#SoftwareLibre20

Programación con

PYTHON

Guido Van Rossum

Es un lenguaje joven comparado con el resto de lenguajes (1991), Python se desarrolla bajo una licencia de Open Source, por lo que se puede usar y distribuir libremente.

CARACTERISTICAS

Interpretado: los lenguajes Interpretados son aquellos en los que el código del programador es traducido mediante un intérprete a medida que es necesario. Que Python sea interpretado nos presenta ventajas:

- Al ser interpretado no necesitamos compilar ahorrándonos mucho tiempo en el desarrollo y prueba de una aplicación.
- Nuestro código fuente puede ser ejecutado en cualquier software siempre y cuando este disponga del intérprete (Windows, Linux, Mac, Web).

CARACTERISTICAS

Multiparadigma: Esto nos dice que Python es un lenguaje que soporta más de un paradigma.
Python admite los siguientes paradigmas.

- Programacion imperativa
- Orientada a objetos
- Estructurada
- Funcional, en menor medida

Tipado Dinámico: Las variables puede tomar diferentes valores de distintos tipos en diferentes momentos. En python las variables son declaradas por su contenido y no por su contenedor, lo que nos va a permitir cambiar el valor y tipo de una variable durante la ejecución sin necesidad de volver a declarar.

Fuertemente tipado: El tipo de valor no cambia repentinamente, cada cambio de tipo requiere una conversión explicita.

Usos de Python

Python como lenguaje de scripting

Tradicionalmente Python ha tenido un uso muy extendido como herramienta de scripting, sustituyendo a scripts escritos en bash, otros lenguajes de script más. Por ello, Python siempre ha sido un buen compañero de los administradores de sistemas y los equipos de operaciones.

Hoy en día, muchas de las herramientas punteras para gestión de despliegues e infraestructura usan o se basan en Python.

Para ese uso se usan módulos (librerías) como OS y SYS que nos permiten acceder a funciones del sistema operativo.

Python en el desarrollo web

Otro de los campos en los que Python ha brillado en los últimos años es en el desarrollo de aplicaciones web, principalmente gracias a frameworks de desarrollo web muy potentes como Django, un framework completo o Flask, un microframework.

Sin embargo, en el ecosistema de desarrollo web existen muchas alternativas y frameworks muy maduros y asentados como Symfony para PHP, Spring para Java, Rails para Ruby. Todos estos frameworks están continuamente tomando ideas entre ellos, inmersos en ofrecer las mejores alternativas para los desarrolladores.

En este caso la ventaja que aporta Django, el principal framework para desarrollo web en Python, es la de ofrecer un marco de trabajo completo y de calidad para desarrollar aplicaciones web muy rápido. Como su leitmotiv dice es: “el framework para perfeccionistas con fechas de entrega”.

Big Data, IA: el boom de Python

Sin embargo, al margen de todas las bondades que hemos comentado del lenguaje, en los últimos años ha ocurrido algo que ha revolucionado y extendido radicalmente el uso de Python.

La generalización del Big Data en los últimos años, seguida de la explosión de la Inteligencia Artificial y el surgimiento de la ciencia de datos como un nuevo área de trabajo con especialistas propios, ha revolucionado el panorama.

Y es que muchas de las nuevas herramientas que han surgido, y que son explotadas por los ingenieros de datos y los científicos de datos, han sido desarrolladas en Python o nos ofrecen Python como la forma predilecta de interactuar con ellas.

Podemos hablar de tecnología para Big Data como PySpark, de herramientas para Data Science como Pandas, NumPy, Matplotlib o Jupyter.

Que necesito para programar en Python

Para programar en Python es necesario tener un editor de texto como puede ser **Bloc de notas** o un IDE(entorno de desarrollo integrado), para poder escribir el código que se ejecutara después

Editores de texto

Sublime Text

IDE

PyDev

Librerias

Una de las grandes ventajas que tiene Python es que las principales librerías (uso habitual) vienen incluidas dentro del interprete, con lo que no hay necesidad de instalarlas. Aunque algunas de las librerías que trae no son del gusto de una parte de la comunidad y se opta por usar librerías de terceros. Estas son algunas:

- Request
- Pillow
- NumPy
- SciPy
- Matplotlib
- Pygame
- Pandas
- Django

#SoftwareLibre20

Zen de Python

Es una colección de 20 principios de software que influyen en el diseño del Lenguaje de Programación Python, 19 escritos por Tim Peters.

Los principios están listados a continuación:

```
Python 3.7.5 (tags/v3.7.5:5c02a39a0b, Oct 15 2019, 00:11:34) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 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.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

- 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!