Resumen Python 3

Víctor Mardones Bravo

Índice general

1. Introducción a Python **1**
1.1. ¿Qué es Python? 1
1.2. Aplicaciones 1
1.3. Componentes principales 2
1.4. Características de Python 2
1.5. Organizaciones que lo usan 3
1.6. Ventajas y desventajas 3
1.7. Sintaxis 4
1.8. Instalación 4
1.9. Hola mundo 5
1.10. El Zen de Python 5

Capítulo 1

Introducción a Python

1.1. ¿Qué es Python?

Python es un lenguaje de programación popular y de alto nivel. Fue creado por Gudo van Rossum y fue lanzado en el año 1991. Se han lanzado diferentes versiones hasta llegar a la versión actual, Python 3.

Según su creador, el nombre de Python viene de la serie de comedia británica "El Circo Volador de Monty Python".

1.2. Aplicaciones

Python tiene aplicaciones en numerosas áreas y generalmente se usa para:

- Scripting
- Desarrollo web
- Desarrollo de software
- Prototipos rápidos
- Desarrollo de aplicaciones para empresas
- Redes y seguridad
- Automatización de tareas
- Programación de bots
- Cálculos complejos
- Estudios científicos y computación científica
- Desarrollo de juegos
- Desarrollo de aplicaciones móviles
- Extracción de datos de sitios web
- Análisis de datos
- Inteligencia artificial y modelos de machine learning
- Procesamiento de imágenes
- Internet de las cosas

1.3. Componentes principales

- **Funciones**: Python tiene funciones predefinidas que son de gran utilidad, por ejemplo las funciones matemáticas. Las funciones son grupos de bloques de código que pueden ejecutarse en cualquier sección del código, según el programador lo necesite.
- **Clases**: Son los planos para construir objetos, los cuales incluyen todas sus características y comportamientos. En Python, prácticamente todo es un objeto, aunque en algunos casos no parezca obvio.
- **Módulos**: Agrupan funciones y clases que sirven para objetivos similares. La biblioteca estándar de Python es muy grande comparada con las de otros lenguajes de programación.
- **Paquetes**: Agrupan módulos usados en aplicaciones grandes, permitiendo su distribución a terceros.

Si estos conceptos no se entienden todavía, no hay problema, pues se verán más a fondo en sus respectivos capítulos.

1.4. Características de Python

- **Soporte de múltiples plataformas**: Python es "platform independent". Esto significa que un mismo código puede ejecutarse en cualquier sistema operativo, ya sea Windows, Unix, Linux o Mac, sin necesidad de hacerle cambios.
- **Interpretado**: El código escrito en Python no necesita ser compilado, como ocurre en lenguajes como C, Java o C++. El intérprete de Python convierte el código en bytes ejecutables línea por línea, lo cual hace el desarrollo de prototipos más conveniente para el programador, pero al mismo tiempo hace que la ejecución del código sea más lenta.
- **Simple**: La sintaxis de Python es simple y fácil de leer, aunque requiere un conocimiento de inglés básico. Esta simplicidad también permite escribir código más corto, obteniendo el mismo resultado en menos líneas que en otros lenguajes.
- **Robusto**: Esto significa que el lenguaje es capaz de manejar los posibles errores que podrían ocurrir durante la ejecución de programas escritos en él.
- **De alto nivel**: Es un lenguaje de nivel usado para scripting. Esto quiere decir que el programador no necesita recordar características de bajo nivel como la arquitectura del sistema o el manejo de memoria. Estas características se abstraen.
- **Multiparadigma**: Python soporta programación estructurada, funcional y orientada a objetos.
- **Gran soporte de librerías**: Python puede integrarse a otras librerías que tengan funcionalidades específicas. No hay necesidad de tener que escribir el código por tu cuenta, si ya existe y funciona correctamente.
- **Integrable**: El código fuente escrito en Python puede usarse dentro de otros lenguajes de programación, lo cual permite expandir las funcionalidades de sus programas con las de programas escritos en otros lenguajes.
- **De código abierto**: Python es de código abierto. Se puede usar sin necesidad de tomar su licencias y se puede descargar fácilmente.
- **Gratuito**: Ninguna persona u organización necesita pagar para usarlo en sus proyectos.
- **Conciso y compacto**: El código escrito en Python es conciso y compacto, lo que sirve para que los programadores puedan entenderlo rápidamente.
- **Dinámicamente tipeado**: El tipo de dato de cada variable se decide durante el tiempo de ejecución, lo que significa que no es necesario declarar su tipo en el código.

