**Buttons**

|  |
| --- |
| The official documentation is [**HERE**](http://python-ev3dev.readthedocs.io/en/latest/other.html#button)**.**  Here are some short scripts demonstrating the use of the Button() class.   * If you run the scripts from the command line then the Brickman interface will continue to be displayed and to respond while the script is running. The print command will print to the terminal. A script can be forced to end with Ctrl-C. * If you run the scripts from Brickman's file browser then the EV3's screen will be cleared so that the script can print to it. The Brickman interface will reappear when the script ends. The script can be forced to end by long-pressing the Backspace button.   Currently, for unknown reasons, these scripts work for me only from the command line - they close immediately if run from Brickman.  **Press any button to exit**  The loop checks whether any button is pressed and if so beeps and exits the script. The script checks the button state every 0.01 second.  #!/usr/bin/env python3  from ev3dev.ev3 import \*  from time import sleep  **btn = Button()**  while True:  **if btn.any()**:    # Checks if any button is pressed.          Sound.beep().wait()  # Wait for the beep to finish.          exit()  # Stop the program.      else:          sleep(0.01)  # Wait 0.01 second  Here's a much neater version of the above script:  #!/usr/bin/env python3  from ev3dev.ev3 import \*  from time import sleep  **btn = Button()**  while **btn.any()==False**: # While no button is pressed.          sleep(0.01)  # Wait 0.01 second  Sound.beep().wait()  **React to button presses and releases**  In this script's loop, the highlighted command btn.process() checks for any change in the state of the buttons. If it detects a change then it triggers the corresponding 'events'. For example, if it detects that the left button has just been pressed then it triggers a 'left button state change' event and a 'button change' event. It also assigns a value of True to the parameter state if the button is pressed and a value of False if the button is released. Event handlers respond to the events. For example, if the left button is pressed then the 'left button state change' event will trigger the highlighted on\_left event handler which will call the function 'left' (the function does not have to have this name).  #!/usr/bin/env python3  from ev3dev.ev3 import \*  from time import sleep  btn = Button()  # Do something when state of any button changes:    def left(state):      if state:          print('Left button pressed')      else:          print('Left button released')    def right(state):  # neater use of 'if' follows:      print('Right button pressed' if state else 'Right button released')    def up(state):      print('Up button pressed' if state else 'Up button released')    def down(state):      print('Down button pressed' if state else 'Down button released')    def enter(state):      print('Enter button pressed' if state else 'Enter button released')    def backspace(state):      print('Backspace button pressed' if state else 'Backspace button released')    **btn.on\_left** = left  **btn.on\_right** = right  **btn.on\_up** = up  **btn.on\_down** = down  **btn.on\_enter** = enter  **btn.on\_backspace** = backspace  while True:  # This loop checks buttons state continuously,               # calls appropriate event handlers  **btn.process()** # Check for currently pressed buttons.      # If the new state differs from the old state,      # call the appropriate button event handlers.      sleep(0.01)  # buttons state will be checked every 0.01 second  # If running this script via SSH, press Ctrl+C to quit  # if running this script from Brickman, long-press backspace button to quit  **Obtain a list of changed button states**  This script has similar functionality to the previous one but uses the on\_change(changed\_buttons) event handler to achieve a much more concise script. Press Ctrl-C (command line) or long-press the backspace button (Brickman) to end the script.  #!/usr/bin/env python3  from ev3dev.ev3 import \*  from time import sleep  btn = Button()  **def change(changed\_buttons):   # changed\_buttons is a list of**  **# tuples of changed button names and their states.**  **print('These buttons changed state: ' + str(changed\_buttons))**  **btn.on\_change = change**  while True:  # This loop checks buttons state  # continuously and calls appropriate event handlers  **btn.process()**      sleep(0.01)  **Check button states**  The previous script showed how process() and its corresponding event handlers can be used to detect and respond to CHANGES in state. There are also functions (down, left, enter etc) which simply detect the STATE of the buttons. The script below checks and displays the state of the left button once per second. You can end the script with a one-second press on the Backspace button.  #!/usr/bin/env python3  from ev3dev.ev3 import \*  from time import sleep  btn = Button()  **while not btn.backspace**:      print(**btn.left**)      sleep(1)  **Other button functions**  The following script demonstrates two more functions: buttons\_pressed and check\_buttons(). Every second a list of the pressed buttons is printed and you can end the script by pressing ONLY the left and right buttons simultaneously.  #!/usr/bin/env python3  from ev3dev.ev3 import \*  from time import sleep  btn = Button()  while True:      print(**btn.buttons\_pressed**)      if **btn.check\_buttons(buttons=['left','right'])**:          exit()      sleep(1) |

