Cheat-Sheet of Serial Communication

@Written by Mahshid mzeinaly@sfu.ca

// ********* Processing:

**Reading:
port.read(); //reads one ascii character
port.readStringUnitl('\n'); // reads one line and you need to extract the integer

**Writing:
port.write()

**Where to put reading part:
one way: in draw function:
if(port.available()>0) {
// then read it!
}

Alternative way:
Considering "Arriving new data" is an event, you could define the following function outside of draw() and it will be called automatically whenever a new data arrives in Serial port.

serialEvent(Serial port){
//read the data here!
}

Note: port is the name of your port

// ********* Arduino:

**Reading:
if(Serial.available()>0){
Serial.read(); // But you have to know if it comes from Processing or the user, to understand its format!
}

**Writing:
*Serial.println(your data)
*Serial.write(your data); This one writes a byte of data. So if your data is less than 255, and you want to pass it easily, you could choose to not to bother and just use this. Then in processing part, simply read the data, using port.read();

For more information look at this: http://arduino.cc/en/Serial/Write

As the page recommend: If your data is integer, you better use print. For more information look at other related discussions in the webct.

Example:

//*********Arduino:

Serial.write(134); //write this in setup. It is just one test

//*******Processing:

```
if(Port.available() > 0){ //write this in draw
val = port.read();
println(val);
}
```

You will see that port.read() reads it correctly. If you change the value in Arduino to be 341, it can not read it correctly anymore!

As a general hint: Always make sure you know which format it is that the data is written in. And which format it is that your method is reading the data. If you did not remember, then just run a test and you will get it!