CT10A0013 Ohjelmointi Pythonilla

L07: Rakenteiset tietorakenteet

Uolevi Nikula

Päivän asiat

- Teoria: rakenteinen tietorakenne
  - Dynaaminen tietorakenne, useita samanlaisia tietoalkioita – lista
  - Yhteenkuuluvia tietoalkioita, erilaisten tietojen yhdistäminen – luokka/olio
- Käytäntö
  - Lista ja luokka/olio
  - Tiedosto ja lista
  - Merkkijonojen käsittely
- Koodiesimerkkejä
- Ohjelmointitehtävien ratkaiseminen
- Lopuksi

Ohjelman alkukommentit / otsikkotiedot

- Laita jokaiseen CG:hen palauttamaasi tehtävään tästä viikosta alkaen seuraavat tiedot alkukommenttiin eli otsikkotietoihin Koodiesimerkit-kalvoilla olevat alkukommentit asianmukaisesti täytettyinä. Katso tiedosto Moodlessa, otsikkotiedot.py
  - Tekijä: oma nimi
  - Opiskelijanumero: oma opiskelijanumero
  - Päivämäärä: ohjelman tekopäivämäärä
  - Oppimateriaalien ulkopuoliset yhteistyö ja lähteet, nimi/lähde ja miten vaikuttanut työhön
- Kaikki kurssin tehtävät ovat henkilökohtaisia eli ne tulee tehdä itse ja tässä voi ilmoittaa, jos joku muu on vaikuttanut koodiin niin, että se voi tulla ilmi esim. vilppitarkistuksissa
- Otsikkotietojen puute voi johtaa toisella periodilla pisteiden nollaamiseen ko. tehtävän osalta
  - Tällä viikolla harjoitellaan asiaa eikä tietojen puuttumisesta menetä pisteitä

Teoria

Uusia rakenteisia tietorakenteita

- Dynaaminen tietorakenne – lista
- Rakenteinen tietoalkio – luokka/olio

Dynaaminen tietorakenne

- Dynaamisuus tarkoittaa sitä, että **tietorakenne** voi muuttua ajon aikana
  - Se voi kasvaa eli siihen voi lisätä alkioita
  - Se voi pienentyä eli siitä voi poistaa alkioita
  - Se tietää, kuinka monta alkiota siinä kulloinkin on
- Pythonissa on useita dynaamisia tietorakenteita, joista **lista** on yleisin
- Kokonaisluku-listan voi muodostaa helposti range-funktion avulla
  - range(4) –käsky tuottaa järjestetyn joukon eli periaatteessa listan [0, 1, 2, 3]
- Lista-muuttuja määritellään seuraavasti
  - lista = []
- Lista-tyyppisellä muuttujalla on jäsenfunktioita, joilla listaa voi muokata ajon aikana eli dynaamisesti

Listan jäsenfunktioita

- Lista on luokka, jolla on tietojen eli **jäsenmuuttujien** lisäksi mm. seuraavat **jäsenfunktiot** eli metodit
  - .append(x) – lisää alkion x
  - .insert(i, x) – lisää alkion x kohtaan i
  - .remove(x) – poistaa alkion x
  - .sort() – järjestää alkiot arvojärjestykseen
  - .reverse() – kääntää listan alkiot päinvastaiseen järjestykseen eli ensimmäinen viimeiseksi jne.
- Tarkempia tietoja löytyy ohjelmointioppaasta ja muista dokumenteista, erityisesti Python-dokumentaatiosta (F1)

Rakenteinen tietoalkio

- Ohjelmissa käsitellään usein monia tietoja, jotka liittyvät samaan asiaan, esim.
  - Ihminen: etunimi, sukunimi, puhelinnumero, …
  - Potilas: nimien lisäksi henkilötunnus, sukupuoli, ikä, pituus, …
  - Myytävä asunto: hinta, osoite, huoneiden lukumäärä, …
- Tällaisia tietoja voidaan yhdistää rakenteiseksi tietoalkioksi
  - Yksi muuttuja pystyy tällöin toimimaan lähtökohtana useisiin yhteen liittyviin tietoihin
- Pythonissa, ja muissa olio-ohjelmointikielissä, tällaista tietoalkiota kutsutaan **olioksi**, jotka määritellään **luokan** avulla

