Informasjon

Prøveeksamen i IN1000 høsten 2018

Tid

Fra tirsdag 6.11 kl. 14:15 til tirsdag 13.11 kl. 12:00
(Normal eksamenstid er 4 timer)

Oppgavene

Oppgave 2b og 2c er flervalgsoppgaver. Her får man det angitte antall poeng om man svarer riktig; ved galt svar eller intet svar får man 0 poeng.

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 du kan benytte funksjoner eller metoder fra oppgaven selv om du ikke har skrevet dem.

Tillatte hjelpemidler

Alle trykte og skrevne hjelpemidler

1

1a) 1 poeng

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

```
tall = 3+1
tall = tall*2
```

Maks poeng: 1

2

1b) 1 poeng

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

```
tekst = "a" + "c"
tekst = tekst + "b"
```

Maks poeng: 1

3

2 poeng

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

```
i = 5
j = 10
while i<j:
    i = i+2
    j = j-1
```

Maks poeng: 2

4 1d) 2 poeng

Hva skrives ut på skjermen når følgende kode utføres?

```
tallene = [1,6,4,2]
a = 0
b = 0
for tall in tallene:
    if tall<3:
        a = a+tall
    else:
        b = b+tall
print(a*b)
```

Maks poeng: 2

5 1e) 2 poeng

VI har en funksjon kalkuler som vist nedenfor:

```
def kalkuler(tall):
    if tall<5:
        return tall*2
    else:
        return tall
```

Hva skrives ut på skjermen når følgende kode utføres?

```
a = kalkuler(4+3)
print(a)
```

Maks poeng: 2

6 1f) 3 poeng

Hva skrives ut på skjermen når følgende kode utføres?

```
class Tall:
    def __init__(self, a, b):
        self._a = a
        self._b = b
    def m1(self, c):
        self._b = self._b + c
    def m2(self):
        self._a = self._a + self._b
    def m3(self):
```

```
        return 2*self._a
t = Tall(3,2)
t.m1(1)
t.m2()
t.m2()
print(4+t.m3())
```

Maks poeng: 3

7 2a) 1 poeng

Hva skrives ut på skjermen når følgende kode utføres?

```
a = [5, 2, 4]
b = a[0]
b = b + 1
print(a[0])
```

Maks poeng: 1

8 2b) 2 poeng

Hva skrives ut på skjemen når følgende kode utføres?

```
a = [10, 8, 7]
b = a
c = b
b[0] = 3
c[1] = 4
print(a)
```

Velg ett alternativ

- [3,4]
- [3,8,7]
- [10,8,7]
- [3,4,7]
- [10,4,7]

Maks poeng: 2

9 2c) 3 poeng

Hvilke fire tall skrives ut på skjermen når følgende kode kjøres?

```
class Person:
  def __init__(self, alder):
     self._alder = alder
  def doble_alder(self):
```

```
            self._alder = self._alder * 2
        def hent_alder(self):
            return self._alder

        def alder_som_maaneder(self):
            return self._alder * 12
p1 = Person(3)
p2 = p1
p3 = Person(5)
p4 = p3
p1.doble_alder()
print(p1.hent_alder())
print(p2.hent_alder())
print(p3.alder_som_maaneder())
print(p4.hent_alder())
```

Velg ett alternativ

- 3,3,60,5
- 6,3,60,60
- 6,3,60,5
- 6,6,60,60
- 6,6,60,5

Maks poeng: 3

10

3a) 5 poeng

Skriv en funksjon **penger(femkroninger, kronestykker)** som tar som argumenter et gitt antall femkroninger og kronestykker, og returnerer hvor mye penger dette tilsvarer. For eksempel skal kallet *penger(2,3)* returnere 13.

Skriv ditt svar her...

```
1
```

Maks poeng: 5

11

3b) 5 poeng

Et togselskap har regel om at et barn kan reise gratis sammen med en voksen. Skriv en funksjon barnMedVoksen(alder1, alder2) som returnerer True dersom én av parameterverdiene er over (eller lik) 18 og den andre er under 18. Det er det samme hvilke av de to oppgitte aldrene som er voksen eller barn. Altså skal for eksempel kallet barnMedVoksen(18,5) returnere True, og det samme skal kallet barnMedVoksen(10,20). Videre skal for eksempel kallet barnMedVoksen(20,30) returnere False.

