python

introduzione al linguaggio

Federico Caboni
<msx2k@mac.com>

GULCh – Gruppo Utenti Linux Cagliari

Linux Day 2003

Giornata nazionale di Linux e del software libero

Python?

- Python è un linguaggio di programmazione...
  - interpretato
  - di altissimo livello
  - semplice
  - da imparare
    - da usare
    - da ricordare
  - **elegante e coerente**
  - potente e produttivo
  - <insert buzzword here> :-)

Si, OK, ma perchè “Python”?

- Il nome viene dai “Monty Python”...
- ...quindi i serpenti non c'entrano...
- no, *davvero*, i serpenti non c'entrano nier

Scopo del talk

- **Non** è un “minicorso di Python”
  - non c'è abbastanza tempo
  - non ci sarebbero superstiti... :-)
    - ...il che potrebbe avere conseguenze penali...
- Introduzione delle caratteristiche del linguaggio, dei suoi utilizzi...
- ...per suscitare la curiosità....
- ...e un sacco di Filosofia :-)

Why should I care?

- Perchè dovrebbe interessarvi un'**altro** linguaggio?
- Python è **semplice**, davvero **semplice**
- e quindi è molto **produttivo**
- ottimo anche come **primo linguaggio**
  - **ma non è un “linguaggio giocattolo” ;-)**
- è elegante (diffidate dalle imitazioni)
- perchè ve lo dico io
  - altrimenti non sareste qui, no?

In più, per le prime 10 telefonate

E' *open source*

- E' multipiattaforma
  - UNIX, ma non solo...
- E' facilmente integrabile con C/C++ e Java
  - ci torneremo dopo...
- **Batteries Included!**
  - anche su questo, promesso...

Ma lo usa qualcuno...vero?

- **RedHat**
  - anaconda (installazione), tool di configurazione grafici, log viewer
- **NASA**
- **www.google.com**
- **Industrial Light and Magic**
  - si, quelli che fanno gli effetti speciali per Star Wars... ;-)
- **and so on...**

Ha un campioncino di prova?

- Python è *interpretato*
- E' distribuito con un ottimo interprete interattivo...
  - potete usarlo anche come calcolatrice...
    - ...ehi, **io** lo faccio! :-)
  - utilissimo per provare le cose “al volo”
  - può diventare una “ultrashell programmabile”
  - ha molte funzioni per l'uso interattivo (**dir()**)
- Il prompt di default è **>>>**

Ma com'è fatto sto' Python?

- OK, cominciamo da qualcosa di semplice:
-
- **thx = 1138 #crea un nome**
- **thx = 'ciao' #riassegna il nome**
-
-
-

ehi, tipo...

- A prima vista sembrerebbe che Python non abbia *tipi*
- Python in realtà è **strongly typed**
- Ogni oggetto ha un tipo, che non può cambiare *mai*
- Le variabili (**references**)**,** invece, sono come delle “etichette” che si appiccicano agli oggetti (**binding**)**:** loro di tipo non ne hanno

Oh no, more Python!

- >>> domanda = risposta = 42
- >>> print domanda, risposta
- *42, 42*
- >>> x = 5; y = 10
-
- >>> x,y = y,x #si, funziona :-)
-
- >>> if risposta < 50: print risposta
-

Nessuno si aspetta un linguaggio con indentazione obbligatoria...

- In Python, non si usano parole chiave (*begin/end*) o simboli ({}) per delimitare i blocchi logici
- si usa solo l'**indentazione**...
- ...che è **obbligatoria**
- se non indentate, il programma **non funziona**
  - se siete pigri come me apprezzerete i caratteri in meno che dovrete scrivere :-)

Se e solo se

```
a = 5
b = 10
if a+b > 1:
        print “spam”        #indentato
elif a >0 and b>1:          #and, or, not.. :-)
        print “eggs”        #indentato
else:
        print #la riga dopo i : è sempre indentata
```

loops

```
a = 0
while a < 10:
      a += 1

for i in 0,1,2,3,4:
      a += i

for i in range(5):
      a += i    #stessa cosa
```

Beh, oggettivamente...