Luokka-tietorakenne

- Luokka voi sisältää useita tietoja eli muuttujia, jotka määritellään samalla tavalla kuin tavallisetkin muuttujat
  - Luokan jäseniin viitataan pistenotaatiolla eli jäsenmuuttujien avulla
- Esimerkiksi
  - Määritellään luokka jäsenmuuttujilla ja niiden oletusarvoilla (OPISKELIJA)
  - Tehdään luokasta esiintymiä eli olioita eli instansseja (Fuksi)
  - Viitataan olion tietoihin jäsenmuuttujia käyttäen, ts. pistenotaatiolla

```
class OPISKELIJA: # määritellään luokka
      Etunimi = "Teemu"  # ja muuttujat oletusarvoilla
      Sukunimi = "Teekkari"
      Numero = 1

Fuksi = OPISKELIJA() # luodaan olio eli luokan instanssi
Fuksi.Etunimi = "Erkki" # tietoja voidaan muuttaa normaalisti
print(Fuksi.Etunimi, Fuksi.Sukunimi, Fuksi.Numero)
```

Tietorakenne luokka ja struct

- Useissa perinteisissä proseduraalisissa ohjelmointikielissä ohjelmoija voi määritellä haluamansa tietorakenteen varatulla sanalla, esim. **struct**
- Pythonissa ei ole pelkästään tietojen yhdistämiseen käytettävää rakennetta, mutta olio-ohjelmointikielenä Pythonissa on **luokka** eli **class** –rakenne
- Olio-ohjelmoinnin luokka-rakenteen avulla voidaan yhdistää sekä tietoa että aliohjelmia **jäsenmuuttujien** ja **jäsenfunktioiden** avulla
- Tähän mennessä olemme käyttäneet Pythonissa jäsenfunktioita
  - tiedostonkäsittelyn yhteydessä: write, readline, close
  - merkkijonojen yhteydessä: split, format, upper, isdigit, …

Olio-ohjelmoinnin terminologiaa

- **Luokka** – olion prototyyppi, vrt. kakkumuotti, eli ohjelmissa voidaan luokka-rakennetta hyväksi käyttäen luoda olioita
  - Tällä kurssilla luokan nimi kirjoitetaan ISOILLA KIRJAIMILLA, jotta se on helpompi erottaa oliosta
- **Olio** – ohjelmassa varsinainen käytettävä muuttuja eli luokan ilmentymä, instanssi
- **Jäsenfunktio** eli metodi – luokkaan/olioon kuuluva aliohjelma
- **Jäsenmuuttuja** eli attribuutti – luokkaan/olioon kuuluva muuttuja
  - Olion jäsenfunktioita ja jäsenmuuttujia käytetään pistenotaatiolla eli tyyliin *Olio.Jasenmuuttuja* tai *Olio.Jasenfunktio()*, vrt. Tiedosto.close()

Muita rakenteisia tietorakenteita

- Listan lisäksi muita Python-kielessä määriteltyjä rakenteisia tietorakenteita ovat tuple ja sanakirja
  - Erilaiset tietorakenteet sopivat erilaisiin tarpeisiin
  - Tässä vaiheessa keskitytään **listaan** ja **luokkaan**
  - Sanakirjaan palataan myöhemmin, se on listan tyylinen dynaaminen tietorakenne
  - Tuple on periaatteessa lista, jota ei voi muokata, ks. tarkemmin opas
- Huom. Mikäli määrittelet listan ja saat esim. seuraavan virheilmoituksen
  - “`AttributeError: 'tuple' object has no attribute 'append'`”
  - Katso tarkemmin oppaan luku 7. Olet todennäköisesti määritellyt listan sijaan tuplen – **lista määritellään hakasuiluilla []** ja tuple kaarisuluilla ()

