StarLAB Python Cheat Sheet

If you need more help, just ask! support@starlab.education

OBELISK SYSTEMS

http://starlab.education
http://obelisksystems.com

Variables, Types, and Printing Things

A variable holds some piece of data for you to use later. They will have a type that is usually handled by Python, but it is useful to know about them.

Integers can be any whole number like -1, 0, 1, 2, 3

```
intNumber = 1             # assigns intNumber as 1
```

Floating point numbers are numbers like 1.01

```
floatNumber = 1.00        # .00 makes it a float
```

Strings are text values and are set by using quotes (" or ')

```
msg = "Hello Space!"      # assign String to msg
```

You can output a variable with **print**

```
msg = "Hello Space!"      # assign String to msg
print msg                 # displays text in window
```

you can output multiple values with a comma (,)

```
firstName = "Ada"         #Assign String to firstName
lastName = "Lovelace"     #Assign String to lastName
print "Countess", firstName, lastName
```

You can change the type of a variable by 'casting'

```
number = "1"              # a String with the 1 character
number2 = 2                          # the integer 2
print number+number2                 # will cause an error
print int(number)+number2            # prints 3
print float(number)+number2          # prints 3.0
print number+str(number2)            # prints 12
```

Functions

Functions let you use one block of code in many places.

```
def add(x, y=2):          # y=2 we set the default value
    return x + y          # add the inputs and return result
    print add(1)          # call function : returns 3
    print add(1,9)        # call function : returns 10
```

Import

You can get extra functions by using **import**, there are many libraries you can import.

```
import time                                    # import library
n=0                                            # initialise counter
while True:                                    # loop forever!
    print n += 1          #add one
    time.sleep(1)         # pauses the loop for 1 second
```

Maths Operators

maths operations can be done using the built-in operators.

```
product = n + 1           # will have the sum of n and 1
subtraction = n – 1       # one subtracted from n
multiply = n * 8          # eight times n
divide = n / 9            # division
divide = n//9             # integer division
remainder = n % 9         # remainder from division
exponent = n ** 8         # n raised to the 8th power
```

Any maths operator can be used with the equals symbol to assign the vale and perform the operation

```
product +=  1             # product = product + 1
```

User Input

You can allow users to interact with your program with inputs. **raw_input** will store the input as a string.

```
name = raw_input("Who. are. you?") # caterpillar question
print "Explain yourself, "+name+"!"  # his response
```

Other data types are gotten with input; it will decide which type to use based on the input.

```
planets = input("How many planets are there? ")    # integer
print planets             # print 8 (we love Pluto, but no)
pi = input("What's the value of pi?")  # floating point
pi = float(pi)     # 3.14159265…How long can this go for?
```

Booleans (True or False)

Booleans are a special type of variable that can either be **True** or **False**

```
Blue = True               # sets variable to True
Blue = False              # sets variable to False
```

Booleans can be used for conditional arguments.

```
test = (n == 7)           # True if n equal 7
test = (n != 7)           # True if n not equal 7
test = (n > 7)            # True if n greater than 7
test = (n >= 7)           # True if n greater than or equal 7
test = (n < 7)            # True if n less than 7
test = (n <= 7)           # True if n less than or equal 7
```

If Statements

If statements Use Booleans to perform small blocks of code if a test is **True** or **False**.

```
check = (temp >= 18)      # check if the temp is above 18
if check:                 # test check
    print "It's too hot!" # run if check is True
```

if statements can run other code if the test is **False**.

```
if temp < 4:              # first test
    print "It's too cold!"
elif temp < 18:           # only tests if the first is False
    print "This is nice!"
else:                     # run if the others are False
    print "It's too hot!"
```

Loops

A **while** loop repeats a block of code until a certain condition is true. **Hint:** If you get stuck in a loop try Ctrl-C

```
counter = 1               # initialise counter
while counter <= 5:       # test if condition is reached
    print counter         # print the current value
    counter += 1          # add one to the counter
```

Setting the condition for a **while** to True will make it loop infinitely.

```
msg = ""                  # assign msg to be an empty string
while True:               # Loop forever!
    msg = raw_input("Speak friend and enter")
    if msg == 'mellon':
        break             # end the loop
```

A **for** loop will run for a set number of times and then exit.

```
for i in range(1, 6):
    print "Loop number", i
```

Working with Files

```
filename = 'newFile.txt'          # set filename
myfile = open(filename, 'r')      # open file for reading
lines = myfile.readlines()        # load lines into a list
for line in lines:                # loop through lines
    print line                    # print each line
```

Writing to a file

```
filename = 'journal.txt'                 # set filename
myfile = open(filename, 'w')             # open file to write
    myfile.write("I love programming.")  # write text to file
```

Appending to a file:

```
filename = 'journal.txt'                 # set filename
myfile = open(filename, 'a')             # open file to write
    myfile.write("\nI love making games.")# write text to file
```

List

A List stores a series of items in particular order. You access items using an index, or within a loop.

