Pensando como Pythonista

La iniciación

Luis Miguel de la Cruz Salas
Mario Arturo Nieto Butrón

Instituto de Geofísica, UNAM

Febrero 2018

Pensando como Pythonista

Un lenguaje es un recurso que hace posible la comunicación.

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 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.

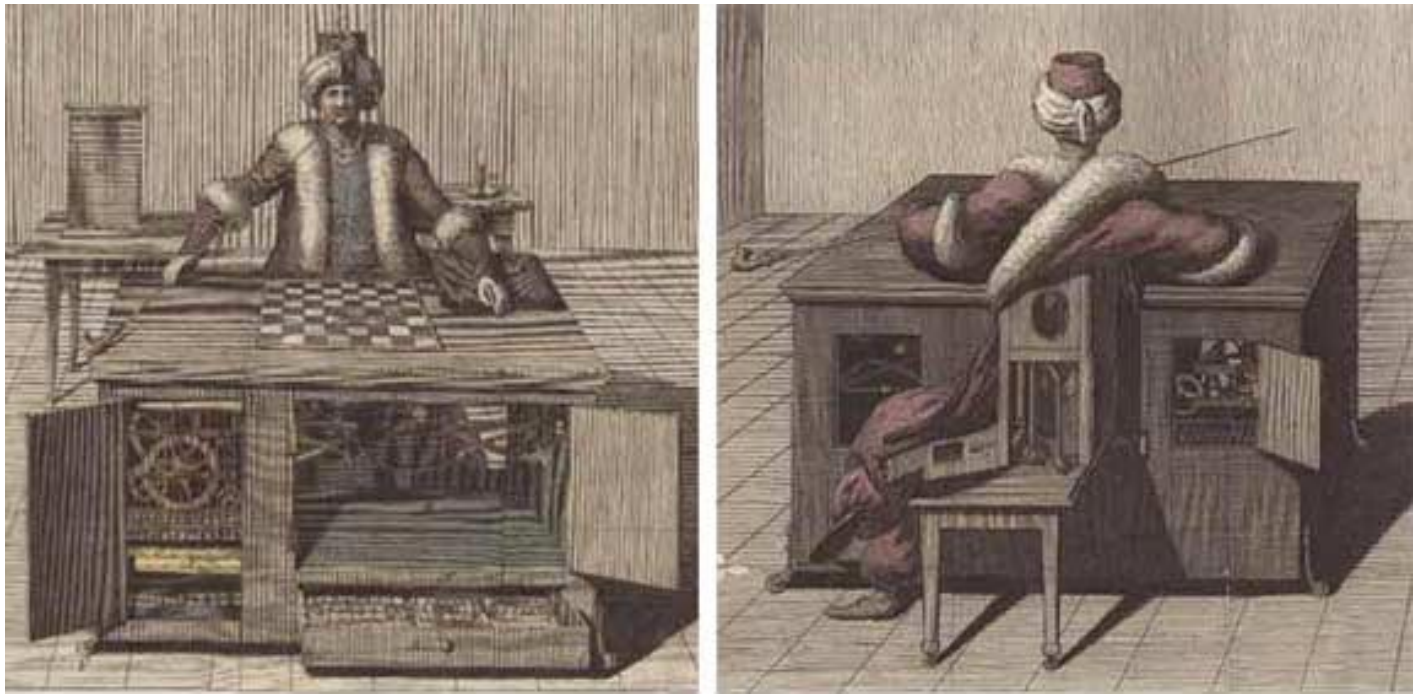
Un lenguaje de programación de alto nivel contiene elementos del lenguaje humano y permite una comunicación simple con una computadora.

Un lenguaje es un recurso que hace posible la comunicación.