**Botones**

|  |
| --- |
| La documentación oficial está [**AQUÍ**](http://python-ev3dev.readthedocs.io/en/latest/other.html#button)**.**  Aquí hay algunos scripts cortos que demuestran el uso de la clase Button ().   * Si ejecuta las secuencias de comandos desde la línea de comandos, la interfaz de Brickman seguirá mostrándose y respondiendo mientras se ejecuta la secuencia de comandos. El comando de impresión se imprimirá en la terminal. Un script puede forzarse a terminar con Ctrl-C. * Si ejecuta las secuencias de comandos desde el explorador de archivos de Brickman, la pantalla del EV3 se borrará para que la secuencia de comandos pueda imprimir en él. La interfaz de Brickman reaparecerá cuando el script finalice. La secuencia de comandos se puede forzar a que termine pulsando prolongadamente el botón Retroceso.   Actualmente, por razones desconocidas, estos scripts solo funcionan para mí desde la línea de comandos; se cierran inmediatamente si se ejecutan desde Brickman.  **Presione cualquier botón para salir**  El ciclo comprueba si se presiona algún botón y, de ser así, emite un sonido y sale del script. El script verifica el estado del botón cada 0.01 segundos.  #! / usr / bin / env python3  de la importación ev3dev.ev3 \*  desde el tiempo de importación de sueño  **btn = Botón ()**  mientras que es cierto:  **if btn.any ()** : # Comprueba si se presiona algún botón.          Sound.beep (). Wait () # Espere a que finalice el pitido.          exit () # Detener el programa.      más:          dormir (0.01) # Esperar 0.01 segundos  Aquí hay una versión mucho más ordenada del script anterior:  #! / usr / bin / env python3  de la importación ev3dev.ev3 \*  desde el tiempo de importación de sueño  **btn = Botón ()**  while  **btn.any () == False**:  # Si bien no se presiona ningún botón.          dormir (0.01) # Esperar 0.01 segundos  Sound.beep () .wait ()  **Reacciona a las presiones y lanzamientos de botones**  En el ciclo de este script, el comando resaltado btn.process () verifica cualquier cambio en el estado de los botones. Si detecta un cambio, desencadena los correspondientes "eventos". Por ejemplo, si detecta que el botón izquierdo acaba de presionarse, activa un evento de "cambio de estado del botón izquierdo" y un evento de "cambio de botón". También asigna un valor de Verdadero al estado de parámetro si se presiona el botón y un valor de False si se suelta el botón. Los manejadores de eventos responden a los eventos. Por ejemplo, si se presiona el botón izquierdo, el evento 'cambio de estado del botón izquierdo' disparará el controlador de evento on\_left resaltado que llamará a la función ' izquierda'(la función no tiene que tener este nombre).  #! / usr / bin / env python3  de la importación ev3dev.ev3 \*  desde el tiempo de importación de sueño  btn = Botón ()  # Haga algo cuando el estado de cualquier botón cambie:    def izquierda (estado):      si estado:          imprimir ('botón izquierdo presionado')      más:          imprimir ('botón izquierdo liberado')    def right (state): # uso más limpio de 'if' a continuación:      print ('Botón derecho presionado' si estado else 'Botón derecho liberado')    def up (estado):      print ('Botón Up presionado' si estado else 'Botón Up liberado')    def down (estado):      print ('Botón Abajo presionado' si estado else 'Botón Abajo lanzado')    def enter (estado):      print ('Entrar botón presionado' si estado else 'botón Enter liberado')    def retroceso (estado):      print ('Botón de retroceso presionado' si estado else 'Botón de retroceso liberado')    **btn.on\_left** = izquierda  **btn.on\_right** = right  **btn.on\_up** = up  **btn.on\_down** = abajo  **btn.on\_enter** = enter  **btn.on\_backspace** = retroceso  while True: # Este loop comprueba el estado de los botones de forma continua,               # llamadas a controladores de eventos apropiados  **btn.process ()** # Compruebe si hay botones actualmente presionados.      # Si el nuevo estado difiere del antiguo estado,      # llamar a los controladores de eventos del botón apropiado.      el estado de los botones sleep (0.01) # se comprobará cada 0.01 segundos  # Si ejecuta este script a través de SSH, presione Ctrl + C para salir  # si ejecuta este script desde Brickman, mantenga presionado el botón de retroceso para salir  **Obtenga una lista de estados de botones modificados**  Este script tiene una funcionalidad similar a la anterior, pero utiliza el controlador de eventos on\_change (changed\_buttons) para lograr un script mucho más conciso. Presione Ctrl-C (línea de comando) o mantenga presionado el botón de retroceso (Brickman) para finalizar el script.  #! / usr / bin / env python3  de la importación ev3dev.ev3 \*  desde el tiempo de importación de sueño  btn = Botón ()  **def change (changed\_buttons): # changed\_buttons es una lista de**  **# tuplas de nombres de botones modificados y sus estados.**  **print ('Estos botones cambiaron de estado:' + str (changed\_buttons))**  **btn.on\_change = cambio**  while True: # Este lazo comprueba el estado de los botones  # continuamente y llama a los controladores de eventos apropiados  **btn.process ()**      dormir (0.01)  **Compruebe los estados del botón**  El script anterior mostraba cómo process () y sus manejadores de eventos correspondientes se pueden usar para detectar y responder a CHANGES en estado. También hay funciones ( abajo , izquierda , entrar , etc.) que simplemente detectan el ESTADO de los botones. El siguiente script verifica y muestra el estado del botón izquierdo una vez por segundo. Puede finalizar el script con un segundo presionando el botón Retroceso.  #! / usr / bin / env python3  de la importación ev3dev.ev3 \*  desde el tiempo de importación de sueño  btn = Botón ()  **while not btn.backspace** :      imprimir ( **btn.left** )      dormir (1)  **Otras funciones de botones**  El siguiente script muestra dos funciones más: buttons\_pressed y check\_buttons () . Cada segundo se imprime una lista de los botones presionados y  puede finalizar el guion presionando SÓLO los botones izquierdo y derecho simultáneamente.  #! / usr / bin / env python3  de la importación ev3dev.ev3 \*  desde el tiempo de importación de sueño  btn = Botón ()  mientras que es cierto:      imprimir ( **btn.buttons\_pressed** )      if **btn.check\_buttons (botones = ['izquierda', 'derecha'])** :          salida()      dormir (1) |