    for b in [3,2,1]:
        a += b
if a<25:
    a = a*2
    while a < 100:
        a += 1
while a<29:
    a = a + 3
print(a)
```

Hva blir skrevet ut her? [ ] (bare verdien)

Maks poeng: 1

1(f) Hvilket helltall mangler i uttrykket på nederste linjen for at variabelen **tall** skal få verdien 10 etter at denne koden er kjørt?

```
tall = 5
while tall < 10:
    tall = tall - 2
    tall = tall * 2

tall = tall -  ?
```

Skriv tallet som skal erstatte spørsmålstegnet her (du skal ikke endre noe i resten av linja):

[ ] .

Maks poeng: 1

1(g) Skriv en verdi i svarfeltet slik at utskriften alltid blir "**ja**".

```
liste = [-1, 2, 322, 95]
s = 0
for tall in liste:
    s = s + liste[1]
if s ==  :
    print("ja")
else:
    print("nei")
```

Maks poeng: 1

1(h) Espens fulle navn er **Espen Askeladd**. Hvordan oppretter du et nytt objekt referert av **helt** slik at navnet blir satt riktig i objektet? Fyll ut svarboksen med høyresiden som mangler i siste kodelinje.

```
class Info:
    def __init__(self, fornavn, etternavn):
        self._navn = fornavn + " " + etternavn

helt =
```

Maks poeng: 1

1(i)

```
1
2  tall = 1+2+3
3  tall = tall*2
4  tall = tall
5  # Fullfør tilordningen over og skriv hele linje 4 i svaret
6
7  print(tall)
```

Hvordan kan du fullføre linje 4 slik at denne koden skriver ut tallet 5? Skriv hele linje 4 i svarboksen (uten linenummer).

Maks poeng: 1

1(j) Hva må stå på høyresiden i første linje for at koden nedenfor skal skrive ut **IN1000**? Fyll ut svarboksen.

```
a= 
if not True:
    a = "I1"
a += "000"
print(a)
```

Maks poeng: 1

2(a) Hvilket uttrykk må fylles inn i svarboksen for at denne koden skal skrive ut en streng der annethvert tegn er et siffer (det vil si et tall fra 0 til 9) og hvert siffer kun forekommer en gang? Variabelen **streng** skal være med i uttrykket.

```
tall = 3
streng = "*" + "3"
for i in range(tall):
    streng = streng + "*"
    streng = 
print (streng)
```

Maks poeng: 2

2(b) Koden i bildet skal skrive ut en streng der annethvert tegn er en bokstav, og annethvert tegn er et siffer (det vil si et tall fra 0 til 9). Hvert siffer skal kun forekomme en gang.

```
1 t = 2
2 streng = "A" + str(t-1)
3 for c in [2, 2*t, 4*t]:
4     streng = streng + "A"
5     streng = ?
6
7 print(streng)
```

Hvilket uttrykk kan erstatte spørsmålstegnet i linje 5 i bildet for at dette skal bli riktig?

Skriv svaret her (kun høyresiden, ikke hele programsetningen):

Maks poeng: 2

2(c)

```
1 i = 0
2 t = "x"
3 while i<5 and len(t)<4:
4     i = i+1
5     t = t+"x"
6
7 assert i == ?
```

Fyll ut assert-setningen slik at programmet i bildet kjører feilfritt.

Skriv svaret her:

Maks poeng: 2

2(d)

```
1
2  def funk(a, b):
3      return a     # fullfør uttrykket slik at koden kjører uten feil
4
5  assert funk(1*2 + 3, 2) * 3 == 21
6  assert funk(1+2, 3) * 2 == 12
7  assert funk(2,2) + 3 == 7
8
9
10
```

Hva må denne funksjonen returnere på linje 3 for at dette programmet aldri skal feile?

Fullfør return-setningen i svarboksen (skriv kun det som mangler): **return a** [ ]

Maks poeng: 2

2(e)