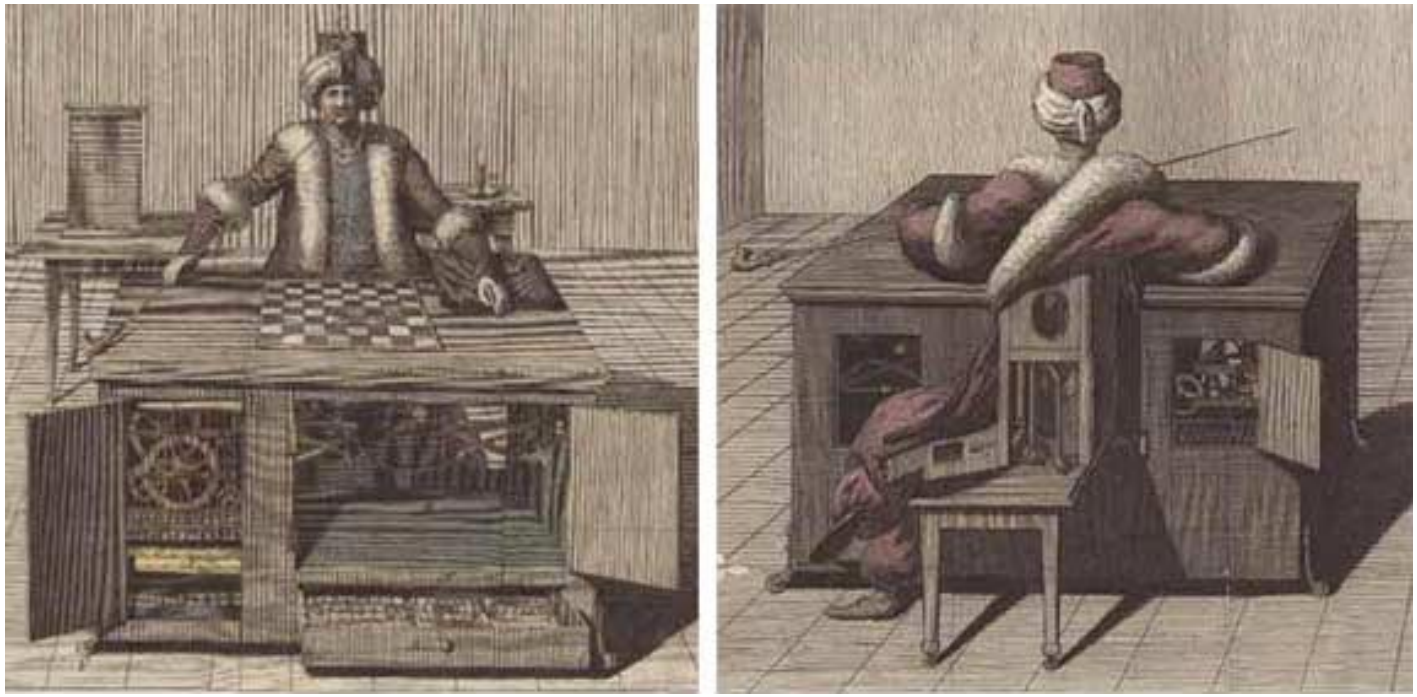
Un lenguaje de programación es aquella estructura que, con una cierta base sintáctica y semántica, permite implementar algoritmos para ejecutarse en una computadora.

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.

