Ohjelmoinnin perusteet Y Python

T-106.1208

8.2.2010

Debuggeri

- Tyypillinen tilanne: ohjelma on kirjoitettu, Python-tulkki ei valita virheistä, mutta ohjelma kuitenkin toimii jotenkin väärin.
- Tällöin ohjelman toimintaa voi tutkia ja virhettä yrittää etsiä *debuggerin* avulla.
- Debuggeri on ohjelma, jonka avulla ohjelmaa voi ajaa käsky kerrallaan ja tutkia samalla ohjelman käyttämien muuttujien arvoja.
- Ohjelmaan voi asettaa pysähdyspisteitä (breakpoint), joissa ohjelma pysähtyy.
- Eclipseen kuuluu debuggeri, jonka avulla voi ajaa mm. Python-ohjelmia.

Debuggerin käytttö Eclipsessä

- Siirry ensin Debug-perspektiiviin painamalla ikkunan oikeasta yläkulmasta `Debug`-painiketta (jos se ei ole näkyvissä, paina yläkulmassa näkyvää kaksoisnuolta tai valitse valikosta `Window->Open Perspective->Other->Debug`).
- Ennen debuggerin käynnistymistä kannattaa asettaa ensimmäinen pysähdyspiste. Valitaan haluttu rivi koodista, painetaan rivin vasemmassa reunassa hiiren oikeaa painiketta ja valintaan esiin tulevasta valikosta `Add breakpoint`.
- Tämän jälkeen debuggerin voi käynnistää valitsemalla `Run->Debug As->Python Run`
- Eclipse ajaa ohjelmaa ensimmäiseen pysähdyspisteeseen saakka. Tämän jälkeen ohjelmassa voi edetä rivi kerrallaan valitsemalla `Run->Step Over`.

Debuggerin käytttö Eclipsessä, jatkoa

- Jos rivillä on funktion kutsu, ja haluaa siirtyä funktion sisälle, on valittava `Run->Step Into`.
- Muuttujien arvot näkyvät joka vaiheessa oikeassa yläkulmassa olevalla välilehdellä.
- Debuggauksesta pääsee takaisin ohjelman editointiin oikeassa yläkulmassa olevan `Debug`-painikkeen viereisestä nuolesta. (Valitse `Pydev`)

Listat

- Esimerkki: halutaan kirjoittaa ohjelma, joka lukee käyttäjältä 30 lämpötilaa.
- Kun lämpötilat on luettu, ohjelma tulostaa luetut lämpötilat ja niiden keskiarvon.
- Tähän asti esitellyillä välineillä tarvittaisiin 30 eri muuttujaa lämpötilojen tallentamiseen.
- Ratkaisu: käytetään *listaa*.
- Lista on rakenne, johon voidaan tallentaa useita eri arvoja.
- Indeksin avulla voidaan määrätä, mitä listan alkiota käsitellään. Esimerkiksi listan `lampotilat` viidettä alkiota päästään käsittelemään kirjoittamalla `lampotilat[4]`.

Listan luominen ja alkioiden lisäys

- Tyhjä uusi lista luodaan kirjoittamalla esimerkiksi

  `lampotilat = []`

  (jolloin `lampotilat` on luotuun listaan viittaavan muuttujan nimi).
- Listan loppuun voi lisätä uuden arvon kirjoittamalla

  `lampotilat.append(arvo)`
- Jos listassa on jo vähintään `i+1` alkiota, voi indeksille `i` sijoittaa uuden arvon kirjoittamalla

  `lampotilat[i] = arvo`

  Tällöin indeksillä `i` ollut vanha arvo häviää.
- Indeksit ovat kokonaislukuja ja listan ensimmäisen alkion indeksi on aina 0.
- Listan yksittäistä alkiota voi käyttää esimerkiksi tulostuskäskyissä tai lausekkeissa:

```
print "5. lampotila", lampotilat[4]
summa = summa + lampotilat[4]
```

Listan läpikäynti

- Listan kaikki alkiot on helppo käydä läpi toistokäskyn avulla. Esimerkiksi seuraava while-käsky laskee yhteen kaikki listan alkiot, jos listassa on `LKM` alkiota.

```
summa = 0
i = 0
while i < LKM:
    summa += lampotilat[i]
    i += 1
```

- Läpikäynti on vielä helpompaa for-käskyn avulla:

```
summa = 0.0
for astemaara in lampotilat:
    summa += astemaara
```

Muuttuja `astemaara` saa arvokseen vuorotellen jokaisen listan alkion.