Skriv ditt svar her...

```
1
```

Maks poeng: 5

12

3c) 8 poeng

```
def allePositive(tallene) :
    for tall in tallene :
        if (tall < 0) :
            return False
    return True
```

Funksjonen **allePositive** ovenfor er ment å kontrollere at alle verdiene i listen **tallene** er positive. I denne oppgaven ønsker vi at du skal:

1) beskrive hva som er problemet med løsningen over

2) gi et eksempel på en liste som ikke gir resultat i henhold til intensjonen hvis den brukes som argument i et kall på funksjonen

3) foreslå endring som gjør at funksjonen oppfører seg i henhold til intensjonen

Skriv ditt svar her...

Format

Words: 0

Maks poeng: 8

13 # 3d) 8 poeng

Skriv en funksjon fyllTilTi(tallene) som tar inn en liste med opptil 10 tall (lengden på listen kan variere, men er maksimalt 10 lang). Funksjonen skal returnere en liste som er nøyaktig 10 lang, hvor alle tall fra listen *tallene* kommer først, og hvor det etterpå er lagt til så mange verdier 0 som trengs for at listen skal bli nøyaktig 10 lang.

Skriv ditt svar her...

```
1
```

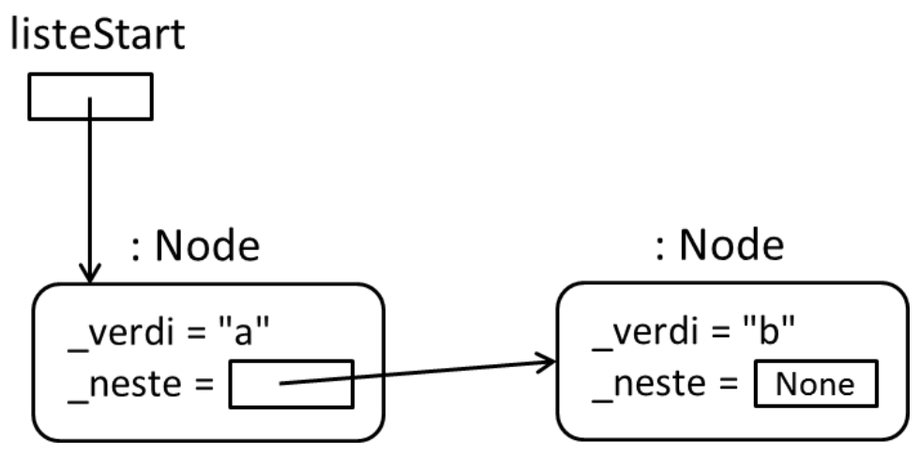
Maks poeng: 8

14 # 3e) 4 poeng

```
class Node :
    def __init__ (self, verdi) :
        self._verdi = verdi
        self._neste = None

    def settInn (self, ny) :
        self._neste = ny

    # ...flere metoder vi ikke trenger..
```



Gitt klassen **Node** som vist, skriv en prosedyre som oppretter 2 objekter med verdiene "a" og "b" i en struktur som vist i figuren. I figuren er variabler vist som en firkantet boks, piler angir objektreferanser, og objektene er vist som rektangler med avrundede hjørne.

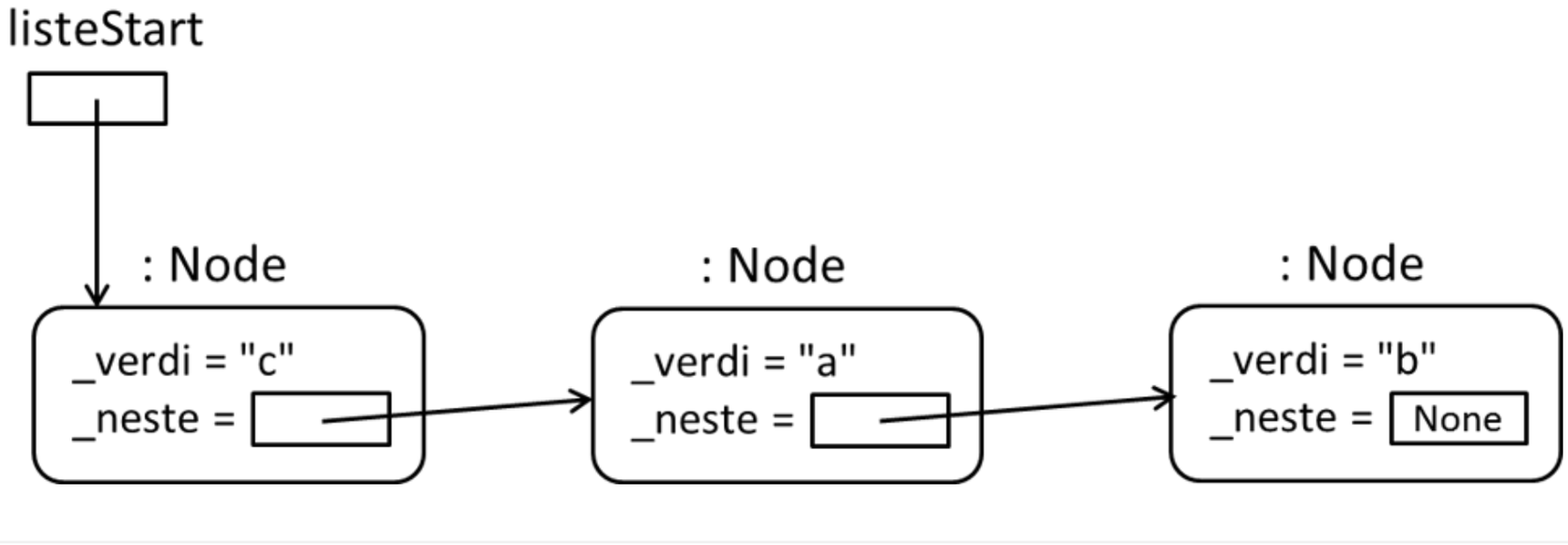
Variabelen listeStart refererer til objektet med verdi "a". Du kan anta at klassen **Node** er importert til programmet ditt.