Make a list:

```
lukeLunch = ['carrot', 'broccoli', 'corn']        # define list
```

Get the first item in a list:

```
first_lukeLunch = lukeLunch[0]                    # lists index from 0
```

Get the last item in a list:

```
last_lukeLunch = lukeLunch[-1]              # -1 is shorthand for last
```

Looping through a list:

```
for veg in lukeLunch:             # veg is the current element
    print(veg)                    # displays element in window
```

Adding items to a list:

```
lukeLunch = []                                 # define empty list
lukeLunch.append('carrot')                        # add Element
lukeLunch.append('broccoli')                      # add Element
lukeLunch.append('corn')                          # add Element
```

Making numerical Lists:

```
squares = []                                      # define empty list
for x in range(1, 11):
    squares.append(x**2)                          # x^2
```

Slicing a list:

```
students = ['grace', 'alan', 'ada','nikola']      # define list
first_two = students[:2]        # ':2' selects everything before 2
```

Copying a list:

```
copy_of_lunch = lukeLunch[:]                   # ':' selects everything
```

Conditional test with lists:

```
'broccoli' in lukeLunch                     # True if broccoli in list
'potato' not in lukeLunch                   # True if potato not in list
```

Connecting to StarLAB

Make sure you copy the **StarLAB.pyc** into the directory that your python script is in. Then you can import the API.

```
import StarLAB                              # import the best library
```

Connect to the StarLAB with the IP on the OLED.

```
myStarLAB = StarLAB.Connect(IP = "192.168.0.1")
```

When connecting to multiple StarLABs use different names for each one.

```
myStarLAB =   StarLAB.Connect(IP = "192.168.0.1")
lukeStarLAB = StarLAB.Connect(IP = "192.168.0.2")
```

StarLAB Spectrum Sensors

The spectrum sensors get information about the light that the StarLAB can see. getSpectrum() returns a list from all the sensors [[Red, Green, Blue], ambient, IR, UV]

```
data = myStarLAB.spectrum.getSpectrum() # all spectrum
```

getRGB returns a list of [Red, Green, Blue] in lux

```
data = myStarLAB.spectrum.getRGB()      # RGB in lux
```

getAmbient, spectrum.getIR, and spectrum.getUV return a Single value in Lux for the first two and μW/cm^2 for UV

```
data1 = myStarLAB.spectrum.getAmbient()          # Lux
data2 = myStarLAB.spectrum.getIR()               # Lux
data3 = myStarLAB.spectrum.getUV()       # μW/cm^2
```

StarLAB Movement Sensors

The IMU returns information about the movement of the starLAB. They all return a list of three dimensions [X,Y,Z]

```
data1 = myStarLAB.IMU.getAccel()                 # m/(s^2)
data2 = myStarLAB.IMU.getGyro()                  # deg/s
data3 = myStarLAB.IMU.getMag()                   # m-Gauss
data4 = myStarLAB.IMU.getOrientation()           # deg
```

StarLAB Atmospheric Sensors

The atmos sensors give you information about the weather and are all single values.

```
data1 = myStarLAB.atmos.getHumidity()  # percentage
data2 = myStarLAB.atmos.getPressure()  # hPa
data3 = myStarLAB.atmos.getAltitudeM() # meters
data4 = myStarLAB.atmos.getTempC()     # celsius
```

StarLAB Hardware Temperature

The temperature of the board can be gotten with

```
data1 = myStarLAB.boardThermo.getTopTempC()
data2 = myStarLAB.boardThermo.getBotTempC()
```

StarLAB LED Lights

The StarLAB has 4 indicator LEDs (LED1-4) and one RGB LED that is controlled by three values (Red, Green, and Blue). To turn on or off any LED you use the **set<name>On** and **set<name>Off**. The RGB brightness is changed with **set<name>** with 0 being off and 255 being maximum.

```
myStarLAB.light.setRedOn()          # turn on Red
myStarLAB.light.setGreenOff()       # turn off Green
myStarLAB.light.setBlue(175)        # set brightness of Blue
```

StarLAB Buzzer

To turn on the buzzer set the frequency using **setFrequency.** This command takes an input between 0-8000Hz. Setting a value of 0 will turn the buzzer off.

```
myStarLAB.buzzer.setFrequency(880)     # buzzer in Hz
myStarLAB.buzzer.setFrequency(0)       # turn buzzer off
```

StarLAB Buttons

The buttons on the StarLAB return a 1 when they are pressed and 0 when they are not. The buttons can be checked all at once with **readButtonALL** and returns the list [Left, Up, Down, Right, Centre, A, B, C]

```
data = myStarLAB.button.readAll()                    # all!
```

Buttons can be checked individually using **readButton<name>** where name is on of A, B, C, Up, Down, Left, Right, Centre.

```
data = myStarLAB.button.readA()                      # just A
```

StarLAB OLED Screen

Write messages on the OED with **writeText** it takes a string input.

```
myStarLAB.OLED.writeText("Hello  Space")     # write hello
```

Write each line of the OLED with **writeTextLine** where each line gets its own string.

```
L1 = "Haikus are easy"                      # String for line 1
L2 = "But sometimes they"                   # String for line 2
L3 = "Refrigerator"                         # String for line 3
myStarLAB.OLED.writeTextLine(Line1=L1,Line2=L2,Line3=L3)
```

Clear the screen with clear

```
myStarLAB.OLED.clear()
```

StarLAB Camera

Pictures can be taken with **takePicture**. The input will be the name of the file in the location of the script.

```
myStarLAB.camera.takePicture("filename") # filename.jpg
```

StarLAB Rover

Take control of the Rover with the new API.

```
myStarLAB.enableRover()   # enable control of the Rover
```

See the power usage of the rover and StarLAB.

```
data1 = myStarLAB.reactor.generator.getVoltage()
                          # battery Level in Volts
data2 = myStarLAB.reactor.engine.getCurrent()
                          # motor current draw in mA
data3 = myStarLAB.reactor.processor.getPower()
                          # power used by the Rover in mW
```

Set the motor power.

```
myStarLAB.motors.setMotorPower(60,60)  # move forward
myStarLAB.motors.setMotorPower(-40,-40)# move backwards
myStarLAB.motors.turnRover(90)         # turn by angle
```

Get the distance from an obstacle.

```
data1 = myStarLAB.ranger.getDistance()   # range in cm
```