Lämpötilaesimerkki, koodi

```
def main():
    LKM = 30
    lampotilat = []
    i = 0
    print "Anna", LKM, "lampotilaa"
    while i < LKM:
        rivi = raw_input("Seuraava lampotila: ")
        lampo = float(rivi)
        lampotilat.append(lampo)
        i += 1
```

Lämpötilaesimerkki, koodi jatkuu

```
    summa = 0.0
    print "Annetut lampotilat"
    for arvo in lampotilat:
        print arvo
        summa += arvo
    keskiarvo = summa / LKM
    print "Lampotilojen keskiarvo", keskiarvo


main()
```

Alkiot listaan listaa luodessa

- Kun luodaan uusi lista, sen ei tarvitse välttämättä olla tyhjä, vaan listaan kuuluvat alkiot voidaan antaa samalla, esimerkiksi

```
numerolista = [2, 4, 6, 8]
nimilista = ['Matti Virtanen', 'Maija Maki', 'Anu Lahti']
```

- Voidaan myös luoda lista, jossa on aluksi haluttu määrä nollia.

```
LKM = 30
lampotilat = [0.0] * LKM
```

Tällöin listaan sijoitettavat lämpötilat voidaan antaa sijoituskäskyllä eikä append-käskyä tarvita:

```
i = 0
while i < LKM:
    rivi = raw_input("Seuraava lampotila: ")
    lampo = float(rivi)
    lampotilat[i] = lampo
    i += 1
```

Listan pituus käyttäjältä

- Luotavan listan pituuden voi myös pyytää käyttäjältä.

```
vast = raw_input("Anna lampotilojen lukumaara")
lkm = int(vast)
lampotilat = [0.0] * lkm
```

- Listan pituuden saa selville Pythonissa valmiina olevalla funktiolla `len`, joten sitä ei välttämättä tarvitse säilyttää muuttujassa listan luonnin jälkeen:

```
i = 0
while i < len(lampotilat):
    rivi = raw_input("Seuraava lampotila: ")
    lampo = float(rivi)
    lampotilat[i] = lampo
    i += 1
```

Merkkijonoja sisältävä lista

- Listan alkioiden ei tarvitse välttämättä olla lukuja, vaan alkioina voi olla esimerkiksi merkkijonoja.
- Saman listan eri alkiot voivat olla keskenään myös erityyppisiä.
- Listan alkiona voi olla myös toinen lista.
- Merkkijonoja sisältävät listat ovat käteviä esimerkiksi silloin, kun käyttäjälle halutaan tulostaa erilaisia valintavaihtoehtoja ja käsitellä käyttäjän valitsemaa vaihtoehtoa.
- Seuraavan kalvon esimerkkiohjelma tulostaa käyttäjälle ruokalistan ja sen jälkeen käyttäjän valitseman ruokalajin.

```
def main():
    ruokalajit = ['makaronilaatikko', 'lihapullat',\
                  'lihakeitto', 'kasvispata',\
                  'pihvi']
    print "Valitse ruokalaji:"
    i = 0
    while i < len(ruokalajit):
        print "%d. %s" % (i + 1, ruokalajit[i])
        i += 1
    vastaus = raw_input()
    valinta = int(vastaus)
    if valinta > 0 and valinta <= len(ruokalajit):
        print "Valitsit ruuan", ruokalajit[valinta - 1]
    else:
        print "Huono valinta"

main()
```

Esimerkki: palkkalista

- Seuraavassa esimerkkiohjelmassa luetaan ensin työntekijöiden työtunnit listaan ja tulostetaan sitten kunkin työntekijän kokonaispalkka.
- Työntekijöiden lukumäärä ja tuntipalkka pyydetään käyttäjältä.
- Kiinnitä huomiota erityisesti seuraaviin asioihin:
  - Listan indeksien läpikäynti for-käskyn, `range`-funktion ja listan pituuden avulla.
  - Tulostuksen muotoilun käyttäminen `raw_input`-käskyssä.
- Ohjelma on (vähän muutettuna) Gaddisin oppikirjasta.

```
def main():
    syote = raw_input("Anna tyontekijoiden lukumaara: ")
    lkm = int(syote)
    tunnit = [0] * lkm
    syote = raw_input("Anna tuntipalkka (euroa): ")
    tuntipalkka = float(syote)

    for i in range(lkm):
        syote = raw_input("Tyontekijan %d tunnit " % (i + 1))
        tunnit[i] = int(syote)

    for i in range(lkm):
        palkka = tunnit[i] * tuntipalkka
        print "Tyontekijan %d palkka: %.2f euroa" %\
              (i + 1, palkka)

main()
```

