

Il Pitone non è un Serpente ovvero Quando Python non è Python

di

Federico Di Gregorio
<fog@initd.org>

- PyConDue – Firenze, 10/5/2008

90° Minuto

(12:24:32) C8E: quanto dura il tuo talk?

(12:27:07) fog: direi da 45m a 1h.

(12:27:14) fog: posso accorciare se vuoi.

...

(19:37:19) C8E: ci saranno scintille

(19:40:32) C8E: ah, e hai 90'

(19:41:17) fog: uh?

(19:41:23) fog: mXXXXXa..90' è tantissimo.

Artisti e Artigiani

Definire cosa sia un artigiano è semplice:

- Possiede degli strumenti
- Possiede delle tecniche
- Trasforma la materia in prodotto
- Se è bravo non è mai lo **stesso** prodotto

Definire un artista è molto più difficile:

- idem, idem, idem, idem

Il cattivo pasticcere

Il cattivo pasticcere lavora alla Nonnino Bianco e non è che faccia biscotti avvelenati ma:

- utilizza le pentole sempre allo stesso modo
- applica le ricette bovivamente, senza tenere conto degli ingredienti
- assaggia sempre allo stesso modo aspettandosi sempre gli stessi gusti
- spesso adora il caffè

Seramente, ora

Python ha introdotto fondamentali innovazioni come strumenti e tecniche.

Un linguaggio di programmazione non deve per forza essere C-like e prolisso o C-like ed oscuro (Java, Perl)

L'introduzione dei tecniche funzionali aumenta le potenzialità dei linguaggi OO.

Lo zucchero sintattico aumenta terribilmente l'espressività in domini specifici.

Quiz (facile!)

Indovinate che linguaggio è questo?

```
# No import? No party!  
def magritte(data, curs):  
    return "Ceçi n'est pas un(e) " + data  
NESTPAS = pe.new_type(  
    (textoid,), "NESTPAS", magritte)  
pe.register_type(NESTPAS)
```

Quiz (difficile?)

Ok, era facile, soprattutto per chi era alla PyCon (al mio talk) l'anno scorso. Ma questo?

```
class Person:
    """Uno, nessuno, centomila."""
    def IsBirthday(date):
        return (date.Month == self.birthdate.Month
                and date.Day == self.birthdate.Day)
```

Quiz (non è Python)

```
class Person
    var birthdate as DateTime
    var age as int

    def PrintBirthday
        print 'Il mio compleanno cade'
        print 'il [birthdate.Day]/[birthdate.Month]!'

    def GetOneYearOlder
        ensure
            .age = old .age + 1
        body
            age += 1
```

Quiz (non è VB!)

```
class Person:  
    def constructor(name, surname, birthdate):  
        with self:  
            _name = name  
            _surname = surname  
            _birthdate = birthdate
```

Però vi sto fregando perché “with” non esiste in nessuno dei linguaggi di cui parliamo.

Quiz (ok, siamo al delirio)

```
var nickies = {  
    "fog", "c8e", "vodka", "flexer", "cgabriel" };  
  
var query =  
    from s in names  
    where s.StartsWith('c')  
    orderby s.Length  
    select s.ToUpper();  
  
foreach (var s in query)  
    // Itera su "C8E" e "CGABRIEL"
```

from Python import *

I Python che non sono python, quelli che si ispirano (ammettendolo o meno):

- Jython
- IronPython
- BOO
- Cobra
- Javascript
- C# 3.0

Il vecchio ed il nuovo

Jython

- Prima versione di Python su VM alternativa
- Compilazione to byte-code JVM nativo
- Buona integrazione con Java

PyPy

- E` il "nuovo" Python
- E` da veri guru
- Andate a sentirvi il talk!

Quello che MS vuole, MS fa

Cinque anni fa, con un gruppo di lavoro che include gli inventori del concetto di IDE, copiando e destra e a manca MS introduce

Common Language Infrastructure

- VM stack-based con istruzioni OO generiche
- Estesa libreria di base (inc. code emission)
- Supporto per linguaggi multipli

IronPython

IronPython è la dimostrazione che il Common Language Runtime è funzionale anche per linguaggi dinamici.

- Open source (almeno come licenza)
- Prestazioni paragonabili a Python
- Accesso alla libreria standard
- Dynamic Language Runtime (open source)

COBRA

Ultimo nato della numerosa famiglia
“pythonici su CLR”.

- Ispirato a Python e Ruby (eresia!)
- Tipizzazione statica o dinamica
- Supporto per il contract-based programming

```
if obj implements Person
    cadeaux = obj.IsBirthday(DateTime.Now)
```

BOO

BOO è un linguaggio CLR-based, tipizzato staticamente nello spirito di Python.

- Tipizzazione statica e dinamica (quack!)
- Type inferencing
- Ereditarietà da qualsiasi linguaggio CLR che generi classi ECMA-compliant
- Pipeline di compilazione dinamica
- Macro sintattiche

Macro Sintattiche

Le macro sintattiche permettono di intervenire a qualsiasi livello dell'AST durante la compilazione:

```
class WithMacro(AbstractAstMacro):
    override def Expand(macro as MacroStatement):
        ReferenceExpression inst =
            macro.Arguments[0] as ReferenceExpression
        block = macro.Block
        ne = NameExpander(inst)
        ne.Visit(block)
        return block
```

Ancora macro sintattiche

```
class NameExpander(DepthFirstTransformer):
    _inst as ReferenceExpression

    def constructor(inst):
        _inst = inst

    override def OnReferenceExpression():
        if node.Name.StartsWith('_'):
            mre = MemberReferenceExpression(
                node.LexicalInfo)
            mre.Name = node.Name[1:]
            mre.Target = _inst.CloneNode()
            ReplaceCurrentNode(mre)
```

C# 3.0

Il C# parte come linguaggio C-like ispirato a C++, pascal, Python e Java (meno di quanto si creda). Il 3.0 aggiunge:

- Costrutti funzionali di alto livello: closures, expressions.
- Classi anonime ed inizializzazione statica di qualsiasi tipo.
- Programmazione funzionale mascherata da SQL: Language INtegrated Query (LINQ)

LINQ

Tutto è dolce con il LINQ:

```
var query =  
    from s in names  
    where s.StartsWith('c')  
    orderby s.Length  
    select s.ToUpper();
```

```
IEnumerable<string> query = names  
    .Where(s => s.StartsWith('c'))  
    .OrderBy(s => s.Length)  
    .Select(s => s.ToUpper());
```

Ma è C# o è LISP?

Python ci ha insegnato a prendere dal dominio funzionale, C# lo fa senza dirlo.

```
Func<string, bool> filter1 =  
    s => s.StartsWith('c');
```

```
Expression<Func<string, bool>> filter2 =  
    s => s.StartsWith('c');
```

```
filter2.Compile().Invoke(...);
```

Conclusioni

Riferimenti

- <http://boo.codehaus.org/>
- <http://cobra-language.com/>
- <http://www.jython.org/Project/index.html>
- (mi rifiuto di citare URL MS)
- fog@initd.org