Skriv ditt svar her...

```
1
```

Maks poeng: 4

15 3f) 3 poeng



Utvid **hovedprogrammet** fra oppgave e) med kode som oppretter et nytt objekt med verdi «c» og og oppdaterer referanser slik at du får en datastruktur som vist i vedlagte figur.

Klassen **Node** er uendret, og objekter av klassen skal kun aksesseres ved hjelp av de oppgitte metodene.

Skriv ditt svar her...

```
1
```

Maks poeng: 3

16 # 4a) 8 poeng

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave). I hver av deloppgavene kan du bruke klassene og metodene fra tidligere deloppgaver, også om du ikke har programmert dem.

Skriv klassen **Hytte** med metoder som angitt i vedlegget, deloppgave a)

Skriv ditt svar her...

Maks poeng: 8

17 **4b) 10 poeng**

Skriv klassen Tur som beskrevet i vedlegget, deloppgave b)

Skriv ditt svar her...

```
1
```

Maks poeng: 10

18 **4c) 10 poeng + 4d) 6 poeng**

Skriv klassen Turplanlegger som beskrevet i vedlegget, deloppgaver c) og d).

Skriv ditt svar her...

```
1
```

Maks poeng: 16

19 **4e) 6 poeng**

Skriv et testprogram som beskrevet i vedlegget, deloppgave e)

Skriv ditt svar her...

```
1
```

Maks poeng: 6

20 **5a) og b) 10 poeng**

I en bedrift har hver ansatt en telefon som kunder kan kontakte dem på. Når en ansatt tar lunsj, setter han/hun opp viderekobling til en annen ansatt. Denne kan igjen ha satt opp viderekobling, noe som gjør at en kunde kan bli viderekoblet flere ganger før han/hun får svar. Dette kan vi representere som en liste hvor hver ansatt tilsvarer en bestemt indeks fra 0 og oppover. Om en ansatt er tilgjengelig for å snakke ligger verdien -1 på dennes lokasjon i lista. Om en ansatt har viderekoblet legges i stedet inn indeksen det er viderekoblet til. F.eks. vil lista [2, -1, -1, 0] representere at man har fire ansatte, hvor ansatt 0 viderekobler til ansatt 2, ansatt 1 og ansatt 2 er tilgjengelig for å snakke, mens ansatt 3 viderekobler til ansatt 0.

a) Skriv en funksjon **ring** som tar inn en viderekoblingsliste (som beskrevet over) og et ansattnummer som kunden opprinnelig ringer til. Funksjonen skal returnere hvilket ansattnummer kunden ender opp med å snakke med. Anta i denne deloppgaven at dette er garantert å gå i orden (at man ikke kan ende opp med å bli viderekoblet i en evig løkke).
F.eks. skal kallet *ring(1, [2, -1, -1, 0])* returnere 1 (ansatt 1 er direkte tilgjengelig), mens kallet *ring(3, [2, -1, -1, 0])* skal returnere 2 (man blir viderekoblet fra ansatt 3 til ansatt 0 til ansatt 2).

b) Skriv en funksjon **gyldig** som tar inn en viderekoblingsliste og sjekker om det finnes mulighet for å havne i en evig løkke av viderekobling i denne listen. Et eksempel er lista [1,-1, 3,4,2] hvor man ved å ringe ansatt 2 blir viderekoblet til ansatt 3, videre til ansatt 4 og så tilbake igjen til ansatt 2 i en evig løkke. Dersom det ikke finnes noen slik evig løkke av viderekobling i lista skal funksjonen returnere **true**. Dersom det finnes minst én slik løkke av viderekobling skal den returnere False. Altså skal kallet *gyldig([2, -1, -1, 0])* returnere **true**, mens kallet *gyldig([1,-1, 3,4,2])* skal returnere **false**.