Arvoparametri vs. muuttujaparametri

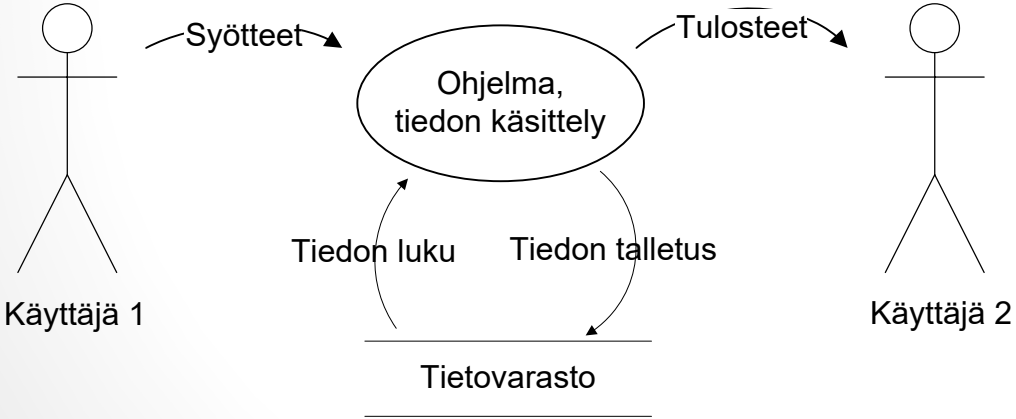
- Pythonissa yksinkertaiset tietotyypit kokonaisluku, liukuluku ja merkkijono ovat aliohjelmissa **arvoparametreja**
  - Aliohjelmissa arvoparametreihin tehdyt muutokset eivät näy kutsuvassa ohjelmassa, koska arvoparametrit ovat alkuperäisten muuttujien kopioita
- Lista on rakenteinen tietotyyppi ja siksi Python välittää aliohjelmiin listan osoitteen, jolloin kyseessä on **muuttujaparametri**
  - Muuttujaparametriin aliohjelmassa tehdyt muutokset näkyvät kutsuvassa ohjelmassa
- Tällä kurssilla arvo- ja muuttujaparametrit käsitellään samalla tavalla yksinkertaisuuden vuoksi
  - **Tieto aliohjelmiin välitetään aina parametrinä ja jos parametrin arvo muuttuu aliohjelmassa, tulee se palauttaa aliohjelmasta paluuarvona**
- Arvo- ja muuttujaparametrit on selitetty tarkemmin oppaan luvussa 7

Käytäntö

Perusohjelman toiminnot
Tiedosto ja lista
Algoritmi valikkopohjaisen ohjelman tekemiseen
Merkkijonojen käsittely

Perusohjelman toiminnot

- Perusohjelma sisältää kolme perustoimintoa
  - **Käyttäjäinteraktion** eli tiedon vaihdon käyttäjän kanssa (syöttö & tulostus; 1 tai useampi käyttäjää)
  - **Tiedon käsittely** itse ohjelmassa
  - **Tiedostonkäsittely** eli tallennus tietovarastoon ja sen luku sieltä
- Nämä kaikki on nyt käyty läpi



- Käyttöliittymä/UI: ”valikko”
- Aliohjelmat: ”laske”, ”analysoi”,…
- Aliohjelmat: ”tallenna”, ”lue”

Tiedoston ja listan käyttö

- L06 tehtävissä lähtökohtaisesti aliohjelma avasi tiedoston, luki ja teki pyydetyn käsittelyn sekä sulki tiedoston
  - Tämä on hidas ja tehoton toimintapa eikä niin kannata toimia enää
- Tyypillisessä toimivassa ohjelmassa on
  - **Aliohjelma tiedoston lukemista varten**, joka tallentaa tiedot *listaan*, tyypillisesti *olioina* eli tiedot jäsenmuuttujiin talletettuina
  - **Tiedonkäsittelyyn tarvittavat aliohjelmat**, yksi tai useita, jotka välittävät tietoa aliohjelmien välillä *listalla* tai useilla
  - **Aliohjelma tiedoston tallentamista varten**, joka tallentaa oheismuistille tarvittavat tiedot *listalta*
  - **Listoja välitetään aliohjelmien välillä tarpeen mukaan *parametreina* ja *paluuarvoina***