Lista parametrina ja funktion palauttamana arvona

- Edellä esitettyä lämpötiloja käsittelevää ohjelmaa voi selkiyttää jakamalla sen useampaan funktioon.
- Tällöin kuitenkin tiedon käsiteltävistä lämpötiloista pitää siirtyä eri funktioiden välillä.
- Tähän voidaan käyttää parametreja ja funktion paluuarvoa.
- Funktio voi palauttaa arvonaan listan ja funktiolle voidaan antaa parametrina lista.
- Jos funktio tekee muutoksia parametrina saadun listan sisältöön, näkyvät muutokset myös silloin, kun samaa listaa käytetään funktion ulkopuolella.

Esimerkki: listan palauttava funktio

```
def kysy_lampotilat():
    LKM = 30
    lampotilat = [0.0] * LKM
    i = 0
    print "Anna", LKM, "lampotilaa"
    while i < LKM:
        rivi = raw_input("Seuraava lampotila: ")
        lampo = float(rivi)
        lampotilat[i] = lampo
        i += 1
    return lampotilat
```

Esimerkki: lista funktion parametrina

```
def tulosta_lampotilat(lammot):
    print "Annetut lampotilat"
    for arvo in lammot:
        print arvo


def laske_keskiarvo(lampotilalista):
    summa = 0.0
    for lampotila in lampotilalista:
        summa += lampotila
    lukumaara = len(lampotilalista)
    if lukumaara > 0:
        keskiarvo = summa / lukumaara
    else:
        keskiarvo = 0.0
    return keskiarvo
```

Esimerkki: pääohjelma

```
def main():
    lampolista = kysy_lampotilat()
    tulosta_lampotilat(lampolista)
    k_arvo = laske_keskiarvo(lampolista)
    print "Lampotilojen keskiarvo", k_arvo


main()
```

Ehdon tarkistus listan läpikäynnissä

- Hyvin usein halutaan etsiä listasta kaikki jonkin ehdon täyttävät arvot tai niiden lukumäärä.
- Esimerkiksi lämpötilalistasta halutaan annetun rajan ylittävien tai yhtäsuurten lämpötilojen lukumäärä.
- Kirjoitetaan toistokäsky, joka käy kaikki listan alkiot läpi. Toistokäskyn sisälle kirjoitetaan if-käsky, joka tarkistaa, onko haluttu ehto käsiteltävän alkion kohdalla tosi. Jos on, kasvatetaan löydettyjen alkioiden lukumäärää.
- Kun koko lista on käyty läpi, palautetaan löydettyjen alkioiden lukumäärä. Huomaa, että palautuskäsky pitää olla toistokäskyn ulkopuolella.

Rajan ylittävät lämpötilat, koodi

```
def montako_yli_rajan(lampotilojenLista, raja):
    ylittavien_maara = 0
    for asteet in lampotilojenLista:
        if asteet >= raja:
            ylittavien_maara += 1
    return ylittavien_maara
```

Halutuimman arvon etsiminen listasta

- Usein toistuva ongelma on myös löytää listasta halutuin arvo, esimerkiksi suurin lämpötila.
- Käytetään apumuuttujaa, johon tallennetaan tähän asti löydetty paras arvo.
- Aluksi apumuuttujan arvoksi alustetaan joko mahdollisimman huono arvo tai listan ensimmäisen alkion arvo.
- Käydään listan alkiot läpi toistokäskyssä. Jos tällä kierroksella tarkasteltava alkio on parempi kuin apumuuttujan arvo, vaihdetaan sen arvo apumuuttujan arvoksi.
- On myös jotenkin käsiteltävä se tilanne, että lista on tyhjä.
- Seuraavan kalvon esimerkkiohjelma etsii listasta suurimman lämpötilan.

Maksimilämpötilan etsiminen, koodi

```
def etsi_maksimi(lampotilalista):
    if len(lampotilalista) == 0:
        return None
    else:
        maksimi = lampotilalista[0]
        for lampotila in lampotilalista:
            if lampotila > maksimi:
                maksimi = lampotila
        return maksimi
```