

# Python Básico

Introducción e Historia



Marzo 2019



- Nombre del curso: Desarrollo Web con Python
- Duración: 45 horas
- Horario: 9:00 - 2:00 (descanso 30 min)
- Fecha de inicio: 02 de Enero
- Fecha de término: xx de Enero



- **Chávez Delgado Jorge Luis (Titular)**
  - Generación 33
  - [jorgechavez.proteco@gmail.com](mailto:jorgechavez.proteco@gmail.com)
- **Martínez Cruz Anallely (Adjunto)**
  - Generación 37
  - [anallely.0710@gmail.com](mailto:anallely.0710@gmail.com)



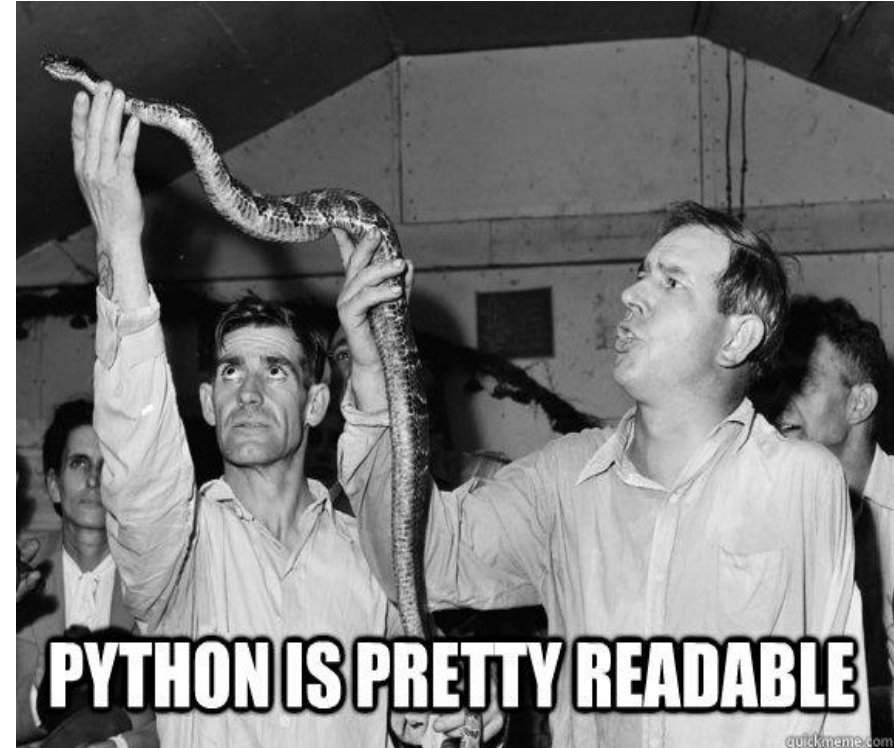
Por favor menciona:

- ❖ Nombre
- ❖ Escuela o trabajo actual
- ❖ Algún otro lenguaje(s) que manejes
- ❖ ¿Por qué quieres aprender desarrollo con Python?
- ❖ ¿Qué esperas de este curso?

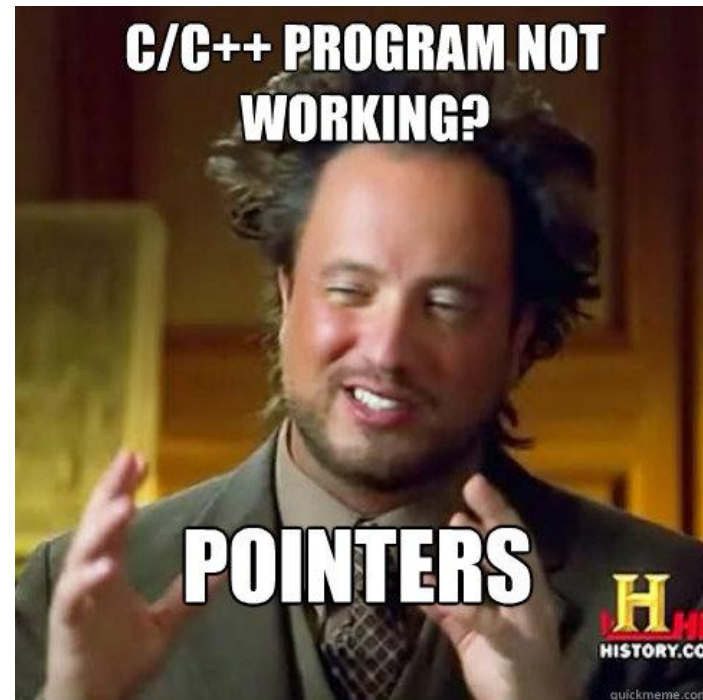


# Introducción

Es un lenguaje cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.



Python es un poderoso lenguaje de programación interpretado y fácil de aprender.