Skriv ditt svar her...

```
1
```

Maks poeng: 10

Question 16
Attached

Oppgave 4: Planlegging av turer fra hytte til hytte (40 poeng)

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave). I hver av deloppgavene kan du bruke klassene og metodene fra tidligere deloppgaver, også om du ikke har programmert dem.

Du skal implementere deler av et system som organiserer en rekke turer. Hver tur går over flere dager med overnatting på en ny hytte hver dag. Den delen av systemet du trenger å kjenne til består av klassene **Hytte**, **Tur** og **Turplanlegger**. Det anbefales å lage en tegning av strukturen for eget bruk.

Klassen **Turplanlegger** holder rede på alle hytter i systemet ved hjelp av en ordbok (Dictionary) med hyttenavnet som nøkkel og referanse til hytta som verdi. Klassen **Turplanlegger** har også en liste med referanser til turer.

I klassen **Tur** finnes det en tekst (som kan inneholde blanke, men ikke linjeskift) som beskriver turen, og en liste med referanser til de hyttene som besøkes i løpet av turen (som alle ligger i **Turplanleggers** ordbok).

Hvert objekt av klassen **Hytte** har et unikt navn, et antall sengeplasser og prisen for overnatting per seng i denne hytta (alltid lik pris for hver seng i en hytte).

Oppgave a) (8 poeng)

Skriv klassen **Hytte.** Konstruktøren skal ha parametere for instansvariablene navn, antall senger og pris for overnatting. Videre skal klassen ha følgende metoder:

hentNavn som returnerer hyttenavnet.

totPris som returnerer prisen for et antall personer (antallet er parameter til metoden) for én natt.

sjekkPlass som returnerer **True** om hytta har nok senger til et antall personer (parameter til metoden), ellers **False**.

__str__ som returnerer en bruker-vennlig streng med hyttas navn, antall senger og pris per seng.

__eq__ som sammenligner objektet den kalles på med et annet objekt (referansen er parameter til metoden). To **Hytte**-objekter er like hvis de har samme navn, da returneres **True**, ellers **False**.

Oppgave b) 10 poeng

Skriv klassen **Tur** med en konstruktør som tar imot en liste med referanser til hytter, og en (en-linjes) tekst som beskriver turen. Videre skal klassen ha følgende metoder:

skrivTur som skriver ut på terminal beskrivelsen av turen, og deretter informasjon om hver hytte turen går innom.

sjekkPrisPlass som går gjennom alle hyttene på turen og sjekker om det er nok senger for et antall personer på hver av hyttene, og om totalprisen for turen (alle personer på alle hyttene) er under et maksbeløp. Antall personer og maksbeløp er parametere til metoden. Metoden returnerer **True** om det er nok plasser og kostnadene blir under maksbeløpet, ellers **False**.

Oppgave c) 10 poeng

Du skal skrive klassen **Turplanlegger** med konstruktør og metodene **_hytterFraFil** og **_turerFraFil**. Metodene kalles fra konstruktøren og leser inn data til henholdsvis ordboken med hytter, og listen med turer. Filnavnene er parametere til konstruktøren og til hver sin metode.

Metoden **_hytterFraFil** åpner og leser fra filnavnet gitt i parameter til metoden. Filen har et ukjent antall linjer på følgende format (to hytter vist)

```
Hyttenavn antSenger pris
Hyttenavn antSenger pris
:
```

der første streng skal leses som en tekst og de to siste strengene som heltall og flyttall. Hver hytte representeres i et objekt av klassen Hytte, som lagres i en ordbok i med hyttenavnene som nøkler og referanse til objektet som verdier. Denne ordboken returneres fra metoden når filen er ferdig lest.

