hyväksymispäivä arvosana

arvostelija

Otsikko

Erkki Heino
Tero Huomo
Eeva Terkki

Helsinki 5.2.2013

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

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 Aritmeettisten operaatioiden laskentajärjestys

COBOLin aritmeettisten lausekkeiden evaluoinnin järjestys riippuu COBOLin versiosta ja kääntäjän toteuttajasta. Compaqin COBOLissa laskentajärjestyksessä on neljä tasoa [Com02]. Ensimmäisenä määritetään muuttujien etumerkit. Tämän jälkeen potenssit, jonka jälkeen kerto- ja jakolaskut. Viimeisenä suoritetaan yhteen- ja vähennyslaskut.

COBOLin viimeisimmissä versioissa kieleen tuli mukaan myös bittioperaatiot. Micro Focus'n Visual COBOL toteutuksessä tämä on nostanut laskentatasoja seitsemään [Mic13]. Bittioperaatioista `NOT` evaluoidaan yhdessä etumerkkien kanssa. Muuten bittioperaatiot evaluoidaan viimeisinä järjestyksessä `AND`, `XOR` ja `OR`.

3.2.2 Valinta

Valinta suoritetaan rakenteella `IF - THEN - ELSE - ENDIF` [].

```
* 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
```

Esimerkistä huomataan, että rakenne vastaa pitkälti Javan If-valintalausetta. Avainsana `THEN` on vapaaehtoinen [Kar13], mutta mahdollistaa luonnollisen englanninkielen kaltaisen lausemuodon. If-rakenne lopetetaan avainsanalla `END-IF`, joka on mahdollista korvata myös pisteellä. COBOL ei kuitenkaan tue monen modernin kielen käyttämää else-if -valintaa. Else-if on korvattava esimerkin kaltaisilla sisäkkäisillä if-else -valintalauseilla.

COBOLin vastaavuus Javan tai C:n Switch-case -rakenteelle on `EVALUATE` -verbi [Cou99a].

```
EVALUATE VALIKKO-SYOTE
    WHEN "0"
        DISPLAY 'VALITSIT 0'
    WHEN "1" THRU "9"
        DISPLAY 'VALITSIT 1-9'
END-EVALUATE.
```

Esimerkissä EVALUATE -verbiä seuraa evaluaation subjekti, `VALIKKO-SYOTE`. Avainsana `WHEN` vastaa pitkälti Switch-case -rakenteen `case`a. Evaluaation lopettaa avainsana `END-EVALUATE`. Esimerkissä jos `VALIKKO-SYOTE` on 0, näytetään käyttäjälle "VALITSIT 0". Jos se on yhden ja yhdeksän väliltä, näytetään "VALITSIT 1-9". Toisin kuin Javan Switch-Case, EVALUATE tukee useita samanaikaisia vertailuja -muuttujia ja ehtoja voi ketjuttaa avainsanan `ALSO` avulla.

3.2.3 Toisto

COBOL tukee pitkälti samoja toistorakenteita kuin Java. Tässä lueteltavat toistorakenteet esittelee Cobol Tutorial [Cou99b]. COBOLin vastaavuudet while-rakenteelle ja do-while -rakenteelle on `PERFORM UNTIL`, jossa toistoehdon voi sijoittaa alkuun tai loppuun.

Javan For-looppia vastaa `PERFORM - TIMES` ja `PERFORM VARYING`. Seuraava esimerkki tulostaa käyttäjälle viisi kertaa sanan "HEI":

```
PERFORM 5 TIMES
    DISPLAY "HEI"
END-PERFORM
```

`PERFORM VARYING` mahdollistaa Javan tapaisen for-loopin, jossa muuttujan arvoa korotetaan yhden kierroksen jälkeen muulla kuin yhdellä.

```
PERFORM VARYING NUMERO FROM 1 BY 2
    UNTIL NUMERO > 5
DISPLAY 'Numero on nyt: ' NUMERO
END-PERFORM
```

Esimerkissä muuttujaa NUMERO korotetaan jokaisen kierroksen jälkeen kahdella.

Esimerkin tulostus:

```
Numero on nyt: 1
Numero on nyt: 3
Numero on nyt: 5
```

3.2.4 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 rekursiosta eksplisiittisesti kertovaa `RECURSIVE`-ehtoa, ohjelman suoritus päättyy.

Lähteet

Cou99b Cobol tutorial: Iteration, http://www.csis.ul.ie/cobol/course/Iteration.htm. [5.2.2013].

Cou99a Cobol tutorial: Selection, http://www.csis.ul.ie/cobol/course/Selection.htm. [5.2.2013].

Com02 Compaq cobol reference manual: Arithmetic expressions, http://h30266.www3.hp.com/odl/vax/progtool/cobol57a/6296/6296_profile_019.html. [5.2.2013].

Kar13 Cobol - if else endif statement, http://code.xmlgadgets.com/2012/03/28/cobol-if-else-endif-statement/. [5.2.2013].

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].

Mic13 Micro focus visual cobol for visual studio 2010: Formation and evaluation rules, http://documentation.microfocus.com/help/index.jsp?topic=%2Fcom.microfocus.eclipse.infocenter.visualcobol.vs%2FHRLHLHCLANU058.html. [5.2.2013].

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].