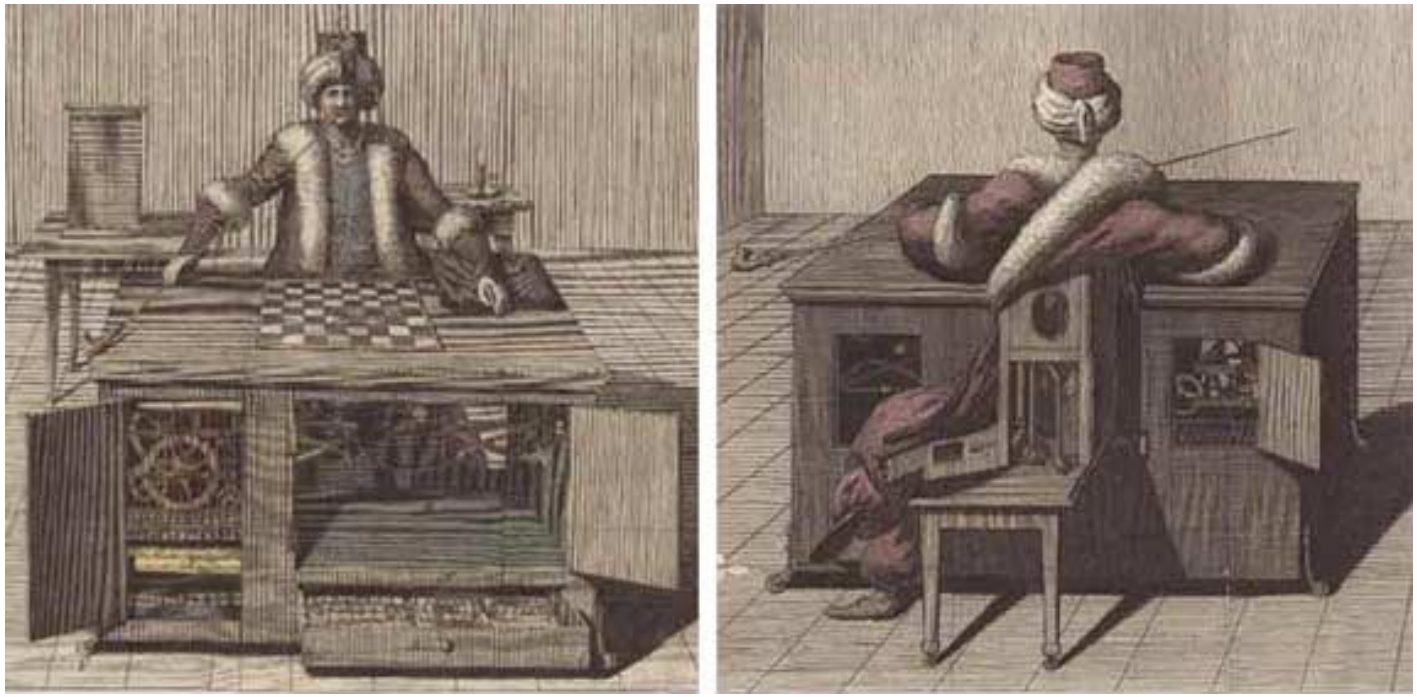


El Turco, (1769). Wolfgang von Kempelen



El Turco, (1769). Wolfgang von Kempelen

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



El Turco, (1769). Wolfgang von Kempelen

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

Observación: existen muchas implementaciones de esta interfaz.

Pensando como Pythonista

El Zen de Python descrito en el PEP[1] 20

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
...
If the implementation is hard to explain, it's a bad idea.
...
```

[1] Python Enhancement Proposals (PEPs)

El Zen de Python descrito en el PEP[1] 20

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
...
If the implementation is hard to explain, it's a bad idea.
...
```

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

[1] Python Enhancement Proposals (PEPs)

El Zen de Python descrito en el PEP[1] 20

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
...
If the implementation is hard to explain, it's a bad idea.
...
```

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

Ejemplo: intercambio de valores

```
>>> b, a = a, b
```

[1] Python Enhancement Proposals (PEPs)

Pensando como Pythonista

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

Milhouse Van Houten, The Simpsons.

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

Milhouse Van Houten, The Simpsons.

Lenguaje C

```
int a = 1;
```


```
a = 2;
```


```
int b = a;
```

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

Milhouse Van Houten, The Simpsons.

Lenguaje C

```
int a = 1;
```


```
a = 2;
```

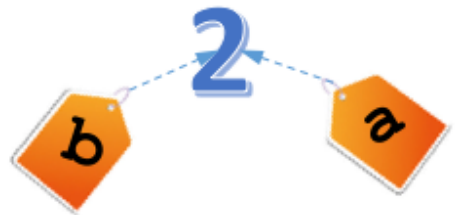

```
int b = a;
```

Python

```
a = 1
```


```
a = 2
```


```
b = a
```



Propiedades de un objeto

- Una identidad única (**id()**)
- Un tipo (**type()**).
- Un estado interno.
- Uno o varios nombres.
- Un comportamiento.

Propiedades de un objeto

- Una identidad única (**id()**)
- Un tipo (**type()**).
- Un estado interno.
- Uno o varios nombres.
- Un comportamiento.