Algoritmi valikkopohjaisen ohjelman tekemiseen

1. **Tulosta valikko** /valinnat print-lauseilla – tämä on ohjelman ”käyttöliittymä”
   - Sisältö esimerkistä tai toimeksiannosta, testaa, Valikko()-aliohjelma
2. **Toteuta toisto- ja valintarakenteet** pääohjelmassa
   - Kaikki valikon vaihtoehdot käydään läpi ml. lopetus
   - Osoita valinta tulostamalla valittu toiminto tekstinä näytölle, testaa kaikkien valintojen toimivuus
3. Vaihda valintarakenteen tulostuskäskyt **aliohjelmakutsuiksi**
   - Siirrä tulostuskäskyt aliohjelmiin ja testaa, että ohjelma käy kaikissa aliohjelmissa
4. Toteuta **aliohjelmien edellyttämä tiedonsiirto** eli parametrit ja paluuarvot
   - Mieti aliohjelmien tarvitsemat ja tuottamat tiedot, lisää parametrit ja paluuarvot sekä tulosta ne aliohjelmien alussa ja lopussa, testaa
5. Lisää jokaiseen **aliohjelmaan** siihen liittyvä varsinainen **toiminnallisuus**
   - Toteuta tarvittava toiminnallisuus, esim. lue/kirjoita tiedosto, kysy tiedot, laske tms. ja testaa jokainen aliohjelma aina heti sen toteuttamisen jälkeen

Merkkijonojen jako osiin

- Merkkijonoja voidaan jakaa osiin
  - `merkkijono[0:2:1]` –notaatiolla merkkejä voidaan erottaa niiden indeksien perusteella, esim. 2 ensimmäistä merkkiä
  - merkkijono voidaan jakaa osiin tietyn merkin kohdalta, ns. ”erotinmerkki”, joka on Suomessa tyypillisesti puolipiste esim.
    - `Etunimi;Sukunimi;Ika;Sukupuoli;Paino;Pituus`
  - **merkkijonon jako osiin onnistuu split-käskyllä**, joka palauttaa **listan**
    - `sarakkeet = merkkijono.split(';')`
  - tämän jälkeen sarakkeisiin voidaan viitata **listan** alkioina, esim.
    - `sarakkeet[0] == ”Etunimi”`
- Katso tarkemmin koodiesimerkeistä

Koodiesimerkkejä

Otsikkotiedot
Listan peruskäyttö
Luokka, olio, lista, format ja split peruskäyttö
Valikkopohjainen ohjelma *listalla*

otsikkotiedot.py

```
##########################################################################
# 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
```

Listan perusoperaatiot

```
Tiedot = [] # lista-tyyppisen muuttujan määrittely

for i in range(10):
    Tiedot.append(i) # tietojen lisäys listaan, append

for Alkio in Tiedot: # listan alkioiden läpikäynti
    print(Alkio)

Tiedot.clear() # listan tyhjennys
```

Tiedosto-lista-tiedonvälitys

```
def kirjoitaTiedosto(Nimi, Data):
    Tdsto = open(Nimi, "w")
    for Alkio in Data:
        Tdsto.write(str(Alkio)+'\n')
    Tdsto.close()
    return None

def lueTiedosto(Nimi, Data):
    Tdsto = open(Nimi, "r")
    Rivi = Tdsto.readline()
    while (len(Rivi) > 0):
        Data.append(int(Rivi))
        Rivi = Tdsto.readline()
    Tdsto.close()
    return Data
```

```
def paaohjelma():
    TiedostoNimi = "L07Demo.txt" # kovakoodattu
    Tiedot = []
    for i in range(10):
        Tiedot.append(i)

    print(Tiedot)
    kirjoitaTiedosto(TiedostoNimi, Tiedot)
    Tiedot.clear()
    print(Tiedot)
    Tiedot = lueTiedosto(TiedostoNimi, Tiedot)
    print(Tiedot)
    Tiedot.clear()
    return None

paaohjelma()
```

