Adopt a reptile! Use Python.

JOKER Ltd.

Chi è questo python?

- Il Python e' un linguaggio di alto livello, adattabile, ad oggetti, con una sintassi chiara ed estremamente leggibile, tipizzato dinamicamente, multipiatttaforma.

Ma come funziona?

- Come il Java e' metacompilato, ma al contrario di Java la compilazione e' implicita. Nel momento in cui si lancia un file python, con estenzione .py, l'interprete/compilatore controlla. Se la versione compilata presente, con estensione pyc solitamente, e' meno recente della versione sorgente, provvede a ricompilare, altrimenti usa direttamente il metacompilato esistente.

Ma e' completo?

- In Python tutto e' un oggetto, e quando dico tutto intendo dire tutto.
- E' un linguaggio "battery included", dove la libreria di sistema e' cosi' ampia da poter affrontare quasi tutte le esigenze che si possono presentare.

Caro, c'e' l'immondizia da buttare

- Ha un sistema di garbage collection potente ed automatico.
  Tenete presente che essendo tutto dinamico e ad oggetti, sarebbe impossibile far andare qualcosa senza impazzire se cosi' non fosse.

E' morto lo zio d'america

- Presenta un meccanismo di ereditarieta' multipla tra le classo, al contrario di Java e PHP che nascono con la singola, l'overloading degli operatori, ed una infinita' di librerie di terze parti, quasi tutte disponibili gratuitamente e in formato sorgente (sotto licenze open source) per fare praticamente ogni cosa.

Non e' bella ma e' un tipo

- La tipizzazione (definizione dei tipi di oggetti e variabili) e' forte ma dinamica, non necessita di dichiarazioni di tipo, si assegna il valore alla variabile e questa viene tipizzata in base al valore, presenta una sintassi semplice e pulita, i blocchi sono definiti dall'indentazione quindi diviene impossibile scrivere il codice in maniera farragginosa.

Usato? Sicuro!

- Qualcuno potrebbe chiedersi: si ma se e' cosi' buono, chi e' che lo usa?
  La risposta, come per Linux, e': “tanti che pero' non te lo vengono a raccontare”.

Mi ritorni in mente

A partire da Google, che ha assunto tra gli altri Guido Van Rossum, il *benevolo tiranno* che ha creato questo linguaggio, alla NASA, a Canonical (molte parti di Linux che lavorano a livello di scripting ora vengono scritte in Python) Yahoo, IBM, Red Hat, e persino i Disney Animation Studios, solo per citare i primi che mi vengono in mente.

Linguaggi Write Only

- E' facile da apprendere (un amico che all'epoca lavorava in PHP, riusci' ad imparare il linguaggio e scriversi il web server che gli serviva in 3 giorni, mentre per farlo in PHP, che conosceva gia' e piuttosto bene, avrebbe impiegato una settimana),e' facile da usare ed e' facile da leggere, dopo.

Ma lo vendono all'ipercoop?

- Lo trovate preinstallato su tutte le versioni di Linux e Mac Os X.
  Se siete dei windows-ari beh dovrete fare la fatica di installarlo voi. Ma sospetto che presto anche microsoft lo adottera'.

Ciascuno si fa del male come crede

- Tra parentesi esiste una versione, Iron Python, per l'ambiente di sviluppo .NET, se proprio non potete farne a meno, come esiste Jthon per chi vuole usare le librerie Java con Python.

Il mondo in una conchiglia

- Presenta una sua shell interattiva, evocabile semplicemente con il comando "python" seguito da invio, che vi permette di fare prove al volo di pezzi di codice.

Ma guarda che tipo!

- Anche se i tipi base non sono vere classi, una classe può ereditare da essi.
  In questo modo è possibile estendere stringhe, dizionari e perfino gli interi.

Fatelo a fette!

- In Python esiste un meccanismo di gestione detto slicing che permette di suddividere stringhe o array, in una maniera differente da quella degli altri linguaggi.

Questione di posizioni

- La posizione 0 in un array tradizionale rappresenta il primo elemento.

```
|C|I|A|O|
 0 1 2 3
```

- In Python e' il separatore che precede il primo elemento.

```
|C|I|A|O|
0 1 2 3 4
```

Impariamo a tagliare

- Di conseguenza in python possiamo scrivere:

```
lista = [1,2,3,4]
print lista[0:1]  => [1]
print lista[0:2] => [1,2]
```

- Ma anche:

```
print lista[-1:0:-1] => [4,3,2]
```

E in pratica?

- Tempo fa mi era stato chiesto in un test di selezione del personale di una azienda di ordinare una lista (array) in ordine inverso senza usare funzioni built-in o librerie.

  Il quesito successivo chiedeva la versione senza for e if.

  La risposta era la stessa in entrambi I casi.

