

Ohjelmoinnin perusteet Y Python

T-106.1208

23.3.2009

Olio metodin parametrina: luokka Tasovektori

- ▶ Halutaan kirjoittaa luokka kaksiulotteisen vektorin $a\vec{i} + b\vec{j}$ kuvaamiseen.
- ▶ Kuvataan kertoimia a ja b kentillä `x_kerroin` ja `y_kerroin`.
- ▶ Määritellään metodit mm. vektorin pituuden ja kahden vektorin pistetulon laskemiseen.
- ▶ Pistetulon laskeva metodi tarvitsee tiedon kahdesta eri `Tasovektori`-oliosta.
- ▶ Toinen olio yksilöidään metodin kutsussa ennen pistettä ja metodin nimeä, toinen annetaan metodille parametrina.

Parametrioion kenttien arvon selvittäminen

- ▶ Parametrina saadun olion kentät voidaan selvittää joko pistenotaation avulla

```
def pistetulo(self, toinen_vektori):  
    tulo = self.__x_kerroin * toinen_vektori.__x_kerroin + \  
           self.__y_kerroin * toinen_vektori.__y_kerroin  
    return tulo
```

- ▶ Toinen vaihtoehto on käyttää luokan metodeita:

```
def pistetulo2(self, toinen_vektori):  
    tulo = self.__x_kerroin * \  
           toinen_vektori.kerro_x_kerroin() + \  
           self.__y_kerroin * \  
           toinen_vektori.kerro_y_kerroin()  
    return tulo
```

Luokka Tasovektori, koodi

```
import math
```

```
class Tasovektori:
```

```
    def __init__(self, eka_kerroin, toka_kerroin):  
        self.__x_kerroin = eka_kerroin  
        self.__y_kerroin = toka_kerroin
```

```
    def kerro_x_kerroin(self):  
        return self.__x_kerroin
```

```
    def kerro_y_kerroin(self):  
        return self.__y_kerroin
```

Luokka Tasovektori, koodi jatkuu

```
def laske_pituus(self):  
    return math.sqrt(self.__x_kerroin ** 2 + \  
                      self.__y_kerroin ** 2)  
  
def kerro_luvulla(self, kertoja):  
    self.__x_kerroin *= kertoja  
    self.__y_kerroin *= kertoja  
  
def pistetulo(self, toinen_vektori):  
    tulo = self.__x_kerroin * \  
           toinen_vektori.__x_kerroin + \  
           self.__y_kerroin * \  
           toinen_vektori.__y_kerroin  
    return tulo
```

Luokka Tasovektori, koodi jatkuu

```
def __str__(self):
    if self.__y_kerroin >= 0:
        miono = "%.3fi + %.3fj" % \
                (self.__x_kerroin, self.__y_kerroin)
    else:
        miono = "%.3fi - %.3fj" % \
                (self.__x_kerroin, - self.__y_kerroin)
    return miono
```

Esimerkki Tasovektori-olioiden käytöstä

- ▶ Seuraavien kalvojen pääohjelma luo kaksi Tasovektori-oliota ja kutsuu niille eri metodeita.
- ▶ Käsiteltävien vektoreiden tiedot pyydetään käyttäjältä.
- ▶ Desimaalilukujen lukemista varten on jälleen määritelty oma apufunktio.

Esimerkki vektoreiden käsittelystä, koodi

```
import tasovektori

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

Esimerkki vektoreiden käsittelystä, koodi jatkuu

```
def main():  
    print "Anna ensimmäisen vektorin i-kerroin."  
    i_kerroin = lue_desimaaliluku()  
    print "Anna ensimmäisen vektorin j-kerroin."  
    j_kerroin = lue_desimaaliluku()  
    a_vektori = tasovektori.Tasovektori(i_kerroin, j_kerroin)  
    print "Antamasi vektori on", a_vektori  
  
    print "Anna toisen vektorin i-kerroin."  
    i_kerroin = lue_desimaaliluku()  
    print "Anna toisen vektorin j-kerroin."  
    j_kerroin = lue_desimaaliluku()  
    b_vektori = tasovektori.Tasovektori(i_kerroin, j_kerroin)  
    print "Antamasi vektori on", b_vektori
```

Esimerkki vektoreiden käsittelystä, koodi jatkuu