```
>>> a = 1
>>> id(a)
4492568576
>>> type(a)
<class 'int'>
```

```
>>> b = a
>>> id(b)
4492568576
>>> type(b)
<class 'int'>
```

En Python:

- Todo es un objeto.
- El término *variable* no existe.
- Existe el tipado dinámico: el tipo de un objeto se evalúa durante la ejecución.
- Existe la generación espontánea: los objetos se crean y se destruyen.
- Los objetos viven en espacios de nombres.

En Python:

- Todo es un objeto.
- El término *variable* no existe.
- Existe el tipado dinámico: el tipo de un objeto se evalúa durante la ejecución.
- Existe la generación espontánea: los objetos se crean y se destruyen.
- Los objetos viven en espacios de nombres.

Observación:

Suele decirse que Python es un lenguaje interpretado, de tipado dinámico, genérico, pero poco eficiente. Pero en realidad estas cuatro características dependen de la implementación de cada interfaz de Python, por lo que NO SON CIERTAS en general!.

Pensando como Pythonista

Pensando como Pythonista

Tamales for dummies

Tamales for dummies

Programación

1. Ingredientes $\sim$ Leng. de prog.

Tamales for dummies

Programación

1. Ingredientes $\sim$ Leng. de prog.
2. Receta $\sim$ Programa.

Tamales for dummies

Programación

1. Ingredientes $\sim$ Leng. de prog.
2. Receta $\sim$ Programa.

Tamales for dummies

Programación

1. Ingredientes $\sim$ Leng. de prog.
2. Receta $\sim$ Programa.

Compilación

1. Preparación $\sim$ Ligado.

Tamales for dummies

Programación

1. Ingredientes $\sim$ Leng. de prog.
2. Receta $\sim$ Programa.

Compilación

1. Preparación $\sim$ Ligado.
2. Cocción $\sim$ Compilado $\Rightarrow$ *machine code* eficiente, no portable.

Tamales for dummies

Programación

1. Ingredientes $\sim$ Leng. de prog.
2. Receta $\sim$ Programa.

Compilación

1. Preparación $\sim$ Ligado.
2. Cocción $\sim$ Compilado $\Rightarrow$ *machine code* eficiente, no portable.
3. Consumo $\sim$ Ejecución.

Tamales for dummies

Programación

1. Ingredientes $\sim$ Leng. de prog.