Luokka, olio, lista, format ja split peruskäyttö

```
# Tämä koodi kuuluu oikealla olevan koodin yläpuolelle
# Luokan määrittely
class HENKILO:
    Nimi = ""
    Ika = 0

# Olioiden/muuttujien määrittely, jäsenmuuttujat
Fuksi1 = HENKILO()
Fuksi1.Nimi = "Kalle"
Fuksi1.Ika = 20
Fuksi2 = HENKILO()
Fuksi2.Nimi = "Kille"
Fuksi2.Ika = 18
```

```
# Lista-muuttujan määrittely ja olioiden lisäys,
# olio-lista
Kurssilaiset = []
Kurssilaiset.append(Fuksi1)
Kurssilaiset.append(Fuksi2)

Rivit = [] #Listan määrittely merkkijonoja varten
# Listan läpikäynti
for Oppilas in Kurssilaiset:
    Rivi = "Nimi;{0:s};Ika;{1:d}".format(
                Oppilas.Nimi, Oppilas.Ika)
    print(Rivi)
    Rivit.append(Rivi)
Kurssilaiset.clear() # Olio-listan tyhjennys

# Merkkijonolistan läpikäynti, rivitiedot
# jäsenmuuttujiksi
for Rivi in Rivit:
    Sarakkeet = Rivi.split(';')
    Oppilas = HENKILO()
    Oppilas.Nimi = Sarakkeet[1]
    Oppilas.Ika = int(Sarakkeet[3])
    Kurssilaiset.append(Oppilas)
Rivit.clear() # Merkkijonolistan tyhjennys

# Olio-listan läpikäynti, tulostus ja tyhjennys
for Oppilas in Kurssilaiset:
    print(Oppilas.Nimi, Oppilas.Ika)
Kurssilaiset.clear()
```

Valikkopohjainen ohjelma listalla

```
def paaohjelma():
    Valinta = 1
    TiedostoLue = "L06Lue.txt"
    TiedostoKirjoita = "L07Kirjoita.txt"
    ListaSyote = []
    ListaTulos = []
    IndeksiMax = None
    Indeksi = 0
    while (Valinta != 0):
        Valinta = valikko()
        if (Valinta == 1):
            if (IndeksiMax == None):
                ListaSyote = lueTiedosto(TiedostoLue, ListaSyote)
                IndeksiMax = len(ListaSyote) - 1
            else:
                Indeksi += 1

            if (Indeksi <= IndeksiMax):
                Merkkijono = ListaSyote[Indeksi]
            else:
                print("Merkkijonot loppuivat, lopeta ohjelma.")
        elif (Valinta == 2):
            ListaTulos.append(Merkkijono)
            print("Lisätty listaan merkkijono '" + Merkkijono + "'.")
        elif (Valinta == 3):
            Merkit = Merkkijono[::-1]
            ListaTulos.append(Merkit)
            print("Lisätty listaan merkkijono '" + Merkit + "'.")
        elif (Valinta == 0):
            print("Lopetetaan.")
        else:
            print("Tuntematon valinta, yritä uudestaan.")
        print()
    tallennaTiedosto(TiedostoKirjoita, ListaTulos)
    ListaSyote.clear()
    ListaTulos.clear()
    print("Kiitos ohjelman käytöstä.")
    return None

paaohjelma()
```

```
def valikko():
    print("1) Lue merkkijono")
    print("2) Lisää listaan merkkijono etuperin")
    print("3) Lisää listaan merkkijono takaperin")
    print("0) Lopeta")
    Syote = input("Anna valintasi: ")
    Valinta = int(Syote)
    return Valinta

def lueTiedosto(Tiedosto, Lista):
    Tdsto = open(Tiedosto, "r")
    Rivi = Tdsto.readline()[:-1]
    while (len(Rivi) > 0):
        Lista.append(Rivi)
        Rivi = Tdsto.readline()[:-1]
    Tdsto.close()
    Tulosta = "Luettu tiedosto '" + Tiedosto + "'."
    print(Tulosta)
    return Lista

def tallennaTiedosto(Tiedosto, Lista):
    Tdsto = open(Tiedosto, "w")
    for Str in Lista:
        Rivi = Str + '\n'
        Tdsto.write(Rivi)
    Tdsto.close()
    Tulosta = "Tallennettu tiedosto '" + Tiedosto +
"'."
    print(Tulosta)
    return None
```

Ohjelmointitehtävien ratkaisemisesta

Ohjelmointitehtävien ratkaisemisesta 1

- **Miten ohjelmointiongelmia voi välttää ennalta ja ratkoa, jos välttäminen ei onnistunut?**