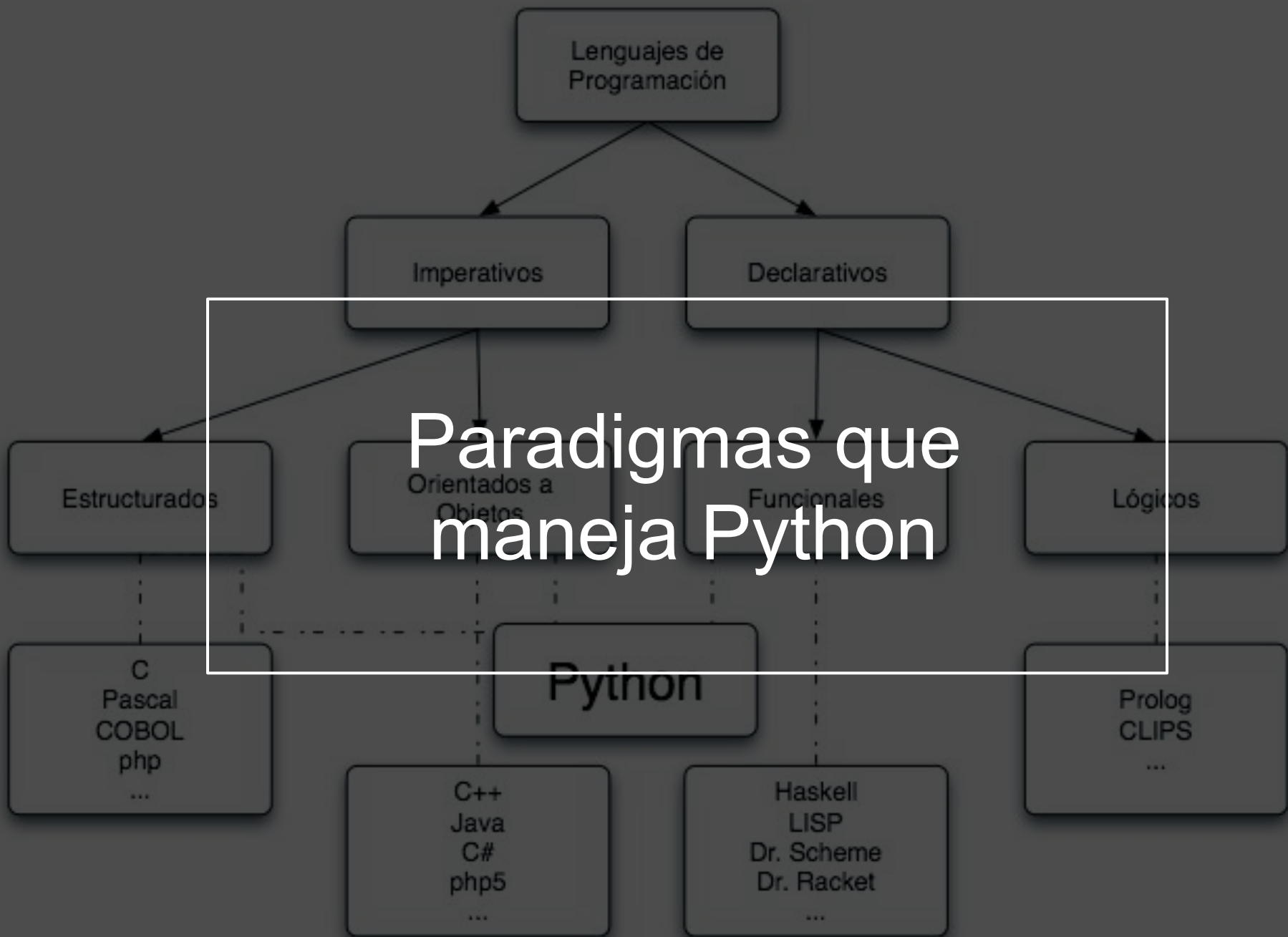
- Es un lenguaje multiparadigma, ya que soporta orientación objetos, programación estructurada y en menor medida, programación funcional.
- Un paradigma representa un enfoque particular o filosofía para diseñar soluciones.