2. Receta $\sim$ Programa.

Compilación

1. Preparación $\sim$ Ligado.
2. Cocción $\sim$ Compilado $\Rightarrow$ *machine code* eficiente, no portable.
3. Consumo $\sim$ Ejecución.

Tamales for dummies

Programación

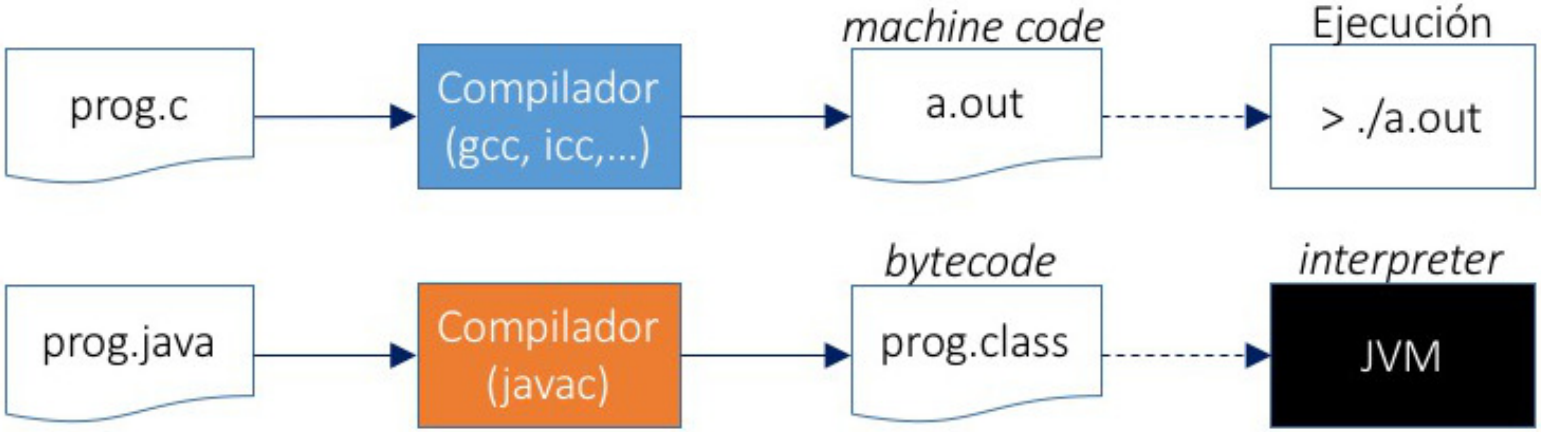
1. Ingredientes $\sim$ Leng. de prog.
2. Receta $\sim$ Programa.

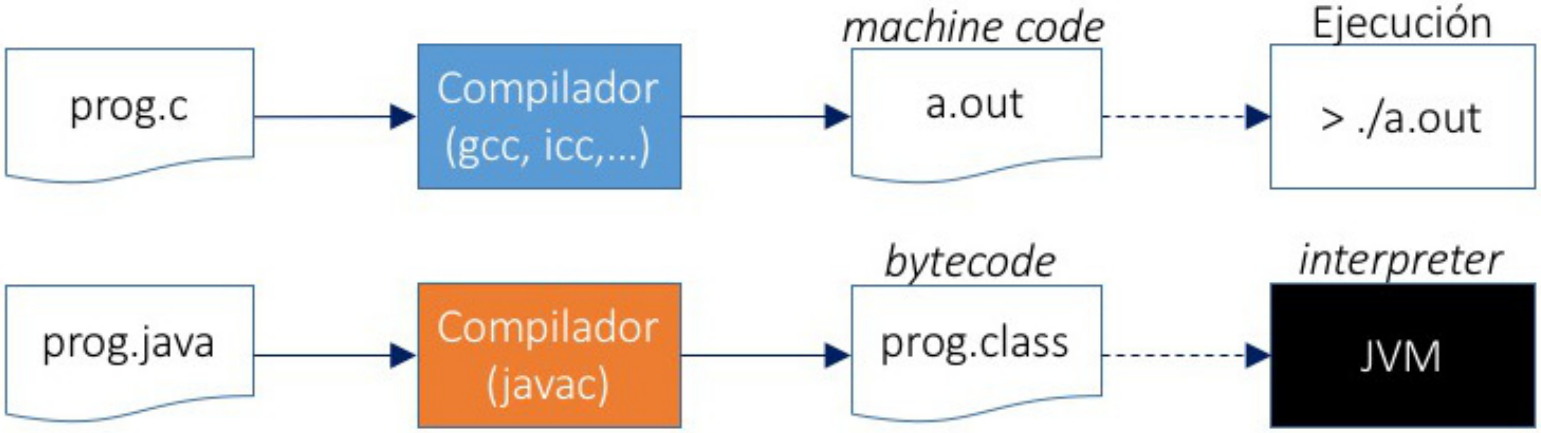
Compilación

1. Preparación $\sim$ Ligado.
2. Cocción $\sim$ Compilado $\Rightarrow$ *machine code* eficiente, no portable.
3. Consumo $\sim$ Ejecución.

Interpretación

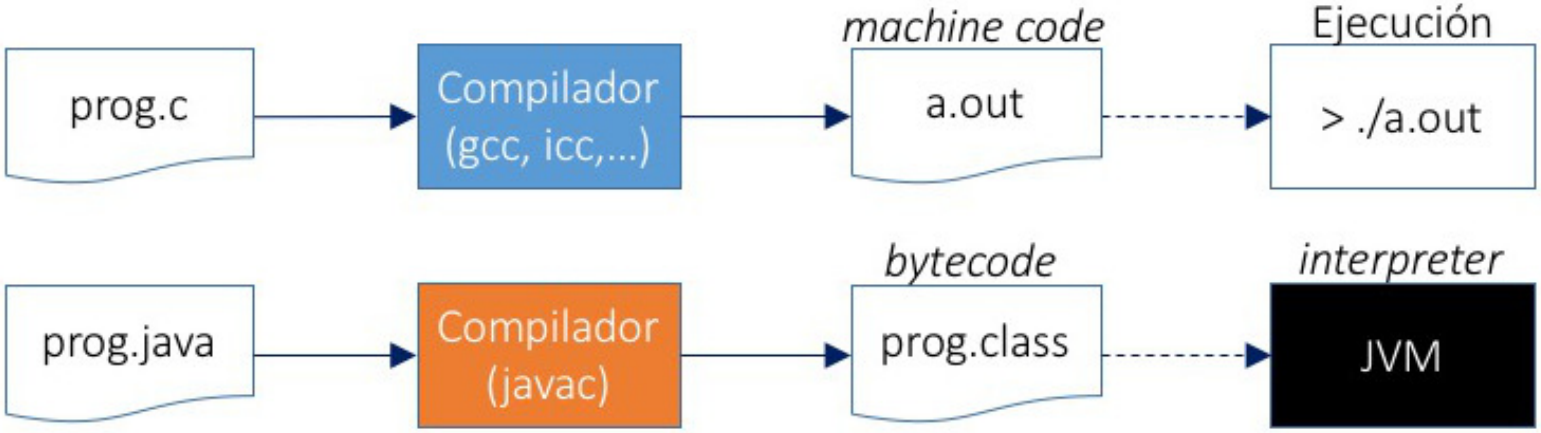
1. El intérprete convierte el código fuente en *bytecode* (.pyc), el cual es portable, pero poco eficiente.
2. Una máquina virtual se encarga del ligado, compilado y ejecución del *bytecode*.


machine code
Ejecución
prog.c
Compilador
(gcc, icc,...)
a.out
> ./a.out
bytecode
interpreter
prog.java
Compilador
(javac)
prog.class
JVM



Implementación	Máquina Virtual	Lenguaje Compatible
CPython	CPython VM	C
Jython	JVM	Java
