**Reading Numbers and strings from the Serial Monitor:**

The serial monitor is usually just used to display data from the [Arduino](https://www.amazon.com/gp/product/B008GRTSV6/ref=as_li_qf_asin_il_tl?ie=UTF8&tag=circbasi-20&creative=9325&linkCode=as2&creativeASIN=B008GRTSV6&linkId=e387d2ea29aa6ca1b2d2f67a8a00c430) on a computer monitor. But it can also be used as an input device that takes input from a user and sends it to the Arduino. This is useful for creating serial monitor based menus, calculators, and password logins, where the user is asked to input information and the Arduino performs an action based on that input.

## **HOW TO GET USER INPUT FROM THE SERIAL MONITOR**

To get a user’s input from the serial monitor, the first step is to prompt the user for information. This could be a question like “how many times do you want the LED to blink” or “choose an option from the menu”. It’s just text to tell the user they need to enter something. The code for the prompt could be a simple Serial.print() function that prints the question to the serial monitor. Users can enter int, float, or string data types, but you will need to know in advance what data type the user will be entering.

The next step is to use the Serial.available() function in an empty while loop. The Serial.available() function returns the number of bytes available to read from the serial port. When there is no user input, Serial.available() will return a value of zero. When the user inputs data and presses Enter, Serial.available() will return a non-zero value. This is used as the condition of an empty while loop to make the program wait until there is an input from the user:

**while (Serial.available() == 0) {**

**}**

The condition of the empty while loop is Serial.available()==0. When there is no input from the user, the Serial.available() function returns a zero value, making the condition true. The sketch stays inside the while loop until the user inputs something and the Serial.available() returns a non-zero value.

The final step is to read the information entered by the user and perform an action based on that input. To do that, we have to parse (read), the information stored in the serial buffer. To parse the information stored in the serial buffer, we can use one of these three functions:

* **Serial.parseInt()**
* **Serial.parseFloat()**
* **Serial.readString()**

The data type of the information input by the user determines which function you should use. If the user will be entering an int, use Serial.parseInt(). If the user will be entering a float, use Serial.parseFloat(). And if the user will be entering a string, use Serial.readString().

Before these functions can be used, we need to declare a variable to store the parsed data. The data type of that variable needs to match the data type of the input data. For example, to parse an integer, you could declare an int variable called integerVariable and set it equal to the Serial.parseInt() function like this:

**int integerVariable = Serial.parseInt();**

To parse a float, you could declare a float variable and set it equal to the Serial.parseFloat() function like this:

**float floatVariable = Serial.parseFloat();**

To parse a String, you could declare a string variable and set it equal to the Serial.readString() function like this:

**String stringVariable = Serial.readString();**

int number;

String msg="Enter the number";

String msg2="ur number is:";

## **HOW TO PARSE INTEGER DATA TYPES**

int number;

String msg="Enter the int number";

String msg2="ur int number is:";

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

}

void loop() {

// put your main code here, to run repeatedly:

Serial.println(msg);

while(Serial.available()==0)

{

}

number=Serial.parseInt();

Serial.print(msg2);

Serial.println(number);

}

## **HOW TO PARSE FLOAT DATA TYPES**

float number;

String msg="Enter the float number";

String msg2="ur float number is:";

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

}

void loop() {

// put your main code here, to run repeatedly:

Serial.println(msg);

while(Serial.available()==0)

{

}

number=Serial.parseFloat();

Serial.print(msg2);

Serial.println(number);

}

## **HOW TO PARSE STRING DATA TYPES**

String myname;

String msg="Enter ur name";

String msg2="ur name is:";

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

}

void loop() {

// put your main code here, to run repeatedly:

Serial.println(msg);

while(Serial.available()==0)

{

}

number=Serial.readString();

Serial.print(msg2);

Serial.println(myname)

}