Piu' facile di cosi'

```
lista = [1,2,3,4,5,6,7,8,9]
print lista[-1::-1]
=> [9,8,7,6,5,4,3,2,1]
```

Capire le liste

- Altra feature davvero potente sono le *list comprehension*. In pratica una maniera di manipolare una lista man mano che la si scorre, applicando ad esempio, un filtro sulla stessa. Qualcosa come:

```
lista = [1,2,3,4,5]
print [2 ** n for n in lista ]
```

che stampera' i quadrati dei vari numeri.

Impariamo il decoro

- Python presenta anche i decoratori, di recente adottati anche da altri linguaggi tra cui il C#, sono oggetti Python chiamabili usati per modificare una funzione, un metodo o una definizione di classe.

  Un decoratore è passato all'oggetto e ritorna l'oggetto modificato.

  Viene distinto dalla parola chiave @ che si prepone al nome del decoratore.

Oppure?

- Altra caratteristica particolare di Python e' l'uso della keyword *ELSE* non solamente con il costrutto *IF* ma anche con il *WHILE* e persino con il *TRY*.

  In pratica si tratta di quella parte di codice che deve venire eseguita "nel caso che nessun altra parte sia stata eseguita".

Back to future

- Va notato infine che scrivere estensioni in C o C++ da utilizare all'interno di Python dove le performance siano davvero importanti.

I need a little help from my friends

- Va tenuto presente che in Python, che ricordo e' un linguaggio battery included, anche l'help e' integrato.
- Infatti nella shell interattive basta scrivere

help(*comando*)

dove comando puo' essere una keyword, una funzione, una classe, o quant'altro e leggeremo le note che chi ha scritto il codice ha inserito nell'apposito formato.

Autodocumentazione automatica

- Questo perche' l'autodocumentazione e' integrata e quindi e' sufficiente quando si scrive ad esempio una classe, subito dopo averla definita bastera' racchiudere tra due coppie di 3 apici, singoli o doppi, il testo che vogliamo appaia quando si invooca l'help, ed ecco che l'interprete si preoccupara' di tutto.

```
''' Sono un commento '''
"""io pure"""
```

E' arrivato il direttore

- Altra utile funzione e' dir(), che ha la stessa sintassi ma ci torna una lista di componenti dell'oggetto. In questo modo anche se si ha a che fare con librerie non note si puo' facilmente capire come siano state strutturate e come vadano utilizzate.

```
import sys
dir(sys)
['__displayhook__', '__doc__', '__egginsert', '__excepthook__',
    '__name__', '__package__', '__plen', '__stderr__', '__stdin__',
    '__stdout__', '_clear_type_cache', ...]
```

Modularita'

- Altro punto, i moduli.
  In pratica si puo' spezzare il codice in diversi moduli che possono venire importati semplicemente importandoli, completi con l'istruzione

```
import nomemodulo
```

  oppure solo le singole componenti, come un metodo di una classe, con la sintassi

```
from nomemodulo import nomemetodo
```

Dove lo avro' messo?

- La sola cosa importante e' che il modulo si trovi o nello stesso package di quello che lo importa oppure all'interno del PYTHONPATH
  ovvero quella lista di directory dentro cui cercare le varie librerie.

Fai da te

- Per creare un modulo e' sufficiente che nella cartella che contiene i files sia presente un file che si chiama
  __init__.py
  Il nome del modulo sara' quello della cartella contenitore.

Lost in the deep namespace

- Se si e' importato un modulo con la sintassi

  import nomemodulo

  per utilizzare i suoi componenti si dovra' indicare il path completo, detto *namespace*, come

  sys.version

  che ci torna le info sulla versione di python utilizzata.

Chiariamoci

- I namespaces sono estremamente importanti perche' permettono di evitare ambiguita'.

  In pratica possiamo avere due funzioni, metodi, classi con lo stesso nome in differenti namespaces ed evitare di chiamare quello sbagliato semplicemnte indicando il namespace completo.

Come lo vuoi?

- In Python ci sono inoltre tantissimi frame-works, tanto per sviluppare applicazioni web based (Django, Turbo Gears, Zope, Flask, Web2Py, etc.) che applicazioni client-server (WxPython, PyQT, PyGtk) e di recente anche per il mobile (Kivy).

E' pasquaaaaaa

- Termino il tutto citandovi lo Zen of python, che potrete leggere semplicemente digitando nella shell interattiva l'istruzione

```
import this
```

un simpatico easter egg.

Zen of Python – part 1

- 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.

Zen of Python – part 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!

Piccolo, spazio, pubblicita'

- Per tutti gli interessati, a breve partira' il progetto chiamato **Pythonisti Anonimi**, una specie di ILS per il Python.
- Un manifesto e non una vera associazione. Per chi fosse interessato scriva a *pythonistianonimi@gmail.com* e vi terremo informati.