Lectura 3: El tutorial de Python (páginas 1 a 7)

Este material de lectura corresponde a las primeras páginas del libro "El tutorial de Python". En la sección de recursos del curso encontrarás el libro completo.

En estas primeras páginas encontrarás una breve introducción a Python y al tutorial, seguido por algunas motivaciones para aprender el lenguaje y se describen algunas características de Python.

Por último se explica cómo utilizar el intérprete de Python en sus diversas variantes.

Cantidad de páginas: 5.

Introducción

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, https://www.python.org/, y puede distribuirse libremente. El mismo sitio contiene también distribuciones y enlaces de muchos módulos libres de Python de terceros, programas y herramientas, y documentación adicional.

El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables.

Este tutorial introduce de manera informal al lector a los conceptos y características básicas del lenguaje y el sistema de Python. Es bueno tener un interprete de Python a mano para experimentar, sin embargo todos los ejemplos están aislados, por lo tanto el tutorial puede leerse estando desconectado.

Para una descripción de los objetos y módulos estándar, mirá *La referencia de la biblioteca*. *La referencia de la biblioteca* provee una definición más formal del lenguaje. Para escribir extensiones en C o C++, leé *Extendiendo e Integrando el Intérprete de Python* y la *Referencia de la API Python/C*. Hay también numerosos libros que tratan a Python en profundidad.

Este tutorial no pretende ser exhaustivo ni tratar cada una de las características, o siquiera las características más usadas. En cambio, introduce la mayoría de las características más notables de Python, y te dará una buena idea del gusto y estilo del lenguaje. Luego de leerlo, serás capaz de leer y escribir módulos y programas en Python, y estarás listo para aprender más de los variados módulos de la biblioteca de Python descriptos en *La referencia de la biblioteca*.

También vale la pena mirar el *glosario*.

Abriendo tu apetito

Si trabajás mucho con computadoras, eventualmente encontrarás que te gustaría automatizar alguna tarea. Por ejemplo, podrías desear realizar una búsqueda y reemplazo en un gran número de archivos de texto, o renombrar y reorganizar un montón de archivos con fotos de una manera compleja. Tal vez quieras escribir alguna pequeña base de datos personalizada, o una aplicación especializada con interfaz gráfica, o un juego simple.

Si sos un desarrollador de software profesional, tal vez necesites trabajar con varias bibliotecas de C/C++/Java pero encuentres que se hace lento el ciclo usual de escribir/compilar/testear/recompilar. Tal vez estás escribiendo una batería de pruebas para una de esas bibliotecas y encuentres que escribir el código de testeo se hace una tarea tediosa. O tal vez has escrito un programa al que le vendría bien un lenguaje de extensión, y no quieres diseñar/implementar todo un nuevo lenguaje para tu aplicación.

Python es el lenguaje justo para ti.

Podrías escribir un script (o programa) en el interprete de comandos o un archivo por lotes de Windows para algunas de estas tareas, pero los scripts se lucen para mover archivos de un lado a otro y para modificar datos de texto, no para aplicaciones con interfaz de usuario o juegos. Podrías escribir un programa en C/C++/Java, pero puede tomar mucho tiempo de desarrollo obtener al menos un primer borrador del programa. Python es más fácil de usar, está disponible para sistemas operativos Windows, Mac OS X y Unix, y te ayudará a realizar tu tarea más velozmente.

Python es fácil de usar, pero es un lenguaje de programación de verdad, ofreciendo mucha más estructura y soporte para programas grandes de lo que pueden ofrecer los scripts de Unix o archivos por lotes. Por otro lado, Python ofrece mucho más chequeo de error que C, y siendo un *lenguaje de muy alto nivel*, tiene tipos de datos de alto nivel incorporados como arreglos de tamaño flexible y diccionarios. Debido a sus tipos de datos más generales Python puede aplicarse a un dominio de problemas mayor que Awk o incluso Perl, y aún así muchas cosas siguen siendo al menos igual de fácil en Python que en esos lenguajes.

Python te permite separar tu programa en módulos que pueden reusarse en otros programas en Python. Viene con una gran colección de módulos estándar que puedes usar como base de tus programas, o como ejemplos para empezar a aprender a programar en Python. Algunos de estos módulos proveen cosas como entrada/salida a archivos, llamadas al sistema, sockets, e incluso interfaces a sistemas de interfaz gráfica de usuario como Tk.

