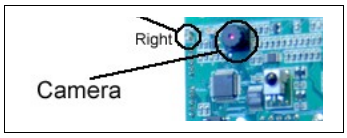
5 Percibiendo el Mundo

Veo todos los obstáculos en mi camino.
De la canción *Puedo ver claramente ahora*,
Johnny Nash, 1972.

Página opuesta: Los Sentidos
La foto es cortesía de Blogosphere (cultura.blogosfere.it)

En el capítulo anterior, aprendieron cómo la percepción del tiempo, el atascamiento y el nivel de batería, pueden ser usados para escribir comportamientos de robot simples pero interesantes. Todos los robots, además, vienen equipados con un conjunto de sensores externos (o extroperceptores) que pueden percibir varias cosas en el ambiente. La percepción hace que el robot sea *consciente* de su entorno y puede ser usada para definir más comportamientos inteligentes. La percepción también está relacionada con otro concepto importante en computación: la entrada o input. Las computadoras actúan a partir de diferente tipo de información: números, textos, sonidos, imágenes, etc. para producir aplicaciones útiles. Generalmente se hace referencia a la adquisición de información para ser procesada como entrada o input. En este capítulo también veremos cómo otras formas de entradas pueden ser adquiridas para ser procesadas por un programa. Primero, focalicemos en los sensores del Scribbler.

Los sensores del Scribbler



El robot Scribbler puede percibir la cantidad de luz en el ambiente, la presencia (o ausencia) de obstáculos a su alrededor, y también toma fotos con su cámara. En el Scribbler hay localizados varios dispositivos (o sensores). A continuación, se presenta una breve descripción de los mismos:

Cámara: La cámara puede tomar una imagen fija de cualquier cosa que el robot esté "visualizando" en ese momento.

Luz: Hay tres sensores de luz en el robot. Están localizados en los tres agujeros de la parte frontal del robot. Estos sensores pueden detectar los niveles de brillo (u oscuridad). Estos pueden ser usados para detectar variaciones en la luz ambiente en una habitación. Además, al usar la información proveniente de la cámara, el Scribbler tiene a disposición un conjunto de sensores de brillo alternativo (izquierda, centro y derecha).

Proximidad: Hay dos juegos de estos sensores en el Scribbler: Sensores IR (izquierda y derecha) en el frente; y Sensores de Obstáculos (izquierda, centro y derecha) en el *Fluke*. Pueden ser usados para detectar objetos en el frente y a los costados.

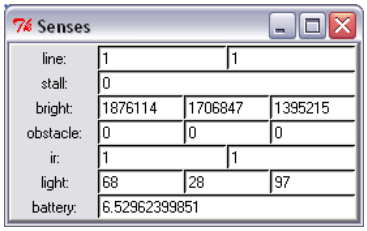
Familiarizándose con los sensores

Percibir a través de los sensores provistos por el Scribbler es sencillo. Una vez que estén familiarizados con los detalles de comportamiento de los sensores, podrán usarlos en sus programas para diseñar comportamientos interesantes con su Scribbler. Pero primero, debemos ocupar un poco de tiempo conociendo estos sensores; cómo acceder a la información que reportan; y cómo se ve esta información. Con respecto a los sensores internos, Myro provee varias funciones que pueden ser usadas para adquirir datos de cada dispositivo de sensor. Allí donde haya disponibles múltiples sensores, tienen la opción de obtener datos de todos los sensores, o selectivamente de un sensor individual.

Realizar la siguiente actividad: quizás la mejor forma de tener una mirada rápida sobre el comportamiento global de los sensores es usando la función Myro senses:

```
>>> senses()
```

Scribbler Sensors

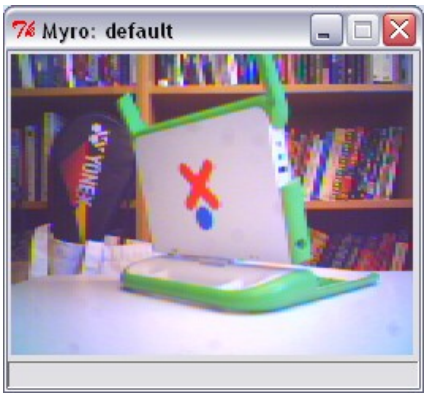


Esto da como resultado una ventana (ver la imagen adjunta) que muestra todos los valores de los sensores (exceptuando la cámara) en tiempo real. Se actualizan a cada segundo. Deberían mover el robot por el entorno par ver cómo cambian los valores de los sensores. La ventana también mostrará los valores del sensor de atascamiento y de nivel de batería. El valor que está en el extremo izquierdo en cada conjunto de sensor (luz, IR, obstáculo y brillo) es el valor del sensor de la izquierda, seguido por el del centro (si existe), y luego el de la derecha.

La Cámara

La cámara puede ser usada para tomar fotos de la vista actual del robot. Como pueden ver, la cámara está localizada en el Fluke. La vista de la imagen tomada por la cámara dependerá de la orientación del robot (y del Fluke). Para tomar fotos con la cámara deberán usar el comando takePicture:

```
takePicture()
takePicture("color")
takePicture("gray")
```



Este comando toma una foto y devuelve una imagen. De manera predeterminada, cuando no se especifican parámetros, la foto es en color. Si se usa la opción "gray", pueden obtener una foto en escala de grises.