1. Opiskele teoria
   - Käy luennoilla, lue luentokalvot ja ohjelmointiopas, tutustu luento- ja ohjelmointivideoihin, vastaa viikkoväittämiin
2. Yritä tehdä tehtävät
   - Tee tutut asiat ensin: aloita selvittämällä syötteet ja tulosteet ja koodaa ne
   - Yritä ratkaista tietojenkäsittelyongelma
   - Pyydä apua tarvittaessa *tiettyyn* ongelmaan
     - Käy neuvontatilaisuuksissa
     - Laita kysymys keskustelupalstalle
     - Älä unohda tutustua asioita käsittelevään kirjallisuuteen!
       - Ohjelmointiin liittyen kurssin oppimateriaalit
       - Ratkaistava ongelma tyypillisesti google, esim. painoindeksi ja keskiarvo

Ohjelmointitehtävien ratkaisemisesta 2

- Mikäli et saa ratkaistua tehtävää, käy läpi seuraavat vinkit
  - Lue ***tehtäväksianto*** huolella, ts. *mitä* oikeasti pitää tehdä
  - Tarkista kirjallisuudesta, miten asia pitäisi ***yleisesti ottaen*** ratkaista; ratkaisun idea eli *miten*, esim. ympyrän pinta-alan laskenta/kaava
  - ***Toteuta ohjelma*** niin selkeästi, että tiedät miten se toimii; toteuta ratkaisu *virheittä*
  - ***Testaa ohjelmaa*** ja yritä paikallistaa ongelma, ts. yritä löytää missä kohdin ohjelma toimii ensimmäisen kerran väärin
  - Pyytäessäsi apua, kerro selkeästi ongelmasi
    - Keskustelupalstalle voi laittaa muutaman ongelmia tuottavan koodirivin mukaan ja selitä ongelma mahdollisimman selkeästi

Ohjelmointitehtävien ratkaisemisesta 3

- Mikäli et pääse alkuun tehtävässä
  - Lue tehtäväksianto, katso luennot, luentodemot, ohjelmointivideot
  - Kysy tarvittaessa keskustelupalstalta vinkkejä ja kerro, mitä saat/et saa tehtyä
- Mikäli ohjelma ei toimi
  - Laita toimimaton koodiosa keskustelupalstalle ja mahdollinen virheilmoitus
  - Kerro mikä toimii ja mikä ei toimi suunnitellulla tavalla
- Ohjelma toimii (IDLEllä), muttei mene CG:stä läpi
  - Laita CG:n testiajo ja virheilmoitus keskustelupalstalle
  - Kerro mikä toimii ja mikä ei toimi
- Mikäli ohjelmasi kirjoittaa tiedoston, tarkista
  - Tiedostonimen kirjoitusasu ml. pääte (.txt) ja isot/pienet kirjaimet
  - Katso tiedoston sisältö IDLEllä, ei Notepad, Word, tms.

Ohjelmien tekemisen perusongelmat

1. **Ei pääse alkuun, ei tiedä mitä tehdä ja miten**
   - Tyypillisesti tehtäväksiannosta löytyy
     - ohjelman rakenne eli tarvittavat pää- ja aliohjelmat
     - ohjelmien välillä lähetettävät parametrit ja paluuarvot
     - ohjelman keskeiset toiminnot
   - Jos ei löydy, voi arvata ja yrittää – tällä kurssilla pari ohjelmarunkoa
2. **Tekeminen ei onnistu: ohjelma/käsky/algoritmi ei toimi oikein**
   - Tyypillisesti ohjelmointivideolla on näytetty oleelliset uudet asiat, esim. pienin/suurin alkio, datan ryhmittely, split, format, …
   - Apua voi kysyä keskustelupalstalla – ehkä kysytty jo
   - Apua voi käydä kysymässä neuvontatilaisuuksissa

Lopuksi

Osaamistavoitteet

Ohjelmointivideot

Osaamistavoitteet

- Lista ja luokka/olio
- Listan käyttö aliohjelmassa eli parametri ja paluuarvo
- Ison ohjelman toteutus vaiheittain, valikkopohjainen ohjelma
- Ohjelmointitehtävien ongelmien välttäminen ja ratkaiseminen
- Tekstitiedostossa olevan data luku ja kirjoitus
  - Merkkijonon pilkkominen osiin listaksi split-käskyllä
  - Merkkijonon muodostaminen format-käskyllä