Metoden **_turerFraFil** åpner og leser all informasjon om alle turene i systemet fra filen med navn oppgitt i parameteren, og oppretter og returnerer en liste over **Tur**-objekter. Denne filen inneholder to linjer per tur. Første linje inneholder turbeskrivelsen (typisk flere ord), andre linje inneholder navnene på alle hytter på turen. Ingen hyttenavn inneholder mellomrom (space). Samme hytte kan inngå i flere turer. Filen har følgende format:

```
En tekst som beskriver en tur
Hyttenavn1 Hyttenavn2 Hyttenavn3
En tekst som beskriver en annen tur
Hyttenavn1 Hyttenavn4
:
```

Oppgave d) 6 poeng

Skriv en metode **finnTurer** i klassen **Turplanlegger** som kan brukes til å identifisere turer som har plass til et gitt antall personer på alle hyttene, der totalkostnad for alle personer alle dager er under et maksimalbeløp, og som ikke varer mer enn et maks antall dager (turene går alltid til en ny hytte hver dag). Metoden skal gå gjennom alle turer, og skrive ut på terminalen all informasjon om de turene som tilfredsstiller kravene. Antall personer, maksbeløp og maksimalt antall dager er parametere til metoden. Du kan bruke metoder fra tidligere oppgaver uansett om du har skrevet dem eller ikke.

Oppgave e) 6 poeng

Skriv et testprogram for klassen **Turplanlegger** som gjør følgende:

- Oppretter et objekt av klassen Turplanlegger som leser inn data fra filene *hytter.txt* og *turer.txt*
- Skriver ut all informasjon om alle turer som tilfredsstiller følgende krav:
    - ikke varer mer enn 5 dager
    - har overnattingsplass på alle hyttene for et følge på 7 personer, ..
    - ..med et overnattingsbudsjett på til sammen 7000,- for alle 7, hele turen.

Question 17
Attached

Oppgave 4: Planlegging av turer fra hytte til hytte (40 poeng)

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave). I hver av deloppgavene kan du bruke klassene og metodene fra tidligere deloppgaver, også om du ikke har programmert dem.

Du skal implementere deler av et system som organiserer en rekke turer. Hver tur går over flere dager med overnatting på en ny hytte hver dag. Den delen av systemet du trenger å kjenne til består av klassene **Hytte**, **Tur** og **Turplanlegger**. Det anbefales å lage en tegning av strukturen for eget bruk.

Klassen **Turplanlegger** holder rede på alle hytter i systemet ved hjelp av en ordbok (Dictionary) med hyttenavnet som nøkkel og referanse til hytta som verdi. Klassen **Turplanlegger** har også en liste med referanser til turer.

I klassen **Tur** finnes det en tekst (som kan inneholde blanke, men ikke linjeskift) som beskriver turen, og en liste med referanser til de hyttene som besøkes i løpet av turen (som alle ligger i **Turplanleggers** ordbok).

Hvert objekt av klassen **Hytte** har et unikt navn, et antall sengeplasser og prisen for overnatting per seng i denne hytta (alltid lik pris for hver seng i en hytte).

Oppgave a) (8 poeng)

Skriv klassen **Hytte.** Konstruktøren skal ha parametere for instansvariablene navn, antall senger og pris for overnatting. Videre skal klassen ha følgende metoder:

hentNavn som returnerer hyttenavnet.

totPris som returnerer prisen for et antall personer (antallet er parameter til metoden) for én natt.

sjekkPlass som returnerer **True** om hytta har nok senger til et antall personer (parameter til metoden), ellers **False**.

__str__ som returnerer en bruker-vennlig streng med hyttas navn, antall senger og pris per seng.

__eq__ som sammenligner objektet den kalles på med et annet objekt (referansen er parameter til metoden). To **Hytte**-objekter er like hvis de har samme navn, da returneres **True**, ellers **False**.

Oppgave b) 10 poeng

Skriv klassen **Tur** med en konstruktør som tar imot en liste med referanser til hytter, og en (en-linjes) tekst som beskriver turen. Videre skal klassen ha følgende metoder:

skrivTur som skriver ut på terminal beskrivelsen av turen, og deretter informasjon om hver hytte turen går innom.

sjekkPrisPlass som går gjennom alle hyttene på turen og sjekker om det er nok senger for et antall personer på hver av hyttene, og om totalprisen for turen (alle personer på alle hyttene) er under et maksbeløp. Antall personer og maksbeløp er parametere til metoden. Metoden returnerer **True** om det er nok plasser og kostnadene blir under maksbeløpet, ellers **False**.

Oppgave c) 10 poeng

Du skal skrive klassen **Turplanlegger** med konstruktør og metodene **_hytterFraFil** og **_turerFraFil**. Metodene kalles fra konstruktøren og leser inn data til henholdsvis ordboken med hytter, og listen med turer. Filnavnene er parametere til konstruktøren og til hver sin metode.

