LUTin Python-ohjelmoinnin tyyliohje

1. Johdanto

Tämä on LUTin Python-ohjelmointikurssien tyyliohje. Nämä ohjeet on tarkoitettu kurssin harjoitustyön valikkopohjaisen ohjelman ja sitä vastaavien ohjelmien tyyliohjeeksi auttamaan toimivan ja ymmärrettävän Python-ohjelman tekemisessä. Tätä ohjetta voi soveltaa muissa vastaavissa projekteissa, mutta lähtökohtaisesti kaikilla ohjelmointiin vakavasti suhtautuvilla organisaatioilla ja projekteilla on omiin tarpeisiin sovitetut tyyliohjeet, joita tulee noudattaa ja soveltaa organisaation sisäisten ohjeiden mukaisesti.

1.1. Kurssin ohjelmien vaatimukset

LUTin Python-ohjelmointikursseilla kaikkia ohjelmia koskee seuraavat vaatimukset:

1. Ohjelman on mentävä automaattitarkastuksesta läpi ja toimittava tehtäväksiannon mukaisesti
2. Tiedostojen avaamisen ja käsittelyn yhteydessä on oltava poikkeustenkäsittely
3. Kaikki ohjelman varaamat resurssit on vapautettava ohjelman lopuksi
4. Globaalit muuttujat ovat kiellettyjä
5. Kaikissa palautettavissa tiedostoissa on oltava tämän ohjeen mukaiset alkukommentit
6. Kaikkien pää- ja aliohjelmien on päätyttävä `return`-käskyyn ja paluuarvoon
7. Kurssilla käsittelemättömien kirjastojen käyttö on kielletty.

Nämä vaatimukset tulevat voimaan sen jälkeen, kun ne on käsitelty luennoilla ja niiden noudattamatta jättäminen johtaa työn korjaamiseen tai hylkäämiseen tilanteesta riippuen.

Kursseilla ei opeteta kaikkien Python-kielen käskyjen käyttöä. Yllä olevassa listassa on nimetty kursseilla kielletyt ohjelmointirakenteet ja muita rakenteita/käskyjä saa käyttää, mutta vastuu niiden oikeasta käytöstä ja ohjelman selkeydestä on rakenteen käyttäjällä. Esimerkkejä tällaisista rakenteista ovat tuple, match-case, f-string ja with-open.

Muutoin tyyliohjeiden noudattaminen on kurssilla opetettavan ymmärrettävän ja ylläpidettävän ohjelmointityylin lähtökohta. Joissain tapauksissa näistä ohjeista tulee poiketa, jolloin asia mainitaan erikseen tehtäväksiannossa. Perustelut näille tyyliohjeille löytyy LUTin Python-ohjelmointioppaasta ja tarvittaessa kurssilla hyvästä ohjelmointityylistä päättää kurssin vastuuopettaja.

2. Rakenne

Kiintoarvojen, luokkien, pää- ja aliohjelmien määrittelyt sekä kirjastojen sisällytykset tulee sijoittaa ohjelman päätasolle. Kaikki muu koodi tulee toteuttaa näiden rakenteiden sisällä.

2.1. Tiedostorakenne, yksi tiedosto

Tiedoston rakenne, kun ohjelma koostuu vain yhdestä tiedostosta (L08):

1. Tiedoston alkukommentti
2. Kirjastojen sisällytykset
3. Kiintoarvojen määrittely
4. Luokkien määrittely
5. Aliohjelmien koodi kutsujärjestyksessä
6. Pääohjelman koodi

2.2. Tiedostorakenne, useista tiedostoista muodostuva ohjelma

Laajassa ohjelmassa on pääohjelma ja yksi tai useampi aliohjelma, esim. harjoitustyössä tyypillisesti vähintään neljä aliohjelmaa. Aliohjelmat tulee jakaa eri tiedostoihin alla olevan

mukaisesti. Kiintoarvot ja luokat määritellään ohjelmassa vain yhden kerran siinä tiedostossa, jossa on niiden pääasiallinen tarve (L08).

