Oppgave 1

Dere har nettopp kjøpt dere ny leilighet. Leiligheten har en stue, et kjøkken, et bad og to soverom. Alle rommene er rektangulære. Stua har en lengde på 5 meter og en bredde på 4 meter. Kjøkkenet har en lengde og en bredde på 3 meter. Badet er 4 meter bredt og 3 meter langt, og begge soverommene er på 2 meter x 4 meter.

For å sette deres eget preg på leiligheten, har dere bestemt dere for å legge nytt gulv i de fem rommene. Dere skal legge samme gulvtype i alle rom. For å finne og skrive ut antall kvadratmeter med gulv dere trenger totalt, skal dere bruke et `Rektangel`-objekt for hvert rom. Tips: det finnes en `Rektangel`-klasse med en `areal`-metode i ukas forelesningsslides.

For å få inn ordbøker og lister med referanser kan dere gå videre med å lage en klasse Hus med instansvariabel _rommene, en ordbok der romnavn som "bad", "stue", "soverom1" etc. er nøkler, verdiene er referanser til Rom-objekter..

Oppgave 2

Tenk at dere har fått utdelt en fil med gruppeoppmøte i IN1000. Filnavn: *oppmøte_in1000.tx*t.

Første linje i filen består av gruppenummer: gruppe_1, gruppe_2, gruppe_3, osv, til og med gruppe_8. Gruppene er atskilte med «;» (semikolon). På linjene nedenfor står antall oppmøtte i hver gruppe per uke. Antallene er atskilte med «;» (semikolon). Andre linje i fila har oppmøteantall for uke 1, tredje linje i fila har oppmøte for uke 2, osv, til og med uke 12. Her er et eksempel på hvordan de to første linjene i oppmøte_in1000.txt kan se ut:

gruppe_1; gruppe_2; gruppe_3; gruppe_ 4; gruppe_5; gruppe_6; gruppe_7; gruppe_8

10; 19; 25; 18; 23; 11; 13; 17

Lag et hovedprogram som oppretter et gruppeobjekt for hver av de åtte gruppene og legger dem inn i ei liste. I hvert objekt skal gruppenummer og totalt oppmøte (alle ukene summert) lagres. Til slutt skal det skrives ut en ryddig oversikt over gruppenummer og totalt oppmøte for hver av gruppene, i tillegg til totalt oppmøte for alle gruppene samlet. Husk innkapsling. Dere trenger altså metoder for å hente og endre data i objektene underveis i programmet.

Oppgave 3

Et barn kan ha ingen, én, to eller flere foreldre. NAV har gitt dere en liste, der hvert element i lista er en streng på formen `antall foreldre-fødselsnummer`. En liste med tre barn kan for eksempel se slik ut: `[3-01012455555, 1-02020277777, 5-03030399999]`. På generelt grunnlag kan lista ha et vilkårlig (hvilket som helst) antall strenger på formen over.

a. NAV ønsker at dere strukturerer informasjonen om hvert barn i objekter. Det er viktig at fødselsnummer og antall foreldre blir lagret. Bindestreken (-) skal ikke tas med. Bruk en prosedyre som tar inn en vilkårlig liste med barn fra NAV. Prosedyren skal kalle en funksjon hver gang det skal opprettes et objekt. Funksjonen skal returnere det opprettede objektet. Hvert element i lista med strenger skal byttes ut med et barneobjekt med tilsvarende informasjon. Dere skal også lage et hovedprogram. Hovedprogrammet skal kalle prosedyren med en vilkårlig liste med barn fra NAV.

b. Med utgangspunkt i eksempellista ovenfor, lag en datastrukturtegning som viser lista med de tre barneobjektene og deres tilhørende data.

Oppgave 4

I en hektisk hverdag er det mye som skal gjøres. For å få bedre oversikt over gjøremål, har dere bestemt dere for å opprette ei gjøremålsliste. I lista finnes en rekke gjøremål, der hvert gjøremål har en tittel, en beskrivelse og en frist (deadline). Det skal kun gå an å hente ut den angitte informasjonen om hvert gjøremål ved bruk av forskjellige metoder.

Når programmet starter, skal dere få valg om å 1) Opprette et gjøremål, 2) Skrive ut informasjon om alle gjøremålene i gjøremålslista og 3) Avslutte programmet. Når dere har skrevet inn tittel, beskrivelse og en frist (deadline), legges et nytt gjøremål til i gjøremålslista. Programmet deres skal kun avslutte hvis 3 skrives inn i terminalen. Ellers skal dere kunne opprette flere gjøremål eller skrive ut gjøremålslista uten at programmet avslutter.

Lag et program med funksjonalitet som oppfyller kravene beskrevet ovenfor.

Oppgave 5

I en boligblokk bor det bare samboerpar. Hver av de to personene i paret identifiseres med et fødselsnummer (fødselsdato og personnummer). I tillegg har hver person et navn, som består av fornavn, mellomnavn og etternavn. `Person`-klassen ser slik ut:

```
class Person:
    def __init__(self, fødselsnummer, navn):
        self._fødselsnummer = fødselsnummer
        self._navn = navn

    def hent_fødselsnummer(self):
        return self._fødselsnummer

    def hent_navn(self):
        return self._navn
```

a. Skill ut fødselsnummer og navn i egne objekter. Et fødselsnummer skal holde på fødselsdato og personnummer. Navn skal holde på fornavn, mellomnavn, etternavn. Person-objektene skal ha referanser til objekter av fødselsnummer og navn.
b. Legg til funksjonalitet for å kunne opprette `Par`-objekter bestående av to personer.
c. Representer boligblokka parene bor i med en samling. Boligblokka består av fem etasjer. I hver etasje bor det fem par. Rekkefølgen på parene i hver av etasjene er vesentlig, siden hvert par alltid bor i den samme leiligheten (og ikke bytter).
d. Lag et testprogram (hovedprogram), som legger par til i boligblokka og som skriver ut navn og fødselsnummer på personene i hvert par i hver etasje. Det er viktig at det kommer tydelig frem i utskriften hvilke personer som er i hvert par samt hvilke par som bor i hvilken etasje.