



Python Reference for Pet Car Alarm

For more on programming the TI-Innovator Hub with TI-Nspire CXII Python follow the links to the TI Hub Menu Map: [TI-Nspire™ Python Programming](#) > [Python Menu Map](#) > TI Hub Menu

Function	Example	Behavior
<code>from module_name import *</code>	<code>from ti_hub import *</code>	Imports all the functions in the ti_hub module for use in the program. The ti_hub module includes all the necessary additions needed for project.
<code># text comment</code>	<code># defines a temperature sensor</code> <code># object named temp_sensor_inside</code>	# at the beginning of a line denotes a comment. Comments are a “best practice” by programmers to annotate their code. Comment statements are ignored when the program is run. In the TI-Nspire CXII Python editor, [ctrl]+[T] toggles the statement of the current cursor location from a comment to a statement that will be run.
<code>sound.tone(frequency,time)</code>	<code>sound.tone(440,1)</code> <code>sleep(1)</code>	Plays a tone through the TI-Innovator Hub speaker. In the example, plays the frequency 440 hertz (Hz) for 1 second. Following sound.tone with a sleep function assures that the tone will complete before the program begins the next sound function. Note: sound.tone() is available from the Hub Built-in Devices>Sound Outputs menu.
<code>name_of output=device_type("port")</code>	<code>left_headlamp=("OUT 1")</code>	Creates an LED object named left_headlamp connected to port OUT 1. led is available from the TI Hub > Add Output Device menu. Note: = is the Python operator for storing or assigning values to a variable.
<code>name_of_led_object.on()</code> <code>name_of_led_object.off()</code>	<code>left_headlamp.on()</code> <code>sleep(1)</code> <code>left_headlamp.off()</code> <code>sleep(1)</code>	Turns on an LED object named left_headlamp then pauses the program for 1 second before turning left_headlamp off then pauses the program again for 1 second. Note: To see options for an object paste the object name from the var key menu then press the period key.
<code>sleep(seconds)</code>	<code>sleep(.5)</code>	Pauses program for .5 seconds. sleep() is found on the Hub Commands menu.
<code>for index in range(stop value):</code> block	<code>for n in range(10):</code> print(n)	Repeats the statements in the block ten times, printing the value of the index variable, n , as 0,1,2,...9. The index variable n starts at 0 and increases by 1 with each loop. If n is less than the stop value, 10, the loop continues to repeat.
<code>name_of_led_object.blink(frequency, time)</code>	<code>left_headlamp.blink(0.5,60)</code>	Blinks (turns on and off) an LED object named left_headlamp once every 2 seconds (one half a blink per second) for 60 seconds. Frequency is typically measured in Hertz, cycles per second. Note: To see options for an object paste the object name from the var key menu then press the "." key.