Pääohjelman sisältävän tiedosto:

1. Tiedoston alkukommentti
2. Pääohjelmassa tarvittavien standardikirjastojen sisällytykset
3. Omien kirjastojen sisällytykset
4. Kiintoarvojen määrittely
5. Luokkien määrittely
6. valikko-aliohjelma
7. Pääohjelman koodi

Muiden Python-tiedostojen rakenne:

1. Tiedoston alkukommentti
2. Kirjastossa tarvittavien standardikirjastojen sisällytykset
3. Kiintoarvojen määrittely
4. Luokkien määrittely
5. Aliohjelmien koodi kutsujärjestyksessä

2.3. Tiedoston alkukommentti

Kaikki kurssin ohjelmointitehtävät ovat henkilökohtaisia ja kaikkiin palautettaviin tiedostoihin tulee laittaa seuraavat tiedot sisältävä alkukommentti luentoviikolta 7 alkaen (L07).

```
#########################################################################
# CT60A0203 Ohjelmoinnin perusteet
# Tekijä:
# Opiskelijanumero:
# Päivämäärä:
# Kurssin oppimateriaalien lisäksi työhön ovat vaikuttaneet seuraavat
# lähteet ja henkilöt, ja se näkyy tehtävässä seuraavalla tavalla:
#
# Mahdollisen vilppiselvityksen varalta vakuutan, että olen tehnyt itse
# tämän tehtävän ja vain yllä mainitut henkilöt sekä lähteet ovat
# vaikuttaneet siihen yllä mainituilla tavoilla.
#########################################################################
# Tehtävä LxTx.py

# eof
```

2.4. Pää- ja aliohjelman rakenne

Nimetyt pää- ja aliohjelmat, pää/aliohjelmat, muodostuvat tyypillisesti seuraavista osista (L05, L02).

1. Pää/aliohjelma alkaa `def`-tunnisteella, jota seuraa ohjelman nimi ja parametrit
2. Muuttujien määrittelyt ja alustukset
3. Tietojen kysyminen käyttäjältä
4. Toiminnallinen osuus, esim. laskenta, lukeminen tai kirjoittaminen
5. Tulosten tulostaminen käyttäjälle
6. Lopetusrutiinit ja käyttäjän informointi pää/aliohjelman loppumisesta
7. Kaikki pää/aliohjelmat päättyvät `return`-käskyyn ja paluuarvoon

2.5. Valikkopohjaisen pääohjelman rakenne

Valikkopohjainen ohjelma perustuu toistorakenteen sisällä olevaan valintarakenteeseen, ks. ohjelmointioppaan **Esimerkki 5.7 Valikkopohjainen ohjelma (L05).**

1. Pääohjelman toistorakenteena käytetään `while`-rakennetta
2. Valintarakenteena käytetään monihaaraista `if-elif-else`-rakennetta
    a. Valintarakenteen valinnat käydään järjestyksessä alkaen valinnasta 1 valintaan N, jonka jälkeen on valinta 0 eli ohjelman normaali lopetus
    b. Valintarakenteen viimeinen haara on `else`-haara, jossa käsitellään tuntemattomat valinnat
    c. Valintarakenteesta poistutaan `while`-ehdon muututtua epätodeksi
3. Ohjelman valikon tulostus ja käsittely tapahtuu omassa valikko-aliohjelmassa, ks. kohta 2.6
4. Valintahaarassa tulee tehdä kaikki valintaan liittyvät toimenpiteet alusta loppuun asti ja tarvittaessa yhdessä valinnassa voidaan kutsua useaa eri aliohjelmaa
5. Mikäli valinnan suorittaminen edellyttää tiettyjä tietoja, esim. oliolistaa, tarkistetaan tämän tiedon olemassaolo ehtolauseella ennen siihen liittyvien operaatioiden suorittamista ja tarvittaessa kerrotaan käyttäjälle, ettei operaatioita voida suorittaa tiedon puuttumisen vuoksi
6. Valintarakenteen jälkeen toistorakenteen viimeinen käsky on tyhjän rivin tulostus
7. Ohjelman kaikki lopetusrutiinit ovat toistorakenteen jälkeen ennen pääohjelman loppua
8. Ohjelma voi päättyä toistorakenteen sisällä poikkeustenkäsittelyn seurauksena, ks. kohta 3.6

Pääohjelman valintarakenteen standardifraasit ovat seuraavat:

1. `"Tuntematon valinta, yritä uudestaan."`
2. `"Lopetetaan."`
3. `"Kiitos ohjelman käytöstä."`

Pääohjelman valintojen tyypillisiä ohjeita ja tiedotteita käyttäjälle ovat seuraavat:

1. `"Anna luettavan tiedoston nimi: "`
2. `"Anna kirjoitettavan tiedoston nimi: "`
3. `"Ei analysoitavaa, lue tiedosto ennen analyysiä."`
4. `"Ei kirjoitettavia tietoja, analysoi tiedot ennen tallennusta."`