```
1  ordbok = {'aa': [0, 0, 4, 1, 4, 1, 4, 1, 4, 5, 0, 0, 4, 4, 7, 8], \
2            'bb': [8, 6, 4, 2, 2, 0, 1, 8, 4, 3, 6, 2], \
3            'cc': [], \
4            'dd': [0, 8, 3, 6, 5, 0, 3, 7, 3, 6, 1, 7, 6, 6], \
5            'ee': [3, 0, 0, 7, 4, 2, 0, 8, 8, 5, 9, 6, 4], \
6            'ff': [4, 7, 4], \
7            'gg': [2, 2, 6, 8, 6, 0], \
8            'hh': [7, 1, 9, 3, 6, 5, 8, 6, 7, 3, 9, 7, 0], \
9            'ii': [], \
10           'jj': [0, 2, 1, 3, 5, 8, 4, 8, 8, 2, 9, 8, 5, 4, 4, 7, 8, 6], \
11           'kk': [4, 5, 8, 8, 4, 8, 9, 6, 4], \
12           'll': [9, 4, 3, 4, 2, 0, 2, 0, 4, 5], \
13           'mm': [3, 9, 2, 3, 9, 6, 0, 2]}
14
15
```

Hva er verdien til **svar** når koden nedenfor er utført? **ordbok** er initialisert som i bildet over.

ordbok["ff"].append(7)
svar = len(ordbok["ff"])

Skriv svaret her: [ ]

Maks poeng: 2

2(f)

```
liste = [1,1,-1,2,-2,3,-1,3,5,-2,3,1,-7,8,-9,3,2,-3,6,5,4,-2,3,2]
total=0
i = 0
while liste[i]>0:
    total += liste[i]
    i += 1
i=len(liste)-1
while liste[i]>0:
    total += liste[i]
    i -= 1
i = 0
while liste[i] > len(liste):
    total += liste[i]
    i += 1
print(total)

```

Hva skrives ut her?

Maks poeng: 2

2(g) *I denne oppgaven skal du vise at du kjenner til hva som skjer når vi bruker funksjonskall i argumentet til et annet funksjonskall.*

```
1  def funk1(a):
2      return a + 1
3
4  def funk2(b):
5      return 2 * b
6
7  def funk3(c):
8      return c + 2
9
10 def funk4(a):
11     return 4
12
13 def funk5(b):
14     return b - 1
15
16 def funk6(c):
17     return c
18
19 i = 1
20 k = 3
21 a = funk1( funk2(i+1) + funk3(k) - 1 )
22 b = funk4( funk5( funk6(2) ) + a )
23 c = funk4( funk1(a+b) + funk2(funk3(funk6(1))) + funk5(b*a) - funk6(b) )
24
25 print(str(a)+','+str(b)+','+str(c))
26
```

Hva blir skrevet ut av koden i bildet?

Skriv svaret her (nøyaktig slik det skrives ut på terminalen):

Maks poeng: 2

2(h)

```
matrise = [ [60, 28, 77, 23, 84, 10, 10, 73, 83, 35, 15, 10, 22, 18, 88],  \
            [92, 94, 31, 55, 63, 85, 62, 85, 10, 80, 54, 96, 80, 16, 68],  \
            [56, 53, 91, 25, 69, 18, 94, 85, 11, 86, 98, 70, 64, 44, 42],  \
            [11, 59, 44, 29, 36, 44, 20, 78, 99, 21, 53, 22, 24, 73, 64],  \
            [18, 11, 75, 82, 27, 52, 69, 76, 61, 50, 91, 24, 49, 82, 20],  \
            [67, 39, 27, 14, 49, 39, 63, 60, 78, 55, 71, 33, 15, 27, 82],  \
            [25, 22, 70, 32, 47, 32, 97, 19, 56, 24, 59, 25, 63, 25, 65],  \
            [88, 23, 34, 34, 91, 18, 28, 73, 40, 58, 95, 86, 59, 75, 86],  \
            [90, 86, 56, 94, 35, 83, 11, 64, 62, 64, 66, 75, 77, 62, 80] ]
```

Hvilken verdi evaluerer uttrykket **matrise[0][2] + matrise[7][1]** til?

Maks poeng: 2

2(i)

