hyväksymispäivä arvosana

arvostelija

COBOL ja Python: Tyypitys ja laskennan kapselointi

Erkki Heino, Tero Huomo, Eeva Terkki

Helsinki 12.2.2013

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Tyyppijärjestelmät

1.1 COBOL

COBOL on vahvasti ja staattisesti tyypitetty kieli [Wik13]. Olio-orientointunut COBOL (Object Oriented COBOL) sisältää sekä vahvan että heikon tyypityksen piirteitä. [AC96]

"In COBOL, there are really only three data types -

- numeric

- alphanumeric (text/string)

- alphabetic

The distinction between these data types is a little blurred and only weakly enforced by the compiler. For instance, it is perfectly possible to assign a non-numeric value to a data item that has been declared to be numeric.

The problem with this lax approach to data typing is that, since COBOL programs crash (halt unexpectedly) if they attempt to do computations on items that contain non-numeric data, it is up to the programmer to make sure this never happens."

- http://www.csis.ul.ie/cobol/course/DataDeclaration.htm

Eli kääntäjä ei tee tyyppitarkistuksia (ei olisi staattinen)? Vai puhuiko teksti käyttäjän syötteistä? (Mikä COBOLin versio?)

1.2 Python

Python on vahvasti tyypitetty kieli [Stat09]. Tietyn tyyppiselle muuttujalle ei voida tehdä toisen tyypin operaatioita ennen eksplisiittistä tyyppimuunnosta.

```
a = 5
b = "9"
c = a + int(b)
```

Esimerkissä `b` sisältää merkkijonon "9", mutta yhteenlaskussa merkkijonosta jäsennetään kokonaisluku. Jos kokonaislukujäsennyksen jättää tekemättä, antaa ohjelma kyseisellä rivillä poikkeuksen.

Suoritusaikana muuttujan tyyppi ei ole sidottu, vaan muuttujaan voi dynaamisesti sitoa eri vaiheessa eri tyyppisiä olioita. Seuraavassa esimerkissä muuttuja `a` saa ensin kokonaislukuarvon 5. Sen jälkeen muuttujan `a` arvoksi muutetaan merkkijono "hei".

```
a = 5
a = "Hei"
```

2 Laskennan kapselointi

2.1 COBOL

Muistilistaa:

- "Support for complexity management is minimal in COBOL, with no language features which support any structures larger than subprogram modules."(http://archive.adaic.com/d
)

- Funktioista COBOL ANSI-85 (http://public.support.unisys.com/aseries/docs/clearpath-mcp-13.0/pdf/86001518-310.pdf)

- Object Oriented COBOL (2002) tukee nimettyjä funktioita ja funktioprototyyppejä, sekä tietysti olioita/perintää/polyformismia (http://en.wikipedia.org/wiki/COBOL)

- Kielessä ei luontaista tukea rinnakkaislaskennalle (http://archive.adaic.com/docs/reports/lawlis/
) mutta osa vendoreista tarjoaa apuvälineitä rinnakkaisuuteen (esim. http://supportline.microfocus
)

2.2 Python

Pythonissa funktioiden parametrit ovat arvoparametreja. Lähestulkoon kaikki esitetään kielessä olioina, ja arvoparametrit ovat viitteitä olioihin; ne siis toimivat samalla tavalla kuin parametrina annetut oliot Javassa.

Funktioiden parametrit ovat tavallisesti ns. *positional-or-keyword*-parametreja [Pyt13a]. Tämä tarkoittaa sitä, että funktiokutsussa voi välittää argumentin joko sen sijainnin perusteella tai käyttämällä avainsanaa. Parametrille voi määritellä oletusarvon. Tällöin vastaavan argumentin voi jättää funktiokutsusta pois ja parametrin arvoksi annetaan määrätty oletusarvo. Ilman oletusarvoa olevat parametrit on listattava

funktion määritelmässä ensimmäisenä, ja niiden jälkeen tulevat oletusarvolliset parametrit.

Funktiokutsussa funktiolle välitettävät argumentit voivat olla avainsana-argumentteja (keyword argument) tai "tavallisia"argumentteja [Pyt13a]. Avainsana-argumentissa argumenttia edeltää tunniste, joka kertoo, mistä funktion parametrista on kyse. Avainsana-argumentit voivat olla funktiokutsussa keskenään eri järjestyksessä kuin vastaavat parametrit funktion määritelmässä. Argumentit, joissa ei käytetä parametrin tunnistetta, on annettava funktion määritelmän parametrilistauksen mukaisessa järjestyksessä.

Seuraavan funktion parametreista kahdella on oletusarvo ja yhdellä ei:

```
def tervehdi(nimi, tervehdys="Hei", viesti=None):
    print tervehdys, nimi
    if viesti:
        print viesti
```

Funktiota `tervehdi` kutsuttaessa vain yksi argumentti on pakollinen. Funktiokutsut erilaisilla argumenteilla tuottavat seuraavanlaisia tulostuksia:

```
>>> tervehdi("Pekka", "Moi")
Moi Pekka
```

```
>>> tervehdi("Pekka", viesti="Kvaak kvaak")
Hei Pekka
Kvaak kvaak
```

```
>>> tervehdi(viesti="Opettele Pythonia", nimi="Pekka", tervehdys="Terve")
Terve Pekka
Opettele Pythonia
```

Suoritusaikaiseen virheeseen johtavia funktiokutsuja puolestaan olisivat esimerkiksi kutsut `tervehdi()`, `tervehdi("Pekka", "Moi", tervehdys="Terve")` sekä `tervehdi(tervehdys="Moi", "Pekka")`.

Suoritusaikaisen virheen sattuessa Python-tulkki nostaa poikkeuksen [Pyt13b]. Poikkeus voidaan nostaa myös ohjelmakoodissa käyttämällä `raise`-lausetta. Poikkeuskäsittelijä määritellään `try..except`-lauseella, joka muistuttaa Javan `try..catch`-rakennetta. Poikkeuskäsittelijällä voi olla useita `except`-haaroja, ja Javan tavoin tilanteeseen sopiva haara valitaan poikkeuksen luokan mukaan.

Pythonissa on tuki rinnakkaiselle ohjelmoinnille. Kielen `threading`-kirjasto tarjoaa tähän esimerkiksi säikeet, lukot ja semaforit. `Queue`-moduuli tarjoaa turvallisen tavan säikeiden käsittelyyn.

3 Etuja ja haittoja

Lähteet

AC96 Arranga, E. C. ja Coyle, F. P., Object-oriented cobol.

Pyt13b Python language reference – execution model, `http://docs.python.org/2/reference/executionmodel.html#exceptions`. [12.2.2013].

Pyt13a Glossary, `http://docs.python.org/2/glossary.html`. [12.2.2013].

Stat09 Static vs. dynamic typing of programming languages, `http://pythonconquerstheuniverse.wordpress.com/2009/10/03/static-vs-dynamic-typing-of-programming-languages/`, 2009. [11.2.2013].

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