Secuencias simples

¿De qué nos ocuparemos?
Comandos simples, el uso de Python como una calculadora, el uso de paréntesis para obtener resultados correctos y el uso de cadenas de formato para combinar texto y números. Finalmente veremos cómo cerrar Python desde dentro de un programa.

Una secuencia simple de instrucciones es el programa más básico que podés escribir. La secuencia más simple es aquella que contiene un solo comando. Ahora probaremos algunos de estos comandos. El texto destacado es el que deberás escribir en la línea de comando (el símbolo >>> que aparece en la consola de Python), en el párrafo siguiente encontrarás las explicaciones necesarias.

>>> print 'Hola a todos!'

Mediante el comando `print` logramos que Python nos muestre un resultado en la pantalla. En este caso el resultado implica imprimir en la pantalla la secuencia de caracteres `H,o,l,a, ,a, ,t,o,d,o,s,!`. Entre los programadores estas secuencias de caracteres se conocen como *cadenas de caracteres* or simplemente como *cadenas* (*string*).

Definimos una cadena al encerrarla entre comillas. Podemos usar tanto comillas simples (como en el ejemplo anterior) como comillas dobles: "una cadena". Esto permite incluir un tipo de comillas dentro de una cadena marcada con el otro tipo, lo cual es útil sobre todo para incluir apóstrofos.

```
>>> print "La frase Monty Python's Flying Circus incluye un ' en su interior..."
```

Pero las cadenas no son lo único que puede imprimirse:

>>>print 6 + 5

En este caso imprimimos el resultado de una operación aritmética: sumamos seis y ocho. Python reconoció los números y el signo de adición, y resolvió el cálculo para nosotros. Luego mostró en pantalla el resultado.

De forma sencilla has encontrado un nuevo uso para Python: una 'calculadora de bolsillo' siempre a mano. Probá con otras sumas o usá los otros *operadores* aritméticos:

- sustracción (-)
- multiplicación (*)
- división (/)

También se pueden combinar expresiones múltiples como:

```
>>> print ((8 * 4) + (7 - 3)) / (2 + 4)
```

Notá que utilicé paréntesis para agrupar los números. ¿Que ocurriría si utilizáramos la misma cadena pero sin paréntesis? El asunto es que Python evalúa la multiplicación y la división antes que la suma y la resta. Esto es lo que uno espera desde un punto de vista matemático, pero quizás no es lo que uno puede esperar como programador. Todos los lenguajes de programación tienen reglas que determinan el orden de evaluación de las operaciones, lo que se conoce con el nombre de *precedencia de los operadores*. Es necesario revisar la documentación de referencia de cada lenguaje para conocer el funcionamiento del orden de precedencia. En el caso de Python generalmente este coincide con lo esperable por lógica e intuición, pero ocasionalmente puede haber diferencias...

Como regla general es más seguro incluir siempre los paréntesis cuando operamos con sumas complejas para obetener un resultado correcto.

Otro asunto a tener en cuenta:

```
>>> print 5/2
```

Da como resultado un número entero (*integer*), en este caso 2. Salvo que le indiquemos lo contrario, Python toma a todos los números como enteros, y da por sentado que así lo desea el usuario. Si necesitás usar decimales, simplemente agregá un número después del punto decimal.

```
>>> print 5/2.0
2.5
```

Python lee 2.0 y se da cuenta de que se trata de una fracción (las cuales se definen como números *reales* o *de punto flotante* en la jerga informática), por consiguiente responde con un resultado también utilizando los decimales correspondientes. Si nos mantenemos dentro de los números enteros, podemos obtener el resto de una división utilizando como operador al signo %. Con este operador Python sólo mostrará el resto:

```
>>> print 7/2
3
>>> print 7%2
1
>>> print 7%4
3
```

El operador % se conoce en otros lenguajes como el operador *módulo* y generalmente se codifica como `MOD` u otro nombre similar.

Si experimentás, pronto clarificarás tus ideas.