2.6. valikko-aliohjelma

Valikkopohjaisen ohjelman valikko-aliohjelma tulee toteuttaa seuraavalla tavalla (L05):

1. Aliohjelma ei saa parametrejä ja se palauttaa käyttäjän valinnan kokonaislukuna
2. Valikon jokainen rivi tulostetaan omalla `print`-käskyllä
3. Käyttäjän valinnat numeroidaan alkaen luvusta 1 ja viimeisenä on Lopeta-valinta numerolla 0, numeron jälkeen on)-merkki ja välilyönti eli `"0) Lopeta"`

valikko-aliohjelman standardifraasit ovat seuraavat:

1. `"Valitse haluamasi toiminto:"`
2. `"Anna valintasi: "`

2.7. Aliohjelmat

Aliohjelmat jakavat ohjelman pienempiin loogisiin kokonaisuuksiin, lisäävät ohjelman ymmärrettävyyttä ja mahdollistavat uudelleenkäytön. Aliohjelmiin liittyy seuraavat yleisohjeet (L05).

1. Uusi aliohjelma tulee tehdä, kun sama/vastaava toiminta tehdään ohjelmassa monta kertaa tai ohjelmaan lisätään uusi erillinen toiminnallisuus kuten tiedoston luku tai kirjoitus
2. Ohjelman toiminnallisuuden lisäys voi kasvattaa ohjelmaa ja tehdä siitä hankalasti ymmärrettävän, jolloin loogisia kokonaisuuksia tulee siirtää omiin aliohjelmiin
3. Uutta aliohjelmaa ei tule tehdä, jos se ei tuo lisäarvoa.

Aliohjelmien tiedonvälitys

1. Tiedot aliohjelmiin välitetään parametrien avulla
2. Kaikki aliohjelmat päättyvät `return`-käskyyn ja paluuarvoon. Mikäli ohjelma ei palauta tietoa, tulee käyttää `return None` -käskyä
3. Aliohjelmaan välitetään usein tietoa erilaisissa tietorakenteissa, esim. listassa. Mikäli tietorakennetta muutetaan aliohjelmassa, tulee se palauttaa paluuarvona ja sijoittaa sopivaan muuttujaan käyttöä varten
4. Tällä kurssilla aliohjelmasta palautetaan vain yksi paluuarvo, useiden tietoalkioiden palautukseen käytetään rakenteisia tietorakenteita, esim. listaa tai oliota
5. Globaalit muuttujat ovat kiellettyjä
6. Globaalien kiintoarvojen käyttö on suositeltavaa.

2.8. Nimeäminen

Ohjelmoinnissa tiedostot ja tunnukset tulee nimetä selkeästi ja johdonmukaisesti.

Yleisesti

1. Skandinaavisia merkkejä (å, ä, ö, Å, Ä, Ö) ei tule käyttää tiedostojen ja tunnusten nimissä. Tyypillisesti nämä korvataan a tai o -kirjaimella
2. Tunnusten tulee olla kuvaavia ja yksikäsitteisiä
3. Samassa nimiavaruudessa olevat tunnukset tulee nimetä toisistaan poikkeavilla nimillä, esim. kiintoarvoilla, muuttujilla ja aliohjelmilla on oltava eri nimet
4. Vakiintuneiden tunnusten käyttö on suositeltavaa. Tällaisia ovat mm. toistorakenteen askeltajat `i`, `j` ja `k` sekä koordinaatit `x`, `y` ja `z`
5. Mitäänsanomattomien ja harhaanjohtavien tunnusten käyttö on kielletty, esim. kirjaimet `a`, `b`, `c` ja `a1`, `a2`, `a3` jne.

