ALIOHJELMAT

Ohjelmoinnin peruskurssi yläkouluun

TURUN YLIOPISTO
Oppimisanalytiikan keskus

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

- Modulaarisuudella tarkoitetaan itsenäisten toiminnallisuuksien siirtämistä omiksi aliohjelmikseen eli **ohjelman pilkkomista pienemmiksi osiksi**.
- Modulaarisuuden myötä ohjelman rakenne ja luettavuus paranee.
- Tällöin usein toistettavia toimintoja ei tarvitse kirjoittaa kuin kerran.

ALIOHJELMA

- Aliohjelma on **itsenäinen osa** ohjelman sisällä.
- Aliohjelmat **määritellään** def-lauseella, mutta niitä **ei suoriteta** määrittelyn yhteydessä. Aliohjelmia suoritetaan vain kutsuttaessa.
- Aliohjelmat kannattaakin kirjoittaa **ohjelman alkuun**, jotta ne on varmasti määritelty ennen kutsumista (ja suorittamista).

ALIOHJELMA

- Aliohjelmien **nimissä noudatetaan samaa käytäntöä kuin muuttujien nimissä** eli nimet alkavat kirjaimella ja koostuvat kirjaimista, numeroista ja alaviivoista.
- Aliohjelman **määrittelyn syntaksi**:

```
def <aliohjelman nimi>():
    <aliohjelman lohko>
```

ESIMERKKI

```
# Määritellään aliohjelma, joka tulostaa ruudulle
# kolmen kertotaulun.
def kolmenKertotaulu():
    kerto = 1
    while kerto <= 10:
        print(kerto, "* 3 =", kerto * 3)
```

ALIOHJELMAN KUTSUMINEN

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

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

# Pääohjelma: kutsutaan aliohjelmaa.
kolmenKertotaulu()
```

ALIOHJELMA

- 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.

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

# Pääohjelma: tulostetaan laatikko.
tulostaVaaka()
tulostaPysty()
tulostaPysty()
tulostaVaaka()
```

PARAMETRIT

- Aliohjelmilla voi olla **parametreja**, joiden avulla niiden suoritusta voidaan ohjata.
- Parametreilla tarkoitetaan aliohjelmalle sulkujen sisällä **syötteenä annettuja arvoja/muuttujia**.

PARAMETRIT

- Parametreja voi olla **nolla, yksi tai useampi**. Jos parametreja on enemmän kuin yksi, ne erotetaan toisistaan **pilkulla**.
- Vaikka parametreja ei ole, aliohjelman nimen jälkeen tulevat **sulut ovat kuitenkin pakolliset sekä määriteltäessä että kutsuttaessa**.

ALIOHJELMAN SYNTAKSI

```
def <aliohjelman nimi>(<parametrit>):
      <aliohjelman lohko>
```

ESIMERKKI

```
def tulostaSumma(lista):
    # aliohjelman lohko

def pidempi(mjono1, mjono2):
    # aliohjelman lohko

def ympyra(x, y, sade):
    # aliohjelman lohko
```

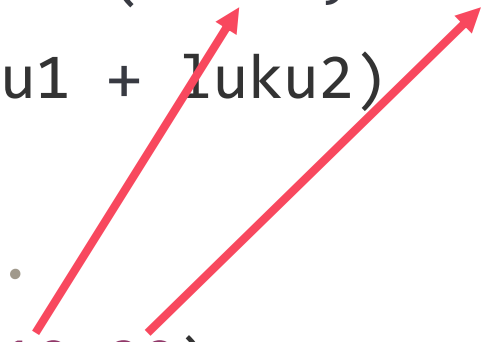
PARAMETRIT

- Parametrimuuttujien nimet (eli **muodolliset parametrit**) kirjoitetaan aliohjelman nimen perään sulkujen sisään.
- Kun aliohjelmaa kutsutaan, parametreille annetaan arvot. Nämä arvot (eli **todelliset parametrit**) asetetaan muodollisten parametrien arvoiksi.
- Tämän jälkeen parametrimuuttujia voi aliohjelman sisällä käyttää kuten mitä tahansa muuttujia.

ESIMERKKI

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

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



ESIMERKKI

- Parametreja voi käyttää kuten mitä tahansa muuttujia.

```
def tulostaLause(lause):
    lause = lause.replace("ä", "a")
    lause = lause.replace("ö", "o")
    print(lause)
    print("Eka merkki:", lause[0])
    print("Vika merkki:", lause[len(lause)-1])
```

PAIKALLISET MUUTTUJAT

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

PAIKALLISET MUUTTUJAT

- Aliohjelmissa ja pääohjelmissa voi kuitenkin olla samannimisiä muuttujia, mutta näillä muuttujilla ei ole nimen lisäksi muuta yhteistä.
- Sekaannuksen välttämiseksi on hyvä pyrkiä nimeämään pääohjelman ja aliohjelmien muuttujat eri nimisiksi.

ESIMERKKI

```
# Aliohjelma, joka tulostaa yhtä suuremman arvon.
def kasvata(a):
    a = a + 1 # Tämä a on aliohjelman oma muuttuja.
    print(a)

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

```
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)
```

Mitä ohjelma tulostaa?

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 **palauttavat arvon kutsukohtaan**.
- Kutsukohdassa tätä palautettua arvoa voi käyttää kuten mitä tahansa muuttujan arvoa tai vakioarvoa.

FUNKTIOT

- Pythonissa **arvon palautus aliohjelmasta** tapahtuu `return`-lauseella.
- Lauseella voi palauttaa **minkä tahansa lausekkeen**, joka kuitenkin **lasketaan ennen palauttamista**.

```
return <lauseke>
```

ESIMERKKI

```
# Lasketaan ja palautetaan parametrien keskiarvo.
def keskiarvo(luku1, luku2, luku3):
    summa = luku1 + luku2 + luku3
    keskiarvo = summa / 3
    # Palautetaan keskiarvo kutsukohtaan.
    return keskiarvo
```

FUNKTION KUTSUMINEN

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

```
def keskiarvo(luku1, luku2, luku3):
    summa = luku1 + luku2 + luku3
    keskiarvo = summa / 3
    return keskiarvo

print(keskiarvo(8,4,6))
```

FUNKTION PALUUARVO

- Paluuarvo sijoitetaan **kutsukohtaan**:

```
def keskiarvo(luku1, luku2, luku3):
    summa = luku1 + luku2 + luku3
    keskiarvo = summa / 3
    return keskiarvo


print(keskiarvo(8,4,6))
```

PALUUARVO LAUSEKKEEN OSANA

```
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
```

PALUUARVO PARAMETRINA

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

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

KUTSUMINEN ALIOHJELMISTA

```
def summa(luku1, luku2, luku3):
    return luku1 + luku2 + luku3
def keskiarvo(luku1, luku2, luku3):
    s = summa(luku1, luku2, luku3) # Kutsutaan summa-funktiota
    return s / 3
print(keskiarvo(1, 3, 4))
```

TURUN
YLIOPISTO
Oppimisanalytiikan keskus