**Sound**

|  |
| --- |
| **Play a standard beep**  Use   Sound.beep()  Examples:  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  # needed for Sound  **Sound.beep()**  or, to **cause the program to wait** until the beep has completed before continuing:  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  Sound.beep()**.wait()**  **Play a SINGLE tone**  To play a SINGLE tone of given frequency (Hz) and duration (milliseconds), use  Sound.tone(*frequency, duration*)  Example:  #!/usr/bin/env python3  from ev3dev.ev3 import \*  # Play a SINGLE 1500Hz tone for 2 seconds  # Wait for the tone to finish playing before continuing  **Sound.tone(1500, 2000).wait()**  **Play a SEQUENCE of tones**  To play a SEQUENCE of tones, use  Sound.tone(*tone\_sequence*)  The tone\_sequence parameter is a list of tuples, where each tuple contains up to three numbers. You should know by now that a tuple is like a list except that a list can be modified and a tuple cannot. The first number is frequency in Hz, the second is duration in milliseconds, and the third is delay in milliseconds between this and the next tone in the sequence.  #!/usr/bin/env python3  # so that script can be run from Brickman  from ev3dev.ev3 import \*  # needed for Sound  # Play a 3000Hz tone for 2 seconds and then wait 0.4 seconds  # before playing the next tone, 800Hz for 1.8 seconds  # followed by a 2 second delay  # Wait for the sequence to finish playing before continuing  **Sound.tone([(3000, 2000, 400),(800, 1800, 2000)]).wait()**  **Play a WAV sound file**  To play a WAV sound file, use     Sound.play(*wav\_file*)  You cannot use mp3 files with the Sound.play command, nor can you use Lego sound files in the RSF or RSO format (but see below).  A good source for free WAV sound files is [**Wavsource.com**](http://wavsource.com/). I did have a problem with a WAV file that I downloaded from there though - I got an error message 'underrun' when I ran my script, which suggests that the sound file was damaged. If you get the same message then delete that sound file and try another. A sample wav file 'cat\_yowl.wav' from wavsource.com can be downloaded using the link at the bottom of this page. I suggest you make a directory called 'sounds' in your 'robot' directory. If you're using MobaXTerm free edition it's easy to make a new directory and upload a file to it using the appropriate buttons. Upload the 'cat\_yowl.wav' file into your sounds directory and run the following script (assumed to be in your robot directory):  #!/usr/bin/env python3  from ev3dev.ev3 import \*  from time import sleep  Sound.play('sounds/cat\_yowl.wav').wait()  Sound.beep()  sleep(4)  Sound.play('sounds/cat\_yowl.wav')  Sound.beep()  sleep(4)  Notice that the first cat yowl plays *before*the beep sound but the second plays *at the same time as* (or even *after*) the beep because there is no .wait() to make program execution pause until the sound has finished playing before continuing.  I said above that Sound.play() can only play sounds that are in the WAV format and therefore cannot play Lego sounds in the RSF or RSO format. I have converted all the Lego sounds into WAV format and made them available via a link at the bottom of this page. All the sounds are in the same folder, not separated into different folders as in EV3-G. All the scripts on this site that make use of these sounds assume that you have put them all in a 'sounds' folder inside the 'robot' folder. Therefore the address of the sounds will be /home/robot/sounds/ . This is similar to the way my scripts assume that you have placed all the Lego images in BMP format in a folder called 'pics' in the 'robot' folder. See [**this page**](https://sites.google.com/site/ev3python/learn_ev3_python/screen).  **Text-to-Speech**  Use  Sound.speak(*text*)  Example at the bottom of the combined example below.  **Combined Example**  An example of a program to demonstrate playing beeps, tones and text-to-speech:  #!/usr/bin/env python3  # so that script can be run from Brickman  from time import sleep   # needed for sleep  from ev3dev.ev3 import \*  # needed for Sound  #play a standard beep  Sound.beep()  sleep(2)  # pause for 2 seconds  # Play a SINGLE 1500 Hz tone for 2 seconds  # Wait for the tone to finish playing before continuing  Sound.tone(2000, 1500).wait()  sleep(2)  # pause for 2 seconds  # To play a SEQUENCE of tones, use a LIST of TUPLES  # The third argument in each tuple is the delay in ms  # before the next tone is played.  # Play a 3000 Hz tone for 2 seconds, then wait 0.4 seconds  # before playing the next tone, 800Hz for 1.8 seconds  # followed by a 2 second delay  # Wait for the sequence to finish playing before continuing  Sound.tone([(3000, 2000, 400),(800, 1800, 2000)]).wait()  # Play a 500 Hz tone for 1 second and then wait 0.4 seconds  # before playing the next tone  # Play the tone three times  Sound.tone([(500, 1000, 400)] \* 3).wait()  #text to speech  Sound.speak('Hello, my name is E V 3!').wait() |

