Ohjelmoinnin perusteet Y Python

T-106.1208

25.3.2009

Listan alkiona viiteitä olioihin

- Halutaan tehdä ohjelma ohjelmointikurssien opiskelijoiden käsittelyyn.
- Yhden opiskelijan tietoja kuvataan yhdellä `Opiskelija`-oliolla.
- Tehdään lista, joka sisältää viitteet kurssin kaikkia opiskelijoita kuvaaviin `Opiskelija`-olioihin.

Olioviitteiden lisääminen listaan

- Listaan voi lisätä uuden `Opiskelija`-olion esimerkiksi seuraavasti:

```
opiskelijalista.append(Opiskelija("Matti Virtanen", \
                                  "11111F"))
```

- Toinen vaihtoehto on tehdä olion luonti ja sen lisääminen listaan eri käskyissä:

```
uusi_opiskelija = Opiskelija("Matti Virtanen", "11111F")
opiskelijalista.append(uusi_opiskelija)
```

- Jos listassa on jo vähintään `i+1` alkiota, voi olioviitteen sijoittaa myös suoraan paikalle `i`:

```
uusi_opiskelija = Opiskelija("Matti Virtanen", "11111F")
opiskelijalista[i] = uusi_opiskelija
```

Metodien kutsuminen listan alkioille

- Indeksillä `i` olevalle `Opiskelija`-oliolle voidaan kutsua `Opiskelija`-luokan metodeita seuraavasti:

```
print "Opiskelijan", opiskelijalista[i].kerro_nimi()
print "kurssiarvosana on", \
      opiskelijalista[i].laske_kokonaisarvosana()
```

- On myös mahdollista käydä koko lista läpi for-käskyllä ja kutsua metodeita vuorotellen jokaiselle listan alkiolle:

```
for kurssilainen in opiskelijalista:
    print "Opiskelijan", kurssilainen.kerro_nimi()
    print "kurssiarvosana on", \
          kurssilainen.laske_kokonaisarvosana()
```

`Opiskelija`-olioita listassa

- Seuraavilla kalvoilla on esimerkkiohjelma, joka lukee käyttäjältä opiskelijoiden tiedot, luo vastaavat `Opiskelija`-oliot ja sijoittaa ne listaan, pyytää listan opiskelijoille arvosanat ja tulostaa kurssin tulokset.
- Eri vaiheita varten on kirjoitettu omat funktionsa.
- Aluksi on jo aikaisemmin esitetty `Opiskelija`-luokan koodi.

Opiskelija-luokka

```
class Opiskelija:

    def __init__(self, annettu_nimi, numero):
        self.__nimi = annettu_nimi
        self.__opiskelijanumero = numero
        self.__tenttiarvosana = 0
        self.__harjoitusarvosana = 0

    def kerro_nimi(self):
        return self.__nimi

    def kerro_opiskelijanumero(self):
        return self.__opiskelijanumero
```

Opiskelija-luokka jatkuu

```
def kerro_tenttiarvosana(self):
    return self.__tenttiarvosana

def kerro_harjoitusarvosana(self):
    return self.__harjoitusarvosana

def muuta_tenttiarvosana(self, arvosana):
    if 0 <= arvosana <= 5:
        self.__tenttiarvosana = arvosana

def muuta_harjoitusarvosana(self, arvosana):
    if 0 <= arvosana <= 5:
        self.__harjoitusarvosana = arvosana
```

Opiskelija-luokka jatkuu

```
    def laske_kokonaisarvosana(self):
        if self.__tenttiarvosana == 0 or \
                self.__harjoitusarvosana == 0:
            arvosana = 0
        else:
            arvosana = (self.__tenttiarvosana +
                        self.__harjoitusarvosana + 1) / 2
        return arvosana

    def __str__(self):
        mjono = self.__nimi + ", " + \
                self.__opiskelijanumero + \
                ", tenttiarvosana: " + \
                str(self.__tenttiarvosana) + \
                ", harjoitusarvosana: " + \
                str(self.__harjoitusarvosana)
        return mjono
```

Opiskelija-olioita listassa, koodi

```
import opiskelija

def lue_kokonaisluku():
    luku_onnistui = False
    while not luku_onnistui:
        try:
            syote = raw_input()
            luku = int(syote)
            luku_onnistui = True
        except ValueError:
            print "Virheellinen kokonaisluku!"
            print "Anna uusi!"
    return luku
```

Opiskelija-olioita listassa, koodi jatkuu

```
def lue_opiskelijatiedot():
    opiskelijat = []
    print "Anna opiskelijoiden nimet ja opiskelijanumerot"
    print "samalla rivilla kauttaviivalla erotettuna."
    print "Lopeta tyhjalla rivilla."
    rivi = raw_input()
    while rivi != "":
        tiedot = rivi.split("/")
        if len(tiedot) != 2:
            print "Virheellinen rivi!"
        else:
            uusi = opiskelija.Opiskelija(tiedot[0], tiedot[1])
            opiskelijat.append(uusi)
        rivi = raw_input()
    print "Opiskelijoiden tiedot luettu!"
    return opiskelijat
```