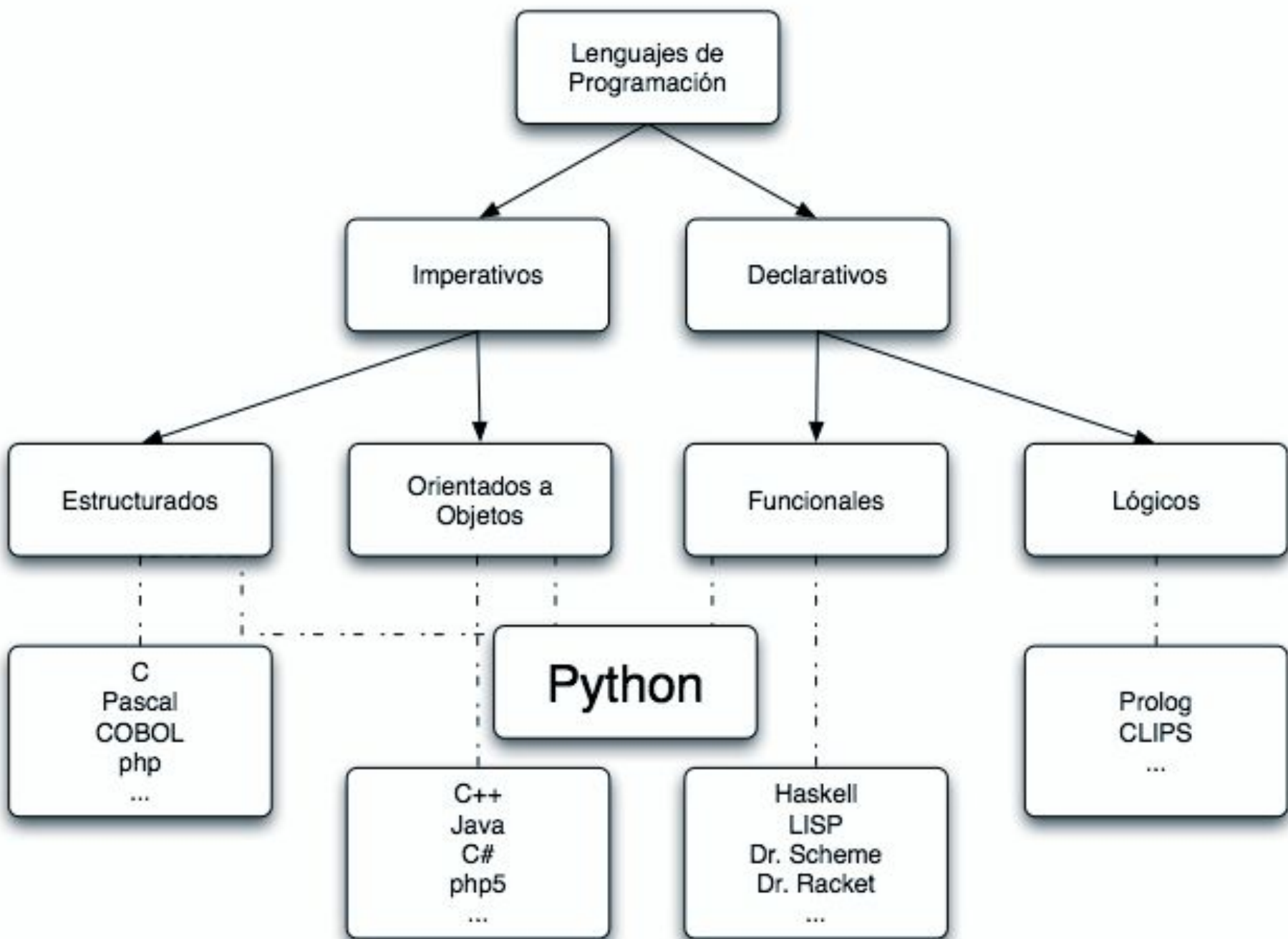


- Los paradigmas son diferentes en la forma de abstraer los elementos involucrados en un problema.
- Generalmente lo usamos porque nos gusta o se nos facilita entender problemas esa manera.









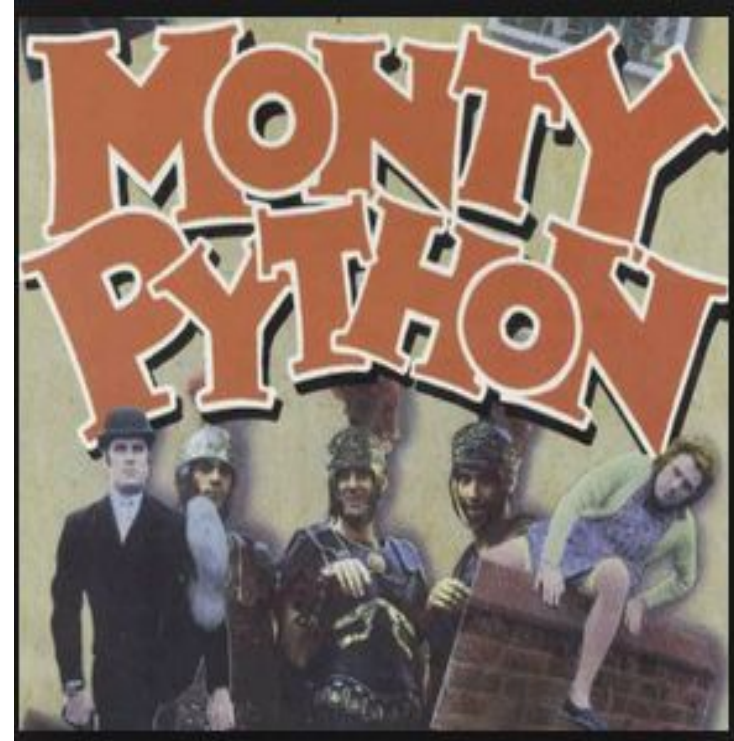
# Creador de python

Python fue creado en 1991 por Guido van Rossum en el Centro para las Matemáticas y la Informática, como un sucesor del lenguaje de programación ABC. Guido nombró al lenguaje por su afición al programa Monty Python.



# Origen del nombre

Monty Python's Flying Circus es una serie de televisión británica, basada en sketches breves que en muchas ocasiones incluían una importante carga de crítica social, y la mayoría de las obras rozaban el absurdo total.



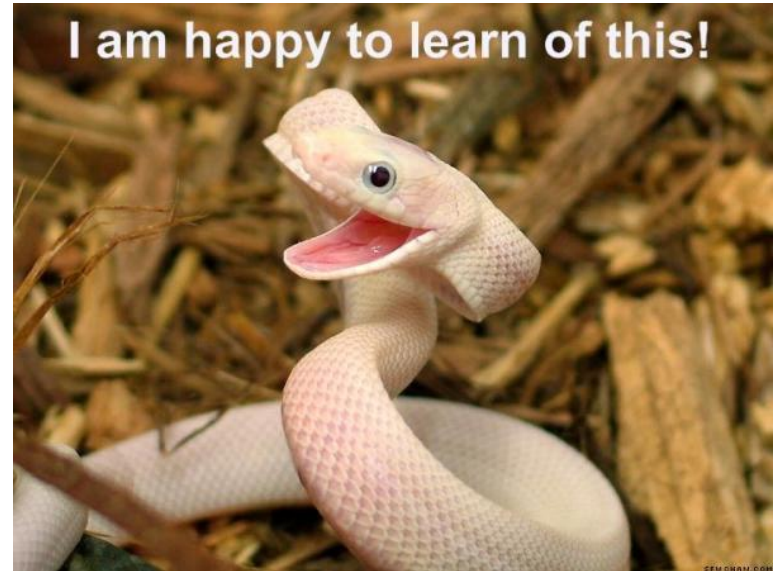
# **BDFL**

Benevolent Dictator for Life (BDFL) es un título informal que se otorga a ciertos individuos de la comunidad de desarrolladores de software de código abierto que tienen la tarea de asignar las directrices generales y, en ciertas situaciones, las decisiones finales dentro del ámbito de un proyecto.



