

Desmitificando Python

Fisa (Juan Pedro Fisanotti)
fisadev@gmail.com
[@fisadev](https://twitter.com/fisadev)



Compilado o interpretado?

**“En Java/C# el código fuente se compila a binario.
Luego ejecutamos el binario.”**

**“En Python el código fuente es interpretado instrucción
por instrucción a medida que se ejecuta.”**

“Python es lento porque es interpretado.”

“En Java/C# el código fuente se compila a binario. Luego ejecutamos el binario.”

“En Python el código fuente es interpretado instrucción por instrucción a medida que se ejecuta.”

“Python es lento porque es interpretado.”

Java/C#: compila a un lenguaje intermedio. Runtime interpreta el lenguaje intermedio.

Java/C#: compila a un lenguaje intermedio. Runtime interpreta el lenguaje intermedio.

Python: compila a un lenguaje intermedio. Runtime interpreta el lenguaje intermedio.

Java/C#: compila a un lenguaje intermedio. Runtime interpreta el lenguaje intermedio.

Python: compila a un lenguaje intermedio. Runtime interpreta el lenguaje intermedio.

La diferencia es cuándo se compila.

Python: al ejecutar el programa.

Java/C#: a mano antes de distribuir el programa.

Dirige nos misericordiam tuam
in corde nostrum. Deus misericordia
nisi et bene. Deus i. id



La lentitud no tiene que ver con esto!

Tarea para el lector: el mundo fuera de CPython

Tipado estático o dinámico?



“En los lenguajes de tipado estático, todas las cosas tienen el tipo de dato definido en código. En los dinámicos, los tipos se saben recién en ejecución”

~~“En los lenguajes de tipo~~ ~~estático~~, todas las cosas tienen el tipo de dato de ~~código~~ en código. En los dinámicos, los tipos se ~~sab~~ recién en ejecución”

“En los lenguajes de tipo de dato estático, todas las cosas tienen el tipo de dato definido en código. En los dinámicos, los tipos se sabe recién en ejecución”



En todos hay info de tipos estática y dinámica.

Estática: mirando el **código fuente**.

Dinámica: al **ejecutar** el programa.

**Python tiene info estática de tipos?
Sí, bastante!**

**Python tiene info estática de tipos?
Sí, bastante!**

La info puede ser inferida.

La info puede usarse para hacer **chequeos** (estáticos o dinámicos).

La info puede usarse para hacer **chequeos** (estáticos o dinámicos).

En esto difieren más!

Python **no hace** chequeos estáticos :(...

La info puede usarse para hacer **chequeos** (estáticos o dinámicos).

En esto difieren más!

Python **no hace** chequeos estáticos :(...

Pero hay muchas herramientas que **lo hacen!**
No usarlas no es culpa de Python :)



Autocompletado copado como en los
lenguajes estáticos?

100% no. Pero se puede tener **mucho.**

Es cuestión de usar un buen editor o IDE :)



Si es dinámico, entonces su tipado es débil?

No, las cosas tienen tipo bien definido!

No, las cosas tienen tipo bien definido!

Python es de **tipado fuerte** (como Java o C#, no como PHP o Javascript).

Un X es un X, y solo puede hacer lo que X sabe hacer.

silns et si
legatus

illo canoma statim infie
rues spoliunt eos pittac



No es muy lento por ser dinámico?

Ni. Técnicamente sí, peeeeero... en la mayoría de los casos **el lenguaje no es el cuello de botella**.

Ni. Técnicamente sí, peeeeero... en la mayoría de los casos **el lenguaje no es el cuello de botella**.





No es muy propenso a errores por ser dinámico?

Con unit tests y las herramientas que conté, no.

Con unit tests y las herramientas que conté, **no**.

En los lenguajes estáticos hacemos **unit tests** y
chequeo estático.

En Python hacemos lo mismo :)



Parámetros por valor o referencia?

**Todas las variables son referencias.
Y los parámetros son variables.**

**Todas las variables son referencias.
Y los parámetros son variables.**

y = 1

x = y

“que x apunte a lo mismo que y”

En Python **todas** las variables son referencias.
Y los parámetros son variables.

y = 1

x = y

“que x apunte a lo mismo que y”

foo(y)

“llamá a foo, y que su primer parámetro apunte a lo mismo que y”

Si hacemos que **y** apunte a otra cosa, **x** sigue igual apuntando a lo mismo que antes.

x = y

y = 2

NO me cambia **x**.

Ludus monstrorum.



Los objetos inmutables no se maneja diferente?

No.

No.

El manejo de referencias es igual para **todos** los objetos, mutables o no.

No.

El manejo de referencias es igual para **todos** los objetos, mutables o no.

Objetos inmutables → tranquilidad de que otros no los modifiquen.

?



Fisa (Juan Pedro Fisanotti)
fisadev@gmail.com
@fisadev