Metoden **_hytterFraFil** åpner og leser fra filnavnet gitt i parameter til metoden. Filen har et ukjent antall linjer på følgende format (to hytter vist)

```
Hyttenavn antSenger pris
Hyttenavn antSenger pris
:
```

der første streng skal leses som en tekst og de to siste strengene som heltall og flyttall. Hver hytte representeres i et objekt av klassen Hytte, som lagres i en ordbok i med hyttenavnene som nøkler og referanse til objektet som verdier. Denne ordboken returneres fra metoden når filen er ferdig lest.

Metoden **_turerFraFil** åpner og leser all informasjon om alle turene i systemet fra filen med navn oppgitt i parameteren, og oppretter og returnerer en liste over **Tur**-objekter. Denne filen inneholder to linjer per tur. Første linje inneholder turbeskrivelsen (typisk flere ord), andre linje inneholder navnene på alle hytter på turen. Ingen hyttenavn inneholder mellomrom (space). Samme hytte kan inngå i flere turer. Filen har følgende format:

```
En tekst som beskriver en tur
Hyttenavn1 Hyttenavn2 Hyttenavn3
En tekst som beskriver en annen tur
Hyttenavn1 Hyttenavn4
:
```

Oppgave d) 6 poeng

Skriv en metode **finnTurer** i klassen **Turplanlegger** som kan brukes til å identifisere turer som har plass til et gitt antall personer på alle hyttene, der totalkostnad for alle personer alle dager er under et maksimalbeløp, og som ikke varer mer enn et maks antall dager (turene går alltid til en ny hytte hver dag). Metoden skal gå gjennom alle turer, og skrive ut på terminalen all informasjon om de turene som tilfredsstiller kravene. Antall personer, maksbeløp og maksimalt antall dager er parametere til metoden. Du kan bruke metoder fra tidligere oppgaver uansett om du har skrevet dem eller ikke.

Oppgave e) 6 poeng

Skriv et testprogram for klassen **Turplanlegger** som gjør følgende:

- Oppretter et objekt av klassen Turplanlegger som leser inn data fra filene *hytter.txt* og *turer.txt*
- Skriver ut all informasjon om alle turer som tilfredsstiller følgende krav:
    - ikke varer mer enn 5 dager
    - har overnattingsplass på alle hyttene for et følge på 7 personer, ..
    - ..med et overnattingsbudsjett på til sammen 7000,- for alle 7, hele turen.

Question 18
Attached

Oppgave 4: Planlegging av turer fra hytte til hytte (40 poeng)

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave). I hver av deloppgavene kan du bruke klassene og metodene fra tidligere deloppgaver, også om du ikke har programmert dem.

Du skal implementere deler av et system som organiserer en rekke turer. Hver tur går over flere dager med overnatting på en ny hytte hver dag. Den delen av systemet du trenger å kjenne til består av klassene **Hytte**, **Tur** og **Turplanlegger**. Det anbefales å lage en tegning av strukturen for eget bruk.

Klassen **Turplanlegger** holder rede på alle hytter i systemet ved hjelp av en ordbok (Dictionary) med hyttenavnet som nøkkel og referanse til hytta som verdi. Klassen **Turplanlegger** har også en liste med referanser til turer.

I klassen **Tur** finnes det en tekst (som kan inneholde blanke, men ikke linjeskift) som beskriver turen, og en liste med referanser til de hyttene som besøkes i løpet av turen (som alle ligger i **Turplanleggers** ordbok).

Hvert objekt av klassen **Hytte** har et unikt navn, et antall sengeplasser og prisen for overnatting per seng i denne hytta (alltid lik pris for hver seng i en hytte).

Oppgave a) (8 poeng)

Skriv klassen **Hytte.** Konstruktøren skal ha parametere for instansvariablene navn, antall senger og pris for overnatting. Videre skal klassen ha følgende metoder:

hentNavn som returnerer hyttenavnet.

totPris som returnerer prisen for et antall personer (antallet er parameter til metoden) for én natt.

sjekkPlass som returnerer **True** om hytta har nok senger til et antall personer (parameter til metoden), ellers **False**.

__str__ som returnerer en bruker-vennlig streng med hyttas navn, antall senger og pris per seng.

__eq__ som sammenligner objektet den kalles på med et annet objekt (referansen er parameter til metoden). To **Hytte**-objekter er like hvis de har samme navn, da returneres **True**, ellers **False**.

