Hablando con el usuario

¿De qué nos ocuparemos?
Cómo pedirle datos al usuario y cómo leer esos datos una vez que han sido ingresados. Veremos cómo leer datos numéricos y de cadena, y cómo recuperar los datos ingresados como argumentos en la línea de comando.

Hasta ahora nuestros programas han manejado datos estáticos, es decir, no podemos cambiar estos datos más que reescribiendo las asignaciones internas del programa. Sin embargo, la mayor parte de los programas no funcionan así, ya que esperan ser controlados por el usuario en tanto éste indique cuál es el archivo que hay que abrir, qué valores utilizar, etc. Veamos cómo podemos lograr esta técnica en nuestros programas.

>>> print raw_input("Escriba algo: ")

El comando `raw_input` simplemente muestra la cadena entre paréntesis y captura lo que el usuario escriba como respuesta. La instrucción print muestra luego la respuesta del usuario a nuestro pedido. También podemos asignar esta respuesta a una variable, lo que nos permite reutilizarla más adelante en nuestro programa:

```
resp = raw_input("¿Cómo te llamás? ")
print "Hola, %s, mucho gusto de conocerte" % resp
```

`raw_input` tiene un primo llamado `input`. La diferencia entre ellos es que `raw_input` acepta los caracteres que el usuario ingresa y los transforma en un cadena, mientras que `input` intenta formar con ellos un valor numérico. Si el usuario tipea los caracteres '1','2','3', `input` convertirá la respuesta en el número *123*.

Usemos `input` para decidir qué tabla de multiplicación vamos a realizar:

```
multiplicador = input("¿Qué tabla de multiplicación desea conocer? Ingrese un número")
for j in range(1,13):
   print "%d x %d = %d" % (j, multiplicador, j * multiplicador)
```

INPUT

En BASIC la instrucción INPUT lee lo ingresado por el usuario:

```
INPUT "¿Qué tabla de multiplicación desea conocer? Ingrese un número";M
FOR J = 1 to 12
    PRINT M "x" J "= " M*J
NEXT J
```

Como puede observarse, es muy similar a Python excepto que debemos poner la variable al final de la instrucción. Además BASIC utiliza `INPUT` tanto para números como para cadenas. Usualmente hay algunas características extra en la instrucción `INPUT` del BASIC, para sacarle un mejor provecho a esta instruccion te recomiendo que revises la documentación de tu versión de BASIC.

Parámetros en la línea de comando

Otro tipo de ingreso de datos es a partir de la línea de comando. Por ejemplo cuando utilizás un editor de texto desde la línea de comando:

```
EDIT Foo.txt
```

¿Cómo sabe el editor el nombre del archivo?

En la mayor parte de los lenguajes el sistema provee un vector o una lista de cadenas que contienen lo tipeado en la línea de comando. En esta lista el primer elemento será el comando mismo, el segundo será el primer argumento, etc. Con frecuencia también presenta algún tipo de variable mágica que nos indica la cantidad de elementos en esta lista.

En Python esta lista la maneja el módulo sys y lleva como nombre `argv` (abreviatura de "*arg*ument *v*alues", valores de argumento). Podemos extraer sus elementos por medio de la utilización de índices o iterando sobre la lista:

```
import sys
for item in sys.argv:
    print item

print "El primer argumento fue:", sys.argv[1]
```

Tcl tiene un esquema similar con tres variables:

- `argv0` - el nombre del comando,
- `argv` - una cadena que contiene el resto de la línea de comando, y
- `argc` - el número de palabras en argv

Un ejemplo de acceso a los argumentos de la línea de comando en Tcl sería:

```
puts "El comando fue: $argv0"
puts "El primer argumento fue:  [lindex $argv 0]"
```

Que yo sepa, BASIC no permite la utilización directa de argumentos en la línea de comandos, aunque sería posible obtenerlos por medio de ciertas opciones del sistema operativo. Sin embargo, este es un tema muy avanzado para este tutorial, por lo que recomiendo que en BASIC se requieran los datos al usuario de manera interactiva.

Respecto de este tema no iremos más allá que lo que hemos visto hasta ahora. Si bien es bastante primitivo, nos permite escribir programas utiles que soliciten datos de manera interactiva. En los viejos tiempos de Unix y de la PC esta era la máxima interacción que uno podía obtener. Actualmente con Python, Tcl y BASIC (en su encarnación 'Visual') es posible escribir sofisticadas interfases gráficas de usuario (GUI) con ventanas, cuadros de diálogo, etc... pero todo esto es muy avanzado para este curso. El caso de estudio que presento al final de este tutorial muestra un breve ejemplo de cómo obtener datos por medio de una GUI, pero no nos dedicaremos especialmente a analizar cómo funciona. Hay muchísimos tutoriales en la Web que explican cómo trabajar con estas interfases una vez que uno ha fijado los conceptos esenciales de la programación general. En las páginas de referencia he colocado una lista de algunos de ellos.

Para recordar

- Usar `input` para leer números y `raw_input` para leer caracteres/cadenas.
- Tanto `input` como `raw_input` pueden mostrar una cadena de mensaje para el usuario.
- El comando `INPUT` de BASIC puede utilizarse para cualquier tipo de datos.
- Los parámetros de la línea de comando pueden obtenerse de la lista `argv` importada desde el módulo `sys` en Python, de la cual el primer item es el nombre del programa.
- TCL usa una lista de nombre similar, `argv`, para guardar los datos de la línea de comando, pero el nombre del programa se guarda en la variable `argv0`.
- La variable `__name__` tomará el valor de `"__main__"` si el módulo ha sido ejecutado desde la línea de comando o mediante un doble click en Windows.

Anterior Próxima Contenido

Si tenés sugerencias o dudas podés enviar un email en inglés a: alan.gauld@btinternet.com o en español a: manilio@xoommail.com