Oppgave 1

```
class Matbutikk:
    def __init__(self):
        self._varer = []
```

a. Utvid klassen `Matbutikk`, slik at den får en metode for å hente lista `varer`.
b. Skrive en metode for å legge til et `Vare`-objekt i lista `varer`.
c. Trenger vi flere parametere enn `self` i konstruktøren over? Begrunn svaret.
d. Skriv klassen `Vare`, med instansvariabler `vareid` og `varenavn`. init-metoden (konstruktøren) tar inn `vareid` og `varenavn`. `Vare` skal ha to metoder; én for å hente `vareid` og én for å hente `varenavn`.
e. Skriv et hovedprogram med en `Matbutikk`, og opprett og legg til tre `Vare`-objekter. Skriv ut `vareid` sammen med `varenavn` for de tre varene på tre forskjellige linjer. Dere blir nødt til å kalle relevante metoder underveis.

Oppgave 2

```
class Student:
    def __init__(self, navn, studieprogram):
        self._navn = navn
        self._studieprogram = studieprogram

    def hent_navn(self):
        return self._navn

    def endre_studieprogram(self, nytt_program):
        self._studieprogram = nytt_program
```

a. Hva er det *self* refererer til i init-metoden (konstruktøren) og i metodene hent_navn og endre_studieprogram?
b. Hva slags variabler er *navn*, *studieprogram* og *nytt_program*?
c. Hvordan skiller variablene i b) seg fra variablene *_navn* og *_studieprogram*?
d. Hvorfor trenger vi *self* i metodene *hent_navn* og *endre_studieprogram*?
e. Hvorfor trenger vi ikke å inkludere *self* som argument når vi kaller metodene hent_navn og endre_studieprogram på et Student-objekt?

Oppgave 3

Gitt hovedprogrammet nedenfor, tegn datastrukturen slik den ser ut etter linjen `forelder.sett_barn(barn)`. En datastrukturtegning viser opprettede objekter og sammenhengen mellom dem. Tips: eksempler på datastrukturtegninger finnes blant annet i ukas forelesningsslides. Sammenlign deres tegning med Python Tutors.

```
class Forelder:
    def __init__(self):
        self._barn = None

    def sett_barn(self, barn):
        self._barn = barn

class Barn:
    def __init__(self, fødselsdato, fødselsvekt):
        self._fødselsdato = fødselsdato
        self._fødselsvekt = fødselsvekt

def hovedprogram():
    forelder = Forelder()
    barn = Barn("01012024", 3500)
    forelder.sett_barn(barn)

hovedprogram()
```

Oppgave 4

```
class A:
    def __init__(self, a):
        self._a = a

class B:
    def __init__(self, b):
        self._b = b

class C:
    def __init__(self, c):
        self._c = c

def hovedprogram():
    c = A("B")
    a = B("C")
    b = C("A")
    b = a
    a = c
    c = b

hovedprogram()
```

a. Hva refererer henholdsvis variablene *a*, *b* og *c* til i hovedprogrammet?

b. Hvorfor bruker vi ikke *self* noe sted i hovedprogrammet?
c. Er argumentene vi bruker gyldige? Forklar hvorfor/hvorfor ikke.

Oppgave 5

```
class KaffeLatte:
    def __init__(self, ant_espressoshots):
        self._antall_espressoshots = ant_espressoshots

    def hent_antall_espressoshots(self):
        return self._antall_espressoshots

def hovedprogram():
    bestillinger = []
    for i in range(5):
        dobbel_latte = KaffeLatte(2)
        bestillinger.append(dobbel_latte)

    for bestilling in bestillinger:
        print(bestilling)

hovedprogram()
```

a. Hva vil skrives ut i programmet? Kom opp med et svar og begrunn hvorfor før dere kjører programmet.
b. Endre programmet, slik at linjen `dobbel_latte = KaffeLatte(2)` står under linjen `bestillinger = []` i stedet for inni for-løkka. Hva skrives ut nå? Kom opp med et svar og begrunn hvorfor før dere kjører programmet.
c. Hvor mange objekter av klassen KaffeLatte finnes i hovedprogrammet i a), og hvor mange finnes det i hovedprogrammet i b)? Begrunn hvorfor.