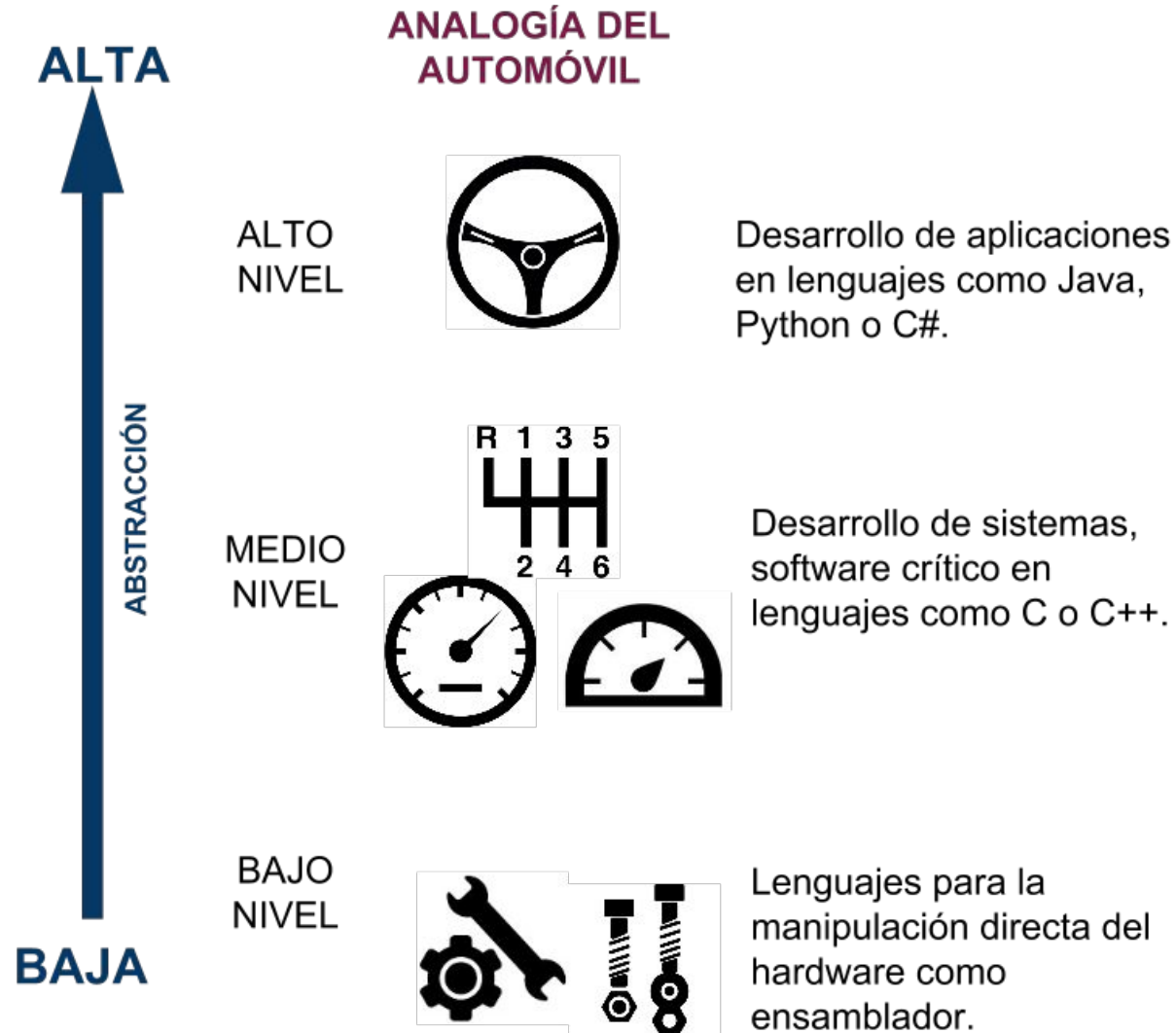
# Características

Ideal para scripts multiplataforma por su naturaleza interpretada, gracias a su elegante sintaxis, tipado fuerte y dinámico y facilidad de aprendizaje fue el lenguaje más usado del 2015 y 2016.