```
pituus1 = a_vektori.laske_pituus()
pituus2 = b_vektori.laske_pituus()
print "Vektorin %s pituus on %.3f" % (a_vektori, pituus1)
print "Vektorin %s pituus on %.3f" % (b_vektori, pituus2)

laskettu_tulo = a_vektori.pistetulo(b_vektori)
print "Vektoreiden pistetulo on %.3f." % (laskettu_tulo)

print "Milla luvulla ensimmäinen vektori kerrotaan?"
kerroin = lue_desimaaliluku()
a_vektori.kerro_luvulla(kerroin)
print "Ensimmäinen vektori kertomisen jälkeen", a_vektori

main()
```

Luokkien kenttien näkyvyys

- ▶ Esimerkeissä luokkien kenttien nimet ovat alkaneet kahdella alaviivalla.
- ▶ Tämä saa aikaan sen, että kenttiin ei pääse suoraan käsiksi luokan ulkopuolelta, vaan olion kenttiä on aina käsiteltävä luokan metodien avulla.
- ▶ Seuraavan esimerkin avulla esitetään, miksi tämä on toivottava tapa käsitellä kenttiä.

Esimerkki kenttien käsittelystä suoraan

- ▶ Olkoon määritelty luokka henkilötietojen käsittelyyn.

```
class Henkilo1:  
  
    def __init__(self, nimi1):  
        self.nimi = nimi1  
        self.ika = 0
```

Luokkaa käytetään apuna monessa eri ohjelmassa, esimerkiksi:

```
oppilas = Henkilo1("Matti")  
oppilas.ika = 15  
if oppilas.ika < 18:  
    print "Oppilas on alaikainen"  
else:  
    print "Oppilas on taysi-ikainen"  
oppilas.ika = 16  
print "Oppilaan ika on", oppilas.ika
```

Esimerkki kenttien käsittelystä jatkuu

- ▶ Jossain vaiheessa kenttä `ika` päätetään korvata kentällä `syntymavuosi`. Luokan määrittely alkaa nyt

```
class Henkilo1:
```

```
    def __init__(self, nimi1):  
        self.nimi = nimi1  
        self.syntymavuosi = 0
```

- ▶ Millaisia muutostarpeita tämä aiheuttaa niissä ohjelmissa, jotka käyttävät tätä luokkaa?

Esimerkki vaihtoehtoisesta tavasta

- Tarkastellaan vaihtoehtoista tapaa: luokan kenttiä voi käsitellä vain luokan metodien avulla:

```
class Henkilo2:
```

```
    def __init__(self, nimi1):  
        self.__nimi = nimi1  
        self.__ika = 0
```

```
    def kerro_ika(self):  
        return self.__ika
```

```
    def muuta_ika(self, uusi_ika):  
        if 0 <= uusi_ika <= 150:  
            self.__ika = uusi_ika
```

Esimerkki vaihtoehtoisesta tavasta jatkuu

- Tätäkin luokkaa käytettäisiin apuna monessa ohjelmassa henkilötietojen käsittelyyn:

```
oppilas = Henkilo2("Matti")
oppilas.muuta_ika(15)
if oppilas.kerro_ika() < 18:
    print "Oppilas on alaikainen"
else:
    print "Oppilas on taysi-ikainen"
print "Oppilaan ika on", oppilas.kerro_ika()
```

Esimerkki vaihtoehtoisesta tavasta jatkuu

- ▶ Luokan kenttä ika korvataan kentällä syntymavuosi:
NYKYINEN_VUOSI = 2009

```
class Henkilo2:
```

```
    def __init__(self, nimi1):  
        self.__nimi = nimi1  
        self.__syntymavuosi = 0
```

```
    def kerro_ika(self):  
        return NYKYINEN_VUOSI - self.__syntymavuosi
```

```
    def muuta_ika(self, uusi):  
        if 0 <= uusi <= 150:  
            self.__syntymavuosi = NYKYINEN_VUOSI - uusi
```

- ▶ Mitä pitää muuttaa tätä luokkaa käyttävissä ohjelmissa?

Lista luokan kenttänä

- ▶ Luokan kenttä voi olla myös esimerkiksi lista, sanakirja tai oliomuuttuja.
- ▶ Seuraavassa esimerkissä on määritelty luokka Bonusasiakas jonkin kaupan kanta-asiakkaan kuvaamiseen.
- ▶ Luokalla on kentät nimi ja ostokset. Jälkimmäinen on lista, joka sisältää kanta-asiakkaan eri kertaostosten arvot.
- ▶ Uutta asiakasta luodessa lista ostokset alustetaan tyhjäksi listaksi. Listaan voi lisätä ostoksia luokan metodin avulla.

Luokka Bonusasiakas, koodi