>>>print 'El total es: ', 23+45

Como te habrás dado cuenta, se pueden imprimir a la vez cadenas y números combinándolos en una misma intrucción print y separándolos con una coma. Podemos ampliar esta característica combinándola con un truco de Python muy útil para mostrar datos denominado *cadena de formato*:

```
>>> print "La suma de %d y %d es: %d" % (7,18,7+18)
```

En este comando la cadena de formato contiene en su interior a los marcadores '%' La letra 'd' después del % le indica a Python que debe colocar en ese lugar un 'número decimal'. Los valores para llenar los marcadores se obtienen de los valores colocados entre paréntesis que siguen al signo % sin modificadores.

Hay otras letras que pueden incluirse después del marcador %. Algunas de ellas son:

- %s - para una cadena
- %x - para un número hexadecimal
- %0.2f - para un número real de hasta dos decimales
- %04d - completa el número con ceros hasta llegar a los cuatro dígitos

La documentación de Python presenta una lista completa de estos modificadores de formato...

De hecho se puede imprimir cualquier *objeto* de Python con el comando print. A veces el resultado no será el que uno espera (tal vez sólo una descripción de qué tipo de objeto es) pero al menos siempre podrá ser impreso.

>>>import sys

Este es un comando extraño. Si lo has probado, habrás notado que aparentemente no hace nada en especial. Pero esto no es cierto. Para entender qué ha ocurrido debemos dar un vistazo a la arquitectura interna de Python (los que no utilizan Python, estarán de acuerdo en que siempre habrá también para ustedes un mecanismo similar).

Cuando se inicia Python hay un conjunto de comandos disponibles denominados *predeterminados (built-in)* que se incluyen internamente en el núcleo de Python. Sin embargo, Python puede ampliar la lista de comandos disponibles mediante la incorporación de modulos de extensión. Es como comprar una nueva herramienta en una ferretería y agregarla a la caja. La herramienta en este caso sería `sys` y la operación `import` sería la acción de ponerla en la caja.

En realidad lo que este comando hace es agregar una serie de nuevas 'herramientas' en forma de comandos de Python, las cuales han sido definidas previamente en un archivo llamado 'sys.py'. Esta es la forma en que Python puede ampliarse para realizar todo tipo de acciones que no estén predeterminadas en el sistema básico. Uno puede incluso crear sus propios *módulos* importándolos y usándolos de la misma manera que los módulos provistos con la instalación de Python.

Pero, ¿Cómo usamos estas nuevas herramientas?

>>>sys.exit()

Eeeeeepa! ¿Qué ha ocurrido? Simplemente ejecutamos el comando `exit` definido en el módulo `sys`. Este comando produce que finalice Python y se cierre. (**Nota**: Normalmente se sale de Python utilizando el caracter de Fin de Archivo (*EOF*) en la línea de comando >>> - CTRL-Z en DOS o CTRL-D en Unix).

Habrás notado que a `exit` le siguen dos paréntesis. Esto se debe a que exit es una *función* definida en sys, y cuando invocamos una función de Python debemos agregar los paréntesis aunque no haya nada entre ellos.

Probá escribir sys.exit sin los paréntesis. Python responde diciendo que exit es una función y no la ejecuta.

Una aclaración final: los dos últimos comandos son útiles sólo en combinación. Es decir, para abandonar Python de otra manera que tipeando EOF es necesario escribir:

```
import sys
sys.exit()
```

Esta es una secuencia de dos comandos. Nos estamos acercando a la programación real... Pero primero deberemos hablar un poco acerca de los *datos*.

Para recordar
• Un solo comando es también un programa • Python realiza calculos matemáticos *casi* como uno esperaría • Para obtener un resultado fraccionario debemos usar un número fraccionario • Se pueden combinar texto y números usando el operador de formato % • Salir de Python con `import sys; sys.exit()`

Si tenés preguntas o sugerencias, enviáme un e-mail a : alan.gauld@btinternet.com