# Lenguaje de alto nivel

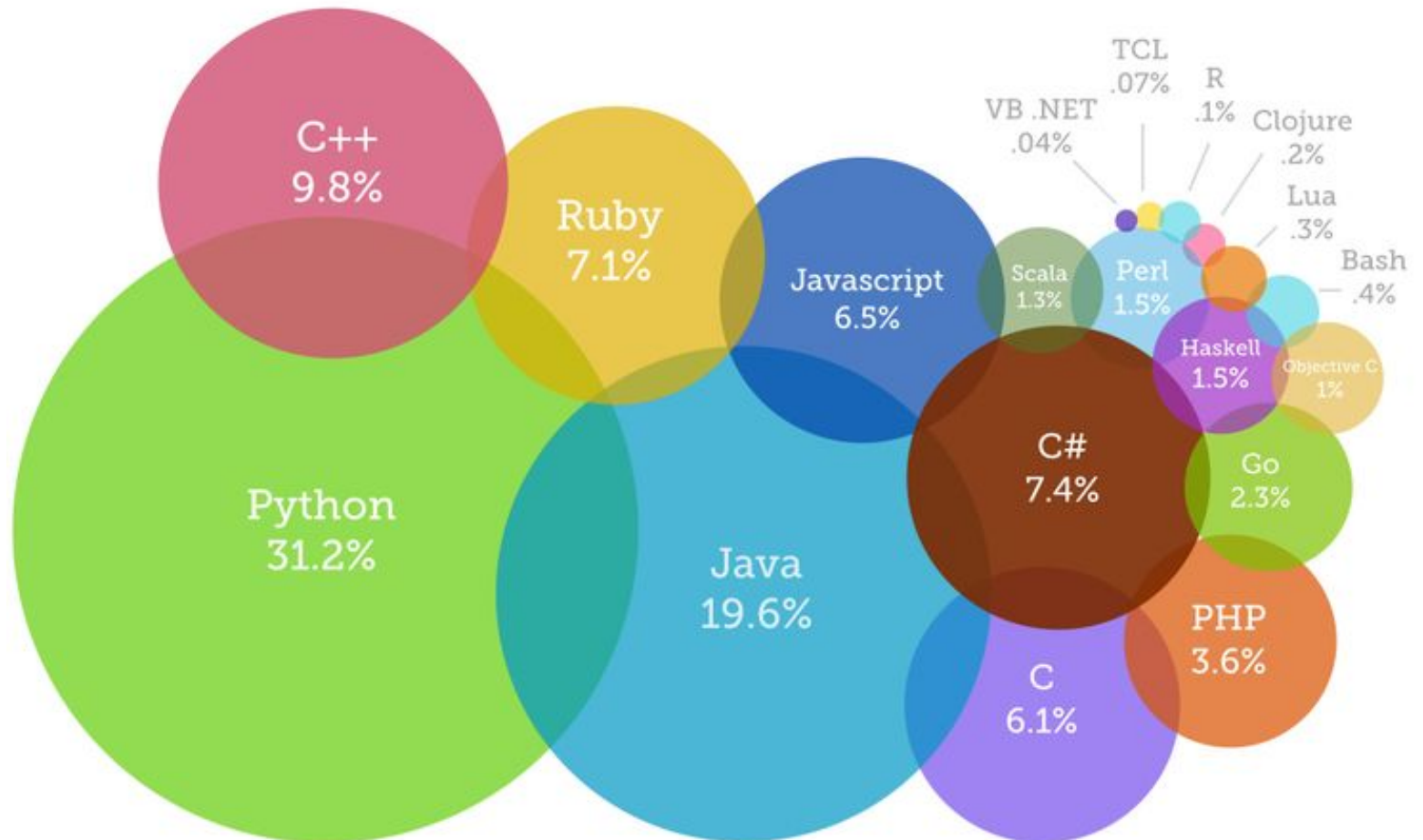




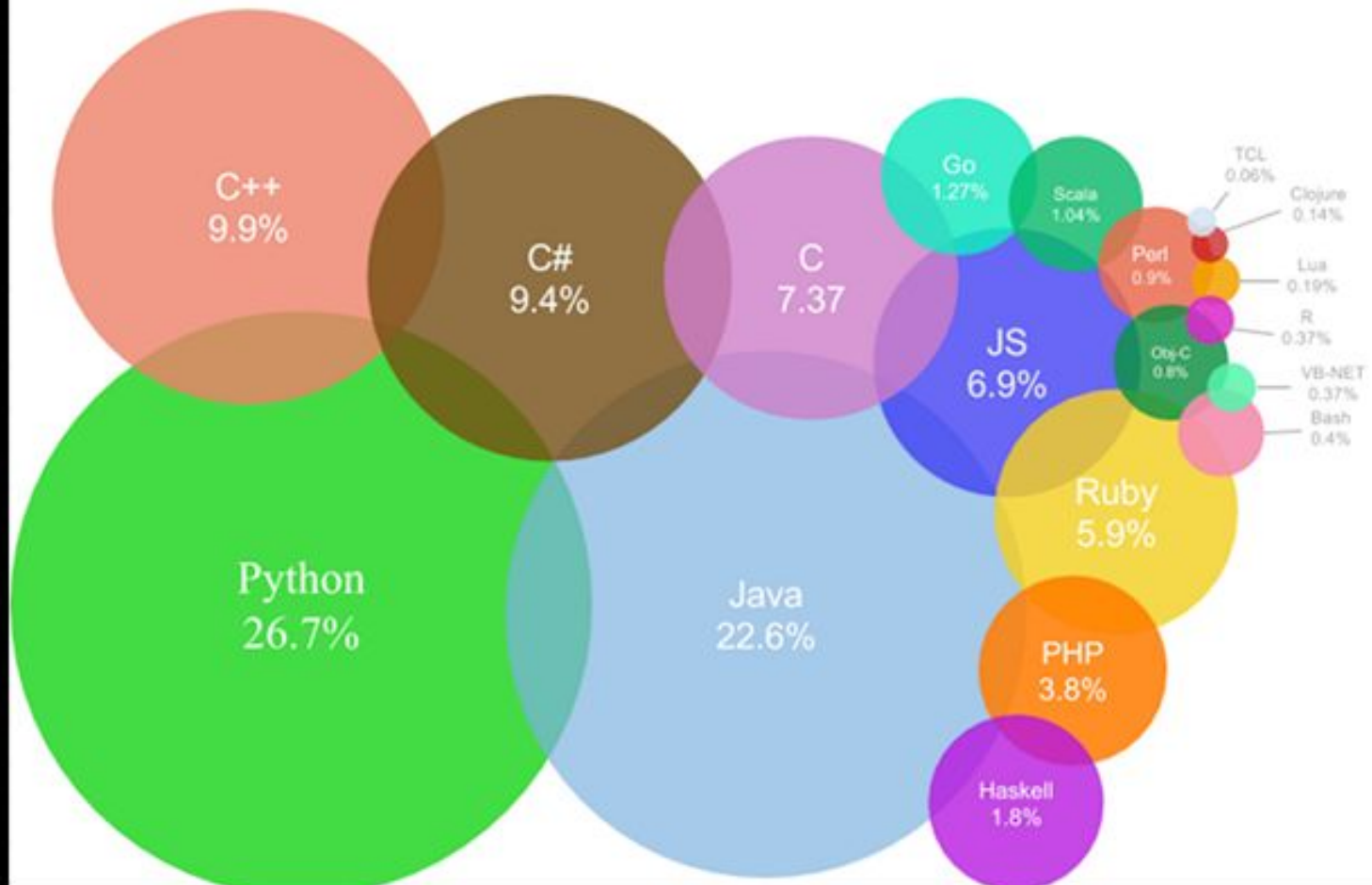
# Most Popular Coding Languages of 2015



## Most Popular Coding Languages of 2015



## Most Popular Coding Languages of 2016



# Ventajas

- Fácil de usar
  - Los tipos se asocian a objetos no a variables
  - Opera en un muy alto nivel de abstracción
  - Las reglas sintácticas son muy sencillas

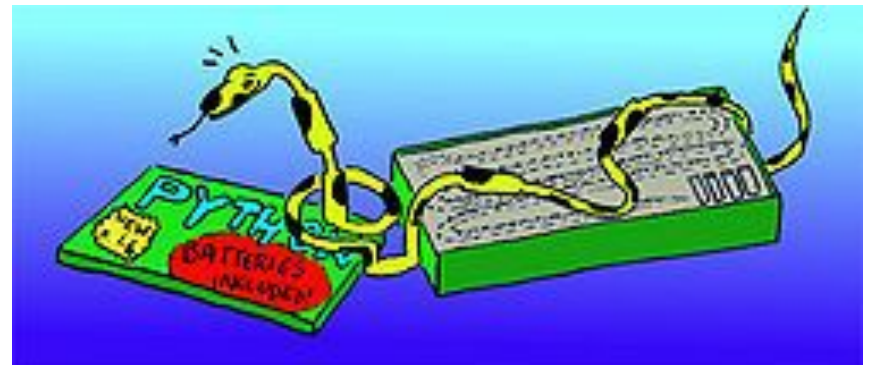
Python es muy conveniente para el desarrollo rápido de aplicaciones.