Oppgave b) 10 poeng

Skriv klassen **Tur** med en konstruktør som tar imot en liste med referanser til hytter, og en (en-linjes) tekst som beskriver turen. Videre skal klassen ha følgende metoder:

skrivTur som skriver ut på terminal beskrivelsen av turen, og deretter informasjon om hver hytte turen går innom.

sjekkPrisPlass som går gjennom alle hyttene på turen og sjekker om det er nok senger for et antall personer på hver av hyttene, og om totalprisen for turen (alle personer på alle hyttene) er under et maksbeløp. Antall personer og maksbeløp er parametere til metoden. Metoden returnerer **True** om det er nok plasser og kostnadene blir under maksbeløpet, ellers **False**.

Oppgave c) 10 poeng

Du skal skrive klassen **Turplanlegger** med konstruktør og metodene **_hytterFraFil** og **_turerFraFil**. Metodene kalles fra konstruktøren og leser inn data til henholdsvis ordboken med hytter, og listen med turer. Filnavnene er parametere til konstruktøren og til hver sin metode.

Metoden **_hytterFraFil** åpner og leser fra filnavnet gitt i parameter til metoden. Filen har et ukjent antall linjer på følgende format (to hytter vist)

```
Hyttenavn antSenger pris
Hyttenavn antSenger pris
:
```

der første streng skal leses som en tekst og de to siste strengene som heltall og flyttall. Hver hytte representeres i et objekt av klassen Hytte, som lagres i en ordbok i med hyttenavnene som nøkler og referanse til objektet som verdier. Denne ordboken returneres fra metoden når filen er ferdig lest.

Metoden **_turerFraFil** åpner og leser all informasjon om alle turene i systemet fra filen med navn oppgitt i parameteren, og oppretter og returnerer en liste over **Tur**-objekter. Denne filen inneholder to linjer per tur. Første linje inneholder turbeskrivelsen (typisk flere ord), andre linje inneholder navnene på alle hytter på turen. Ingen hyttenavn inneholder mellomrom (space). Samme hytte kan inngå i flere turer. Filen har følgende format:

```
En tekst som beskriver en tur
Hyttenavn1 Hyttenavn2 Hyttenavn3
En tekst som beskriver en annen tur
Hyttenavn1 Hyttenavn4
:
```

Oppgave d) 6 poeng

Skriv en metode **finnTurer** i klassen **Turplanlegger** som kan brukes til å identifisere turer som har plass til et gitt antall personer på alle hyttene, der totalkostnad for alle personer alle dager er under et maksimalbeløp, og som ikke varer mer enn et maks antall dager (turene går alltid til en ny hytte hver dag). Metoden skal gå gjennom alle turer, og skrive ut på terminalen all informasjon om de turene som tilfredsstiller kravene. Antall personer, maksbeløp og maksimalt antall dager er parametere til metoden. Du kan bruke metoder fra tidligere oppgaver uansett om du har skrevet dem eller ikke.

Oppgave e) 6 poeng

Skriv et testprogram for klassen **Turplanlegger** som gjør følgende:

- Oppretter et objekt av klassen Turplanlegger som leser inn data fra filene *hytter.txt* og *turer.txt*
- Skriver ut all informasjon om alle turer som tilfredsstiller følgende krav:
    - ikke varer mer enn 5 dager
    - har overnattingsplass på alle hyttene for et følge på 7 personer, ..
    - ..med et overnattingsbudsjett på til sammen 7000,- for alle 7, hele turen.

Question 19
Attached

Oppgave 4: Planlegging av turer fra hytte til hytte (40 poeng)

Denne oppgaven består av en rekke deloppgaver der du skal implementere komponenter av et større system. Om du hopper over en deloppgave er det likevel viktig at du leser hele denne teksten (den ligger samlet som en pdf-fil vedlagt hver deloppgave). I hver av deloppgavene kan du bruke klassene og metodene fra tidligere deloppgaver, også om du ikke har programmert dem.

Du skal implementere deler av et system som organiserer en rekke turer. Hver tur går over flere dager med overnatting på en ny hytte hver dag. Den delen av systemet du trenger å kjenne til består av klassene **Hytte**, **Tur** og **Turplanlegger**. Det anbefales å lage en tegning av strukturen for eget bruk.

Klassen **Turplanlegger** holder rede på alle hytter i systemet ved hjelp av en ordbok (Dictionary) med hyttenavnet som nøkkel og referanse til hytta som verdi. Klassen **Turplanlegger** har også en liste med referanser til turer.