Challenge: Design Pet Car Alarm System

TI-NSPIRE™ CXII PYTHON

<pre>name_of_device=output_device_type("port")</pre>	<pre>window_motor=continuous_servo("OUT 3")</pre>	<p>Creates a continuous servo device object named window_motor connected to port OUT 3.</p> <p>Continuous Servo is available from the TI Hub > Add Output Device menu. The Servo motor requires 5 volts and must be connected to port OUT 3. An external battery must be plugged into the HUB PWR port and turned on before running programs that include the Servo motor. Note: = is the Python operator for storing or assigning values to a variable.</p>
<pre>continuous_servo_variable.set_cw (speed, time) continuous_servo_variable.set_ccw (speed, time)</pre>	<pre>window_motor.set_cw(20,1) sleep(2) window_motor.set_ccw(20,1) sleep(2)</pre>	<p>Sets the continuous servo motor named window_motor to turn clockwise at a speed of 20 (speed inputs range between 0 for off to 255 for maximum speed) for 1 second, then pauses the program for 2 seconds before moving on to set the motor to turn counter clockwise at a speed of 20 for 1 second. Follow the motor set functions with sleep() functions to assure that the motor completes the desired action before proceeding to the next function.</p>
<pre>name_of_sensor=sensor_type("port")</pre>	<pre>temp_sensor_inside=temperature("IN 1")</pre>	<p>Creates a temperature sensor object named temp_sensor_inside connected to port IN 1. temperature is available from the TI Hub > Add Input Device menu. Note: = is the Python operator for storing or assigning values to a variable.</p>
<pre>var=name_of_sensor.measurement()</pre>	<pre>temp_inside=temp_sensor_inside.measurement()</pre>	<p>Reads and stores the current measurement value of the temp_sensor_inside object into variable temp_inside. Note: .measurement() returns the current measured value of a sensor object. To see options for an object paste the object name from the var key menu then press the period key.</p>
<pre>text_at(row,"text","align")</pre>	<pre>text_at(3,"temperature inside car= ` +str(temp_inside)+" °C","left")</pre>	<p>This text_at() function displays a text string on a specified row with an alignment of left, center or right. When variable temp_inside has a value of 26, the following is displayed on row 3, aligned to the left:</p> <p>temperature in car= 26 °C</p> <p>text_at() is available from the TI Hub>Commands menu.</p> <p>Note: The str() function converts a numeric value to a string. The + operator is used to join two strings. str() is available from the Built-ins> Type menu.</p> <p>Note: Degree, percent and other special characters are available from the ?! key menu in the lower right of the TI-Nspire keyboard.</p>
<pre>var="text string"</pre>	<pre>window="closed"</pre>	<p>Sets the value of a variable named window to be the text string, "closed".</p> <p>Variables can contain many different data types, including floating point numbers, integers, text strings and lists.</p>



Challenge: Design Pet Car Alarm System

TI-NSPIRE™ CXII PYTHON

<pre>while get_key() != "esc": block</pre>	<pre>while get_key() != "esc": temp_inside=temp_sensor_inside.measurement() text_at(3,"temperature inside car=" "+str(temp_inside)+" °C","left") sleep(0.5)</pre>	<p>Defines a while loop that will continue until the escape key is pressed.</p> <p>While loops repeat the statements in the block if the condition at the top of the loop is true. In the example, looping continues until the escape key is pressed. Not pressing a key or pressing any key but escape means that get_key() will return a value that is not equal to "esc". The loop condition is true and looping continues. If the escape key is pressed, get_key() returns "esc". The condition will evaluate as "esc" not equal to "esc", which is false. A false result means that the loop statements are not repeated. Program execution skips to the statement just after the loop. Note: The block starts with a colon and includes the indented lines that follow. while get_key() != "esc": is available from the TI Hub > Commands menu.</p>
<p><Boolean expression> value 1 operator value 2</p>	<pre>2+3==6 (result is false) x+4>=y (if x=1 and y=3, the result is true) "enter"!="esc" (result is true)</pre>	<p>Boolean expressions evaluate to either true or false. The examples show some of the relational operators available from the Built-ins > Ops menu.</p> <p>Note: == is the Python operator to check equality. >= is the Python operator to check whether the value to the left is greater than or equal to the value on the right. != is the Python operator to check inequality.</p>
<pre>if <Boolean expression>: block else: block</pre>	<pre>if magnet_value <200: text_at(5,"magnet is present ","left") else: text_at(5,"magnet is not present","left")</pre>	<p>Checks to determine if the value of variable magnet_value is less than 200. If the statement is "true" then the statements in the if block are executed. If the statement is "false" then the statements in the else block are executed. In the example, when magnet_value is less than 200, the text "magnet is present" will be displayed on the calculator screen. When the value of magnet_value is 200 or greater the text "magnet is not present" will be displayed. Note: if..else.. is available from the Built-Ins > Control menu.</p>
<pre>if <Boolean expression> and <Boolean expression>: block</pre>	<pre>if temp_inside>25 and magnet_value<200: text_at("pet in danger")</pre>	<p>If both expressions are true the and function is "true", then the block is executed. Otherwise, the and function returns false, and the block is skipped. In the example, when the temperature value is greater than 25 and the magnet value is less than 200, the message "pet is in danger" will be printed on the calculator screen. Note: if.. is available from the Built-Ins > Control menu.</p>