# Ventajas

- Expresividad
  - Python es un lenguaje muy expresivo
    - Una línea en Python puede hacer más que una línea en cualquier otro lenguaje.
    - Menor líneas de código -> Menor tiempo en escribir y ejecutar.
    - Menor líneas de código -> Mayor facilidad para mantener y depurar los programas.

# Ventajas

- Legibilidad
  - Facilidad de lectura
- Filosofía de Baterías incluidas
  - La librería estándar de Python es muy amplia



# Ventajas

- Multiplataforma
  - Dado que es interpretado, el mismo código puede ejecutarse en cualquier plataforma que tenga instalado un intérprete de Python.
- Open Source
  - Está desarrollado con el modelo open source y está disponible libremente bajo una licencia pública general de GNU .



# Características

- Tipado Dinámico
  - La característica de tipado dinámico se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, Python para todos sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.

# Dinámico VS Estático

## Static vs Dynamic Typing

Java

Static typing:

```
String name;
```

```
name = "John";
```

```
name = 34;
```

Variables have types

Values have types

Variables cannot change type

JavaScript

Dynamic typing:

```
var name;
```

```
name = "John";
```

```
name = 34;
```

Variables have no types

Values have types

Variables change type dynamically

# Características

- Tipado Fuerte
  - No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, si tenemos una variable que contiene un texto (variable de tipo cadena o string) no podremos tratarla como un número (sumar la cadena "9" y el número 8). En otros lenguajes el tipo de la variable cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores.

# Fuerte VS Débil

## Strong vs Weak Typing

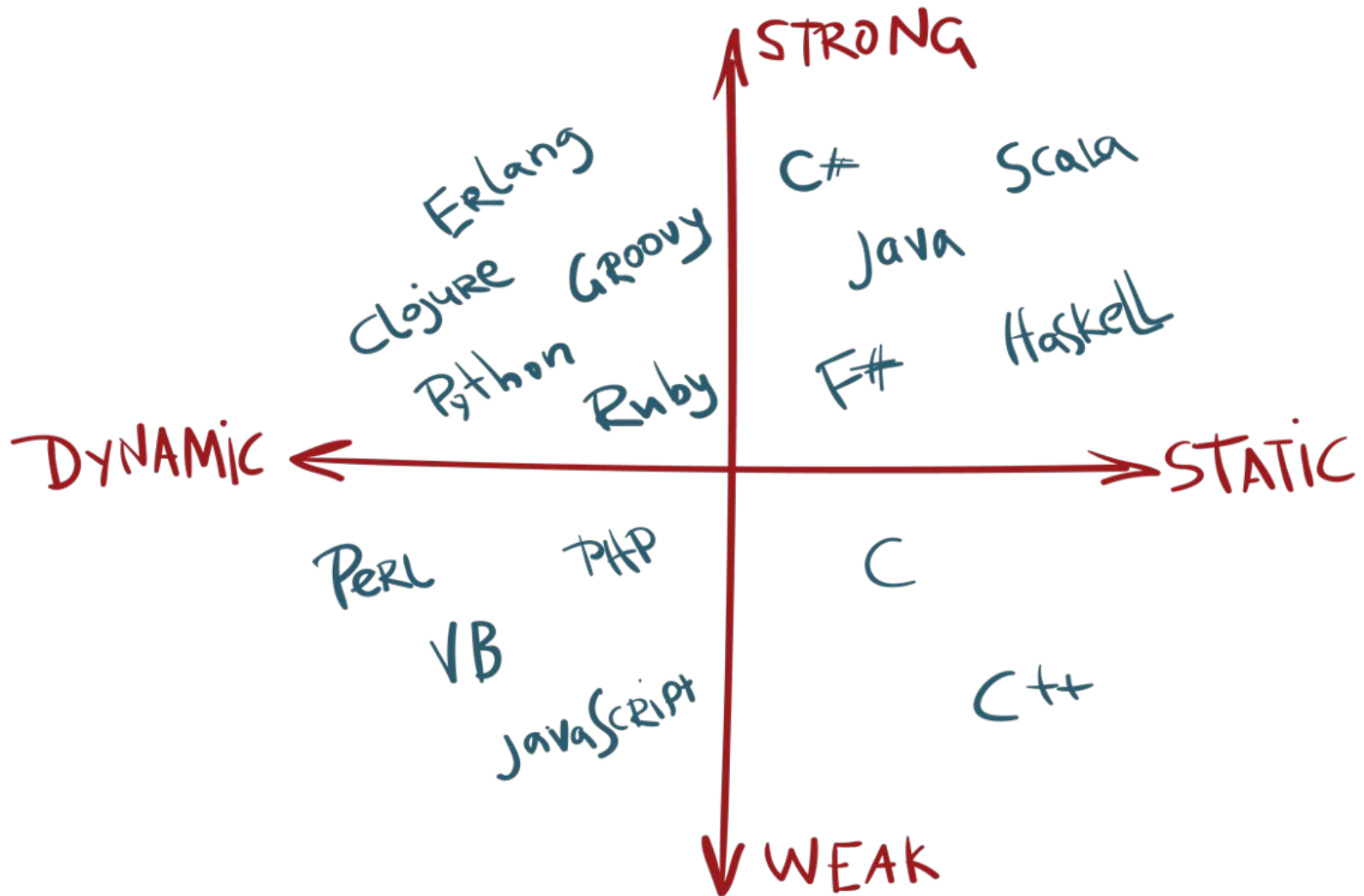
Java

<code>int intVar = 4;</code>	Pretty strong typing
<code>"" + intVar;</code>	Implicit type conversion
<code>intVar + 0.0;</code>	Implicit, widening type conv.
<code>intVar + (int)0.0;</code>	Explicit type conversion
<del><code>intVar + null;</code></del>	Not allowed

