Oppgave 3a (5 poeng)

```
def pris_inkl_frakt(varepris):
    if varepris > 1000:
        return varepris
    elif varepris >= 500:
        return varepris + 50
    else:
        return varepris + 80

assert pris_inkl_frakt(300) == 380
assert pris_inkl_frakt(600) == 650
assert pris_inkl_frakt(1300) == 1300
```

Komponenter:
- Se behov for if: 1p
- Riktig bruk av parameter og returverdi: 1p
- Riktig logikk: 2p
- Fungerer: 1p
- (siden oppgaven er ment for å skille de laveste karakterene trekker vi ikke for overflødige conditions, som i "elif varepris >= 500 and varepris <= 1000")

(alternative fungerende løsninger gis også full uttelling. Om slike inkluderer mindre feil trekkes poeng etter skjønn)

Oppgave 3b (5 poeng)

```
def fjern_utsolgte(handleliste, utsolgte):
    nyliste = []
    for vare in handleliste:
        if not vare in utsolgte:
            nyliste.append(vare)
        else:
            print(vare)
    return nyliste

assert fjern_utsolgte( ["melk", "brus", "pasta"], ["kanel","brus"]) ==
["melk", "pasta"]
```

Komponenter:
- Se behov for løkke: 1p
- Riktig håndtering av lister: 1p
- Skriver ut varer som fjernes: 1p
- Riktig logikk: 1p
- Fungerer: 1p

Bestemte tilfeller:
- å endre listen mottatt som argument i stedet for å lage ny liste: 1 poeng trekk
- å ikke klare å skille printing fra returnering, og dermed korrekt konstruere liste med ikke-utsolgte varer men printe denne i stedet for å returnere: 3 poeng for hele oppgave

(alternative fungerende løsninger gis også full uttelling - om slike inkluderer mindre feil trekkes poeng etter skjønn)

Oppgave 3c (5 poeng)

```
def samlet_vaksinasjon(krav_hvert_land):
    vaksiner = []
    for krav in krav_hvert_land:
        for vaksine in krav:
            if not vaksine in vaksiner:
                vaksiner.append(vaksine)
    return vaksiner

assert samlet_vaksinasjon([["difteri","tyfoid"], ["hepatit","difteri"]]) ==
['difteri', 'tyfoid', 'hepatit']
```

Komponenter:
- Se behov for nøstet løkke: 2p
- Riktig logikk: 2p
- Fungerer: 1p

(alternative fungerende løsninger gis også full uttelling - om slike inkluderer mindre feil trekkes poeng etter skjønn)

Oppgave 3d (5 poeng)

```
def forkort_setning(setning, fjern):
    ny_setning = ""
    for ord in setning.split():
        if not ord==fjern:
            ny_setning = ny_setning + ord + " "
    return ny_setning

setning = "en krabbe skal en dag ut av skallet "
setning_v2 = forkort_setning(setning, "en")
setning_v3 = forkort_setning(setning_v2, "skal")

assert setning_v3 == "krabbe dag ut av skallet "
```

Komponenter:
- Legge relevant del av koden i en funksjon: 1p

- Ha ordet som skal fjernes som parameter: 1p
- Kalle funksjonen to ganger med meningsfulle argument og tilordninger: 2p
- At en riktig oppsatt struktur virkelig fungerer: 1p

Bestemte tilfeller (total poeng for hele oppgaven):
- Løsning hvor essensielt hele den redundante koden er lagt inn i en funksjon som kalles én gang: 0 poeng
- Løsning hvor den definerte funksjonen kun kalles én gang, men hvor koderedundans likevel reduseres betydelig på andre måter: 2 poeng

(alternative løsninger gis også full uttelling så lenge de baserer seg på en rimelig måte å definere funksjon på som er i tråd med hva oppgaven ber om, inkludert at denne funksjonen blir kalt to ganger for å oppnå ønsket oppførsel. Om slike løsninger ikke er i tråd med alt det bes om i oppgaveteksten eller inkluderer mindre feil trekkes poeng etter skjønn. (læringsmålet som oppgaven tester er om de klarer å trekke ut likhet i kode i form av en funksjon som tilpasses med parametre)

Oppgave 5 (6 poeng)

```
def sjekk_om_fyord(setning, fyord, synonym_liste):
    biter = setning.split()
    for bit in biter:
        for synonymer in synonym_liste:
            if bit in synonymer and fyord in synonymer:
                return True
    return fyord in biter

assert sjekk_om_fyord("spis masse godsaker", "snop",
[["saft","lemonade"],["snacks","snop","godsaker"],["mye","masse"]]) ==
True
assert sjekk_om_fyord("spis masse godsaker", "godsaker",
[["saft","lemonade"],["snacks","snop","godsaker"],["mye","masse"]]) ==
True
assert sjekk_om_fyord("spis masse godsaker", "godsaker", []) == True
assert sjekk_om_fyord("spis masse godsaker", "lemonade",
[["saft","lemonade"],["snacks","snop","godsaker"],["mye","masse"]]) ==
False
assert sjekk_om_fyord("spis masse godsaker", "agurk", [["mye","masse"],
["spis","gomle"],["snacks","snop","godsaker"]]) == False
```

Komponenter:
- Se behov for å gå gjennom ett ord av setningen av gangen (med en løkke eller implisitt gjennom “in” el.l): 1p

- Se behov for å gå gjennom hver samling av synonymer (gå gjennom hvert element av synonym_liste, som igjen er en liste): 1p
- Se behov for å sjekke at både fyordet og et gitt setningsord er del av en gitt underliste av synonymer (direkte vha in og and som i v2 eller på en mer løkkete måte): 1p
- Riktig logikk: 2p
- Fungerer: 1p

Bestemte tilfeller:

- Dersom en løsning ikke er i stand til å oppdage at et fyord direkte er i setningen uten å være del av noen synonymliste, trekkes det 1 poeng)
- Dersom en løsning antar at samme ord ikke kan være del av mer enn én gruppe synonymer (subliste), trekker vi ikke noe for dette).

(alternative fungerende løsninger gis også full uttelling - om slike inkluderer mindre feil trekkes poeng etter skjønn)