Python "robusto" e "fluente"?

Discussione su alcune recenti novità introdotte nel linguaggio Python.

Giuseppe Chellini

GULP - Gruppo Utenti Linux Pisa

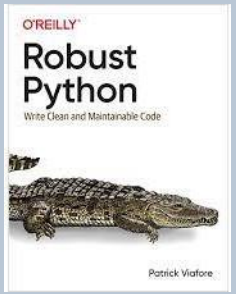
28 Ottobre 2023

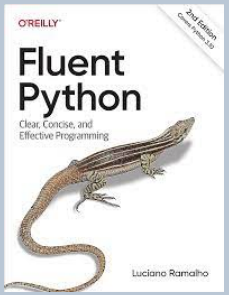
Di cosa parleremo...

Python "Robusto" e "Fluente"...

Questi sono gli aggettivi con cui è stato descritto in alcuni testi di programmazione.

Alcune delle fonti di informazione





Di cosa parleremo...

L'origine di questi aggettivi è legata ad alcune caratteristiche introdotte gradualmente negli ultimi anni. (*"Recenti" è una parola relativa...*)

Python è un linguaggio che evolve secondo linee discusse mediante i PEP.

Documenti di discussione

PEP : *Python Enhancement Proposal*
`https://peps.python.org/`

Di cosa parleremo...

Nelle ultime versioni sono state introdotte e sviluppate, tra le altre, alcune evoluzioni del linguaggio che riguardano i seguenti aspetti:

Gestione dei tipi

PEP 484 : *Type Hynting* (`https://peps.python.org/pep-0484/`)

Sintassi e strutture dati per la gestione dei tipi

PEP 526 : *Syntax for Variable Annotations* (`https://peps.python.org/pep-0526/`)
PEP 557: *Data Classes* (`https://peps.python.org/pep-0557/`)

Structural Pattern Matching

PEP 622 : *Structural Pattern Matching* (`https://peps.python.org/pep-0622/`)

Python è un linguaggio "tipato"?

La semplicità di utilizzo di Python, che non richiede definizione dei tipo per l'esecuzione di codice, porta a dire di **no**.

Questo non è corretto!!!

Python effettua il controllo di tipo che viene chiamato "**DUCK TYPING**"

Duck test

"If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck."

Se assomiglia a un'anatra, nuota come un'anatra e starnazza come un'anatra, allora probabilmente è un'anatra.

In effetti il termine giusto per la modalità adottata da Python è **Tipizzazione Dinamica**, effettuata dall'interprete a *tempo di esecuzione*.

Esempi di esecuzione del codice

bash

IPython: home/giuseppe

IPython: home/giuseppe 110x20

```
[giuseppe@fedora ~]$ ipython
Python 3.11.6 (main, Oct  3 2023, 00:00:00) [GCC 13.2.1 20230728 (Red Hat 13.2.1-1)]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.14.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: def somma(x,y):
   ...:     return x+y
   ...:

In [2]: somma(1,2)
Out[2]: 3

In [3]: somma("primo","secondo")
Out[3]: 'primosecondo'

In [4]: somma([1,2,3],[4,5,6])
Out[4]: [1, 2, 3, 4, 5, 6]

In [5]:
```

Pro/Contro della tipizzazione dinamica

Pro

- il tempo necessario per la scrittura di codice funzionante è ridotto rispetto agli altri linguaggi;
- il codice può essere utilizzato indifferentemente su diverse tipologie di dato;
- consente l'utilizzo di una grande quantità di forme sintattiche che rendono semplice la scrittura di algoritmi anche di elevata complessità.

Contro

- il codice può risultare non funzionante in funzione del tipo di dato fornito;
- permette di scrivere codice poco comprensibile e poco auto-documentato;
- l'utente deve saper gestire il funzionamento del codice in caso di input non corretto.

Introduzione del controllo dei tipi

PEP 484

Annotazione con indicazione dei tipi

PEP 484 – Type Hints

Author:	Guido van Rossum <guido at python.org>, Jukka Lehtosalo <jukka.lehtosalo at iki.fi>, Łukasz Langa <lukasz at python.org>
BDFL-Delegate:	Mark Shannon
Discussions-To:	Python-Dev list
Status:	Final
Type:	Standards Track
Topic:	Typing
Created:	29-Sep-2014
Python-Version:	3.5
Post-History:	16-Jan-2015, 20-Mar-2015, 17-Apr-2015, 20-May-2015, 22-May-2015
Resolution:	Python-Dev message

▸ **Table of Contents**

Esempi di esecuzione del codice

bash

IPython: home/giuseppe

IPython: home/giuseppe 120x20

```
[giuseppe@fedora ~]$ ipython
Python 3.11.6 (main, Oct  3 2023, 00:00:00) [GCC 13.2.1 20230728 (Red Hat 13.2.1-1)]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.14.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: def somma(x:int,y:int)->int:
   ...:     return x+y
   ...:

In [2]: somma(1,2)
Out[2]: 3

In [3]: somma("primo","secondo")
Out[3]: 'primosecondo'

In [4]: somma([1,2,3],[4,5,6])
Out[4]: [1, 2, 3, 4, 5, 6]

In [5]:
```

Come funziona?