JavaScript

<code>var dynVar = 4;</code>	Very weak typing
<code>dynVar + "";</code>	Implicit type conversion
<code>dynVar + null;</code>	Implicit type conversion
<code>dynVar + [];</code>	Implicit type conversion
<code>dynVar + [0];</code>	Implicit type conversion

# Clasificación



# ¿Cómo funciona un compilador?

```
19 /*Funciones*/
20 void error(void){
21     perror("error: insuficiente espacio de memoria");
22     exit(1);
23 }
24 nodo *NuevoNodo(){
25     nodo *q = (nodo *)malloc(sizeof(nodo));
26     if(!q) error();
27     return (q);
28 }
29 void buscar(int, nodo **);
30 void visualizar_arbol(nodo *,int);
31 void borrar_arbol(nodo *a);
32 void borrar(int, nodo **);
33 void borrar_nodo(nodo **, nodo **);
34
35 int main(){
36     nodo *raiz = NULL;
37     int k;
38
39     printf("Introducir claves. Finalizar con eof\n\n");
40     printf("clave: ");
41
42     while (scanf ("%d", &k) != EOF){
43         buscar(k, &raiz);
44         printf("clave: ");
```



```
E1 FF 34 00 00 00 F1
16 A1 A1 00 00
10 3E 01 EA F1 00
```

En un compilador el código completo se analiza y se reportan todos los errores y advertencias. El compilador genera el código máquina mientras que el enlazador se encarga de poner las funciones externas para generar un ejecutable específico para ese sistema.

# ¿Cómo funciona un intérprete?

```
class Animal():
    def __init__(self, especie, tipo, nombre):
        self.especie=especie
        self._tipo=tipo
        self._nombre=nombre

    def setEspecie(self, especie):
        self.especie=especie

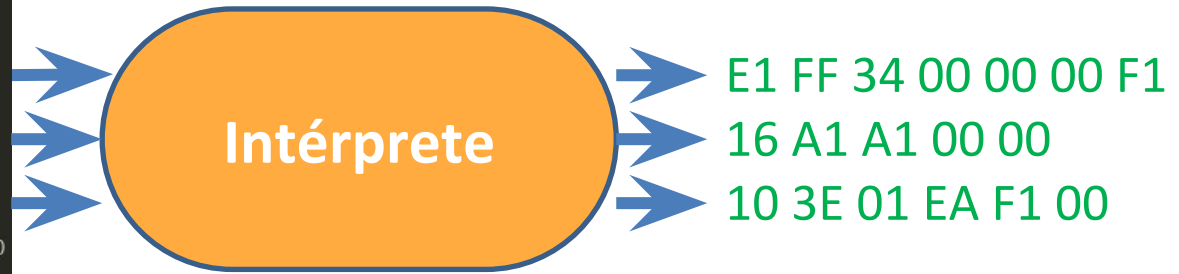
    def getEspecie(self):
        return self.especie

    def setTipo(self, tipo):
        self.tipo=tipo

    def getTipo(self):
        return self.tipo

    def reproducir(self):
        print(self._nombre, "se esta reproduciendo")

    def moverse(self):
        if self._tipo=="acuatico":
            print("Yo nado")
        elif self._tipo=="terreste":
            print("Yo camino o me arrastro")
```



En un intérprete código se analiza y ejecuta línea por línea, por lo regular compila a un lenguaje intermedio (bytecode) y después lo traduce al código máquina. Si alguna línea falla no interrumpe la operación y sigue ejecutando las demás.



