Comenzando con Python

Luis Miguel de la Cruz Salas

Contenido

1. Brevísima historia de Python.
    a. ¿Qué es Python realmente?
2. Pensando como *Pythonista*: las bases.
    a. ¿Variables, objetos o etiquetas?
    b. Tipado dinámico.
    c. Primer vistazo a la POO.

Brevísima historia de Python

"Over six years ago, in December 1989, I was looking for a 'hobby' programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus)."

https://en.wikipedia.org/wiki/Guido_van_Rossum

Guido

Monty Python

Propuesta de Van Rossum a DARPA[1]
Computer Programming for Everybody (CP4E) [2]

- Python debería ser fácil, intuitivo y tan potente como sus principales competidores.
- El proyecto sería de Código Abierto para que cualquiera pudiera colaborar.
- El código escrito en Python sería tan comprensible como cualquier texto en inglés.
- Python debería ser apto para las actividades diarias permitiendo la construcción de prototipos en poco tiempo.

1 Defense Advanced Research Projects Agency, 1999
2 https://www.python.org/doc/essays/cp4e/

Versiones de Python

	1994	2000	2008	2010	2020	2021	2023	2024
Python 1	1.0 …	… 1.6						
Python 2		2.0 …	… 2.6	… 2.7	Jan			
Python 3			3.0 …	3.1 …	3.9.1 Dec	3.7 - 3.9 3.6 Dec	3.7 Jun	3.8 Oct

¿Qué versión debería usar?

R: cualquiera desde la **3.7** hasta la **3.9.1**

En este curso usaremos la versión 3.8.5

Fuentes:

- https://es.wikipedia.org/wiki/Historia_de_Python (https://en.wikipedia.org/wiki/History_of_Python)
- https://www.python.org/downloads/
- Información histórica.

¿Qué es Python realmente?

- Un **lenguaje** es un recurso que hace posible la comunicación.
- Un **lenguaje de programación** es aquella estructura que, con una cierta base sintáctica y semántica, permite implementar algoritmos para ejecutarse en una computadora.
- Un **lenguaje de programación de alto nivel** contiene elementos del lenguaje humano y permite una comunicación simple con una computadora.
- Una **interfaz** es la conexión funcional entre dos sistemas que proporciona una comunicación de distintos niveles permitiendo el intercambio de información.

¿Qué es Python realmente?

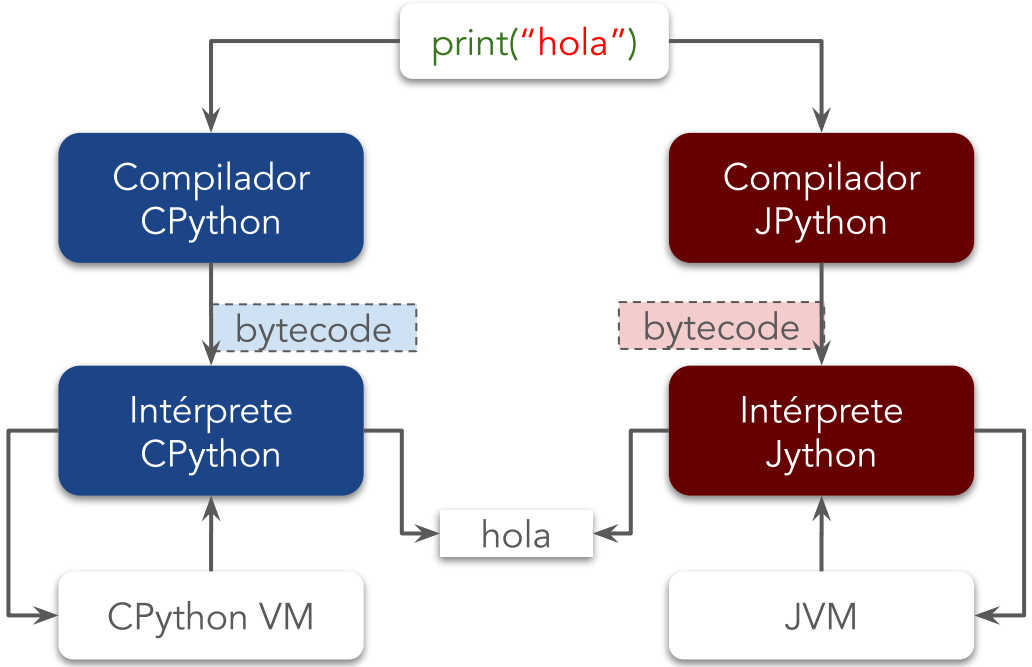
Python es una interfaz entre el ser humano y una computadora.

