Aliohjelmat

Ohjelmoinnin perusteet

6

Turun yliopisto
University of Turku

Modulaarisuus

- Tähän mennessä kaikki esitetyt ja kirjoitetut ohjelmat ovat koostuneet yhdestä "pääohjelmasta"

- Usein monimutkaisemmat ohjelmat on kuitenkin hyvä jakaa erillisiin aliohjelmiin. Tätä kutsutaan modulaarisuudeksi.

Modulaarisuuden etuja

- Itsenäisen toiminnallisuuden siirtäminen omiksi aliohjelmikseen
- Ohjelman rakenne ja luettavuus paranee
- Usein toistettavia toimintoja ei tarvitse kirjoittaa kuin kerran

Aliohjelma

- Aliohjelma on itsenäinen osa ohjelman sisällä.
- Aliohjelmaa voidaan **kutsua** pääohjelmasta tai muista aliohjelmista. Kutsuminen suorittaa aliohjelman sisälle kirjoitetut lauseet.
- Aliohjelmalla on omat **paikalliset muuttujat**. Lisäksi aliohjelmalle voidaan antaa **syötteitä**, ja se voi **palauttaa arvon**

Aliohjelma Pythonissa

- Aliohjelma voidaan määritellä `def`-avainsanalla.

Syntaksi:

```
def <aliohjelma> ( <parametrit> ):
    <suoritettava lohko>
```

Aliohjelmien kirjoittamisesta

- Aliohjelmat kannattaa kirjoittaa selkeyden (ja kutsuttavuuden) takia ohjelman alkuun.
- Aliohjelmalla ei ole pakko olla parametreja. Nimen jälkeen tulevat sulut ovat kuitenkin pakolliset.
- Aliohjelmien nimissä noudatetaan samaa käytäntöä kuin muuttujien nimissä.

Esimerkki

- Aliohjelma, joka tulostaa ruudulle kolmen kertotaulun:

```
def kolmenKertotaulu():
    kerto = 1
    while kerto <= 10:
        print kerto,"* 3 =",kerto * 3
```

Aliohjelman kutsuminen

- Aliohjelmaa kutsutaan (eli suoritetaan se) käyttämällä sen nimeä:

```
def kolmenKertotaulu():
    kerto = 1
    while kerto <= 10:
        print kerto,"* 3 =",kerto * 3

# Pääohjelma: kutsu aliohjelmaa
kolmenKertotaulu()
```

Aliohjelman kutsuminen (2)

- Samaa aliohjelmaa voidaan kutsua useita kertoja pääohjelmasta tai muista aliohjelmista.

- Näin ollen usein tarvittava koodi kannattaa kirjoittaa omaksi aliohjelmakseen. Tällä tavalla ohjelmakoodista tulee siistimpää, ja ohjelmien ylläpito on helpompaa.

Esimerkki

```
def tulostaVaaka():
    print "+-----+"

def tulostaPysty():
    print "|     |"

# Pääohjelma: Tulosta "laatikko" ruudulle
tulostaVaaka()
tulostaPysty()
tulostaPysty()
tulostaVaaka()
```

Aliohjelmien parametrisointi

- Aliohjelmilla voi olla parametreja, joiden avulla niiden suoritusta voidaan ohjata.

- Parametrit ovat aliohjelman paikallisia muuttujia, ja niille annetaan arvot aliohjelmaa kutsuttaessa.

Parametreista

- Parametrimuuttujien nimet (eli **muodolliset parametrit)** kirjoitetaan aliohjelman nimen perään sulkujen sisään.

- Jos parametreja on enemmän kuin yksi, ne erotetaan toisistaan pilkulla.

Esimerkkejä

```
def tulostaSumma(luku1, luku2):
    # aliohjelman runko...

def poimiVokaalit(mjono):
    # aliohjelman runko...

def ympyranAla(x, y, sade):
    # aliohjelman runko...
```

Todelliset parametrit

- Kun aliohjelmaa kutsutaan, parametreille annetaan arvot.
- Nämä arvot (eli **todelliset parametrit**) asetetaan parametrimuuttujien (eli **muodollisten parametrien**) arvoiksi.
- Tämän jälkeen parametrimuuttujia voi aliohjelman sisällä käyttää kuten mitä tahansa muuttujia.

