

SPyppolare o non sPyppolare?

a script by

Federico Di Gregorio

Pycon Italia Qu4ttro - Firenze 8,9 maggio 2010

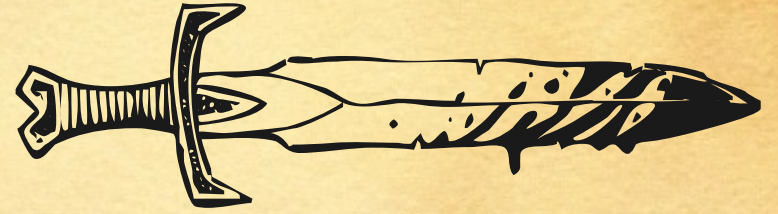


sPyppolare o non sPyppolare?

a script by

Federico Di Gregorio

Pycon Italia Qu4ttro - Firenze 8,9 maggio 2010



Nella bella Firenze,
dove noi collochiam la nostra scena,
molti linguaggi di pari nobiltà;
ferocemente gli uní agli altri oppone
da vecchia ruggine nuova contesa,
onde codice sublime va macchiando
chiaríssimi fosforí.



Molti Linguaggi

- Perché la azienda (per la quale, ovviamente, lavoriamo) ce lo impone.
- Perché non tutti i problemi sono chiodi.
- Perché non tutti i chiodi sono problemi.
- Perché “Erlang fa ma se vuoi fare questo e quello allora non puoi farne a meno...”

Relax Programming



Dramatis Personae - Basso Livello



Dramatis Personae – Java XP



Dramatis Personae – ???



Dramatis Personae – Sigh!



Programmare “Enterprise”



Programmare “Enter-Pool”



PyPremessa

Quelle parti del linguaggio che già sono Pythoniche non ci interessano più di tanto.

```
[x * x for x in get_all_xs()]
```

```
from x in get_all_xs() select x * x;
```



PyVantaggi

Andiamo ad elencare i vantaggi del Python...

- Compatto e leggibile
- Mix imperativo e funzionale
- Doolce! (nel senso di zucchero sintattico)
- Se non puoi risolvere un problema: aggiralo!
 - “Sei il GIL non si scrosta, scrostiamo i thread.”

PyEsempi

Tre esempi che a me piacciono molto:

- `nevow stan`, per la generazione di HTML
- `Stackless Python`
- `multiprocessing`, per parallelizzare
- `execnet` (solo perché l'ho visto ieri...)



Domínare il DOM

```
head = document.createElement("head")
```

```
title = document.createElement("title")
```

```
title.innerHTML = "Questo NON è stile!"
```

```
head.appendChild(title)
```

```
document.rootElement.appendChild(head)
```

```
body = document.createElement("body")
```

```
document.rootElement.appendChild(body)
```

```
...
```

PyStile

```
from nevw import flat, tags as T
```

```
html = T.html[
```

```
    T.head["Questo è Stile!"],
```

```
    T.body[T.p(class="red")]
```

```
        "E questo è un paragrafo rosso e stiloso...",
```

```
        lambda ctx, data: "...e funzionale!" ]]
```

```
print flat.flatten(html)
```

```
"<html><head>Questo è stile!</head><body>..."
```

PyStile#

```
using Tesseract.Html;
var T = new HtmlTemplate(); var html =
    T.head()["Questo è Stile#!"],
    T.body()[T.p("class", "red")["E questo è un paragrafo rosso e stiloso...",
        (ctx, data) => "...e funzionale!"]]]
Console.WriteLine(html.ToString());
"<html><head>Questo è stile!</head><body>..."
```

Stackless

- Modifiche all'interprete per supportare “tasklets”.
- Ovviamente non possiamo modificare XXX.
- Il pattern d'uso più comune consiste nell'avere un alto numero di tasklets che collaborano.
- Twisted fa più o meno la stessa cosa coi generator.

Mumble mumble...

My Name is I, IEnumerable

```
public class Agent007 {  
    Public Agent007(Scheduler s, Queue kills) {  
        this.kills = kills; this.scheduler = s; s.AddTask(Investigate());  
    }  
    IEnumerable Investigate() {  
        if (LocateSpectreAgent()) {  
            this.kills.Push(1); yield return Kill();  
        }  
        yield return DrinkMartini();  
    }  
}
```

My Name is I, IEnumerable (2)

```
IEnumerable Kill() {  
    this.schedule.AddTask(new KillAnimation(this).Start());  
}
```

```
IEnumerable DrinkMartini() {  
    yield return new Timespan(2);  
}
```

```
IEnumerable MeetBeautifulWoman() {  
    yield return new Signal("randezvous");  
}
```

```
}
```



multiprocessing

- Utilizzo delle caratteristiche esistenti del sistema
- API simile a quelle già esistenti (threading)
- Rimozione dello stato condiviso
- Primitive di sincronizzazione esplicite

MA

Di nascosto la libreria fa tutto ciò che voi non fate!

“square-reduce”

```
def square_range(q, a, b):  
    for x in xrange(a, b):  
        q.put(x*x)
```



```
q = Queue()  
p1 = Process(target=square_range, args=(q,0,1000))  
p2 = Process(target=square_range, args=(q,1000,2000))  
...  
# Recuperiamo i dati qui...
```

“square-reduce-faster”

```
def square_range(q, a, b):  
    result = [x * x for x in xrange(a, b)]  
    q.put(result)  
  
q = Queue()  
p1 = Process(target=square_range, args=(q,0,1000))  
p2 = Process(target=square_range, args=(q,1000,2000))  
...  
# Recuperiamo i dati qui...
```

“square-reduce-done-right”

- Incredibile quanto in passato le librerie per il multithreading si siano concentrate sulle primitive.
- Semantica che non richieda lock espliciti.

```
Parallel.Range(1, 1000000)
```

```
    .Select(x => x * x)
```

```
    .Sum();
```

Duck Typing...



PyQuack!
PyQuack!

...is sexy!

Quack! Quack!

“When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.”

Hiss! Hiss!

Se vedo un serpente che striscia come un Pytone, che sibila come un Pyton e che si programma come un Pytone, non é detto che sia un Pytone.

Ma il fatto che ci assomigli mi rende molto, molto, molto felice!

The End

Federico Di Gregorio

⟨fog@initd.org⟩