- In Python tutto è un **oggetto**...
- ...e ogni oggetto ha un **tipo**...
- ...e tutti gli oggetti sono **uguali**...
- anche le *funzioni*, ad esempio...
- ... potete passarle come argomenti di altre funzioni, cambiare loro nome, metterle in una lista, **assegnarvi attributi**...

Tipi di oggetti

- *Numeri*
  - int, long, float, complex
- *Sequenze*
  - str, list, tuple
- *Mapping*
  - dict
- *e tanti altri...*
  - file, function, class, instance, None, ...

per esempio?

- [1,2,”pippo”,3,0]
  - lista, come gli array ma anche con tipi diversi
- {“thx” : 1138, “foo” : “bar”}
  - dizionario, coppie di chiave/valore...
- [[0,1],[1,2],[3,4]]
  - lista di liste: ecco a voi un array bidimensionale

list comprehension

-
- **[x**2 for x in range(10)]**
  - *[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]*
-
- **[x for x in range(1,10) if not x%2]**
  - *[2,4,6,8]*

ma funziona?

```
def subadd(x,y): #ritorna val. multipli
    return x+y, x-y

def adder(*args):    #parametri variabili
    res = 0
    for num in args:  #args: lista degli argomenti
        res += num
    return res
```

Un tocco di classe

- class Pinocchio(Burattino):
- 	def __init__(self):
- 		self.legnetto = 0
- 		self.naso = 45.0
- 	def bugia(self):
- 		self naso += 50
-
- p = Pinocchio()
- p.padre = “Geppetto”

operatore?

- E' facile ridefinire gli operatori e il comportamento di un oggetto ridefinendo alcuni metodi particolari:
  - __add__ *per la somma*
  - __len__ *per calcolare la lunghezza*
  - __cmp__ *per comparare oggetti*
  - *etc. etc. etc.*
- E' facilissimo fare oggetti che supportino gli operatori come quelli di builtin
  -

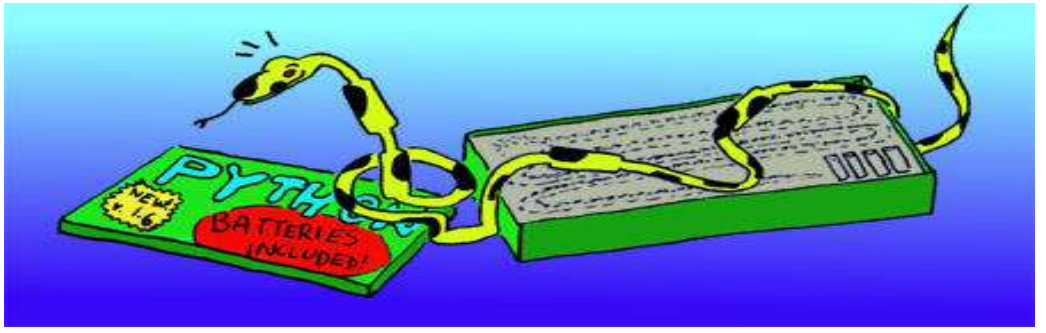
E ora, qualcosa di completamente diverso...

A cosa serve ?

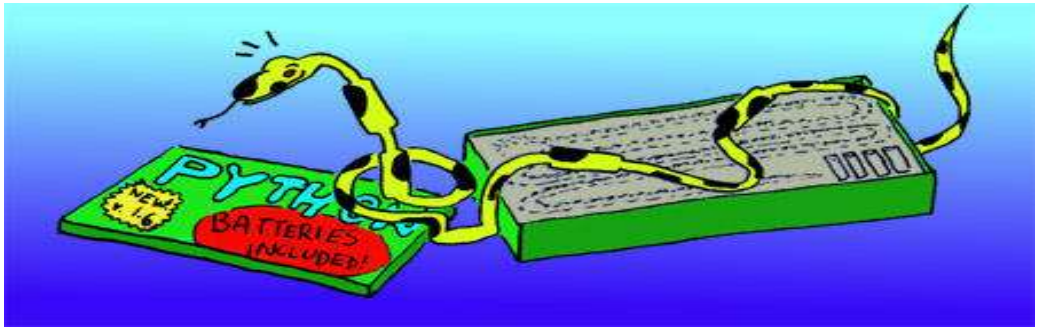
```
import sys
import string

f = file(sys.argv[1])

words = {}
punt = ";,:."

for line in f:
   for word in line.split():
      try:
         words[word.strip(punt)] += 1
      except KeyError:
         words[word.strip(punt)] = 1

keys = words.keys()
keys.sort()

for i in keys:
   print i, ":", words[i]
```

Batteries Included!

