Challenge: Model the Human Four-Chamber Heart

Python Quick Reference for The Plumber: Model the Human Four-Chamber Heart

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	`# Define an external LED output device` `# with variable name led1`	# 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 output=device_type("port")	`led1=("bb1")`	Creates an LED object named **led1** connected to breadboard port 1, "bb1". led is available from the TI Hub > Add Output Device menu. The drop down menu for port does not include breadboard ports. It is necessary to escape from the menu and type in bb1, bb2, etc. Note: = is the Python operator for storing or assigning values to a variable.

name_of_led_object.on()
name_of_led_object.off()
	`led1.on()`
	`sleep(1)`
	`led1.off()`
	`sleep(1)`
		Turns on an LED object named **led1** then pauses the program for 1 second before turning **led1** 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.

sleep(seconds)	`sleep(.5)`	Pauses program for .5 seconds. sleep() is found on the Hub Commands menu.
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. Note: for index in range is found on the Built-ins>Control menu.

for index in list:
block
	`for c in [led1,led2,led3,led4]:`
	`c.on()`
	`sleep(1)`
	`c.off`
	`sleep(1)`
		Loops through the elements of the list one at a time. The index variable **c** is replaced with the list element. In this example, the list elements are LED output objects that were defined earlier in the program. First led1 is turned on then off, then led2 the same and so one through all of the list elements. Note: for index in list is found on the Built-ins>Control menu. List elements are enclosed in square brackets. Values, text strings, objects are some of the data types that can be used in a list.

sound.tone(frequency,time)	`sound.tone(440,1)`	Plays a tone through the TI-Innovator Hub speaker. In the example, plays the frequency 440 hertz (Hz) for 1 second. Note: sound.tone() is available from the Hub Built-in Devices>Sound Outputs menu.
name_of sensor=sensor_type("port")	`temp_sensor=temperature("IN 1")`	Creates a temperature sensor object named **temp_sensor** 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.
var=name_of_sensor.measurement()	`t=temp_sensor.measurement()`	Reads and stores the current measurement value of the **temp_sensor** object into variable **t**. 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.
text_at(row,"text","align")	`text_at(3,"temperature= " +str(t)+" °C","left")`	This text_at() function displays a text string on a specified row with an alignment of left, center or right. When variable **t** has a value of 26, the following is displayed on row 3, aligned to the left: temperature= 26 °C 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":`
	`t=temp_sensor.measurement()`
	`text_at(3,"temperature= "+str(t),"left")`
		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.

<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 t<=27:`
	`period=.15`
	`else:`
	`period=.05`
		Checks to determine if the value of variable **t** is less than or equal to 27. 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 **t** is less than or equal to 27, the value .15 is stored to the variable **period**. When the value of **t** is or greater than 27, the value .05 is stored to the variable **period**. Note: **if..else..** is available from the Built-Ins > Control menu.