Opiskelija-olioita listassa, koodi jatkuu

```
def lisaa_arvosanatiedot(opiskelijalista):
    for kurssilainen in opiskelijalista:
        print "Anna opiskelijan %s tenttiarvosana:" % \
              (kurssilainen.kerro_nimi())
        tentti_as = lue_kokonaisluku()
        print "harjoitusarvosana:"
        harjoitus_as = lue_kokonaisluku()
        kurssilainen.muuta_tenttiarvosana(tentti_as)
        kurssilainen.muuta_harjoitusarvosana(harjoitus_as)
    print "Arvosanat lisatty!"
```

Opiskelija-olioita listassa, koodi jatkuu

```
def tulosta_tulokset(opiskelijat):
    print "numero nimi                tentti harj   kurssi"
    for i in range(len(opiskelijat)):
        print "%-6s %-15s %-6d %-6d %-6d" % \
              (opiskelijat[i].kerro_opiskelijanumero(), \
               opiskelijat[i].kerro_nimi(), \
               opiskelijat[i].kerro_tenttiarvosana(), \
               opiskelijat[i].kerro_harjoitusarvosana(), \
               opiskelijat[i].laske_kokonaisarvosana())
```

Opiskelija-olioita listassa, koodi jatkuu

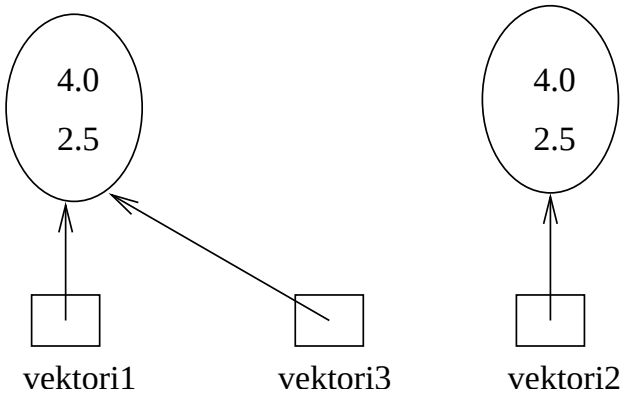
```
def main():
    opiskelijatiedot = lue_opiskelijatiedot()
    lisaa_arvosanatiedot(opiskelijatiedot)
    tulosta_tulokset(opiskelijatiedot)
    print "Ohjelma paattyy."


main()
```

Mitä tämä ohjelma tulostaa?

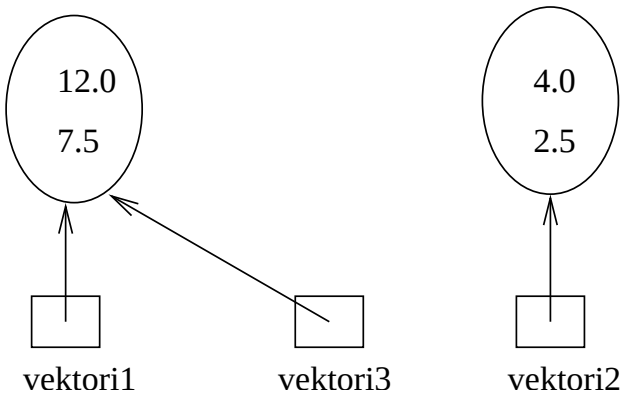
```
import tasovektori

def main():
    vektori1 = tasovektori.Tasovektori(4.0, 2.5)
    vektori2 = tasovektori.Tasovektori(4.0, 2.5)
    vektori3 = vektori1
    print vektori1
    print vektori2
    print vektori3
    vektori1.kerro_luvulla(3.0)
    print vektori1
    print vektori2
    print vektori3

main()
```

Useampi muuttuja viittaa samaan olioon

- Edellisen kalvon koodissa muuttujat `vektori1` ja `vektori2` viittaavat samaan olioon:



- Kun oliota muutetaan muuttujan `vektori1` kautta, näkyy muutos myös, kun samaa oliota katsotaan muuttujan `vektori3` kautta.

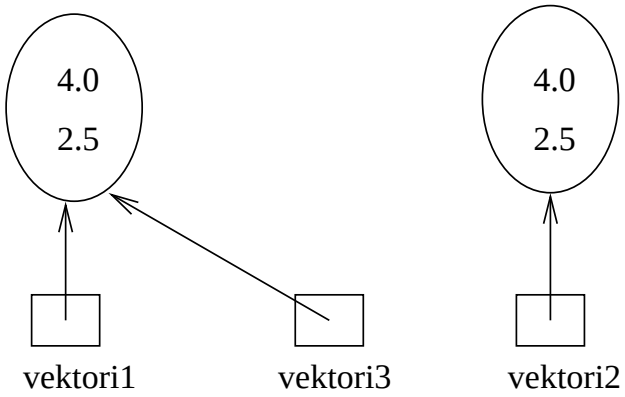


Mitä tämä ohjelma tulostaa 2?