# Intérprete de Python

```
class Animal():
    def __init__(self, especie, tipo, nombre):
        self.especie=especie
        self._tipo=tipo
        self.__nombre=nombre

    def setEspecie(self, especie):
        self.especie=especie

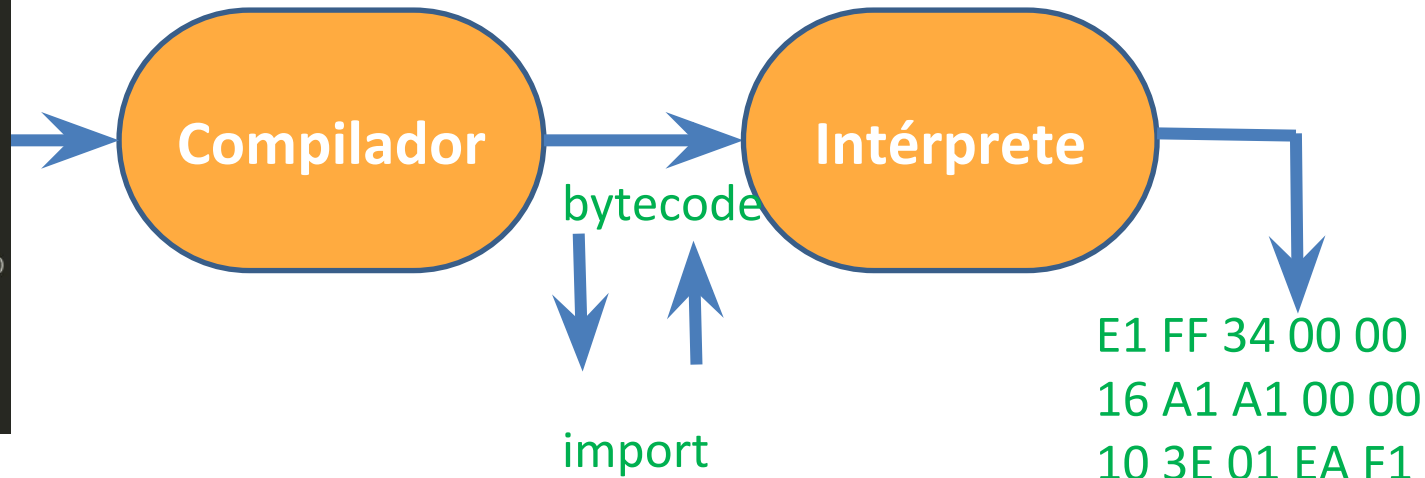
    def getEspecie(self):
        return self.especie

    def setTipo(self, tipo):
        self.tipo=tipo

    def getTipo(self):
        return self.tipo

    def reproducir(self):
        print(self.__nombre, "se esta reproduciendo")

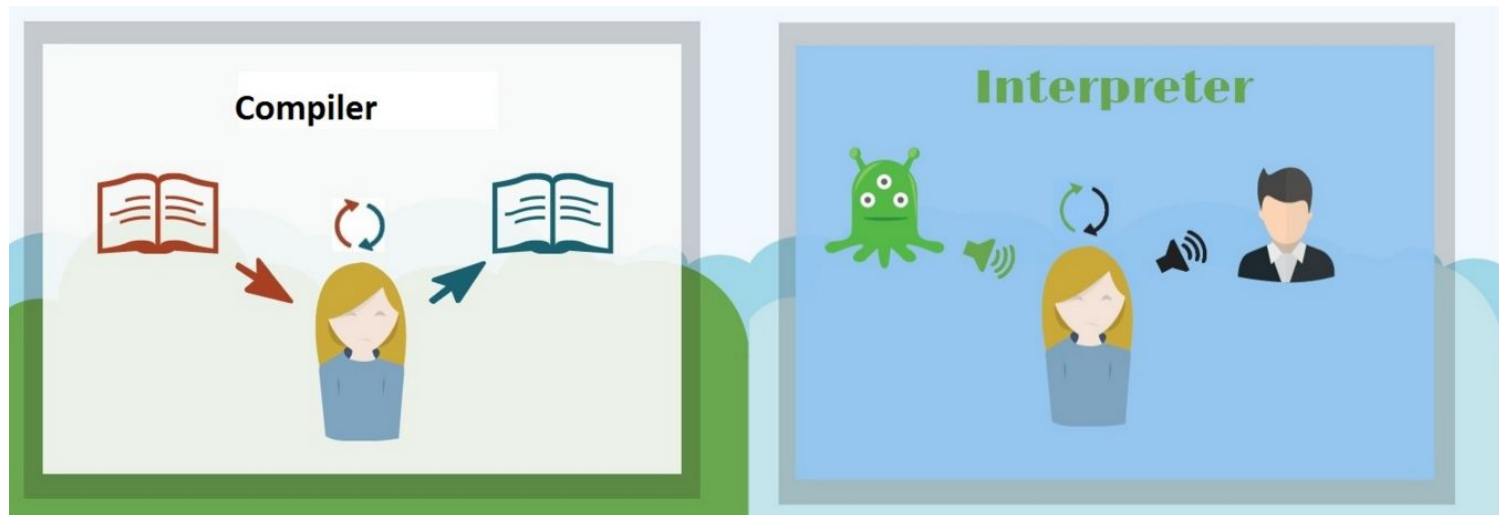
    def moverse(self):
        if self._tipo=="acuatico":
            print("Yo nado")
        elif self._tipo=="terreste":
            print("Yo camino o me arrastro")
```



Un programa en python se pasa a bytecode primero, si hay alguna referencia que no esté definida el compilador lo indicará, luego la máquina virtual de ese sistema ejecuta el bytecode con las dependencias ya resueltas directamente en código máquina

# En Resumen

- Un compilador traduce el código completo y si no está bien escrito no puede completar la traducción, en cambio un intérprete sólo traduce lo que entiende y se va de corrido



# Desventajas

- No es el lenguaje más rápido (por su naturaleza interpretada)
- No posee las librerías más extensas (no es Matlab)
- No tiene revisión de tipos (desperdicia memoria)

# ¿Quién usa Python?



# ¿Quién usa Python?

- <http://www.python.org/about/apps>
- <http://www.python.org/about/success>

# Visualización de código

## VISUALIZE [Python](#), [Java](#), [JavaScript](#), [TypeScript](#), [Ruby](#), [C](#), and [C++](#) programs

**Python Tutor**, created by [Philip Guo](#), helps people overcome a fundamental barrier to learning programming: understanding what happens as the computer executes each line of a program's source code.

Python 2.7

```

1 def listSum(numbers):
2     if not numbers:
3         return 0
4     else:
5         (f, rest) = numbers
6         return f + listSum(rest)
7
8 myList = (1, (2, (3, None)))
9 total = listSum(myList)

```

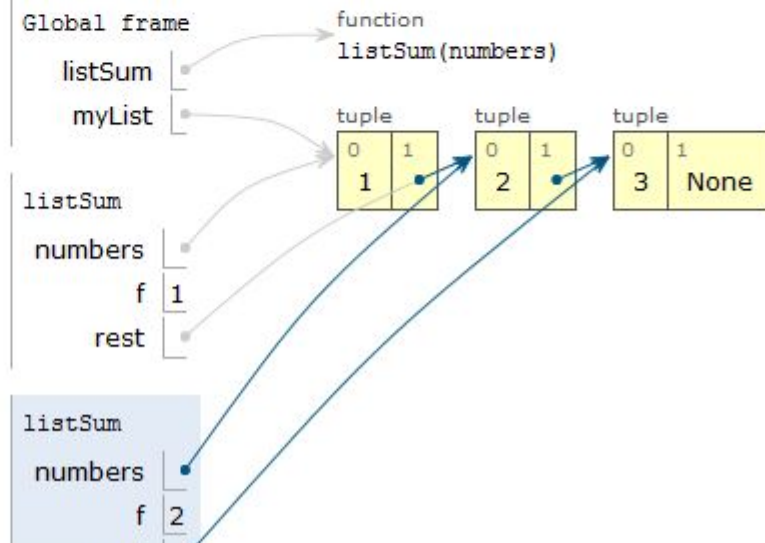
[Edit code](#)

→ line that has just executed

➡ next line to execute

## Frames

## Objects



<http://pythontutor.com/>

# Enlaces de ayuda

- Ejercicios y un poco de teoría  
<https://learnpythonthehardway.org>
- Ayuda específica (pero en inglés)  
<http://stackoverflow.com/>
- Documentación oficial (un poco complicada)  
<https://docs.python.org/3/tutorial/index.html>