Por ejemplo:

```
>>> p = takePicture()
>>> show(p)
```

De manera alternativa, también pueden hacer lo siguiente:

```
>>> show(takePicture())
```

Una vez que han tomado una foto con la cámara, pueden hacer muchas cosas con ella. Por ejemplo, pueden querer ver si hay una computadora portable presente en la imagen. El procesamiento de información es un vasto sub-campo de las ciencias de la computación y tiene

aplicaciones en muchas áreas. Comprender una imagen es bastante complejo, pero es algo que hacemos con naturalidad. Por ejemplo, en la imagen de arriba, no tenemos problemas para localizar la portable la estantería de libros de fondo, e incluso un estuche de una raqueta. La cámara del Scribbler es su dispositivo más complejo y requerirá mucho esfuerzo computacional y energía para ser usado en el diseño de comportamientos. En el caso más simple, la cámara les puede servir como sus “ojos” remotos en el robot. Quizás no lo mencionamos antes, pero el rango del Bluetooth inalámbrico en el robot es de 100 metros. En el capítulo 9 aprenderemos acerca de varias maneras de usar las fotos. Por ahora, si toman una foto con la cámara y desean guardarla para un futuro uso, utilicen el comando Myro, savePicture, como en el siguiente ejemplo:

```
>>> savePicture(p, “office-scene.jpg”)
```

El archivo office-scene.jpg será guardado en la misma carpeta que su carpeta Start Python. También pueden usar savePicture para guardar una serie de imágenes de la cámara y convertirla en una “película” animada (como una imagen gif animada). Esto se ilustra con el ejemplo de abajo.

Realizar la siguiente actividad: En primer lugar, prueben todos los comandos para sacar y guardar fotos. Asegúrense de sentirse cómodos usándolos. Prueben sacar fotos en escala de grises también. Supongan que el robot se ha internado en un lugar donde no pueden verlo pero está aún en un rango de comunicación con la computadora. Ustedes desearían mirar alrededor para ver dónde está. Quizás está en un cuarto nuevo. Pueden pedirle al robot que se dé vuelta y tomar varias fotos que les muestren el entorno. Pueden hacerlo usando una combinación de los comandos rotate y takePicture, como se muestra abajo:

```
while timeRemaining(30):
  show(takePicture())
  turnLeft(0.5, 0.2)
```

Esto significa tomar una foto y luego girar durante 0.2 segundos, repitiendo los dos pasos durante 30 segundos. Si observan la ventana de imagen que aparece, verán las sucesivas vistas del robot. Prueben esto varias veces y observen si pueden contar cuántas imágenes diferentes aparecen. Seguidamente, cambien el comando takePicture para obtener fotos en escala de grises. ¿Pueden contar la cantidad de fotos que ha tomado esta vez? Hay, por supuesto, una manera más sencilla de hacerlo:

Píxeles

Cada imagen está hecha de pequeños elementos de imagen o píxeles. En una imagen color, cada píxel contiene información sobre el color que está compuesta por la cantidad de rojo, verde y azul (RGB). Cada uno de estos valores está en el rango de 0..255 y por lo tanto toma 3 bytes o 24 bits para guardarlo en la información contenida en un solo píxel. Un píxel de color rojo puro tendrá los valores RGB (255, 0, 0). Una imagen en escala de grises, por otro lado, sólo contiene el nivel de gris en un píxel, que puede ser representado por un solo byte (u 8 bits) como un número dentro del rango de 0..255 (donde 0 es negro y 255 es blanco).

```
N = 0
while timeRemaining(30):
   show(takePicture())
   turnLeft(0.5, 0.2)
   N = N + 1
print N
```

Ahora mostrará la cantidad de imágenes que toma. Notarán que es capaz de tomar muchas más imágenes en escala de grises que en color. Esto se debe a que las imágenes en color tienen mucha más información que las grises (ver el texto arriba). Una imagen color de 256x192 requiere 256x192x3 (= 147,456) bytes de datos, mientras que una imagen en escala de grises requiere solamente 256x192 (= 49, 152) bytes. Cuantos más datos se deban transferir del robot a la computadora, más tiempo tardará.

También pueden guardar un GIF animado de las imágenes generadas por el robot usando el comando savePicture, a través de la acumulación de una serie de imágenes en una lista. Esto se muestra abajo:

```
Pics = []
while timeRemaining(30):
   pic = takePicture()
   show(pic)
   Pics.append(pic)
   turnLeft(0.5, 0.2)
savePicture(Pics, “office-movie.gif”)
```

Primero creamos una lista vacía llamada Pics. Luego le anexamos las sucesivas imágenes tomadas por la cámara a la lista. Una vez que se han reunido todas las imágenes, usamos savePicture para guardar el conjunto completo como un GIF animado. Simplemente carguen el archivo en un navegador Web y mostrará las imágenes como en una película.

Hay muchas otras formas interesantes de usar las imágenes de la cámara. En el capítulo 9 exploraremos las imágenes con mayor detalle. Por ahora, veamos los otros sensores del Scribbler.

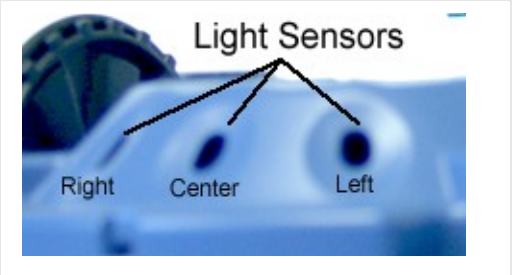
La percepción de la luz

Las siguientes funciones están disponibles para obtener valores de sensores de luz:

`getLight()`: Devuelve una lista que contiene los tres valores de todos los sensores de luz.

`getLight(<POSITION>)` Devuelve el valor actual en el sensor de luz <POSITION>. <POSITION> puede ser uno de 'left' , 'center' , 'right' (izquierda, centro y derecha), o uno de los números 0, 1, 2. Las posiciones 0, 1 y 2 corresponden a los sensores de la izquierda, el centro y la derecha. Por ejemplo:

```
>>> getLight()
[135, 3716, 75]
>>> getLight('left')
135
>>> getLight(0)
135
>>> getLight('center')
3716
>>> getLight(1)
3716
>>> getLight('right')
75
>>> getLight(2)
75
```



Los valores que se reportan desde estos sensores pueden oscilar en el rango de [0..5000] donde los valores bajos implican una luz brillante y los valores altos implican oscuridad. Los valores de arriba fueron tomados con luz ambiental, donde un dedo tapaba completamente al sensor del centro. Por ende, cuanto más oscuro está, más alto es el valor que se reporta. Más adelante, veremos cómo podemos transformar fácilmente estos valores de muy distintas maneras para afectar el comportamiento del robot.

Sería una buena idea usar la función senses para jugar un poco con los sensores de luz y observar sus valores. Prueben mover al robot por el entorno para ver cómo se modifican los valores. Apaguen las lucen en la habitación, o cubran los sensores con los dedos, etc.