IronPython	CLR	C#
Brython	Motor Javascript	JavaScript
RubyPython	Ruby VM	Ruby

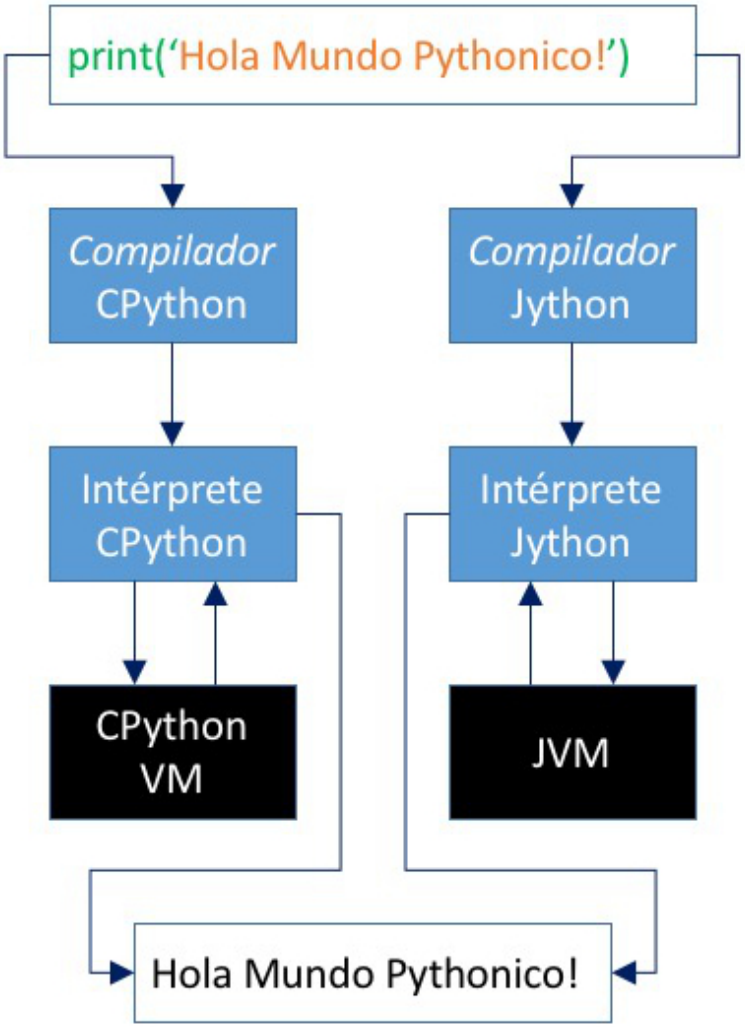
Especificación: The Python Language Reference

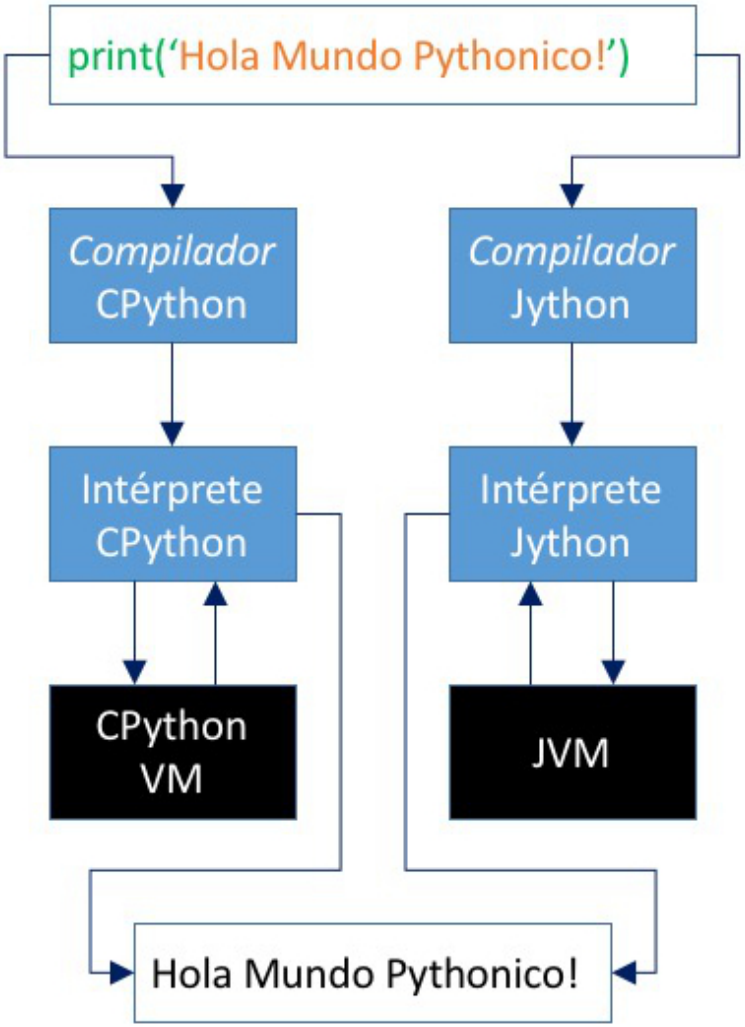


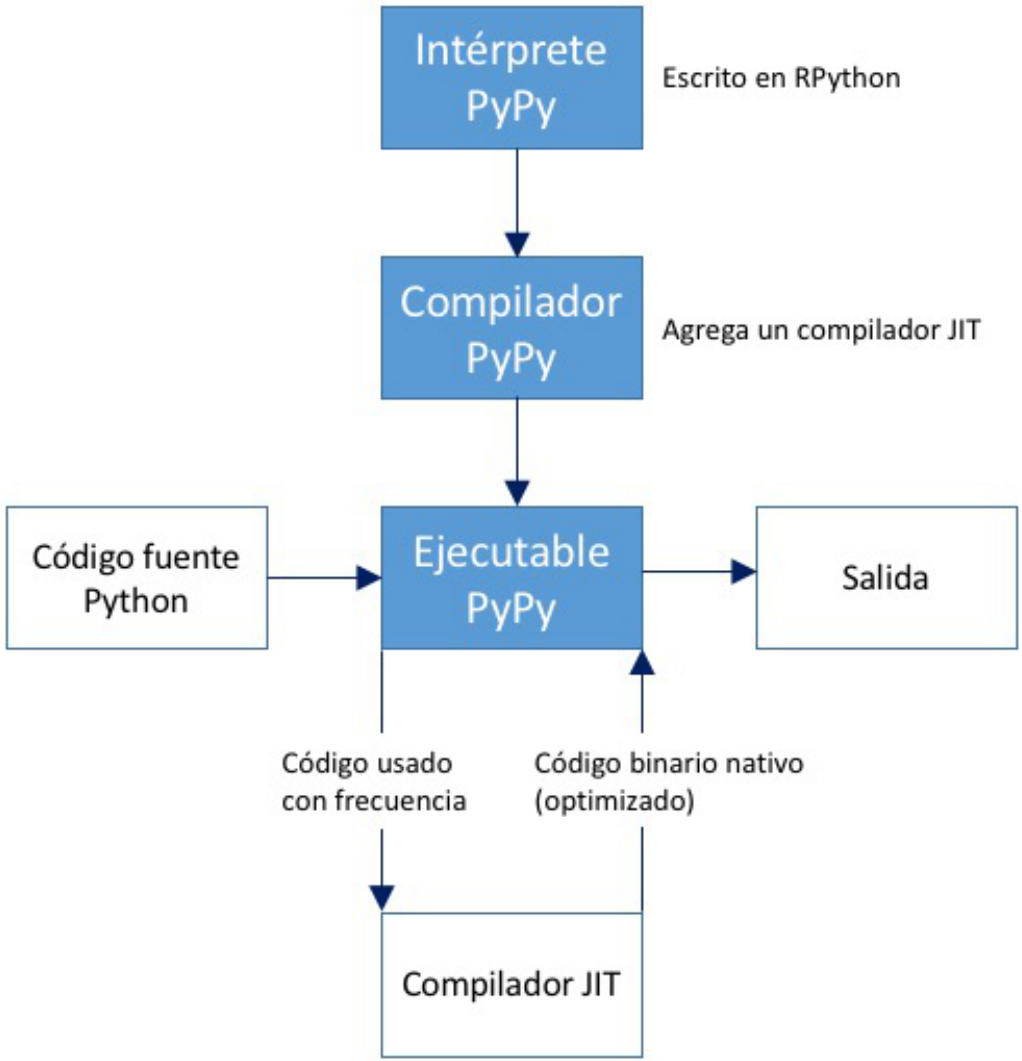
Implementación	Máquina Virtual	Lenguaje Compatible
CPython	CPython VM	C
Jython	JVM	Java
IronPython	CLR	C#
Brython	Motor Javascript	JavaScript
RubyPython	Ruby VM	Ruby

Especificación: The Python Language Reference

- PyPy : implementación escrita en Python (RPython: subconjunto de Python que puede compilarse estáticamente)


print('Hola Mundo Pythonico!')
Compilador CPython
Compilador Jython
Intérprete CPython
Intérprete Jython
CPython VM
JVM
Hola Mundo Pythonico!


print('Hola Mundo Pythonico!')
Compilador CPython
Compilador Jython
Intérprete CPython
Intérprete Jython
CPython VM
JVM
Hola Mundo Pythonico!


Intérprete PyPy
Escrito en RPython
Compilador PyPy
Agrega un compilador JIT
Código fuente Python
Ejecutable PyPy
Salida
Código usado con frecuencia
Código binario nativo (optimizado)
Compilador JIT

Pensando como Pythonista


Python Developer's Guide
https://devguide.python.org/


Python Enhancement Proposals (PEPs)
https://www.python.org/dev/peps/


PEP 20 – The Zen of Python
https://www.python.org/dev/peps/pep-0020/


PEP 20 bye example
http://artifex.org/~hblanks/talks/2011/pep20_by_example.html


A brief analysis of The Zen of Python.
https://medium.com/@Pythonidaer/a-brief-analysis-of-the-zen-of-python-2bfd3b76edbf


The Hitchhiker's Guide to Python: Code Style.
http://docs.python-guide.org/en/latest/writing/style/


CPython
https://github.com/python/cpython


Jython
http://www.jython.org/

IronPython
http://ironpython.net/

Brython
https://brython.info/

RubyPython
http://www.rubydoc.info/gems/rubypython/0.6.3/RubyPython

The Python Language Reference
https://docs.python.org/2/reference/index.html

PyPy
https://pypy.org/

RPython
http://rpython.readthedocs.io/en/latest/getting-started.html