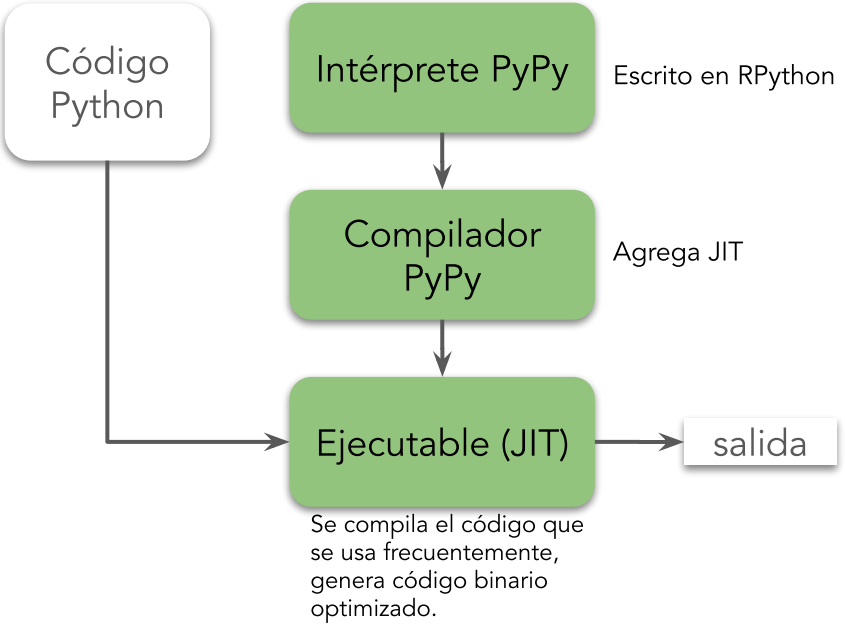
Implementación	Máquina Virtual	Compatibilidad
CPython	CPython VM	C
JPython	JVM	Java
IronPython	CLR	C#
Brython	Motor Javascript	Javascript
RubyPython	Ruby VM	Ruby
PyPy	Compilado	RPython

Especificación: The Python Language Reference

¿Python es interpretado o compilado?

Esto es una propiedad de la implementación, no de la interfaz (Python), entonces la respuesta es: "depende ..."





Fuente: aquí.

JIT: Just-In-Time

2. Pensando como *Pythonista* 1

Pensando como *Pythonista* 1

a. ¿Variables, objetos o etiquetas?
b. Tipado dinámico
c. Primer vistazo a la POO

Notebook:

- 01_Pensando_como_pythonista_1.ipynb

Resumen

Lenguaje C

```
int a = 1;
```


```
a = 2;
```


```
int b = a;
```

Python

```
a = 1
```


```
a = 2
```


```
b = a
```

"No solo no estoy aprendiendo nada, sino que estoy olvidando lo que ya sabía"

Milhouse Van Houten, The Simpsons.

Resumen

- En Python todo es un objeto.
- Existe el término *etiqueta* que es usado de manera "similar" al término *variable* de otros lenguajes.
- Existe el tipado dinámico:
  - el tipo de un objeto se evalúa durante la ejecución.
- Existe la generación espontánea:
  - los objetos se crean, se transforman y se destruyen.
- Los objetos viven en espacios de nombres.

Resumen

- Propiedades de un objeto
  - Una identidad única (id())
  - Un tipo (type()).
  - Un estado interno.
  - Uno o varios nombres (etiquetas).
  - Un comportamiento.

Huevo de pascua (digital)

1. Bello es mejor que feo.
2. Explícito es mejor que implícito.
3. Simple es mejor que complejo.
4. Complejo es mejor que complicado.
5. Plano es mejor que anidado.
6. Espaciado es mejor que denso.
7. La legibilidad es importante.
8. Los casos especiales no son lo suficientemente especiales como para romper las reglas.
9. Sin embargo la practicidad le gana a la pureza.
10. Los errores nunca deberían pasar silenciosamente.
11. A menos que se silencien explícitamente.
12. Frente a la ambigüedad, evitar la tentación de adivinar.
13. Debería haber una, y preferiblemente solo una, manera obvia de hacerlo.
14. A pesar de que esa manera no sea obvia a menos que seas Holandés.
15. Ahora es mejor que nunca.
16. A pesar de que nunca es muchas veces mejor que *ahora* mismo.
17. Si la implementación es difícil de explicar, es una mala idea.
18. Si la implementación es fácil de explicar, puede que sea una buena idea.
19. Los espacios de nombres son una gran idea, ¡tengamos más de esos!

El Zen de Python

```
import this
```

- pep20 by example
- A Brief Analysis of “The Zen of Python”
- Code Style