Python Reference for Smart Irrigation Project

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
from module_name import *	`from ti_hub import *`	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.

# text comment	`# defines a light_level sensor` 	
	`# object named lightsensor`
		# 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.

name_of sensor=sensor_type("port")	`lightsensor=light_level("IN 1")`	Creates a light_level sensor object named **lightsensor** connected to port IN 1. light_level is available from the TI Hub > Add Input Device menu. Note: = is the Python operator for storing or assigning values to a variable.
name_of_sensor.range(min,max)	`lightsensor.range(0,100)`	Scales the measured values read from the **light.sensor** object to the range of 0 to 100. **Note:** to see options for an object select the object name from the var key menu then press the "." key.
var=name_of_sensor.measurement()	`light=lightsensor.measurement()`	Reads and stores the current measurement value of the **lightsensor** object into variable **light**. Note: **.measurement()** returns the current measured value of a sensor object. To see options for an object select the object name from the var key menu then press the "." key.
text_at(row,"text","align")	`text_at(3,"light level= " +str(light),"left")`	This text_at() function displays a text string on a specified row with an alignment of left, center or right. When variable **light** has a value of 26, the following is displayed on row 3, aligned to the left: light level = 26 text_at() is available from the TI Hub>Commands menu. **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. **Note:** Degree, percent and other special characters are available from the ?! key menu in the lower right of the TI-Nspire keyboard.

while get_key() != "esc":
block
	`while get_key() != "esc":`
	`light=lightsensor.measurement()`
	`text_at(3,"light level= "+str(t),"left")`
	`sleep(1)`
		Defines a while loop that will continue until the escape key is pressed. 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.

sleep(seconds)	`sleep(.5)`	Pauses program for .5 seconds.

for index in range(stop value):
block
	`for n in range(10):`
	`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.

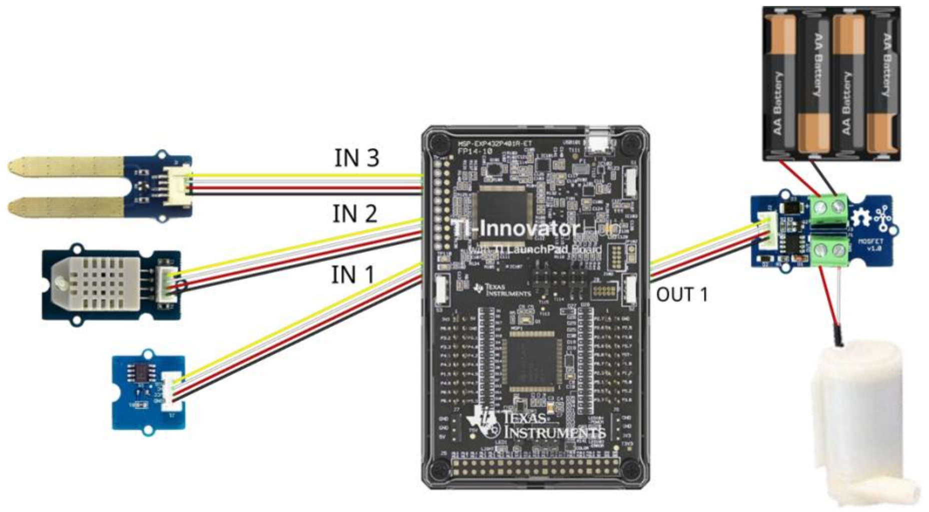
<Boolean expression>
value 1 operator value 2
	`2+3==6 (result is false)`
	`x+4>=y (if x=1 and y=3, the result is true)`
	`"enter"!="esc" (result is true)`
		Boolean expressions evaluate to either true or false. The examples show some of the relational operators available from the Built-ins > Ops menu. 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.

if <Boolean expression>:
block
else:
block
	`if moisture < 10:`
	`text_at(5,"The soil is dry","left")`
	`else:`
	`text_at(5,"The soil is moist","left")`
		Checks to determine if the value of variable **moisture** is less than 10. 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 the **moisture** value is less than 10, the text "The soil is dry" will be displayed on the calculator screen. When the value of moisture is 10 or greater the text "The soil is moist" will be displayed. Note: **if..else..** is available from the Built-Ins > Control menu.

if <Boolean expression> and <Boolean expression>:
block
	`if temperature<= 25 and humidity>=80:`
	`text_at("it is cool and humid.")`
		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 is less than or equal to 25 and the humidity is greater than or equal to 80, the message "It is cool and humid" will be printed on the calculator screen.

name_of_device=device_type("port")	`pump=analog_out("OUT 1")`	Creates an analog_out object named **pump** connected to port OUT 1. analog_out is available from the TI Hub > Add Output Device menu. Note: = is the Python operator for storing or assigning values to a variable.
name_of_output_device.set(value)	`pump.set(255)` `sleep(10)` `pump.set(0)`	Sets the value of an analog_out device named **pump** to 255 (full power). The pump continues at full power until a value of 0 (off) is received after the sleep function causes the program to pause for 10 seconds. **Note:** The output device will continue using a setting value until a new setting value is received or until the power to the system is removed. Make sure to use the .set(0) or .off() output objective methods to turn the device off. To see options for an object select the object name from the var key menu then press the "." key.
var=input("prompt text")	`speed=int(input("pump speed (0-255)= "))`	Prompts the user to enter a value that will be stored as an integer to the variable **speed**. input() prompts the user to enter a text string. Int() from the. Built-Ins>Type menu changes the text string to an integer data type. The integer value is stored to the variable **speed** for later use in functions that require numeric values as inputs.

Setup Project:



Supplies:

- TI-Innovator Hub and cable
- TI Nspire CXII calculator
- Grove - Temperature & Humidity Sensor
- Grove – Soil Moisture Sensor
- Grove – Light Sensor
- Grove – MOSFET for power control module with 4xAA battery holder
- 4 x AA batteries to provide power to the pump (required)
- Water Pump with the plastic tube
- Possible supplies for building garden model:
- Drinking straws
- Duct Tape (always useful)
- Container for the plants, such as a 1-gallon milk
- Soil, perlite or some other growth medium