```
def kalkulator(operasjoner): # operasjoner må ha type str
    tall = 0
    for operasjon in operasjoner:
        if operasjon == "d":
            tall = tall * 2
        elif operasjon == "1":
            tall = tall + 1
        elif operasjon == "2":
            tall = tall + 2
        elif operasjon == "3":
            tall = tall + 3
        elif operasjon == "4":
            tall = tall + 4
        elif operasjon == "5":
            tall = tall + 5
        elif operasjon == "6":
            tall = tall + 6
        elif operasjon == "7":
            tall = tall + 7
        elif operasjon == "8":
            tall = tall + 8
        else:
            tall = tall
    return tall
```

Hva blir returnert hvis vi kaller denne funksjonen med argumentet **"1d7d8"** ?

kalkulator("1d7d8") returnerer verdien

Maks poeng: 2

2(j)

Hvor mange Ukedag-objekter eksisterer (kan man få tak i) etter at koden på bildet har kjørt?

```
1  class Ukedag:
2      def __init__(self, dag):
3          self.dag = dag
4
5  torsdag = Ukedag(4)
6  imorgen = Ukedag(4+1)
7  fredag = imorgen
8  helg = Ukedag(6)
9  fredag = None
10 helg = None
11 lordag = helg
12
```

Skriv svaret her

Maks poeng: 2

2(k) Hvor mange bil-objekter eksisterer (kan man få tak i) etter at koden på bildet har kjørt?

```
1  class Bil:
2      def __init__(self, merke):
3          self._merke = merke
4
5  opel = Bil("Opel")
6  toyota = Bil("Toyota")
7  jaguar = Bil("Jaguar")
8  jaguar = opel
9  opel = toyota
10 toyota = None
11 elbil = opel
12
```

Skriv svaret her:

Maks poeng: 2

2(I)

```python
def kalkulator(operasjoner): # operasjoner må ha type str
    tall = 0
    for operasjon in operasjoner:
        if operasjon == "d":
            tall = tall * 2
        elif operasjon == "1":
            tall = tall + 1
        elif operasjon == "2":
            tall = tall + 2
        elif operasjon == "3":
            tall = tall + 3
        elif operasjon == "4":
            tall = tall + 4
        elif operasjon == "5":
            tall = tall + 5
        elif operasjon == "6":
            tall = tall + 6
        elif operasjon == "7":
            tall = tall + 7
        elif operasjon == "8":
            tall = tall + 8
        else:
            tall = tall
    return tall

```

Finn det korteste (lengden av strengen) argumentet til funksjonen kalkulator slik at den returnerer verdien **23**.

Skriv verdien til argumentet her: " " "

Maks poeng: 2

3(a) Skriv en funksjon **fridager(julaften)** som returnerer antall fridager fra og med 25. desember, til og med 31. desember. Det er alltid fri lørdag og søndag, dessuten 25. og 26. desember. Om noen av disse faller på samme dag blir det færre enn 4 dager fri. Julaften er 24. desember.

Parameteren **julaften** skal inneholde en streng som angir hvilken ukedag julaften faller på. Det vil si at kallet **fridager("sondag")** skal evaluere til **4,** mens **fridager("fredag")** skal evaluere til **2.**

Skriv funksjonen her:

```
1
```

Maks poeng: 6

3(b) Skriv en funksjon **beOmNavn(navneliste)** med en liste av strenger som parameter. Funksjonen skal be bruker oppgi navn inntil navnet som oppgis finnes i **navneliste**, da returneres dette navnet. Navnelisten skal ikke skrives ut. Du kan anta at alle navn i listen og som oppgis av bruker kun inneholder små bokstaver (lowercase).

Skriv ditt svar her

```
1
```

Maks poeng: 5

3(c) Funksjonen *stringFraInt* i koden nedenfor returnerer en tekst med 6 tegn ved å fylle inn '0'-tegn foran heltallet i parameteren tall.

Skriv en ny funksjon **intTilString(tall, antallSiffer)**, som returnerer en tekst med en lengde som er gitt av parameteren **antallSiffer**. Dersom lengden (antall siffer) av parameteren **tall** er mindre enn **antallSiffer**, så skal funksjonen fylle på '0'-tegn foran heltallet i parameteren **tall**, slik som det er gjort i *stringFraInt*.