Al usar la función getLight sin ningún parámetro, obtienen una lista de tres valores de sensores (izquierda, centro y derecha). Pueden asignarlos a variables individuales de muchas maneras:

```
>>> L, C, R = getLight()
>>> print L
135
>>> Center  = getLight(“center”)
>>> print Center
3716
```

Las variables pueden, por lo tanto, utilizarse de muchas maneras para definir comportamientos del robot. Veremos varios ejemplos de esto en el siguiente capítulo.

La cámara que está en el Fluke también puede ser usada como un tipo de sensor de brillo. Esto se hace promediando los valores de brillo de diferentes zonas de la imagen en la cámara. De alguna manera, pueden considerarlo un sensor virtual. Es decir, no existe físicamente, pero está embebido en la funcionalidad de la cámara. La función getBright es similar a la de getLight por el modo en que puede ser usada para obtener valores de brillo.

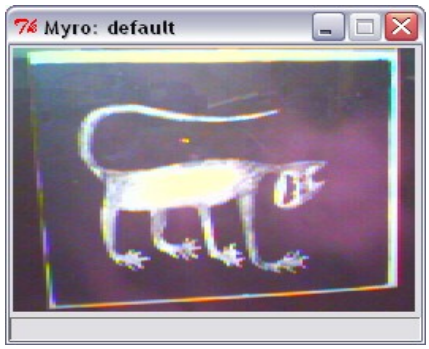
`Getbright():` Devuelve una lista que contiene los tres valores de todos los sensores de luz.

`getBright(<POSITION>):` Devuelve el valor actual en el sensor de luz <POSITION>. <POSITION> puede ser de izquierda, centro, derecha o uno de los números 0, 1, 2. Las posiciones 0, 1 y 2 corresponden a los sensores de izquierda, centro y derecha. Por ejemplo:

```
>>> getBright()
[2124047, 1819625, 1471890]
>>> getBright('left')
2124047
>>> getBright(0)
2124047
>>> getBright('center')
1819625
>>> getBright(1)
1819625
>>> getBright('right')
1471890
>>> getBright(2)
1471890
```

Los valores de arriba pertenecen a la imagen de cámara del póster de Firefox (ver imagen arriba). Los valores que reportan estos sensores pueden variar dependiendo de la vista de la cámara y de los niveles de brillo resultantes en la imagen. Pero notarán que los valores más altos implican segmentos brillantes y los más bajos implican oscuridad. Por ejemplo, aquí hay otra serie de valores basado en la imagen que se muestra a continuación.

```
>>> getBright()
[1590288, 1736767, 1491282]
```



Como podrán observar, es más probable que una imagen oscura produzca valores de brillo más bajos. Pueden ver claramente que el centro de la imagen es más brillante que sus sectores izquierdo o derecho.

También es importante notar las diferencias en la naturaleza de la información que se reporta a través de los sensores getLight y getBright. El primero reporta la cantidad de luz ambiental que percibe el robot (incluyendo la luz sobre el robot). El segundo es un promedio del brillo obtenido de la imagen vista por la cámara. Estos pueden ser usados de muchas maneras, como veremos más adelante.

Realizar la siguiente actividad: El programa que se muestra abajo usa una función de normalización para normalizar los valores del sensor de luz en el rango de [0.0..1.0] relativo a los valores de la luz ambiental. Entonces, los valores de sensores de luz de izquierda y derecha son utilizados para manejar los motores de izquierda y derecha del robot.

```
# registra  un promedio de los valores de luz ambiental
Ambiente = sum(getLight())/3.0

# Esta funcion normailiza los valores de los sensores de luz a valores en el
#rango de 0.0..1.0
def normalize(v):
    if v > Ambiente:
        v = Ambiente

    return 1.0 - v/Ambiente

def main():
    # Run the robot for 60 seconds
```

```
    while timeRemaining(60):
        L, C, R = getLight()
        # motors run proportional to light
        motors(normalize(L), normalize(R))
```

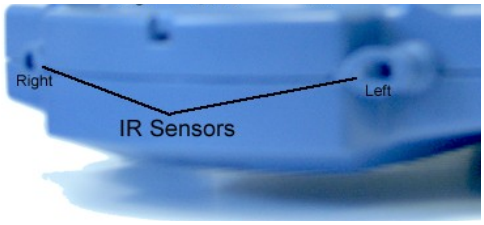
Ejecuten el programa de arriba en su robot Scribbler y observen su comportamiento. Necesitarán una linterna para provocar mejores reacciones. Mientras que el programa esté corriendo, intenten alumbrar uno de los sensores de luz con la linterna (el izquierdo o derecho). Observen el comportamiento. ¿Creen que el robot se está comportando como un insecto? ¿Cuál? Estudien el programa de arriba cuidadosamente. Hay algunas funciones nuevas de Python que discutiremos pronto. También volveremos sobre la idea de hacer que los robots se comporten como insectos en el siguiente capítulo.

Percepción de Proximidad

El Scribbler tiene dos conjuntos de detectores de proximidad. Hay dos sensores infrarrojos IR en el frente del robot y hay tres sensores IR de obstáculos adicionales en el Fluke dongle. Las siguientes funciones están disponibles para obtener valores de los sensores IR del frente:

`getIR():` Devuelve una lista que contiene los dos valores de todos los sensores IR.

`getIR(<POSITION>)` Devuelve el valor actual del sensor IR `<POSITION>`. `<POSITION>` puede ser izquierda o derecha o uno de los números 0, 1. Las posiciones 0 y 1 corresponden a los sensores de izquierda, centro y derecha.



Ejemplos:

```
>>> getIR()
[1, 0]
>>> getIR('left')
1
>>> getIR(0)
1
>>> getIR('right')
0
>>> getIR(1)
0
```

Los sensors IR devuelven ya sea un 1 o un 0. El valor 1 implica que no hay nada en una proximidad cercana frente a ese sensor, y el 0 implica que hay algo frente a él. Estos sensores pueden ser usados para detectar la presencia o ausencia de obstáculos frente al robot. Los sensores IR de izquierda y derecha están ubicados lo suficientemente alejados como para poder detectar obstáculos individuales en cada lado.