I klassen **Tur** finnes det en tekst (som kan inneholde blanke, men ikke linjeskift) som beskriver turen, og en liste med referanser til de hyttene som besøkes i løpet av turen (som alle ligger i **Turplanleggers** ordbok).

Hvert objekt av klassen **Hytte** har et unikt navn, et antall sengeplasser og prisen for overnatting per seng i denne hytta (alltid lik pris for hver seng i en hytte).

Oppgave a) (8 poeng)

Skriv klassen **Hytte.** Konstruktøren skal ha parametere for instansvariablene navn, antall senger og pris for overnatting. Videre skal klassen ha følgende metoder:

hentNavn som returnerer hyttenavnet.

totPris som returnerer prisen for et antall personer (antallet er parameter til metoden) for én natt.

sjekkPlass som returnerer **True** om hytta har nok senger til et antall personer (parameter til metoden), ellers **False**.

__str__ som returnerer en bruker-vennlig streng med hyttas navn, antall senger og pris per seng.

__eq__ som sammenligner objektet den kalles på med et annet objekt (referansen er parameter til metoden). To **Hytte**-objekter er like hvis de har samme navn, da returneres **True**, ellers **False**.

Oppgave b) 10 poeng

Skriv klassen **Tur** med en konstruktør som tar imot en liste med referanser til hytter, og en (en-linjes) tekst som beskriver turen. Videre skal klassen ha følgende metoder:

skrivTur som skriver ut på terminal beskrivelsen av turen, og deretter informasjon om hver hytte turen går innom.

sjekkPrisPlass som går gjennom alle hyttene på turen og sjekker om det er nok senger for et antall personer på hver av hyttene, og om totalprisen for turen (alle personer på alle hyttene) er under et maksbeløp. Antall personer og maksbeløp er parametere til metoden. Metoden returnerer **True** om det er nok plasser og kostnadene blir under maksbeløpet, ellers **False**.

Oppgave c) 10 poeng

Du skal skrive klassen **Turplanlegger** med konstruktør og metodene **_hytterFraFil** og **_turerFraFil**. Metodene kalles fra konstruktøren og leser inn data til henholdsvis ordboken med hytter, og listen med turer. Filnavnene er parametere til konstruktøren og til hver sin metode.

Metoden **_hytterFraFil** åpner og leser fra filnavnet gitt i parameter til metoden. Filen har et ukjent antall linjer på følgende format (to hytter vist)

```
Hyttenavn antSenger pris
Hyttenavn antSenger pris
:
```

der første streng skal leses som en tekst og de to siste strengene som heltall og flyttall. Hver hytte representeres i et objekt av klassen Hytte, som lagres i en ordbok i med hyttenavnene som nøkler og referanse til objektet som verdier. Denne ordboken returneres fra metoden når filen er ferdig lest.

Metoden **_turerFraFil** åpner og leser all informasjon om alle turene i systemet fra filen med navn oppgitt i parameteren, og oppretter og returnerer en liste over **Tur**-objekter. Denne filen inneholder to linjer per tur. Første linje inneholder turbeskrivelsen (typisk flere ord), andre linje inneholder navnene på alle hytter på turen. Ingen hyttenavn inneholder mellomrom (space). Samme hytte kan inngå i flere turer. Filen har følgende format:

```
En tekst som beskriver en tur
Hyttenavn1 Hyttenavn2 Hyttenavn3
En tekst som beskriver en annen tur
Hyttenavn1 Hyttenavn4
:
```

Oppgave d) 6 poeng

Skriv en metode **finnTurer** i klassen **Turplanlegger** som kan brukes til å identifisere turer som har plass til et gitt antall personer på alle hyttene, der totalkostnad for alle personer alle dager er under et maksimalbeløp, og som ikke varer mer enn et maks antall dager (turene går alltid til en ny hytte hver dag). Metoden skal gå gjennom alle turer, og skrive ut på terminalen all informasjon om de turene som tilfredsstiller kravene. Antall personer, maksbeløp og maksimalt antall dager er parametere til metoden. Du kan bruke metoder fra tidligere oppgaver uansett om du har skrevet dem eller ikke.

Oppgave e) 6 poeng

Skriv et testprogram for klassen **Turplanlegger** som gjør følgende:

- Oppretter et objekt av klassen Turplanlegger som leser inn data fra filene *hytter.txt* og *turer.txt*
- Skriver ut all informasjon om alle turer som tilfredsstiller følgende krav:
    - ikke varer mer enn 5 dager
    - har overnattingsplass på alle hyttene for et følge på 7 personer, ..
    - ..med et overnattingsbudsjett på til sammen 7000,- for alle 7, hele turen.