```
def stringFraInt(tall):
    tall=str(tall)
    returtall=tall
    if len(tall) == 1:
        returtall = "00000"+tall
    elif len(tall) == 2:
        returtall = "0000"+tall
    elif len(tall) == 3:
        returtall = "000"+tall
    elif len(tall) == 4:
        returtall = "00"+tall
    elif len(tall) == 5:
        returtall = "0"+tall
    return returtall


def intTilString(tall, antallSiffer): # Svaret skal være hele denne funksjonen
#
assert stringFraInt(87) == "000087"
assert intTilString(87, 4) == "0087"
assert intTilString(87, 2) == "87"
assert intTilString(87, 1) == "87"
assert intTilString(1,11) == "00000000001"
assert intTilString(0, 2) == "00"
assert intTilString(0, 0) == "0"
print("alt ok")
```

Skriv ditt svar her

```
1
```

Maks poeng: 6

3(d) Mange irriterer seg over overdrevet bruk av utropstegn. Skriv en funksjon **dempDeg (tekst)** som returnerer en kopi av parameteren **tekst**, men der alle utropstegn ("!") bakerst i teksten er fjernet (om det er noen) og erstattet av et punktum (".").

For eksempel skal kallet **dempDeg("Hei!!!")** returnere strengen "Hei.", mens kallet **dempDeg(" Hei du! Og du.")** skal returnere strengen " Hei du! Og du." .

Skriv ditt svar her

```
1
```

Maks poeng: 7

4 I denne oppgaven skal du skrive et større objektorientert system. Du skal skrive løsning til alle deloppgaver i den vedlagte PDF-filen i svar-feltet til denne oppgaven. Marker gjerne hvilken deloppgave du har løst underveis som kommentarer.

Du finner også PDF'en HER (klikk på "HER") , dersom du ønsker å åpne den i eget vindu.

Du står fritt til å innføre egne variabler, metoder og klasser utover de som er beskrevet i oppgaveteksten. Husk å unngå særnorske tegn og bokstaver i koden din.

Skriv ditt svar her

```
1
```

Maks poeng: 55

5 Du har kommet over ei liste over synonymer (ord som betyr omtrent det samme). Lista er ei liste av lister, hvor ei av synonymlistene f.eks. kan være:

["godt", "bra", "flott", "ok", "fint", "vel"]

Du ønsker å lage en papirversjon av synonymordboka slik at man kan slå opp på et gitt ord for å finne alle ordets synonymer. Du skal lage en funksjon som returnerer ei Python-ordbok (dict) som hjelper deg i dette arbeidet:

lagSynonymordbok(listeAvLister)

Vær oppmerksom på homonymer. Det er ord som skrives likt, men betyr forskjellige ting. F.eks. kan ordet *gift* både være en relasjon mellom to personer, eller betegne noe som er farlig å få i seg. For å komme rundt dette problemet lages synonymorboka slik at den inneholder ei synonymliste *for hver betydning av et gitt ord* der ordet er et homonym. Dette betyr at ordlista har en streng som nøkkel og ei liste av lister av strenger som verdi. Et eksempel:

synonymordbok["blad"] = [["løv"], ["magasin"]]

fordi ordet *blad* både kan være en del av en plante, men også noe man kan kjøpe i en bladkiosk. Funksjonaliteten framgår forøvrig av testprogrammet nedenfor.

```
17
18  synonymer = [
19                ["a", "e", "i", "o", "u"], \
20                ["HOM", "c", "d"], \
21                ["y", "HOM"], \
22                ["k", "l", "m", "n", "p", "q"], \
23                ["x"]
24              ]
25  # Ordet "HOM" (homonym) har i én betydning synonymene "c" og "d",
26  # i en annen betydning har "HOM" synonymet "y"
27
28  synonymordbok = lagSynonymordbok(synonymer)
29
30  assert synonymordbok["e"]   ==  [ ["a", "i", "o", "u"] ]
31  assert synonymordbok["a"]   ==  [ ["e", "i", "o", "u"] ]
32  assert synonymordbok["u"]   ==  [ ["a", "e", "i", "o"] ]
33  assert synonymordbok["c"]   ==  [ ["HOM", "d"] ]
34  assert synonymordbok["HOM"] ==  [ ["c", "d"], ["y"] ]
35  assert synonymordbok["x"]   ==  [ [] ]
36
37  print("Alt ok")
```

Skriv hele funksjonen lagSynonymordbok:

```
1
```

Maks poeng: 5

Question 27
Attached

Deltakelse på undervisningsaktiviteter ved Ifi

Oppgaven som PDF finner du også på semestersiden til IN1000

Et vanlig problem for mange emner, er at oppmøte til grupper og aktiviteter endrer seg i løpet av semesteret. Noen grupper får flere studenter enn planlagt, mens andre grupper får færre. I denne oppgaven skal vi lage et system som hjelper undervisere med å undersøke oppmøtet i de ulike gruppene og aktivitetene for emnene vi har på Ifi. På den måten kan undervisningen planlegges bedre.

Gjennom hele oppgaven står du fritt til å innføre egne metoder, variabler og eventuelt klasser, der du selv ser behovet for dette slik du løser oppgaven.

Oppgave A (2 poeng)

Skriv klassen Student. Ved opprettelse av objekter av klassen, skal det opprettes en tom liste over emner studenten følger. Konstruktøren skal ta inn studentens brukernavn, og lagre dette i en instansvariabel.

Du skal skrive én metode i klassen, *hentBrukernavn(self)* som skal returnere studentens brukernavn.

Oppgave B (4 poeng)

Du skal også skrive klassen Emne. Ved opprettelse av objekter av klassen, skal en tom liste over aktiviteter (grupper o.a.) tilknyttet emnet opprettes. Konstruktøren skal ta inn kode på emnet, og lagre dette i en instansvariabel.

Klassen skal ha én metode. Denne metoden skal legge en aktivitet til listen over aktiviteter.

Oppgave C (12 poeng)

Du skal opprette en klasse Dato. Denne klassen skal ta tre heltall som argumenter i sin konstruktør: *dag*, *maaned* og *aar*. År skal representeres ved to siffer; 2020 vil for eksempel representeres som 20. Du kan anta at programmet kun skal brukes etter år 2000 og før år 2100. Klassen skal ha følgende metoder:

- *absoluttDato(self)*, som returnerer datoen som ett heltall, der rekkefølgen blir år, måned og dag. Dette sikrer at man alltid kan hente ut datoer kronologisk. For eksempel vil 19. november 2020 representeres som 201119. Merk at du kan trenge å legge til ekstra nuller for å få tallet på seks siffer.

- *__str__(self)*, som returnerer datoen på et pent og leselig format, som en streng. 19/11-20 vil kunne returneres som “19. november 2020”. Du kan anta at programmet kun brukes i høstsemesteret, altså trengs bare september-desember.

Oppgave D (12 poeng)

Klassen Aktivitet representerer en aktivitet knyttet til et emne. Dette er typisk en gruppetime eller lab-øvelse.

Du skal skrive klassen Aktivitet. Ved opprettelse skal konstruktøren ta følgende parametre:
- *emne*: et objekt av klassen Emne, som sier hvilket emne aktiviteten er tilknyttet.
- *dato*: et objekt av klassen Dato, riktig dato for aktiviteten.
- *aktivitetsnummer*: et heltall som indikerer hvilket nummer aktiviteten har.

Konstruktøren skal også opprette to tomme lister, den ene for studenter som er registrert på aktiviteten i StudentWeb, og den andre for studenter som faktisk har møtt opp til denne aktiviteten (oppmøteregistrering).

Klassen skal inneholde følgende metoder:
- *leggTilRegistrertStudent(self, student)*: legger et student-objekt til listen over registrerte studenter.
- *registrerOppmote(self, student)*: studenten har møtt opp til aktiviteten, og objektet legges til listen for oppmøtte studenter.
- *skrivUtOppmotteStudenter(self)*: skriver ut navnet på alle studenter som har møtt til aktiviteten.
- *absoluttDato(self)*: returnerer dato for aktivitet på absolutt-format, som beskrevet i Dato-klassen.
- *oppmote(self)*: returnerer antall oppmøtte studenter til aktiviteten.
- *__str__(self)*: returnerer en streng som inneholder aktivitetens emne, nummer og antall oppmøtte til aktiviteten.

Oppgave E (3 poeng)