Python es un lenguaje interpretado, lo cual puede ahorrarte mucho tiempo durante el desarrollo ya que no es necesario compilar ni enlazar. El intérprete puede usarse interactivamente, lo que facilita experimentar con características del lenguaje, escribir programas descartables, o probar funciones cuando se hace desarrollo de programas de abajo hacia arriba. Es también una calculadora de escritorio práctica.

Python permite escribir programas compactos y legibles. Los programas en Python son típicamente más cortos que sus programas equivalentes en C, C++ o Java por varios motivos:

- los tipos de datos de alto nivel permiten expresar operaciones complejas en una sola instrucción
- la agrupación de instrucciones se hace por sangría en vez de llaves de apertura y cierre
- no es necesario declarar variables ni argumentos.

Python es *extensible*: si ya sabes programar en C es fácil agregar una nueva función o módulo al intérprete, ya sea para realizar operaciones críticas a velocidad máxima, o para enlazar programas Python con bibliotecas que tal vez sólo estén disponibles en forma binaria (por ejemplo bibliotecas gráficas específicas de un fabricante). Una vez que estés realmente entusiasmado, podés enlazar el intérprete Python en una aplicación hecha en C y usarlo como lenguaje de extensión o de comando para esa aplicación.

Por cierto, el lenguaje recibe su nombre del programa de televisión de la BBC "Monty Python's Flying Circus" y no tiene nada que ver con reptiles. Hacer referencias a sketches de Monty Python en la documentación no sólo esta permitido, ¡sino que también está bien visto!

Ahora que ya estás emocionado con Python, querrás verlo en más detalle. Como la mejor forma de aprender un lenguaje es usarlo, el tutorial te invita a que juegues con el intérprete de Python a medida que vas leyendo.

En el próximo capítulo se explicará la mecánica de uso del intérprete. Esta es información bastante mundana, pero es esencial para poder probar los ejemplos que aparecerán más adelante.

El resto del tutorial introduce varias características del lenguaje y el sistema Python a través de ejemplos, empezando con expresiones, instrucciones y tipos de datos simples, pasando por funciones y módulos, y finalmente tocando conceptos avanzados como excepciones y clases definidas por el usuario.

Usando el intérprete de Python

Invocando al intérprete

Por lo general, el intérprete de Python se instala en `/usr/local/bin/python3.6` en las máquinas dónde está disponible; poner `/usr/local/bin` en el camino de búsqueda de tu intérprete de comandos Unix hace posible iniciarlo ingresando la orden:

```
python3.6
```

...en la terminal. [1] Ya que la elección del directorio dónde vivirá el intérprete es una opción del proceso de instalación, puede estar en otros lugares; consultá a tu Gurú Python local o administrador de sistemas. (Por ejemplo, `/usr/local/python` es una alternativa popular).

En máquinas con Windows, la instalación de Python por lo general se encuentra en `C:\Python36`, aunque se puede cambiar durante la instalación. Para añadir este directorio al camino, podes ingresar la siguiente orden en el prompt de DOS:

```
set path=%path%;C:\python36
```

Se puede salir del intérprete con estado de salida cero ingresando el carácter de fin de archivo (Control-D en Unix, Control-Z en Windows) en el prompt primario. Si esto no funciona, se puede salir del intérprete ingresando:`quit()`.

Las características para editar líneas del intérprete incluyen edición interactiva, sustitución usando el historial y completado de código en sistemas que soportan readline. Tal vez la forma más rápida de detectar si las características de edición están presentes es ingresar Control-P en el primer prompt de Python que aparezca. Si se escucha un beep, las características están presentes; ver Apéndice *Edición de entrada interactiva y sustitución de historial* para una introducción a las teclas. Si no pasa nada, o si aparece `^P`, estas características no están disponibles; solo vas a poder usar backspace para borrar los caracteres de la línea actual.

La forma de operar del intérprete es parecida a la línea de comandos de Unix: cuando se la llama con la entrada estándar conectada a una terminal lee y ejecuta comandos en forma interactiva; cuando es llamada con un nombre de archivo como argumento o con un archivo como entrada estándar, lee y ejecuta un *script* del archivo.