Realizar la siguiente actividad: hagan correr la función `senses` y observen los valores de los sensores IR. Ubiquen varios objetos frente al robot y observen los valores de los sensores de

proximidad IR. Tomen su *notebook* y ubíquenla frente al robot a aproximadamente 60cm de distancia. Lentamente muevan la *notebook* más cerca del robot. Observen cómo cambian los valores del sensor de 1 a 0 y luego alejen nuevamente la *notebook*. ¿Pueden averiguar cuán lejos (o cerca) del obstáculo debe estar para ser detectado? Intenten mover la *notebook* de lado a lado. Nuevamente observen los valores de los sensores IR.

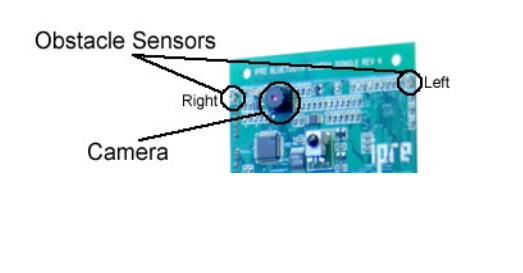
El Fluke tiene un grupo de sensores adicionales. También son sensores IR pero se comportan de manera muy diferente en términos de los tipos de valores de reportan. Las siguientes funciones están disponibles para obtener valores de los sensores IR de obstáculos:

`getObstacle():` Devuelve una lista que contiene los dos valores de todos los sensores IR.

`getObstacle(<POSITION>):` Devuelve el valor actual del sensor IR `<POSITION>`. `<POSITION>` puede ser izquierda, centro o derecha, o uno de los números 0, 1 ó 2. Las posiciones 0, 1 y 2 corresponden a los sensores de la izquierda, el centro y la derecha.

Ejemplos:

```
>>> getObstacle()
[1703, 1128, 142]
>>> getObstacle('left')
1703
>>> getObstacle(0)
1703
>>> getObstacle('center')
1128
>>> getObstacle(1)
1128
>>> getObstacle('right')
142
>>> getObstacle(2)
142
```



Los valores reportados por estos sensores varían del 0 al 7000. Un 0 implica que no hay nada frente al sensor, mientras que un número alto implica la presencia de un objeto. Los sensores a los costados pueden ser usados para detectar la presencia (o ausencia) de paredes a los costados.

Realizar la siguiente actividad: Ubiquen al Scribbler en el piso, enciéndanlo, inicien Python y conéctense a él. También conecten el controlador del game pad e inicien la operación de manejo manual (gamepad()). A continuación, emitan el comando senses para obtener la visualización del sensor en tiempo real. Nuestro objetivo aquí es “meternos en la cabeza del robot” y manejarlo por el entorno sin mirar al robot en ningún momento. También deben resistir la tentación de sacar una foto. Pueden usar la información brindada por los sensores para navegar el robot. Intenten conducirlo a un punto oscuro de la habitación. Intenten conducirlo de manera que en ningún momento se choque contra un objeto. ¿Pueden detectar cuándo hay algo? Si se atasca, ¡traten de maniobrar para sacarlo del atascamiento! Este ejercicio les dará una idea bastante cercana a lo que percibe el robot, al modo en que usa sus sensores, y al rango de comportamiento del que es capaz. Encontrarán que es un ejercicio difícil, pero les dará una buena idea de lo que ocurre en los cerebros de este tipo de robots cuando intentan diseñarlos. Intentaremos volver sobre este escenario a medida que construimos varios programas de robots.

También realicen la siguiente actividad: Prueben el programa de abajo. Es muy similar al programa de arriba que usaba los sensores de luz normalizados.

```
def main():
    # Run the robot for 60 seconds
    while timeRemaining(60):
        L, R = getIR()
        # motors run proportional to IR values
        motors(R, L)
main()
```

Dado que los sensores IR reportan valores 0 ó 1, no necesitan normalizarlos. También observen que estamos poniendo el valor del sensor izquierdo (L) en el motor derecho y el valor del sensor derecho (R) en el motor izquierdo. Ejecuten el programa y observen el comportamiento del robot. Tengan a mano una *notebook* y traten de ubicarla frente al robot. También ubíquenla levemente a la izquierda o a la derecha. ¿Qué sucede? ¿Pueden sintetizar lo que está haciendo el robot? ¿Qué ocurre cuando cambian los valores R y L de los motores?

Pueden ver cómo programas simples como los que vimos arriba pueden resultar ser interesantes estrategias de control automático para robots. También pueden definir comportamientos completamente automatizados e incluso una combinación de comportamientos automatizados y manuales para robots. En el siguiente capítulo exploraremos varios comportamientos de robots. Primero, es hora de que aprendan acerca de las listas es Python.

Las listas en Python

Han visto arriba que varias funciones de sensores devuelven listas de valores. También utilizamos listas para acumular una serie de fotos de la cámara para generar un GIF animado. Las listas son una manera muy útil de coleccionar un grupo de información y Python provee un conjunto de operaciones y funciones útiles que permiten la manipulación de listas. En Python, una lista es una secuencia de objetos. Los objetos pueden ser cualquier cosa: números, letras, *strings*, imágenes, etc. La lista más simple que pueden tener es una lista vacía:

```
>>> []
[]
```

o

```
>>> L = []
>>> print L
[]
```

Una lista vacía no contiene nada. Aquí hay algunas listas que contienen objetos:

```
>>> N = [7, 14, 17, 20, 27]
>>> Ciudades = [“La Plata”, “Rosario”, “Carlos Paz”]
>>> NumerosPrimos = [1,2,3,5]
>>> MiAuto = [“Citroen”, 2007, “Azul”]
```