- libreria standard :
  - **180** moduli “comuni”
    - math, sys, os, sets, re, random, pydoc, gzip, threading, csv...
    - socket, select, xmlrpclib, ftplib, rfc822
  - **8** packages con altri **70** moduli
    - bsddb, compiler, curses, distutils, email, logging, xml
  - **80** moduli di encoding, **280** moduli di unit-test**,** **180** moduli in Demo/ e **180** in Tools...
  -
    -

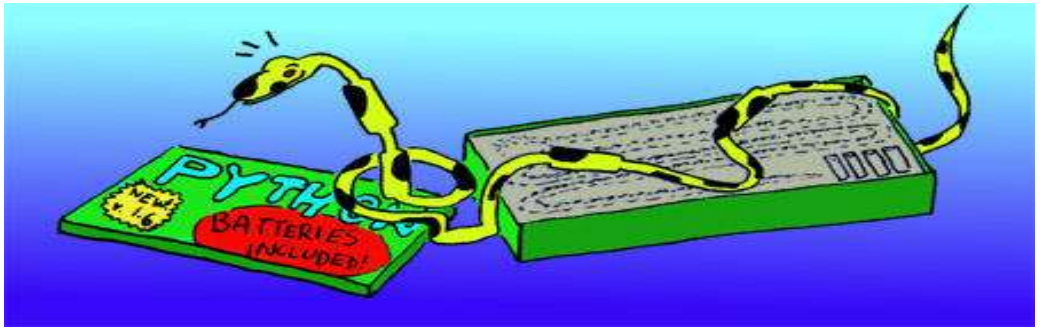


altre batterie – GUI

- **Interfacce grafiche**:
  - Tkinter (*incluso*, usa Tcl/Tk)
  - wxPython
  - PyQt
  - PyGTK
  - Pythonwin (*solo per windows ;-p*)
  - Cocoa,Swing, AWT, PyUI, PicoGUI, EasyGUI...
-
  -
    - 

altre batterie – vari

- **reportlab**
  - generazione di PDF
- **Twisted Matrix**
  - networking asincrono
- **PIL**
  - image processing
- **Numeric, SciPy**
-
  -
    -



Tool di sviluppo

- **Emacs** ha un ottimo *python-mode*
- **vi**, beh..fa del suo meglio..tsk :-p
- Free (as in speech and as in beer) IDEs: **IDLE** (incluso!)
- **Boa Constructor**
- **Pythonwin** (solo per windows ;-p)

Integrazione C/C++

- estensione:
  - Python C API
  - SWIG
  - Boost Python
  - Pyfort, Pyrex, CXX, SCXX, sip, COM...
- embedding:
  - Python C API
  - Boost Python
  -

Integrazione Java

- Esiste una versione di Python scritta IN JAVA
- Si possono usare classi Java da Python, e viceversa
- Il codice Python viene byte-compilato in bytecode JVM
- L'interprete Jython è accessibile da Java
-

And now.... the meaning of life...

The Zen of Python *(T. Peters)*

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
-

The Zen of Python *(T. Peters) – 2*

- In the face of ambiguity, refuse the temptation to guess.
- There should be one-- and preferably only one --obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea -- let's do more of those
-

Risorse

-
- **www.python.org**
  - sito ufficiale: download, documentazione, TUTTO...
- **www.python.it**
  - sito italiano... al momento down :-(
-

Ringraziamenti

- Ringrazio **Alex Martelli** per i suggerimenti e il materiale fornito
- Ringrazio **Guido von Rossum** per aver creato Python
- Ringrazio **voi** per l'attenzione :-)

Domande?

- 42