Una segunda forma de iniciar el intérprete es `python -c comando [arg] ...`, que ejecuta las sentencias en *comando*, similar a la opción *-c* de la línea de comandos. Ya que las sentencias de Python suelen tener espacios en blanco u otros caracteres que son especiales en la línea de comandos, es normalmente recomendado citar *comando* entre comillas dobles.

Algunos módulos de Python son también útiles como scripts. Pueden invocarse usando `python -m module [arg] ...`, que ejecuta el código de *module* como si se hubiese ingresado su nombre completo en la línea de comandos.

Cuando se usa un script, a veces es útil correr primero el script y luego entrar al modo interactivo. Esto se puede hacer pasándole la opción *-i* antes del nombre del script.

Todas las opciones de línea de comandos están se descriptas en *Línea de comandos y entorno*.

Pasaje de argumentos

Cuando son conocidos por el intérprete, el nombre del script y los argumentos adicionales son entonces convertidos a una lista de cadenas de texto asignada a la variable `argv` del módulo `sys`. Podés acceder a esta lista haciendo `import sys`. El largo de esta lista es al menos uno; cuando ningún script o argumentos son pasados, `sys.argv[0]` es una cadena vacía. Cuando se pasa el nombre del script con `'-'` (lo que significa la entrada estándar), `sys.argv[0]` vale `'-'`. Cuando se usa *-c command*, `sys.argv[0]` vale `'-c'`. Cuando se usa *-m module*, `sys.argv[0]` toma el valor del nombre completo del módulo. Las opciones encontradas luego de *-c command* o *-m module* no son consumidas por el procesador de opciones de Python pero de todas formas almacenadas en `sys.argv` para ser manejadas por el comando o módulo.

Modo interactivo

Se dice que estamos usando el intérprete en modo interactivo, cuando los comandos son leídos desde una terminal. En este modo espera el siguiente comando con el *prompt primario*, usualmente tres signos mayor-que (`>>>`); para las líneas de continuación espera con el *prompt secundario*, por defecto tres puntos (`...`). Antes de mostrar el prompt primario, el intérprete muestra un mensaje de bienvenida reportando su número de versión y una nota de copyright:

```
$ python3.6
Python 3.6 (default, Sep 16 2015, 09:25:04)
[GCC 4.8.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Las líneas de continuación son necesarias cuando queremos ingresar un constructor multilínea. Como en el ejemplo, mirá la sentencia `if`:

```
>>> el_mundo_es_plano = True
>>> if el_mundo_es_plano:
...     print("¡Tené cuidado de no caerte!")
...
¡Tené cuidado de no caerte!
```

Para más información sobre el modo interactivo, ve a *Modo interactivo*.

El intérprete y su entorno

Codificación del código fuente

Por default, los archivos fuente de Python son tratados como codificados en UTF-8. En esa codificación, los caracteres de la mayoría de los lenguajes del mundo pueden ser usados simultáneamente en literales, identificadores y comentarios, a pesar de que la biblioteca estándar usa solamente caracteres ASCII para los identificadores, una convención que debería seguir cualquier código que sea portable. Para mostrar estos caracteres correctamente, tu editor debe reconocer que el archivo está en UTF-8 y usar una tipografía que soporte todos los careacteres del archivo.

Para especificar una codificación distinta de la por defecto, un línea de comentario especial debe ser agregada como la *primera* línea del archivo. La sintaxis es como sigue:

```
# -*- coding: encoding -*-
```

Donde *encoding* es uno de los `codecs` válidos soportados por Python.

Por ejemplo, para indicar que el encoding Windows-1252 es el usado, la primera línea de tu código fuente debe ser:

```
# -*- coding: cp-1252 -*-
```

Una excepción a la regla de la *primera línea* es cuando el código fuente comienza con *UNIX "shebang" line*. En este caso, la declaración del encoding debe ser agregada como la segunda línea del archivo. Por ejempo:

```
#!/usr/bin/env python3
# -*- coding: cp-1252 -*-
```

1 En Unix, el intérprete de Python 3.x no se instala por default con el ejecutable llamado `python` para que no conflictúe con un ejecutable de Python 2.x que esté instalado simultaneamente.