Tiedostot

1. Viikkotehtävät tulee nimetä luennon ja tehtävän perusteella, esim. `L01T1.py`
2. Kirjastotiedostot nimetään `xxKirjasto.py`, esim. `L08T1Kirjasto.py`
3. Harjoitustyötiedostot tulee nimetä tehtäväksiannon mukaisesti, esim. `HTPerus.py`, `HTPerusKirjasto.py`
4. Tentissä tiedostot tulee nimetä tentin ohjeiden mukaisesti

Kiintoarvot

1. Kiintoarvot kirjoitetaan suuraakkosilla, esim. `IKA_MAX = 18`, `EROTIN = ';'`. Kiintoarvoja tulee käyttää esim. vakiokokoisten listojen ja matriisien koon määrittelyyn

Muuttujat

1. Muuttujat tulee nimetä systemaattisesti niiden kuvaaman tiedon perusteella, esim. `Lukumaara`, `Nimi`, `Paino`, `Valinta`

2. Nimet kannattaa muodostaa tarvittaessa useista sanoista ja sanat tulee liittää toisiinsa uudet sanat suuraakkosilla kirjoittaen, esim. `SummaSuurin`, `ListaTulokset`, `TiliNumero`
3. Nimissä voi käyttää yleisesti käytettyjä selkeitä lyhenteitä, esim. `Lkm`, `Pvm`, `Nro`, `PainoMin`, `PituusMax`

Aliohjelmat

1. Aliohjelman nimen tulee kertoa, mitä aliohjelmassa tapahtuu ja tarvittaessa käytetään useita sanoja, esim. `analysoiTiedot`, `kirjoitaTiedosto`, `tulostaTiedot`
2. Yhdestä sanasta muodostuvat aliohjelmanimet tulee kirjoittaa kaikki kirjaimet pienellä ja jos sanoja on useita, seuraavien sanojen ensimmäiset kirjaimet kirjoitetaan suuraakkosilla, esim. `paaohjelma`, `valikko`, `kysyNimi`, `lueTiedosto`

Luokka ja olio

1. Luokan nimi tulee kirjoittaa suuraakkosilla, esim. `AUTO`, `DATA`
2. Luokasta luotava olio tulee nimetä samalla sanalla kuin luokka, mutta muuttujan nimeämisohjeilla, esim. `Auto`, `Data`, `Auto1`, `Auto2`

2.9. Näkyvyys

Tunnusten näkyvyyteen liittyen tulee muistaa seuraavat lähtökohdat:

- Globaalit muuttujat ovat kiellettyjä
- Muuttujat määritellään lokaaleina pää/aliohjelmissa
- Kiintoarvot, luokat ja aliohjelmat määritellään globaaleina

3. Perusoperaatiot

3.1. Sulkujen käyttö

Pythonissa kaarisulkuja (ja) tulee käyttää seuraavissa tapauksissa

1. aliohjelmien määrittelyssä sulkuihin tulee vastaanotettavat parametrit. Jos parametrejä ei ole, sulut jätetään tyhjiksi
2. aliohjelmia/jäsenfunktioita kutsuttaessa sulkuihin laitetaan lähetettävät parametrit. Jos parametrejä ei ole, sulut jätetään tyhjiksi, esim. `print("Sana")` ja `Tiedosto.close()`
3. oliota luodessa, esim. `Data = DATA()`
4. Laskentakaavoissa ja ehtolausekkeissa ryhmittelyyn normaalien matemaattisten sääntöjen mukaisesti, esim. lasku ”2 * (3 + 4)” johtaa tulokseen 14.

3.2. Tiedostonkäsittely

Katso tarkemmin ohjelmointioppaan luvun 6 ja 7 tyyliohjeet.

Tiedostonkäsittely tehdään omassa aliohjelmassa

1. Aliohjelma saa ensimmäisenä parametrinä luettavan tiedoston nimen
2. Käytettävät käskyt ja jäsenfunktiot
    a. Tiedosto avataan ja suljetaan `open`/`close` -käskyillä
    b. Jos tiedostossa on kirjainmerkkejä, tulee avauksessa määritellä UTF-8 koodaus. Pelkän numeerisen datan yhteydessä tätä ei käytetä
    c. Tiedosto luetaan `readline`-käskyllä
    d. Mahdollinen otsikkorivi ohitetaan erillisellä `readline`-käskyllä ennen toistorakenteessa olevaa datarivien lukemista

    e. Tiedostojen tulee päättyä aina tyhjään riviin ja sitä käytetään tiedoston lukemisen lopetusehtona
    f. Tiedostosta luetun rivin lopusta tulee poistaa rivinvaihtomerkki
    g. Kirjoitettaessa tiedosto avataan kirjoitustilassa, `'w'`
    h. Tiedosto kirjoitetaan `write`-käskyllä, usein kirjoitettava merkkijono kannattaa muodostaa erillisenä käskynä ennen kirjoittamista
3. Lähtökohtaisesti yhdeltä riviltä luetuista tiedoista muodostetaan yksi olio, joka tallennetaan oliolistaan
4. Tiedostoa luettaessa listalla olevat aiemmat tiedot poistetaan ennen uusien lisäämistä.