1.5. Organizaciones que lo usan

Python es muy popular y es usado por organizaciones y aplicaciones como:

- Microsoft
- Google
- Yahoo
- Mozilla
- Cisco
- Facebook
- Spotify
- OpenStack
- NASA
- CIA
- Disney

1.6. Ventajas y desventajas

Ventajas:

- Es de código abierto y fácil de empezar a usar.
- Es fácil de aprender y explorar.
- Se pueden integrar módulos de terceros con facilidad.
- Es un lenguaje de alto nivel y orientado a objetos.
- Es interactivo y portable.
- Las aplicaciones pueden ejecutarse en cualquier plataforma.
- Es dinámicamente tipeado.
- Tiene una comunidad grande y foros activos.
- Tiene una sintaxis intuitiva.
- Tiene librerías con amplio soporte.
- Es un lenguaje interpretado.
- Se puede conectar con bases de datos.
- Aumenta la productividad debido a su simplicidad.

Desventajas:

- No puede usarse para desarrollo de aplicaciones móviles.
- Su acceso a bases de datos es limitado.
- Consume más memoria por ser dinámicamente tipeado.
- Su ejecución es lenta comparada con lenguajes compilados.
- Las aplicaciones y código requieren mayor mantenimiento, debido a su gran grado de abstracción.
- Debido a su abstracción, algunos conceptos de programación importantes podrían ser omitidos durante su aprendizaje.

1.7. Sintaxis

Python fue diseñado para ser fácil de leer y tiene similitudes con el idioma inglés, con influencia de las matemáticas.

A diferencia de otros lenguajes de programación, Python termina las líneas con un salto de línea, en vez de un punto y coma `;`.

Para delimitar bloques de código en estructuras de control como `if`, `while`, `for` o funciones, se utiliza indentación con espacios en blanco en vez de usar llaves `{}` como en otros lenguajes.

1.8. Instalación

Muchos PCs vienen con Python preinstalado. Para revisar la versión que se tiene instalada se puede ingresar el siguiente comando en el terminal.

```
python --version
```

Salida:

```
Python 3.9.2
```

Siempre se puede descargar la última versión gratuitamente desde el sitio web oficial. Después de instalarlo, existen muchas formas de utilizar este lenguaje.

Se puede abrir la consola de Python en el terminal ingresando el siguiente comando.

```
python
```

Salida:

```
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit
↪  (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Si no funciona, existe otro comando que también puede abrir la consola.

```
py
```

Dentro de la consola, se puede ingresar el código línea por línea, y es interpretado después de pulsar `Intro`. Para salir de la consola, se debe llamar a la función `exit()`.

```
exit()
```

Para ejecutar un archivo de extensión `.py`, se puede ingresar `python` seguido de la ruta relativa (desde el directorio de trabajo) o absoluta del archivo.

```
python archivo.py
```

También existen muchos IDEs (Integrated Development Environment) que permiten programar en Python. Estos contienen muchas herramientas que pueden ser de utilidad.

Algunos IDEs y editores de texto recomendados son los siguientes:

- IDLE, que viene preinstalado con Python.
- Thonny
- Pycharm
- Visual Studio Code, que requiere instalar esta extensión.
- Sublime Text

1.9. Hola mundo

Para mostrar el texto "Hola mundo" en pantalla se puede usar la función `print()`.

```
print('Hola mundo')
```

Salida:

```
Hola mundo
```

Cada declaración de impresión `print()` genera texto en una nueva línea.

```
print('Hola')
print('Mundo')
```

Salida:

```
Hola
Mundo
```

Si no se le entrega ningún texto a `print()`, se imprimirá una línea vacía. El texto puede escribirse entre comillas simples o dobles.

```
print("Hola")
print()
print("Mundo")
```

Salida:

```
Hola

Mundo
```

1.10. El Zen de Python

El Zen de Python es una colección de 20 "principios", 19 de ellos escritos por Tim Peters. Estos principios han servido como inspiración para muchos programadores de todo el mundo a la hora de crear software.

Se mostrará en pantalla como un "huevo de pascua" la primera vez que se ejecute la siguiente linea, para importar el módulo `this`.

```
import this
```

Después se mostrará el siguiente texto, en inglés.

```
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!
```

Una traducción posible sería la siguiente:

```
El Zen de Python, escrito 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.
Espaciado es mejor que denso.
La legibilidad es importante.
Los casos especiales no son lo suficientemente especiales como para romper
↪  las reglas.
Sin embargo la practicidad le gana a la pureza.
Los errores nunca deberían pasar silenciosamente.
A menos que se silencien explícitamente.
Frente a la ambigüedad, evitar la tentación de adivinar.
Debería haber una, y preferiblemente solo una, manera obvia de hacerlo.
A pesar de que esa manera no sea obvia a menos que seas Holandés.
Ahora es mejor que nunca.
A pesar de que nunca es muchas veces mejor que *ahora* 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 son una gran idea, ¡tengamos más de esos!
```

Nota: Si se observa con claridad, se puede ver que el principio 20 “no existe”. Tim Peters, quien lo escribió, dejó este último principio para que Guido (el creador de Python) lo llene. El origen de este texto puede encontrarse en esta conversación.