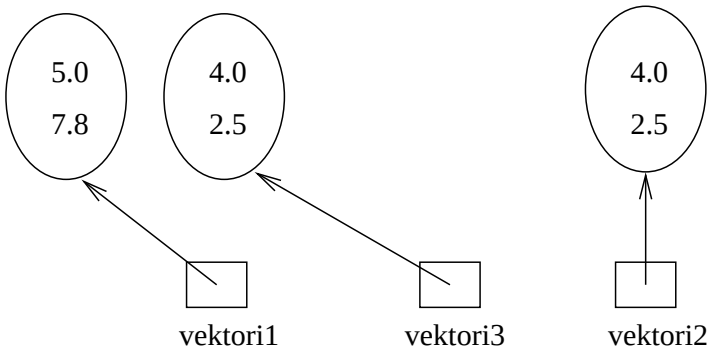
```
import tasovektori

def main():
    vektori1 = tasovektori.Tasovektori(4.0, 2.5)
    vektori2 = tasovektori.Tasovektori(4.0, 2.5)
    vektori3 = vektori1
    print vektori1
    print vektori2
    print vektori3
    vektori1 = tasovektori.Tasovektori(5.0, 7.8)
    print vektori1
    print vektori2
    print vektori3

main()
```

Tulostuksen selitys

- Jälleen muuttujat `vektori1` ja `vektori3` viittaavat aluksi samaan olioon:



- Muuttujaan `vektori1` tehtävä sijoitus ei kuitenkaan muuta itse oliota, vaan panee muuttujan viittaamaan uuteen olioon. Muuttuja `vektori3` jää viittaamaan samaan olioon kuin aikaisemminkin:



Oliomuuttuja luokan kenttänä

- Luokan kenttänä voi olla viite saman tai toisen luokan olioon.
- Halutaan kirjoittaa luokka `Kellonaytto`, jonka avulla voidaan esittää kellonaikoja muodossa `hh:mm`.
- Luokassa on metodi ajan asettamista varten sekä metodi, joka kasvattaa aikaa minuutilla.
- Luokka pitää huolen siitä, että tunnit ovat aina välillä 0–24 ja minuutit välillä 0–60.
- Määritellään luokka `Numeronaytto`, jonka avulla voidaan esittää lukuja kahdella numerolla. Luokassa on metodi luvun kasvatukseen. Numeronäytön arvo voi olla välillä 0 – (raja - 1), missä raja on määritelty `Numeronaytto`-oliota luodessa.
- `Kellonaytto`-olion kenttinä on kaksi `Numeronaytto`-oliota.
- Esimerkin idea on kirjasta Barnes and Kölling: Objects first with Java.

Numeronaytto, koodi

```
class Numeronaytto:

    def __init__(self, nollausraja):
        self.__arvo = 0
        if 1 <= nollausraja <= 100:
            self.__raja = nollausraja
        else:
            self.__raja = 1

    def kerro_arvo(self):
        return self.__arvo

    def kerro_raja(self):
        return self.__raja
```

Numeronaytto, koodi jatkuu

```
def aseta_arvo(self, uusi_arvo):
    if 0 <= uusi_arvo < self.__raja:
        self.__arvo = uusi_arvo


def kasvata_arvoa(self):
    self.__arvo = (self.__arvo + 1) % self.__raja


def __str__(self):
    if self.__arvo < 10:
        return "0" + str(self.__arvo)
    else:
        return str(self.__arvo)
```

Kellonaytto, koodi

```
import numeronaytto

class Kellonaytto:

    def __init__(self):
        self.__tunnit = numeronaytto.Numeronaytto(24)
        self.__minuutit = numeronaytto.Numeronaytto(60)
        self.aseta_aika(0, 0)


    def aseta_aika(self, uudet_tunnit, uudet_min):
        self.__tunnit.aseta_arvo(uudet_tunnit)
        self.__minuutit.aseta_arvo(uudet_min)
```

Kellonaytto, koodi jatkuu

```
def lisaa_minuutilla(self):
    self.__minuutit.kasvata_arvoa()
    if self.__minuutit.kerro_arvo() == 0:
        self.__tunnit.kasvata_arvoa()


def __str__(self):
    return str(self.__tunnit) + ":" + \
           str(self.__minuutit)
```

Pääohjelma, koodi

```
import kellonaytto

def main():
    kello1 = kellonaytto.Kellonaytto()
    print "Kello aluksi:", kello1
    kello1.aseta_aika(12, 45)
    print "Muutoksen jalkeen:", kello1
    for i in range(128):
        kello1.lisaa_minuutilla()
    print "Lisattiin 128 min:", kello1
```

Pääohjelma, koodi jatkuu

```
    kello2 = kellonaytto.Kellonaytto()
    kello2.aseta_aika(23, 55)
    print "Toinen kello aluksi:", kello2
    for i in range(10):
        kello2.lisaa_minuutilla()
    print "keskiyon jalkeen:", kello2


main()
```