Tiedostonkäsittelyn tyypillisiä ohjeita ja tiedotteita käyttäjälle ovat mm. seuraavat:

1. `"Tiedosto 'tiedoston_nimi' luettu."`
2. `"Tiedosto 'tiedoston_nimi' luettu ja tulostettu."`
3. `"Tiedosto 'tiedoston_nimi' kirjoitettu."`

3.3. Tietojen analysointi

Tietojen analyysi tehdään tyypillisesti oliolistassa oleville tiedoille (L08).

1. Analyysi-aliohjelma saa tyypillisesti parametrina kaksi tietorakennetta: analysoitavan datan sisältävän tietorakenteen ja tulostietorakenteen, johon tallennetaan tulokset. Analysoitava data on tyypillisesti listassa ja tulostietorakenne on tyypillisesti yksi olio tai lista
    a. Mikäli tulostietorakenne on lista ja analyysi suoritetaan kerralla suurelle tietomäärällä, esim. luetaan kokonainen tiedosto, tulee aiemmat analyysin tulokset poistaa tyhjentämällä lista ennen analyysiä. Vastaavassa tilanteessa matriisin kaikki arvot tulee asettaa nolliksi
2. Tee analyysi ja sijoita tulokset tulostietorakenteeseen
3. Etsittäessä datasta tiettyjä arvoja, esim. minimi tai maksimi, tulee tilapäismuuttujat alustaa datasetin ensimmäisen alkion/olion arvoilla
4. Mikäli datassa on useita ehdon täyttäviä arvoja, esim. useita yhtä suuria minimi-/maksimiarvoja, valitaan näistä
    1. erikseen mainitun ehdon täyttävä arvo, jos tällainen ehto on olemassa
    2. vanhin, jos datassa on aikaleima
    3. alkuperäisen datan ensimmäinen arvo (rivinumeron mukaan)
5. Mikäli tehtävässä käytetään datasetin ominaisuuksia, tulee ne selvittää datasta, esim. alkioiden lukumäärä tai ensimmäisen/viimeisen alkion aikaleima
6. Kaikki analyysit tulee suorittaa alkuperäisissä yksiköissä ja mahdollinen tulosten pyöristys tehdään vasta muotoiltaessa lopullisia tulosteita

3.4. Varattujen resurssien vapauttaminen

Ohjelman päättyessä varatut resurssit kuten tiedostokahvat ja tietorakenteiden muistialueet on vapautettava lopetusrutiineissa ennen ohjelman päättymistä. Tällä kurssilla se tarkoittaa seuraavaa:

1. Tiedostokahvat tulee vapauttaa sulkemalla avatut tiedostot siinä pää/aliohjelmassa, jossa ne avataan
2. Rakenteiset tietorakenteet lista, matriisi ja sanakirja tulee tyhjentää sen pää/aliohjelman lopussa, jossa ne luodaan
    a. Tyypillisesti listat luodaan pääohjelmassa esim. `DataLista`, `NimiLista` ja `OlioLista` ja ne välitetään parametrinä aliohjelmiin. Nämä listat

tyhjennetään pääohjelman lopussa. Mikäli ohjelman toiminta niin vaatii, listat voi tyhjentää muuallakin, esim. luku- tai analyysi-aliohjelmien alussa

b. Apulistat ja muut väliaikaiset tietorakenteet luodaan siinä aliohjelmassa, jossa niitä tarvitaan ja ne tyhjennetään saman aliohjelman lopussa.

3.5. Rakenteiset tietorakenteet

Kurssilla käytettävät rakenteiset tietorakenteet ovat lista, sanakirja, luokka, olio ja oliolista, jotka käsitellään tarkemmin oppaan luvussa 7, sanakirja luvussa 10.