Como pueden ver arriba, una lista puede ser una colección de cualquier tipo de objeto. Python provee varias funciones útiles que permiten la manipulación de estas listas. Abajo mostraremos algunos ejemplos usando las variables definidas arriba:

```
>>> len(N)
5
>>>len(L)
```

```
0
>>> N + NumerosPrimos
[7, 14, 17, 20, 27, 1, 2, 3, 5]
>>> Ciudades[0]
La Plata
>>> NumerosPrimos[1:3]
[2, 3]
>>> 2 in NumerosPrimos
True
>>> 190 in NumerosPrimos
False
```

Del ejemplo de arriba, pueden ver que la función `len` toma una lista y devuelve el largo o el número de objetos en la lista. Una lista vacía tiene cero objetos en ella. También pueden acceder a elementos individuales en la lista usando la operación de indexación (como en Ciudades[0]). El primer elemento de una lista tiene índice 0 y el último elemento de una lista con `n` elementos tendrá un índice `n-1`. Pueden concatenar las dos listas usando el operador `'+'` para producir una nueva lista. También pueden especificar una franja en la operación de indexación (como en NumerosPrimos[1:3]) para hacer referencia a la sublista que contiene elementos del índice 1 al 2 (uno menor que 3). También pueden formar condiciones `True` / `False` para chequear si el objeto está en una lista o no, usando el operador `in`. Estas operaciones se sintetizan con más detalle al final del capítulo.

Además de las operaciones de arriba, Python también provee varias otras operaciones de listas. Aquí presentamos ejemplos de algunas operaciones de listas útiles: `sort, reverse y append.`

```
>>> NumerosPrimos
[1, 2, 3, 5]
>>> NumerosPrimos.sort()
>>> NumerosPrimos
[1, 2, 3, 5]
>>> NumerosPrimos.reverse()
>>> NumerosPrimos
[5, 3, 2, 1]
>>> NumerosPrimos.append(11)
>>> NumerosPrimos
[5, 3, 2, 1, 11]
```

`sort` reacomoda los elementos en la lista en forma ascendente. `reverse` revierte el orden de los elementos en la lista, y `append` agrega un elemento al final de la lista. Algunas otras operaciones de listas útiles se presentan al final del capítulo. Recuerden que las listas son también secuencias y por lo tanto pueden ser usadas para llevar a cabo repeticiones. Por ejemplo:

```
>>> Ciudades = ["La Plata", "Rosario", "Carlos Paz"]
>>> for ciudad in Ciudades:
        print ciudad

La Plata
Rosario
Carlos Paz
```

La variable ciudad toma valores subsecuentes en la lista Ciudades y las sentencias dentro del loop se ejecutan una vez por cada valor en ciudad. Recuerden que escribimos loops o iteraciones de la siguiente manera:

```
for I in range(5):
    <hacer algo>
```

La función range devuelve una secuencia de números:

```
>>> range(5)
[0, 1, 2, 3, 4]
```

Por lo tanto la variable `I` toma valores en la lista `[0, 1, 2, 3, 4]` y, como en el ejemplo de abajo, el *loop* se ejecuta 5 veces:

```
>>> for I in range(5):
    print I

0
1
2
3
4
```

También recuerden que los *strings* son secuencias. Esto significa que el *string:*

```
ABC = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

es una secuencia de 26 letras. Pueden escribir un *loop* que tome cada letra individual del *string* y lo diga en voz alta de la siguiente manera:

```
>>> for letra in ABC:
      speak(letra)
```

También hay algunas funciones útiles que convierten *strings* en listas. Digamos que tenemos un *string* que contiene el enunciado:

```
>>> oracion = "Mi planta de naranja lima"
```

Pueden convertir el *string* de arriba en palabras individuales usando la operación `split`.

```
>>> oracion.split()
['Mi', 'planta', 'de', 'naranja', 'lima']
```

Observando la lista de operaciones presentadas arriba, revisen algunos de los ejemplos de percepción de los inicios de este capítulo. Estaremos usando listas en muchos ejemplos de lo que queda del texto. Por ahora, volvamos al tema de la percepción.

¿Percepción Extrasensorial?

Han visto muchas maneras de adquirir información sensorial usando los sensores del robot. Además del robot en sí mismo, deberían ser conscientes de que su computadora también tiene varios "sensores"o dispositivos para adquirir todo tipo de datos. Por ejemplo, ya han visto cómo, usando la función de input, pueden hacer input de algunos valores en sus programas Python:

```
>>> N = input("Ingresa un numero: ")
Ingresa un numero: 42
>>> print N
42
```

Sin duda, hay otras maneras en que pueden adquirir información para sus programas Python. Por ejemplo, pueden obtener datos de un archivo en su carpeta. En el capítulo 1 también han visto cómo pudieron controlar el robot usando el controlador de *game pad*. El *game pad* fue efectivamente enchufado en su computadora y funcionó como un dispositivo de entrada. Adicionalmente, su computadora está seguramente conectada a Internet, a través de la cual pueden acceder a muchas páginas Web. También es posible obtener el contenido de cualquier página Web usando Internet. Tradicionalmente, en ciencias de la computación se hace referencia a esto como un proceso de entrada o *input*. Desde esta perspectiva, obtener información sensorial de un robot es sólo un tipo de entrada. Dado que tenemos a nuestra disposición todas las facilidades de entradas provistas por la computadora, podemos de la misma manera obtener entradas fácilmente de cualquiera de las modalidades y combinarlas con el comportamiento de robot que deseemos. Si consideran a esto como *percepción extrasensorial* o no es una cuestión de opinión. De todas maneras, el poder obtener entradas de diversos tipos de fuentes puede conducir a ciertos aplicativos de computadora y de robot muy interesantes y útiles.

Controladores Game Pad

El controlador *game pad* que usaron en el capítulo 1 es un dispositivo típico que provee facilidades de interacción al jugar con juegos de computadora. Estos dispositivos han sido lo suficientemente estandarizados como para que, al igual que con el *mouse* de una computadora o un teclado, ustedes puedan comprarlos en un negocio y conectarlo a un puerto USB de su computadora. Myro provee algunas funciones de entradas muy útiles que pueden ser utilizadas para obtener input del controlador *game pad*. Los *game pads* vienen de todos los sabores y configuraciones con números variados de botones, ejes y otros dispositivos incluidos. En los ejemplos de abajo, nos restringiremos al *game pad* básico que se muestra en la figura.

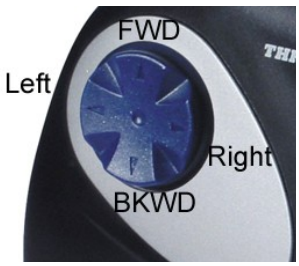
El game pad básico tiene ocho botones (numerados del 1 al 8 en la imagen) y un eje controlador (ver imagen). Los botones pueden ser presionados o soltados) on/off), lo cual se representa con 1 (para on – *encendido*) y 0 (para off –*apagado*). El eje puede ser presionado hacia muchas orientaciones distintas, representadas por un par de valores (para el eje-x y el eje-y) que van del –1.0 al 1.0, donde [0.0, 0.0] representa a ninguna actividad en el eje. Dos funciones Myro son provistas para acceder a los valores de los botones y el eje:

```
getGamepad(<device>)
getGamepadNow(<device>)
```

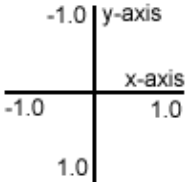
devuelve los valores que indican el estado del `<device>` (dispositivo) especificado. `<device>` puede ser `"axis"` o `"button"`.

La función `getGamepad` regresa solamente después de que `<device>` ha sido utilizado por el usuario. Esto significa que espera a que el usuario presione o utilice el dispositivo y luego devuelve los valores asociados con ese dispositivo al instante. `getGamepadNow` no espera y simplemente devuelve el estado del dispositivo enseguida. Aquí hay algunos ejemplos:

Control de eje del Game Pad



Eje del Game Pad




```
>>> getGamepadNow("axis")
[0.0, 0.0]
>>> getGamepad("axis")
[0.0, -1.0]
>>> getGamepadNow("button")
[0, 0, 0, 0, 1, 1, 0, 0]
```

Ambos, `getGamepad` y `getGamepadNow`, devuelven el mismo grupo de valores: los valores del eje (*axis*) son devueltos en forma de lista [x-axis, y-axis] (ver imagen para orientarse), y los valores son devueltos como una lista de 0 y 1. El primer valor en la lista es el estado del botón #1, seguido por 2, 3 y así sucesivamente. (Ver la imagen arriba con la numeración de los botones).

Realizar la siguiente actividad: Conecten el controlador *game pad* a su computadora, inicien Python e importen el módulo Myro. Prueben los comandos del *game pad* de arriba y observen los valores. Aquí hay otra manera de comprender mejor la operación del *game pad* y las funciones del mismo.

```
while timeRemaining(30):
    print getGamepad("button")
```

Prueben diferentes combinaciones de botones. ¿Qué sucede cuando presionan más de un botón? Repitan lo de arriba para el control del eje y observen los valores que devuelve (tengan a mano el diagrama del eje para poder orientarse).

El controlador *game pad* puede ser usado para todo tipo de fines interactivos, especialmente para el control del robot, así como también para escribir juegos de computadoras (ver capítulo X). Escribamos un controlador de robot basado en el *game pad*. Ingresen el siguiente programa y háganlo correr:

```
def main():
    # A simple game pad based robot controller
    while timeRemaining(30):
        X, Y = getGamePadNow("axis")
        motors(X, Y)
```

El programa de arriba correrá por 30 segundos. En ese tiempo repetidamente tomará muestras del controlador del eje y dado que esos valores están en el rango –1.0..1.0, los usará para manejar los motores. Cuando ejecuten ese programa, observen cómo se mueve el robot respondiendo a la presión de distintas partes del eje. ¿Los movimientos del robot corresponden a las direcciones que se muestran en la foto del *game pad* de la página anterior? Intenten modificar el comando de `motors` a `move` (recuerden que el movimiento requiere dos valores: traslación y rotación). ¿Cómo se comporta con respecto a los ejes? Intenten modificar el comando a `move(-X, -Y)`. Observen el comportamiento.

Como pueden ver a partir del ejemplo sencillo de arriba, es fácil combinar las entradas de un *game pad* para controlar al robot. ¿Pueden expandir el programa de arriba para que se comporte exactamente como la función del controlador `gamepad` que usaron en el capítulo 1? (ver ejercicio YY).

La World Wide Web

Si su computadora está conectada a Internet, también pueden usar las facilidades Python para acceder al contenido de cualquier página Web y utilizarlo como input de su programa. Las páginas Web están escritas usando lenguaje *markup* como HTML, por lo tanto al acceder al contenido de una página Web, obtendrán el contenido con los *markups* incluidos. Es esta sección les mostraremos cómo acceder al contenido de una simple página Web e imprimirlo. Más adelante, veremos cómo podrían usar la información que contiene para realizar futuros procesamientos.

Accedan a un navegador Web y entren a la página:

```
http://www.fi.edu/weather/data/jan07.txt
```

Esta página Web está *hosteada* por el Instituto Franklin de Filadelfia y contiene el registro diario de datos meteorológicos de Filadelfia para enero 2007. Pueden navegar desde la dirección de arriba a otras páginas del sitio y mirar los datos del clima de otras fechas (¡los datos llegan hasta 1872!). Más abajo, les mostraremos cómo, usando la librería Python llamada `urllib`, pueden acceder fácilmente al contenido de cualquier página Web. La librería `urllib` provee una función útil llamada `urlopen`; usando esta función, pueden acceder a cualquier página de Internet de la siguiente manera:

```
>>> from urllib import *
>>> Data = urlopen("http://www.fi.edu/weather/data/jan07.txt")
>>> print Data.read()

January 2007

Day   Max   Min   Liquid      Snow  Depth

1     57    44    1.8   0     0

2     49    40    0     0     0
```

```
3      52     35     0      0      0

…      …      …      …      …      …

31     31     22     0      0      0

#days 31

Sum    1414   1005   4.18   1.80   1.10
```

Los siguientes comandos son importantes:

```
>>> Data = urlopen("http://www.fi.edu/weather/data/jan07.txt")
>>> print Data.read()
```

El primer comando usa la función `urlopen` (la cual es importada del `urllib`) para establecer una conexión entre el programa y la página Web. El segundo comando enuncia un `read` (leer) para leer de esa conexión. Lo que sea leído de esa página se imprime como resultado del comando print.

Algo más acerca de las funciones Python

Antes de que prosigamos, sería bueno refrescar la memoria sobre la escritura de comandos y funciones Python. En el capítulo 2 aprendimos que la sintaxis básica para definir nuevos comandos/funciones es:

```
def <FUNCTION NAME>(<PARAMETERS>):
    <SOMETHING>
    ...
    <SOMETHING>
```

El módulo Myro les provee varias funciones útiles (`forward`, `turnRight`, etc.) que permiten un fácil control de los comportamientos básicos del robot. Además, usando la sintaxis de arriba, aprendieron a combinar estos comportamientos básicos para generar comportamientos más complejos (como `wiggle`, `yoyo`, etc.). Usando parámetros, pueden personalizar aún más el comportamiento de las funciones proveyendo diferentes valores para los parámetros (por ejemplo, `forward(1.0)` moverá el robot más rápido que `forward(0.5)`). También deberán notar una diferencia crucial entre los comandos de movimiento como `getLight` o `getStall, etc.` Los comandos sensoriales también siempre devuelven un valor donde sea que se hayan enunciado. Esto significa:

```
>>> getLight('left')
221
>>> getStall()
0
```

Los comandos que devuelven un valor al ser invocados se llaman funciones, dado que de hecho se comportan de manera muy similar a las funciones matemáticas. Ninguno de los comandos de

movimiento devuelve un valor, pero son útiles de otras maneras. Por ejemplo, hacen que el robot haga algo. En cualquier programa, en general se necesitan ambas funciones: aquellas que hacen algo pero no devuelven nada como resultado; y aquellas que hacen algo y devuelven un valor. En Python todas las funciones devuelven un valor. Pueden ver la utilidad de tener estos dos tipos de funciones de los ejemplos que han visto hasta ahora. Las funciones son una parte integral o crítica de cualquier programa y parte del aprendizaje de ser un buen programador es aprender a reconocer las abstracciones que pueden ser empaquetadas en funciones individuales (como `dibujoPoligono` o `giro`) que pueden ser usadas una y otra vez.

Escribir funciones que devuelven valores

Python provee una sentencia de devolución (`return`) que pueden usar dentro de una función para devolver o retornar los resultados de una función. Por ejemplo:

```
def triple(x):
   # Retorna x*3
   return x * 3
```

La función de arriba puede ser usada igual que las que han venido usando:

```
>>> triple(3)
9
>>> triple(5000)
15000
```

La forma general de la sentencia `return es:`

```
return <expression>
```

Es decir, la función en la cual se encuentra esta sentencia devolverá el valor de <expression>. Por lo tanto, en el ejemplo de arriba, la sentencia return devuelve el valor de la expresión 3*x, tal como se muestra en las invocaciones de ejemplo. Otorgándole distintos valores al parámetro x, la función simplemente lo triplica. Esta es la idea que usamos en la normalización de valores de sensores de luz en los ejemplos anteriores en los que definimos la función normalizar para tomar los valores de los sensores de luz y normalizarlos al rango de 0.0..1.0 relativo a los valores de luz ambiental observados:

```
# This function normalizes light sensor values to 0.0..1.0
def normalize(v):
    if v > Ambient:
        v = Ambient

    return 1.0 – v/Ambient
```

Al definir la función de arriba, también estamos usando una nueva sentencia Python: la sentencia-if. Esta sentencia permite toma de decisiones simples dentro de programas de computadora. La forma más simple de una sentencia-if tiene la siguiente estructura:

```
if <CONDITION>:
   <do something>
   <do something>
   ...
```

Es decir, si la condición especificada por <CONDITION> es True, entonces lo que sea que esté especificado en el cuerpo de la sentencia-if se lleva a cabo. En el caso en que <condition> sea False, todos las sentencias para el comando if se saltean.

Las funciones pueden tener cero o más sentencias-return. Algunas de las funciones que han escrito, como wiggle, no tienen ninguno. Técnicamente, cuando una función no tiene una sentencia-return que devuelva un valor, la función devuelve un valor especial llamado None (ninguno). Esto ya está definido en Python.

Las funciones, como han visto, pueden ser usadas para empaquetar cómputos útiles y pueden ser utilizadas una y otra vez en muchas situaciones. Antes de concluir esta sección, les daremos otro ejemplo de una función. Recuerden, del capítulo 4, el comportamiento del robot que permite que el robot se mueva hacia delante hasta chocarse contra una pared. Uno de los fragmentos de programas que usamos para especificar este comportamiento se muestra abajo:

```
while not getStall():
   forward(1)
stop()
```

En el ejemplo de arriba, estamos usando el valor devuelto por `getStall` para ayudarnos a tomar la decisión de continuar avanzando o parar. Tuvimos suerte de que el valor devuelto es directamente utilizable para nuestra toma de decisión. A veces, se debe hacer un poco de interpretación de los valores del sensor para averiguar exactamente lo que el robot está percibiendo. Podrán observar esto en el caso de los sensores de luz. Aunque las sentencias de arriba son fáciles de leer, podemos mejorarlas escribiendo una función llamada `stuck()` de la siguiente manera:

```
def stuck():
   # Is the robot stalled?
   # Returns True if it is and False otherwise.

   return getStall() == 1
```

La función de arriba es bastante simple, dado que `getStall` ya nos brinda un valor utilizable (`0`/`False` o or `1`/`True`). Pero ahora, si usáramos `stuck` para escribir el comportamiento del robot, se leería así:

```
while not stuck():
   forward(1)
stop()
```

Como pueden observar, se lee mucho mejor. En este sentido, programar se parece mucho a escribir. Como en la escritura, hay varias maneras de expresar las ideas en palabras. Algunas son mejores y más legibles que otras. Algunas son directamente poéticas. De la misma manera, en programación, la expresión de algo en un programa puede ser formulada de distintas maneras, algunas mejores y más legibles que otras. La programación no se trata sólo de funcionalidad, *puede* haber poesía en el modo en que uno escribe un programa.

Resumen

En este capítulo han aprendido todo acerca de la obtención de datos del sistema perceptivo del robot para percepción visual (fotos), percepción de luz y de proximidad. El Scribbler provee un rico conjunto de sensores que pueden ser usados para diseñar interesantes comportamientos del robot. También aprendieron que la percepción es equivalente a la operación de entrada básica en una computadora. También han aprendido cómo obtener input de un *game pad*, de la World Wide Web, y de archivos de datos. Los programas pueden ser escritos para usar creativamente las modalidades de entradas disponibles para definir comportamientos de robot, juegos de computadora, e incluso para el procesamiento de datos. En el resto del libro aprenderán cómo escribir programas que usan estas modalidades de input de muy distintas maneras.

Revisión Myro

`getBright()`
Devuelve una lista que contiene los tres valores de todos los sensores de luz.

`getBright(<POSITION>)`
Devuelve el valor actual del sensor de luz `<POSITION>`. `<POSITION>` puede ser izquierda, centro o derecha o uno de los números 0, 1, 2.

`getGamepad(<device>)`
`getGamepadNow(<device>)`
Devuelve los valores que indican el estado del `<device>` especificado. `<device>` puede ser `"axis" o "button". La función getGamepad` espera un evento antes de devolver valores. `getGamepadNow` inmediatamente devuelve el estado actual del dispositivo.

`getIR()`
Devuelve una lista que contiene los dos valores de todos los sensores IR.

`getIR(<POSITION>)`
Devuelve el valor actual en el sensor IR `<POSITION>, <POSITION>` puede ser izquierda o derecha o uno de los números 0, 1.

`getLight()`
Devuelve una lista que contiene los tres valores de todos los sensores de luz

`getLight(<POSITION>)`
Devuelve el valor actual en el sensor de luz `<POSITION>. <POSITION>` puede ser izquierda, centro, derecha o uno de los números 0, 1, 2. Las posiciones 0, 1 y 2 corresponden a los sensores de la izquierda, el centro y la derecha.

`getObstacle()`
Devuelve una lista que contiene los dos valores de los sensors IR.

`getObstacle(<POSITION>)`
Devuelve el valor actual en el sensor IR `<POSITION>. <POSITION> puede ser` izquierda, centro o derecha, o uno de los números 0, 1, 2.

`savePicture(<picture>, <file>)`
`savePicture([<picture1>, <picture2>, …], <file>)`
Guarda la foto en el archivo especificado. La extensión del archivo debe ser `".gif"` o `".jpg"`. Si el primer parámetro es una lista de fotos, el nombre del archivo debe tener la extensión `".gif"` y se creará un GIF animado usando las fotos provistas.

`senses()`
Muestra los valores de los sensores del Scribbler en una ventana. La vista se actualiza cada segundo.

`show(<picture>)`
Muestra la foto en una ventana. Pueden clickear el mouse izquierdo en cualquier lugar de la pantalla para mostrar los valores (x, y) y (r, g, b) del punto en la barra de estado de la ventana.

```
takePicture()
takePicture("color")
takePicture("gray")
```
Toma una foto y devuelve un objeto foto. Cuando no se especifican parámetros, la foto es color.

Revisión Python

```
if <CONDITION>:
   <statement-1>
  ...
   <statement-N>
```

Si la condición se evalúa como True, todos los enunciados se llevan a cabo. De lo contrario, todos los enunciados se saltean.

`return <expression>`
Puede ser usada dentro de cualquier función para devolver el resultado de la función.

`<string>.split()`
Divide a `<string>` en una lista.

`urlopen(<URL>)`
Establece una conexión *stream* con la <URL>. Esta función se importa del módulo Python `urlopen`.

`<stream>.read()`
Lee los contenidos enteros del `<stream>` como un *string*.

Listas:
[] es una lista vacía..

`<list>[i]`
Devuelve el elemento `iceavo` de la lista. La indexación comienza de 0.

`<value> in <list>`
Devuelve `True` si `<value>` está en la `<list>`, si no, `False`.

`<list1> + <list2>`
Concatena `<list1>` y `<list2>`.

`len(<list>)`
Devuelve el número de elementos en una lista.

`range(N)`
Devuelve una lista de números de 0...N

`range(N1, N2)`
Devuelve una lista de números comenzando desde N1..(N2-1)

`range(N1, N2, N3)`
Devuelve una lista de números iniciada en N1 y menor que N3, incrementándose por N3.

`<list>.sort()`
Distribuye la `<list>` en orden ascendente.

`<list>.append(<value>)`
Agrega el `<value>` al final de la `<list>`.

`<list>.reverse()`
Da vuelta los elementos de la lista.

Ejercicios

1 Además de texto, el comando speak también puede vocalizar números. Prueben `speak(42)` y también `speak(3.1419)`. Prueben algún número entero muy largo, como `speak(4537130980)`. ¿Cómo lo vocaliza? ¿Pueden averiguar el límite para la vocalización numérica?

2. La cámara Fluke devuelve fotos con 256x192 (= 49, 152) píxeles. Las típicas cámaras digitales se caracterizan por lo general por el número de píxeles que usan para sus imágenes. Por ejemplo, 8 mega píxeles. ¿Qué es un mega píxel? Realicen un poco de investigación por la Web para averiguar el resultado.

3. Todas las imágenes tomadas y guardadas usando la cámara Fluke también pueden ser mostradas en una página Web. En su navegador favorito, usen la opción de File Open -*abrir archivo*- (bajo el menú de File) y naveguen a la foto guardada por el Scribbler. Selecciónenla y visualícenla a través de la ventana del navegador. Inténtenlo con un archivo GIF animado.

4 ¿Qué produce la siguiente expresión?:

```
L = [5]*10
```

Ingrésenla como un comando Python y observen el valor de `L`.

5. Modifiquen el programa de input del *game pad* en este capítulo para hacer que el controlador del eje se comporte de tal manera que cuando se presiona la parte superior del eje, el robot se adelante, y cuando se presiona la parte de abajo, vaya hacia atrás.