Lag en klasse Undervisningsadministrasjon. Klassen skal ikke ta noen argumenter til sin konstruktør. Ved opprettelse av objekter, skal følgende instansvariabler opprettes:
- *emner*: en ordbok, der en streng med emnekode vil være nøkkel, og objekter av Emne skal være verdi.
- *datoer*: en ordbok, dato på absolutt format (som er lik returverdien til *absoluttDato*) skal være nøkkel, mens Dato-objekter er innholdsverdi.
- *studenter*: en ordbok over studenter. Brukernavn er nøkkel, Student-objekt er verdi.

Oppgave F (10 poeng)

Lag en metode i Undervisningsadministrasjon som leser inn filer og oppretter emner og deres aktiviteter. Merk at det kun skal være ett dato-objekt for hver dato. Det skal også kun være ett objekt per emne. Filen som skal leses vil være på følgende format (eksempel):

IN1000 Gruppe1 20 09 18
IN1000 Gruppe2 20 09 22
IN1000 Gruppe3 20 10 08
IN5090 Lab1 21 01 12

Obs: husk at hvert emne kun skal representeres som ett objekt, og at aktiviteter legges til dette objektet. Objektene legges til sine respektive ordbøker.

Oppgave G (7 poeng)

Lag en metode i Undervisningsadministrasjon som leser informasjon fra fil om studenter, deres emner og hvilke aktiviteter de er registrert for. Dataene du leser, skal legges inn i datastrukturen, og Student-objekter skal opprettes for hver enkelt student. Et eksempel kan være som følger:

henrihlo IN1000 Gruppe1
henrihlo IN1010 Gruppe4
siriamj IN2010 Gruppe3
siriamj IN2010 Lab1

Du kan anta at emnet eksisterer i ordboken over emner, og at gruppe/aktivitet eksisterer i emnenes lister over aktiviteter.

Oppgave H (5 poeng)

Skriv en metode i Undervisningsadministrasjon *skrivGrupperMedLavtOppmoete(self, antall)* som skriver ut en liste over de gruppene som har hatt færre enn et antall oppmøtte studenter gitt som parameter.

Question 28

Attached

Deltakelse på undervisningsaktiviteter ved Ifi

Oppgaven som PDF finner du også på semestersiden til IN1000

Et vanlig problem for mange emner, er at oppmøte til grupper og aktiviteter endrer seg i løpet av semesteret. Noen grupper får flere studenter enn planlagt, mens andre grupper får færre. I denne oppgaven skal vi lage et system som hjelper undervisere med å undersøke oppmøtet i de ulike gruppene og aktivitetene for emnene vi har på Ifi. På den måten kan undervisningen planlegges bedre.

Gjennom hele oppgaven står du fritt til å innføre egne metoder, variabler og eventuelt klasser, der du selv ser behovet for dette slik du løser oppgaven.

Oppgave A (2 poeng)

Skriv klassen Student. Ved opprettelse av objekter av klassen, skal det opprettes en tom liste over emner studenten følger. Konstruktøren skal ta inn studentens brukernavn, og lagre dette i en instansvariabel.

Du skal skrive én metode i klassen, *hentBrukernavn(self)* som skal returnere studentens brukernavn.

Oppgave B (4 poeng)

Du skal også skrive klassen Emne. Ved opprettelse av objekter av klassen, skal en tom liste over aktiviteter (grupper o.a.) tilknyttet emnet opprettes. Konstruktøren skal ta inn kode på emnet, og lagre dette i en instansvariabel.

Klassen skal ha én metode. Denne metoden skal legge en aktivitet til listen over aktiviteter.

Oppgave C (12 poeng)

Du skal opprette en klasse Dato. Denne klassen skal ta tre heltall som argumenter i sin konstruktør: *dag*, *maaned* og *aar*. År skal representeres ved to siffer; 2020 vil for eksempel representeres som 20. Du kan anta at programmet kun skal brukes etter år 2000 og før år 2100. Klassen skal ha følgende metoder:

- *absoluttDato(self)*, som returnerer datoen som ett heltall, der rekkefølgen blir måned, år og dag. Dette sikrer at man alltid kan hente ut datoer kronologisk ut fra hvilken måned aktiviteten skal være i. For eksempel vil 19. november 2020 representeres som 111920. Merk at du kan trenge å legge til ekstra nuller for å få tallet på seks siffer.

- *__str__(self)*, som returnerer datoen på et pent og leselig format, som en streng. 19/11-20 vil kunne returneres som “19. november 2020”. Du kan anta at programmet kun brukes i høstsemesteret, altså trengs bare september-desember.

Oppgave D (12 poeng)

Klassen Aktivitet representerer en aktivitet knyttet til et emne. Dette er typisk en gruppetime eller lab-øvelse.

Du skal skrive klassen Aktivitet. Ved opprettelse skal konstruktøren ta følgende parametre:
- *emne*: et objekt av klassen Emne, som sier hvilket emne aktiviteten er tilknyttet.
- *dato*: et objekt av klassen Dato, riktig dato for aktiviteten.
- *aktivitetsnummer*: et heltall som indikerer hvilket nummer aktiviteten har.

Konstruktøren skal også opprette to tomme lister, den ene for studenter som er registrert på aktiviteten i StudentWeb, og den andre for studenter som faktisk har møtt opp til denne aktiviteten (oppmøteregistrering).

Klassen skal inneholde følgende metoder:
- *leggTilRegistrertStudent(self, student)*: legger et student-objekt til listen over registrerte studenter.
- *registrerOppmote(self, student)*: studenten har møtt opp til aktiviteten, og objektet legges til listen for oppmøtte studenter.
- *skrivUtOppmotteStudenter(self)*: skriver ut navnet på alle studenter som har møtt til aktiviteten.
- *absoluttDato(self)*: returnerer dato for aktivitet på absolutt-format, som beskrevet i Dato-klassen.
- *oppmote(self)*: returnerer antall oppmøtte studenter til aktiviteten.
- *__str__(self)*: returnerer en streng som inneholder aktivitetens emne, nummer og antall oppmøtte til aktiviteten.

Oppgave E (3 poeng)

Lag en klasse Undervisningsadministrasjon. Klassen skal ikke ta noen argumenter til sin konstruktør. Ved opprettelse av objekter, skal følgende instansvariabler opprettes:
- *emner*: en ordbok, der en streng med emnekode vil være nøkkel, og objekter av Emne skal være verdi.
- *datoer*: en ordbok, dato på absolutt format (som er lik returverdien til *absoluttDato*) skal være nøkkel, mens Dato-objekter er innholdsverdi.
- *studenter*: en ordbok over studenter. Brukernavn er nøkkel, Student-objekt er verdi.

Oppgave F (10 poeng)

Lag en metode i Undervisningsadministrasjon som leser inn filer og oppretter emner og deres aktiviteter. Merk at det kun skal være ett dato-objekt for hver dato. Det skal også kun være ett objekt per emne. Filen som skal leses vil være på følgende format (eksempel):

IN1000 Gruppe1 20 09 18
IN1000 Gruppe2 20 09 22
IN1000 Gruppe3 20 10 08
IN5090 Lab1 21 01 12

Obs: husk at hvert emne kun skal representeres som ett objekt, og at aktiviteter legges til dette objektet. Objektene legges til sine respektive ordbøker.

Oppgave G (7 poeng)

Lag en metode i Undervisningsadministrasjon som leser informasjon fra fil om studenter, deres emner og hvilke aktiviteter de er registrert for. Dataene du leser, skal legges inn i datastrukturen, og Student-objekter skal opprettes for hver enkelt student. Et eksempel kan være som følger:

henrihlo IN1000 Gruppe1
henrihlo IN1010 Gruppe4
siriamj IN2010 Gruppe3
siriamj IN2010 Lab1

Du kan anta at emnet eksisterer i ordboken over emner, og at gruppe/aktivitet eksisterer i emnenes lister over aktiviteter.

Oppgave H (5 poeng)

Skriv en metode i Undervisningsadministrasjon *skrivGrupperMedHoytOppmoete(self, antall)* som skriver ut en liste over de gruppene som har hatt flere enn et antall oppmøtte studenter gitt som parameter.