Il sistema di annotazione del codice consente di effettuare il controllo di correttezza formale prima della sua esecuzione, in una sorta di *pre-compilazione*. Tale operazione è effettuata mediante l'ausilio di strumenti appositi (es. **mypy**).

mypy

https://mypy-lang.org/

Per installare:

$ python3 -m pip install mypy

Per eseguire:

$ mypy program.py

Diventerà obbigatoria l'annotazione del codice?

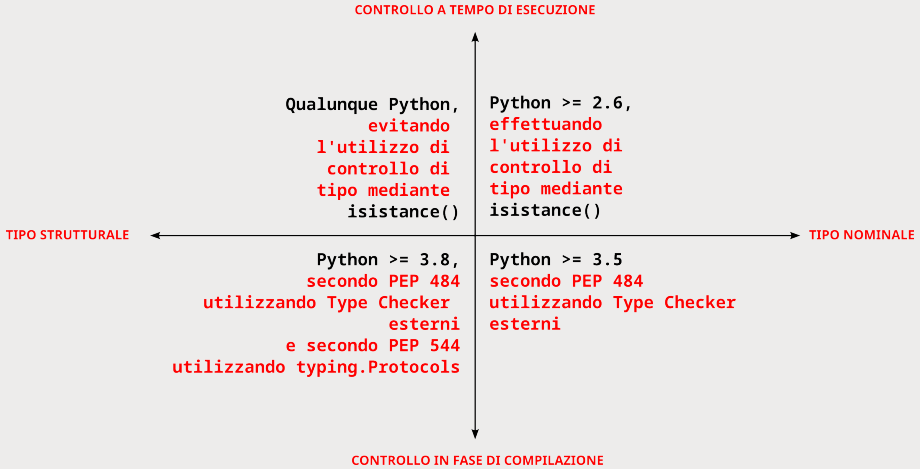
L'annotazione del codice costituisce una **forma di documentazione** ed un mezzo per poter fare **analisi statica** del codice prima della sua esecuzione.

PEP 484

It should also be emphasized that Python will remain a dynamically typed language, and the authors have no desire to ever make type hints mandatory, even by convention.

Va anche sottolineato che Python rimarrà un linguaggio tipizzato dinamicamente, e gli autori non hanno alcun desiderio di rendere obbligatori i suggerimenti di tipo, anche per convenzione.

Possibili modalità di gestione dei tipi



typing.Protocols : suggerimenti di tipo a tempo di esecuzione

Il sistema di annotazioni consente di effettuare la verifica della correttezza dei tipi mediante analisi statica. Posso effettuare controlli a tempo di esecuzione (mantenendo efficace la tipizzazione dinamica?

PEP 544

PEP 544 – Protocols: Structural subtyping (static duck typing)
`https://peps.python.org/pep-0544/`
"Structural subtyping is natural for Python programmers since it matches the runtime semantics of duck typing: an object that has certain properties is treated independently of its actual runtime class."

Structural Pattern Matching

Python ha introdotto nuove forme sintattiche mirate a semplificare la scrittura di controlli complessi.

match

Syntactically, a match statement contains:

- a subject expression
- one or more case clauses.

Each case clause specifies:

- a pattern (the overall shape to be matched)
- an optional “guard” (a condition to be checked if the pattern matches)
- a code block to be executed if the case clause is selected

Structural Pattern Matching

Posso avere diverse tipologie di Pattern Matching:

tipologie

- su singoli valori
- su tuple
- su valori alternativi (OR)
- su sequenze
- su dizionari
- su classi

Software per la presentazione

IMPORTANTE!!!

La presentazione è stata realizzata interamente mediante SOFTWARE LIBERO.

- Python ovviamente...

  (https://www.python.org/)
- Latex con pacchetto Beamer

  (https://www.ctan.org/tex-archive/macros/latex/contrib/beamer)
- editor per LaTeX: TexStudio (http://www.texstudio.org/)
- Beamer Template: Laughlin by Mohamed El Morabity

  (https://fedoraproject.org/wiki/Templates_for_Presentations)

Saluti

Domande?

Buon Linux Day!