

# Python



## Primeros Pasos

### Parte I

Algoritmos y Programación I - FIUBA  
Lic. Gustavo Bianchi

1

## Qué veremos?

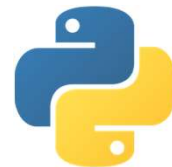
- El origen del lenguaje
- Sus características fundamentales
- Filosofía del diseño de Python
- Compilador Vs Intérprete
- Instalando el Intérprete de Python

Algoritmos y Programación I  
Lic. Gustavo Bianchi

2

## Un poco de historia

- **Guido Van Rossum**, en el año **1989**, viendo la necesidad de disponer de un lenguaje con características que le permita desarrollar rápida y fácilmente aplicaciones para el proyecto del S.O. Amoeba, en el que estaba interviniendo.
- En base a su experiencia adquirida en el diseño e implementación del lenguaje ABC; **logra en 3 meses el desarrollo de un primer prototipo** para mostrar en el Centro para las Matemáticas y la Informática (CWI), en los Países Bajos.
- En **febrero de 1991, se lanza la primer versión**, comenzando a recibir aportes de terceros para mejorarlo.
- Su nombre proviene del grupo de humoristas británicos Monty Python
- Videos en Youtube:
  - The early years of Python
  - The modern era of Python
  - Otros



Algoritmos y Programación I  
Lic. Gustavo Bianchi

3

## Características

- **Es interpretado interactivamente** (permite el ingreso de comandos mediante el uso de una terminal o consola), a diferencia de los lenguajes compilados o que usan compiladores.
- **Multiparadigma** (programación orientada a objetos, programación imperativa y programación funcional)
- **Fuertemente tipado** (una variable con un valor de un tipo determinado, no se puede usar como si fuera de otro tipo, a menos que se haga una conversión)
- **Tipado dinámico** (una variable puede almacenar valores de distinto tipo en distintos momentos)
- **Utiliza conteo de referencias** (técnica que permite contabilizar la cantidad de referencias que tiene un determinado recurso)

Algoritmos y Programación I  
Lic. Gustavo Bianchi

4

## Filosofía del diseño de Python

(Por Tim Peters)

- **Bello es mejor que feo.**
- **Explícito es mejor que implícito.**
- **Simple es mejor que complejo.**
- **Complejo es mejor que complicado.**
- Plano es mejor que anidado.
- Disperso es mejor que denso.
- **La legibilidad cuenta.**
- **Los casos especiales no son tan especiales como para quebrantar las reglas.**
- Lo práctico gana a lo puro.
- Los errores nunca deberían dejarse pasar silenciosamente.
- A menos que hayan sido silenciados explícitamente.
- Frente a la ambigüedad, rechaza la tentación de adivinar.
- Debería haber una -y preferiblemente sólo una- manera obvia de hacerlo.
- Aunque esa manera puede no ser obvia al principio a menos que usted sea holandés.
- Ahora es mejor que nunca.
- Aunque *nunca* es a menudo mejor que *ya mismo*.
- **Si la implementación es difícil de explicar, es una mala idea.**
- **Si la implementación es fácil de explicar, puede que sea una buena idea.**
- Los espacios de nombres (*namespaces*) son una gran idea ¡Hagamos más de esas cosas!

Nota: Desde la consola se puede acceder escribiendo "import this"

Algoritmos y Programación I  
Lic. Gustavo Bianchi

5

## Compilador Vs Intérprete

Compilador		Intérprete	
Es el encargado de traducir nuestro programa fuente, usualmente a lenguaje de máquina		Es el encargado de traducir instrucciones de código a medida que se solicitan, instrucción por instrucción	
Ventajas	Desventajas	Ventajas	Desventajas
Listo para ser Ejecutado	No es multiplataforma	Es multiplataforma	Se requiere intérprete instalado
Usualmente más rápido	Poco flexible	Fácil de testear	Más lento
El código fuente es privado	Es necesario realizar pasos adicionales	Fácil de buscar y corregir errores (Debugging)	El código fuente es público

Algoritmos y Programación I  
Lic. Gustavo Bianchi

6

# Instalando el intérprete de Python