```
class Bonusasiakas:

    def __init__(self, asiakkaan_nimi):
        self.__nimi = asiakkaan_nimi
        self.__ostokset = []

    def kerro_nimi(self):
        return self.__nimi

    def lisaa_ostos(self, arvo):
        if arvo > 0.0:
            self.__ostokset.append(arvo)
            return True
        else:
            return False
```

Luokka Bonusasiakas, koodi jatkuu

```
def laske_keskiarvo(self):  
    lkm = 0  
    summa = 0.0  
    for ostoksen_arvo in self.__ostokset:  
        summa += ostoksen_arvo  
        lkm += 1  
    if lkm == 0:  
        return 0.0  
    else:  
        return summa / lkm  
  
def laske_rajan_ylittaneet(self, alaraja):  
    ylittaneiden_lkm = 0  
    for arvo in self.__ostokset:  
        if arvo > alaraja:  
            ylittaneiden_lkm += 1  
    return ylittaneiden_lkm
```

Luokka Bonusasiakas, koodi jatkuu

```
def laske_bonus(self):
    PIENIBONUS = 1.0
    SUURIBONUS = 2.5
    BONUSRAJA = 400.0
    summa = 0.0
    for ostos in self.__ostokset:
        summa += ostos
    if summa >= BONUSRAJA:
        bonus = SUURIBONUS * summa / 100.0
    else:
        bonus = PIENIBONUS * summa / 100.0
    return bonus

def nollaa_ostokset(self):
    self.__ostokset = []
```

Luokka Bonusasiakas, koodi jatkuu

```
def __str__(self):  
    mjono = "Asiakas: " + self.__nimi + "\nOstot:\n"  
    for arvo in self.__ostokset:  
        mjono += "%.2f eur\n" % (arvo)  
    return mjono
```

Bonusasiakas-olioita käsittelevä pääohjelma

```
import bonusasiakas

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

Bonusasiakas-olioita käsittelevä pääohjelma jatkuu

```
def main():
    OSTOSKERRAT = 4
    nimi1 = raw_input("Anna 1. asiakkaan nimi: ")
    asiakas1 = bonusasiakas.Bonusasiakas(nimi1)
    nimi2 = raw_input("Anna 2. asiakkaan nimi: ")
    asiakas2 = bonusasiakas.Bonusasiakas(nimi2)
    for i in range(OSTOSKERRAT):
        print "Anna asiakkaan %s yhden kertaoston arvo." % \
            asiakas1.kerro_nimi()
        luettu_arvo = lue_desimaaliluku()
        if asiakas1.lisaa_ostos(luettu_arvo):
            print "Ostoksen lisays onnistui."
        else:
            print "Ostoksen lisays ei onnistunut."
```

Bonusasiakas-olioita käsittelevä pääohjelma

```
for i in range(OSTOSKERRAT):
    print "Anna asiakkaan %s yhden kertaoston arvo." % \
          asiakas2.kerro_nimi()
    luettu_arvo = lue_desimaaliluku()
    if asiakas2.lisaa_ostos(luettu_arvo):
        print "Ostoksen lisays onnistui."
    else:
        print "Ostoksen lisays ei onnistunut."

print "1. asiakkaan ostosten keskiarvo: %.2f eur" % \
      (asiakas1.laske_keskiarvo())
print "2. asiakkaan ostosten keskiarvo: %.2f eur" % \
      (asiakas2.laske_keskiarvo())
```

Bonusasiakas-olioita käsittelevä pääohjelma jatkuu

```
print "1. asiakkaan bonus: %.2f eur" % \
      (asiakas1.laske_bonus())
print "2. asiakkaan bonus: %.2f eur" % \
      (asiakas2.laske_bonus())

print "Anna raja, jonka ylittävät ostokset haetaan."
raja = lue_desimaaliluku()
ylitykset1 = asiakas1.laske_rajan_ylittaneet(raja)
ylitykset2 = asiakas2.laske_rajan_ylittaneet(raja)
print "1. asiakkaalla oli %d suurempaa ostosta" % \
      (ylitykset1)
print "2. asiakkaalla oli %d suurempaa ostosta" % \
      (ylitykset2)
```

Bonusasiakas-olioita käsittelevä pääohjelma jatkuu

```
print "Asiakkaiden tiedot:"  
print asiakas1  
print asiakas2  
  
asiakas1.nollaa_ostokset()  
print "1. asiakas nollauksen jälkeen:"  
print asiakas1
```

```
main()
```