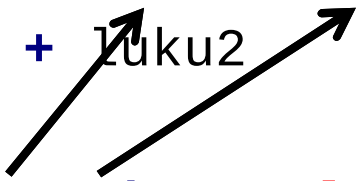
Esimerkki

```
def tulostaSumma(luku1, luku2):
    print luku1 + luku2

# Pääohjelma
tulostaSumma(10,20) # Tulostaa 30
tulostaSumma(2, 3 + 3) # Tulostaa 8

n = input("Anna luku:")
tulostaSumma(n,n)
```

Parametrien arvot

- Metodia kutsuttaessa parametrien todelliset arvot siis asetetaan parametrimuuttujien arvoiksi:

```
def tulostaSumma(luku1, luku2):
    print luku1 + luku2

tulostaSumma(10,20) # Tulostaa 30
```



Parametrit ovat muuttujia

- Parametreja voidaan käyttää aliohjelmassa kuten mitä tahansa muuttujia:

```
def tulostaKertoma(n):
    k = n - 1
    while k > 1:
        n = n * k
        k = k - 1
    print n

tulostaKertoma(5)
```

Paikalliset muuttujat

- Aliohjelmassa määritellyt muuttujat (mukaan lukien parametrit) ovat näkyvissä ainoastaan aliohjelman sisällä.
- Tällaisia muuttujia nimitetään aliohjelman **paikallisiksi muuttujiksi**.
- Paikallista muuttujaa ei siis voi käyttää aliohjelman ulkopuolella.

Paikalliset muuttujat (2)

- Aliohjelmissa ja pääohjelmassa voi kuitenkin olla samannimisiä muuttujia.
- Näillä muuttujilla ei ole nimen lisäksi muuta yhteistä.
- Sekaantumisen välttämiseksi on hyvä pyrkiä nimeämään pääohjelman ja aliohjelmien muuttujat eri nimisiksi.

Esimerkki

- Pää- ja aliohjelmassa on samanniminen muuttuja a. Nämä ovat kuitenkin eri muuttujia.

```
def kasvata(a):
    a = a + 1 # Tämä on aliohjelman muuttuja
    print a

a = 3 # Tämä on pääohjelman muuttuja
kasvata(a) # Tulostaa 4
print a # Tulostaa 3
```

Esimerkki 2

- Mitä seuraava ohjelma tulostaa?

```
def kasvataTulosta(luku):
    luku = luku + 10
    print luku

def vahennaTulosta(luku):
    luku = luku - 10
    print luku

luku = 20
print luku
kasvataTulosta(luku)
vahennaTulosta(luku)
print luku
```

Funktiot ja proseduurit

- Aliohjelmat voivat myös palauttaa arvon
- Tällaisia aliohjelmia kutsutaan **funktioiksi**
- Aliohjelmia, jotka eivät palauta mitään (kuten tähänastiset esimerkit) kutsutaan **proseduureiksi.**

Funktiot

- Funktio siis **palauttaa arvon kutsujalle**

- Kutsuja voi käyttää tätä arvoa kuten mitä tahansa muuttujan arvoa tai vakioarvoa.

Arvon palautus pythonissa

- Pythonissa arvon palautus aliohjelmasta tapahtuu `return`-lauseella

- Syntaksi:

```
return <lauseke>
```

Esimerkki

- Funktio, joka laskee parametrien keskiarvon ja palauttaa sen kutsujalle:

```
def keskiarvo(n1, n2, n3):
    summa = n1 + n2 + n3
    return summa / 3.0
```

Funktion kutsuminen

- Funktiota kutsutaan kuten mitä tahansa aliohjelmaa, käyttäen sen nimeä ja antaen oikea määrä parametreja:

```
def keskiarvo(n1, n2, n3):
    summa = n1 + n2 + n3
    return summa / 3.0

print keskiarvo(7,5,6)
```

Funktion paluuarvo

- Paluuarvo ikään kuin "sijoitetaan" kutsukohtaan:

```
def keskiarvo(n1, n2, n3):
    summa = n1 + n2 + n3
    return summa / 3.0

print keskiarvo(7,5,6)
```

Funktion paluuarvo (2)

- Paluuarvoa voi käyttää normaalisti osana lausekkeita:

```
def poimiEkaSana(lause):
    valiIndeksi = lause.find(" ")
    ekaSana = lause[0:valiIndeksi]
    return ekaSana

a = "Pekka Python"
b = "Olli Ohjelmointi"
c = poimiEkaSana(a) + " " + poimiEkaSana(b)
print c # Tulostaa Pekka Olli
```

Funktion paluuarvo (3)

- Paluuarvoa voi tietysti käyttää myös parametrina:

```
def nelio(luku):
    return luku * luku

n = 10
n2 = nelio(nelio(n))
print n2 # Tulostaa 10000
```

Funktiokutsut aliohjelmista

- Funktiot ja proseduurit voivat myös kutsua toisiaan:

```
def summa(n1,n2,n3):
    return n1 + n2 + n3

def keskiarvo(n1,n2,n3):
    s = summa(n1,n2,n3)
    return s / 3.0

print keskiarvo(1,3,4)
```