1. Dynaamisia rakenteita, lista tai sanakirja, tulee käyttää, jos samanlaisia muuttujia on 5 tai enemmän. Näissä rakenteissa olevien tietoalkioiden tulee olla samaa tyyppiä
2. Listan läpikäyntiin tulee käyttää `for Alkio in Lista` –rakennetta, paitsi jos indeksin käyttö tuo lisäarvoa
3. Luokat määritellään tiedoston alussa. Luokassa tulee olla vähintään kaksi jäsenmuuttujaa, jotka ovat tyypillisesti perustietotyyppejä eli merkkijono, kokonaisluku tai desimaaliluku ja ne alustetaan `None`:lla. Luokkaa käytetään vain olioiden luomiseen
4. Olioihin ei saa lisätä uusia jäsenmuuttujia ohjelman suorituksen aikana
5. Oliota käytetään yhteenkuuluvien tietojen säilyttämiseen, esim. yhden henkilön tiedot
6. Oliolista tarkoittaa listaa, jonka kaikki alkiot ovat olioita. Oliolistaa tulee käyttää perus- ja tavoitetason ratkaisuissa
7. Sanakirja on perus- ja tavoitetason rakenne, jota kannattaa käyttää tietyissä tehtävissä.

3.6. Poikkeustenkäsittely

Tällä kurssilla poikkeustenkäsittely on toteutettava aina tiedostonkäsittelyn yhteydessä. Asiaa käsitellään laajemmin ohjelmointioppaassa ja tässä keskitytään vain poikkeustenkäsittelyyn tiedostonkäsittelyn yhteydessä (L09).

1. Tiedostonkäsittely tulee toteuttaa aina poikkeustenkäsittelyn sisällä
2. Poikkeustenkäsittelijä sijoitetaan samaan aliohjelmaan tiedostonkäsittelyn kanssa ja tiedoston avaus, luku/kirjoitus ja sulkeminen ovat saman poikkeustenkäsittelijän sisällä
3. Tarkkailtavaksi poikkeukseksi määritellään aina `Exception`

Poikkeustapauksen sattuessa ilmoitetaan käyttäjälle ongelmasta ja lopetetaan ohjelma `except`-haarassa olevilla `print`- ja `sys.exit(0)`-käskyillä.

Tyypillisimmät virheilmoitukset ovat seuraavat:

1. `"Tiedoston 'tiedoston_nimi' käsittelyssä virhe, lopetetaan."`

3.7. Ylimääräiset koodirivit

Ohjelmassa ei tule olla ylimääräisiä koodirivejä, jotka eivät tee mitään, kumoavat toisensa tai joita ei voi koskaan saavuttaa. Tällaisia ovat mm. seuraavat:

1. Muuttujat, kiintoarvot, aliohjelmat ja luokat, jotka määritellään, mutta joita ei käytetä sen jälkeen
2. Käskyjen `break`, `continue`, `return` ja `sys.exit` jälkeen samassa koodilohkossa olevat käskyt
3. Peräkkäiset `int` ja `str` -käskyt, jotka tyypillisesti kumoavat toisensa
4. Tyyppimuunnokset kuten `int(123)` ja `str(input(…))`, joilla ei ole vaikutusta.

3.8. Yksittäisiä huomioita

1. Ohjelmarivit on sisennettävä loogisesti samalla tyylillä koko ohjelmassa. Suositus on sisentää koodia aina neljää (4) välilyöntiä, joka onnistuu useimmissa koodieditoreissa sarkainnäppäimellä, esim. IDLE ja Visual Studio Code
2. Tulosteissa rivinvaihtomerkit tulee olla merkkijonon loppussa tai omina erillisinä käskyinäTiedot tulee kysyä käyttäjältä `input`-käskyllä ja tarvittaessa syöte muutetaan oikeaan tietotyyppiin heti kysymisen jälkeen seuraavalla rivillä esim. `int` tai `float`-käskyllä
3. Lähtökohtaisesti toistorakenteen tulee päättyä normaalisti, jotta sen jälkeen olevat lopetusrutiinit tulevat suoritettua. Normaali lopetus tarkoittaa, että kaikki läpikäytävät arvot on käsitelty tai ohjelman suoritus päättyy poikkeustenkäsittelijään, ks. kohta 3.6
4. Askeltavaa toistoa (`for`) käytetään, kun käydään läpi listaa tai kierrosmäärä on tiedossa etukäteen. `while`-rakennetta käytetään, kun läpikäytävien tietojen lukumäärä ei ole tiedossa etukäteen tai lopetusehtoja on useita
5. Rekursiota ei tule käyttää, ellei tehtäväksiannossa ole niin erikseen kehotettu
6. Mikäli ohjelman suorituksessa tulee ongelmia, tulee ohjelman kertoa käyttäjälle selkeästi, mikä ongelma on ja miten sen voi ratkaista. Tämä koskee erityisesti virheiden ennaltaehkäisyä ja poikkeustenkäsittelyä, ks. kohdat 2.5 ja 3.6.