**Sonar**

|  |
| --- |
| **Reproduzca un pitido estándar**  Use    Sound.beep ()  Ejemplos:  #! / usr / bin / env python3  # para que el script pueda ejecutarse desde Brickman  de ev3dev.ev3 import \*  # needed for Sound  **Sound.beep ()**  o, para **hacer que el programa espere** hasta que el pitido se haya completado antes de continuar:  #! / usr / bin / env python3  # para que el script pueda ejecutarse desde Brickman  de la importación ev3dev.ev3 \*  Sound.beep () **.wait ()**  **Juega un tono SIMPLE**  Para reproducir un tono SIMPLE de dar n frecuencia (Hz) y duración (milisegundos), use  Sonar. tono ( *frecuencia, duración*)  Ejemplo:  #! / usr / bin / env python3  de la importación ev3dev.ev3 \*  # Reproduce un tono SINGLE de  1500 Hz  durante 2 segundos  # Espere a que el tono termine de reproducirse antes de continuar  **Sound.tone (1500, 2000) .wait ()**  **Juega una SECUENCIA de tonos**  Para reproducir una SECUENCIA de tonos, use  Sonar. tono ( *tono\_secuencia* )  El parámetro tone\_sequence es una lista de tuplas, donde cada tupla contiene hasta tres números. Debería saber ahora que una tupla es como una lista, excepto que una lista puede modificarse y una tupla no. El primer número es la frecuencia en Hz, el segundo es la duración en milisegundos, y el tercero es el retraso en milisegundos entre este y el siguiente tono de la secuencia.  #! / usr / bin / env python3  # para que el script pueda ejecutarse desde Brickman  de ev3dev.ev3 import \* # needed for Sound  # Reproduce un  tono de 3000 Hz  durante 2 segundos y luego espera 0.4 segundos  # antes de reproducir el siguiente tono, 800Hz durante 1.8 segundos  # seguido de un retraso de 2 segundos  # Espere a que la secuencia termine de reproducirse antes de continuar  **Sound.tone ([(3000, 2000, 400), (800, 1800, 2000)]). Wait ()**  **Reproduce un archivo de sonido WAV**  Para reproducir un archivo de sonido WAV, use      Sonido. play ( *wav\_file*)   No puede usar archivos mp3 con el  comando  Sound.play , ni puede usar archivos de sonido Lego en el formato RSF o RSO (pero vea más abajo).  Una buena fuente de archivos de sonido WAV gratuitos es  [**Wavsource.com**](http://wavsource.com/) . Sí, tuve un problema con un archivo WAV que descargué de allí. Obtuve un mensaje de error 'underrun' cuando ejecuté mi script, lo que sugiere que el archivo de sonido estaba dañado. Si obtiene el mismo mensaje, elimine ese archivo de sonido y pruebe con otro. Se puede descargar un archivo de muestra wav 'cat\_yowl.wav' de wavsource.com usando el enlace al pie de esta página. Te sugiero que hagas un directorio llamado 'sonidos' en tu directorio 'robot'. Si está utilizando la edición gratuita de MobaXTerm, es fácil crear un nuevo directorio y cargar un archivo usando los botones correspondientes. Suba el archivo 'cat\_yowl.wav' en su directorio de sonidos y ejecute el siguiente script (se supone que está en su directorio de robots):  #!/usr/bin/env python3  from ev3dev.ev3 import \*  from time import sleep  Sound.play('sounds/cat\_yowl.wav').wait()  Sound.beep()  sleep(4)  Sound.play('sounds/cat\_yowl.wav')  Sound.beep()  sleep(4)  Observe que el primer aullido de gato se reproduce *antes* del pitido, pero el segundo toca *al mismo tiempo* (o incluso *después* ) el pitido porque no hay .wait () para hacer que la ejecución del programa se detenga hasta que el sonido haya terminado antes de continuar.  Dije anteriormente que Sound.play () solo puede reproducir sonidos que están en formato WAV y, por lo tanto, no puede reproducir sonidos de Lego en el formato RSF o RSO. He convertido todos los sonidos de Lego en formato WAV y los he hecho disponibles a través de un enlace en la parte inferior de esta página. Todos los sonidos están en la misma carpeta, no separados en carpetas diferentes como en EV3-G. Todos los scripts en este sitio que hacen uso de estos sonidos suponen que los has puesto todos en una carpeta de 'sonidos' dentro de la carpeta 'robot'. Por lo tanto, la dirección de los sonidos será  / home / robot / sounds /. Esto es similar a la forma en que mis scripts suponen que ha colocado todas las imágenes de Lego en formato BMP en una carpeta llamada 'pics' en la carpeta 'robot'. Vea [**esta página**](https://sites.google.com/site/ev3python/learn_ev3_python/screen) .  **Texto a voz**  Use   Sound.speak ( *texto* )  Ejemplo en la parte inferior del ejemplo combinado a continuación.  **Ejemplo combinado**  Un ejemplo de un programa para demostrar pitidos, tonos y texto a voz en reproducción:  #! / usr / bin / env python3  # para que el script pueda ejecutarse desde Brickman  desde el momento de importar el sueño    # necesario para dormir  de ev3dev.ev3 import \* # needed for Sound  # reproducir un pitido estándar  Sound.beep ()  sleep (2) # pausa por 2 segundos  # Reproduce un tono SENCILLO de  1500 Hz  durante 2 segundos  # Espere a que el tono termine de reproducirse antes de continuar  Sound.tone (2000, 1500) .wait ()  sleep (2) # pausa por 2 segundos  # Para reproducir una SECUENCIA de tonos, use una LISTA de TUPLES  # El tercer argumento en cada tupla es el retraso en ms  # antes de que se reproduzca el siguiente tono.  # Reproduce un  tono de 3000 Hz  durante 2 segundos, luego espera 0,4 segundos  # antes de reproducir el siguiente tono, 800Hz durante 1.8 segundos  # seguido de un retraso de 2 segundos  # Espere a que la secuencia termine de reproducirse antes de continuar  Sound.tone ([(3000, 2000, 400), (800, 1800, 2000)]). Wait ()  # Reproduce un  tono de 500 Hz  durante 1 segundo y luego espera 0.4 segundos  # antes de reproducir el siguiente tono  # Reproduce el tono tres veces  Sound.tone ([(500, 1000, 400)] \* 3) .wait ()  #texto a voz  Sound.speak ('Hola, mi nombre es EV 3!'). Wait () |