Ohjelmointivideot

- **Katso ohjelmointioppaasta kokoava esimerkki 7.10**
- Ohjelman teko tyhjästä – valikkopohjaisen perusohjelman toteutus
  - Huomaa toteutus useissa vaiheissa välillä testaten
  - Tiedostoon talletetaan tietoa riveille puolipisteellä eroteltuna, huomaa tiedoston luku, rivin paloittelu tietoalkioiksi ja niiden sijoittaminen olioihin sekä listaan
  - Tiedostossa olevien tietojen luku listaan, listalla olevien tietojen tallennus tiedostoon ja listan välitys parametrina ja paluuarvona

Täydennyksiä oppaan lukuun 7

Tyyliohjeita pienille Python-ohjelmille
ASPAn tarkistukset
Oppaan esimerkit ja käsitellyt asiat

Pienen Python-ohjelman tyyliohjeet 1

- Luokat määritellään globaaleiksi kiintoarvojen jälkeen ennen aliohjelmia
- Luokan nimet kirjoitetaan suuraakkosin, esim. class OPISKELIJA, jotta ne on helpompi erottaa olioista, jotka nimetään samalla tavoin kuin muutkin muuttujat, esim. Opiskelija1
- Tällä kurssilla tieto menee aliohjelmiin aina parametreillä
- Tällä kurssilla tieto aliohjelmista tulee aina palauttaa paluuarvona
  - Näin kannattaa toimia myös muuttujaparametrien kanssa virheiden välttämiseksi
- Dataa sisältävän tekstitiedoston tyypillinen muoto on seuraava
  - Yhdellä rivillä on yhteenkuuluvat tiedot erotettuina sarakkeisiin erotinmerkillä, esim. yhden henkilön tiedot, yhden tunnin kulutus-/tuottotiedot jne.
  - Eri riveillä on eri tapausten tietoja – eri henkilöiden tietoja, eri tuntien tietoja jne.
  - Kun tiedosto luetaan, tulee yhdestä rivistä yksi alkio merkkijonoja sisältävään listaan tai yksi olio oliolistaan
  - Kun tiedosto kirjoitetaan, yhden olion tiedot menevät yhdelle riville tiedostoon
- Ohjelmat lukevat datatiedoston yhdessä aliohjelmassa listaan ja käyttävät jatkossa listaa
- Ohjelman päättyessä, tai käyttäjän halutessa, listalla olevat tiedot kirjoitetaan tiedostoon
- Ohjelman suorituksen aikana suuret datamäärät pidetään tyypillisesti listassa

Pienen Python-ohjelman tyyliohjeet 2

- Oliolista
  - Tarkoittaa tällä kurssilla, että luodaan olioita ja ne kaikki lisätään listaan erillisiä olioina, jolloin olioista muodostuu lista eli oliolista (esimerkin 7.10 mukaisesti)
- Lista, luokka, olio
  - Katso suositeltu käyttö oppaan esimerkistä 7.10
  - Esim. tiedostoa luettaessa toistorakenteessa
    - Luo luokasta uusi olio ja aseta sen jäsenmuuttujille arvot
    - Lisää olio listaan

ASPAn L07 tarkistukset

- Listoihin liittyen ASPAssa ei ole erillisiä tarkastuksia
  - Yleisin ongelma listan kanssa on sen määrittely globaalina muuttujana
    - ASPA varoittaa globaaleista muuttujista L05 mukaisesti
  - Tällä kurssilla tulee käyttää listaa (tai muita dynaamisia tietorakenteita), mikäli samanlaisia muuttujia on yli 5 kpl
    - Henkilökunta tarkistaa
- Luokkiin liittyen ASPA tarkistaa seuraavat asiat
  - Luokat pitää määritellä globaaleina kiintoarvojen jälkeen ennen aliohjelmia
  - Luokan nimi tulee kirjoittaa isoilla kirjaimilla, esim. class AUTO:
  - Luokasta pitää aina tehdä olio, tai useita, ja sijoittaa arvot niiden jäsenmuuttujiin

Käsitellyt asiat oppaan luvussa 7

- Lista: Esimerkki 7.1, 7.2, 7.3, 7.4, 7.8, 7.10
- Luokka/olio: Esimerkki 7.6, 7.7, 7.8, 7.10
- Parametrit ja paluuarvo: Esimerkki 7.3, 7.4, 7.5, 7.10
- Tuple: Esimerkki 7.9

- **Huom. Kokoava esimerkki 7.10, valikkopohjainen ohjelma listalla, luokalla ja olioilla**