hyväksymispäivä arvosana

arvostelija

Otsikko

Erkki Heino
Tero Huomo
Eeva Terkki

Helsinki 5.2.2013

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Näkyvyysalueet **1**

1.1 COBOL 1

1.2 Python 1

2 Sidonta **3**

2.1 Cobol 3

2.2 Python 3

3 Kontrollin ohjaus **3**

3.1 Python 3

3.1.1 Valinta 3

3.1.2 Toisto 3

3.1.3 Rekursio 3

3.2 COBOL 3

3.2.1 Valinta 3

3.2.2 Toisto 5

3.2.3 Rekursio 5

Lähteet **6**

1 Näkyvyysalueet

1.1 COBOL

1.2 Python

Pythonin lohkorakenne on syvä, ja ohjelman suoritusaikana käytössä on ainakin kolme sisäkkäistä näkyvyysaluetta [Pyt13b]. Näkyvyysalueita käytetään dynaamisesti. Sisimmällä näkyvyysalueella ovat paikalliset nimet. Mahdollisilla funktioita ympäröivillä funktioilla on omat näkyvyysalueensa, joiden sisältämät nimet eivät ole paikallisia eivätkä globaaleja. Toisiksi uloimmalla näkyvyysalueella ovat moduulin globaalit nimet ja kaikkein uloimmalla kieleen rakennetut nimet.

Pythonissa luokan näkyvyysalueella määritellyt nimet eivät näy luokan metodeille [Pyt13a]. Metodin ensimmäinen argumentti, jolle on tapana antaa nimi `self`, edustaa luokan ilmentymää. Sen kautta metodi voi käyttää luokan ilmentymän muita metodeja ja attribuutteja.

Seuraavassa esimerkissä on kaksi sisäkkäistä funktiota:

```
def f1():
    a =  1
    def f2():
        b = 2
        print a + b
    print a
    f2()

f1()
```

Funktio `f1` määrittelee muuttujan `a` ja funktion `f2`, tulostaa `a`:n arvon ja kutsuu määrittelemäänsä funktiota. Funktio `f2` määrittelee muuttujan `b` ja tulostaa muuttujien `a` ja `b` arvojen summan. Ohjelma tulostaa luvut 1 ja 3. Muuttuja `a` on näkyvissä funktion `f1` ja sen sisäisten funktioiden sisällä. Muuttuja `b` puolestaan on paikallinen muuttuja, joka on näkyvissä vain `f2`-funktion sisällä. Siihen viittaaminen `f2`-funktion ulkopuolella johtaisi virhetilanteeseen.

Seuraavassa esimerkissä käytetään globaalia muuttujaa:

```
g = 1

def f3():
    global g
    g = 2
    print g

f3()
print g
```

Esimerkissä globaalin muuttujan `g` arvoksi alustetaan ensin luku 1. Funktio `f3` asettaa `g`:n arvoksi luvun 2 ja myös tulostaa muuttujan arvon. Kun funktiota `f3` kutsutaan ja sen jälkeen vielä tulostetaan `g`:n arvo, ohjelma tulostaa kaksi kertaa luvun 2. Funktio `f3` siis käsittelee globaalia muuttujaa. Avainsana `global` on tärkeä, sillä se ilmaisee, että kyseinen tunnus tulkitaan globaalin muuttujan tunnukseksi. Ilman koodiriviä `global g` funktion määritelmän sisällä oleva muuttuja olisi paikallinen muuttuja, ja esimerkkiohjelma tulostaisi luvut 2 ja 1.

Pythonissa kaikki asiat, jotka voidaan nimetä, ovat ensimmäisen luokan arvoja – myös funktiot, metodit ja moduulit [Gui09].

2 Sidonta

2.1 Cobol

2.2 Python

3 Kontrollin ohjaus

3.1 Python

3.1.1 Valinta

3.1.2 Toisto

3.1.3 Rekursio

3.2 COBOL

3.2.1 Valinta

Valinta suoritetaan rakenteella `IF - THEN - ELSE - ENDIF` []. COBOL ei tue esimerkiksi Javassa tunnettua `else if` -valintaa. Usean vaihtoehdon välinen valinta

saadaan aikaan sisäkkäisillä if-else lauseilla.

```
* Yksinkertainen valinta
    IF NUMERO = 2
      DISPLAY 'KAKSI'
    END-IF
* IF ELSE lause
    IF NUMERO IS EQUAL TO 2 THEN
        DISPLAY 'KAKSI'
    ELSE
        DISPLAY 'EI KAKSI'
    END-IF
* sisäkkäinen IF ELSE lause
    IF NUMERO = 1
        DISPLAY 'NUMERO'
    ELSE
        IF NUMERO = 2
            DISPLAY 'KAKSI'
        ELSE
            DISPLAY 'EI YKSI EIKÄ KAKSI'
        END-IF
    END-IF
```

COBOLin vastaavuus Javan tai C:n Switch-case -rakenteelle on `EVALUATE` -verbi [].

```
EVALUATE  VALIKKO-SYOTE
    WHEN "0"
        PERFORM INIT-PROC
    WHEN "1" THRU "9"
        PERFORM PROCESS-PROC
    WHEN "R"
        PERFORM READ-PARMS
    WHEN "X"
        PERFORM CLEANUP-PROC
    WHEN OTHER
        PERFORM ERROR-PROC
END-EVALUATE.
```

Esimerkissä EVALUATE -verbiä seuraa evaluaation subjekti eli kohde, esimerkissä `VALIKKO-SYOTE`. Avainsana `WHEN` vastaa . Evaluaation lopettaa avainsana `END-EVALUATE`.

3.2.2 Toisto

3.2.3 Rekursio

Alunperin COBOL ei mahdollistanut rekursiota lainkaan [Wik13]. Uudemmista COBOLin versioista ainakin IBM:n COBOL-toteutus tukee rekursiota. IBM:n toteutuksessa rekursiivisesti kutsuttavan ohjelman tai aliohjelman Program-ID:ssä on annettava ehto `RECURSIVE` [IBM04]. Mikäli ohjelmaa kutsuu rekursiivisesti ilman ehtoa, ohjelman suoritus päättyy.

Lähteet

IBM04 Making recursive calls.

Pyt13a The python language reference – execution model, http://docs.python.org/2/reference/executionmodel.html#naming-and-binding. [4.2.2013].

Pyt13b The python tutorial – classes, http://docs.python.org/2/tutorial/classes.html. [2.2.2013].

Gui09 van Rossum, G., First-class everything, http://python-history.blogspot.fi/2009/02/first-class-everything.html, 2009. [2.2.2013].

Wik13 Cobol, http://en.wikipedia.org/wiki/COBOL. [5.2.2013].