Engineering Technology (ENGR 101)

Programming Basics

Mohammad Nekooei

Engineering and Computer Science

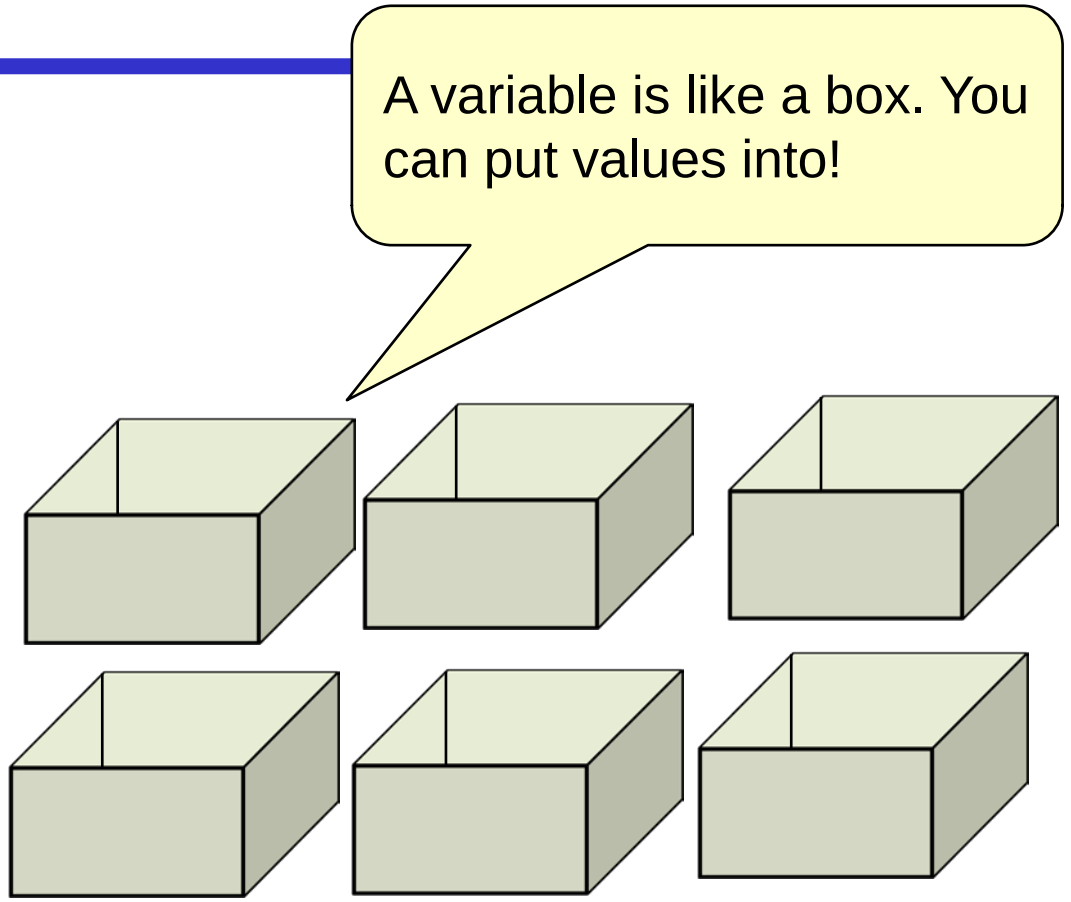
Victoria University of Wellington

Admin

- Lab 5 has been released
  - Due date is 17 May, 19:00 (Xiamen Time)

Variables

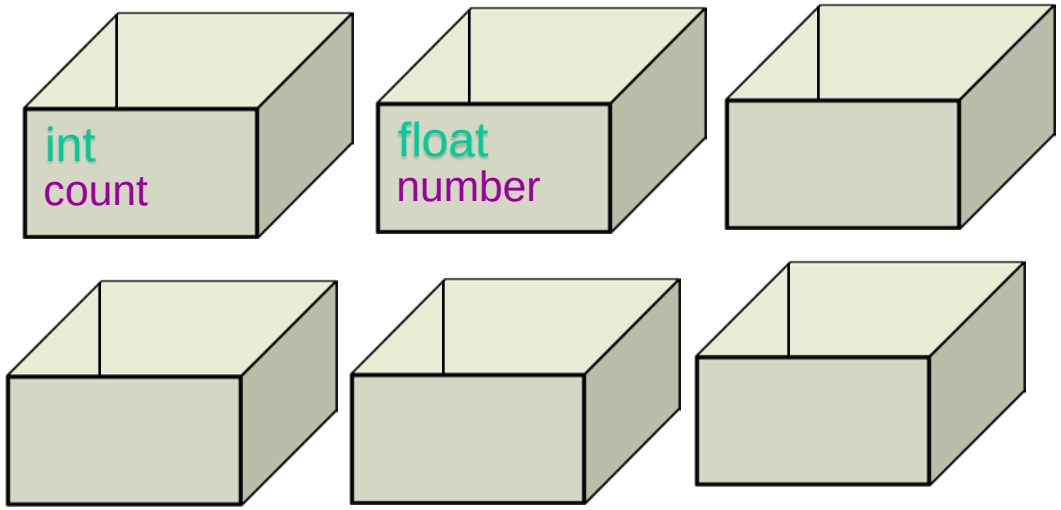
- The power of computer programming
  - Variables
  - Operations with variables
  - Input/output commands
- Variable is not the same as variables in math
- Variable in computer science is like a box that you can put different values into it



Arduino structure

- Processor
  - Central Processing Unit (CPU): It does calculations.
- Memory
  - A place where program and variables live
  - Memory is connected to the CPU

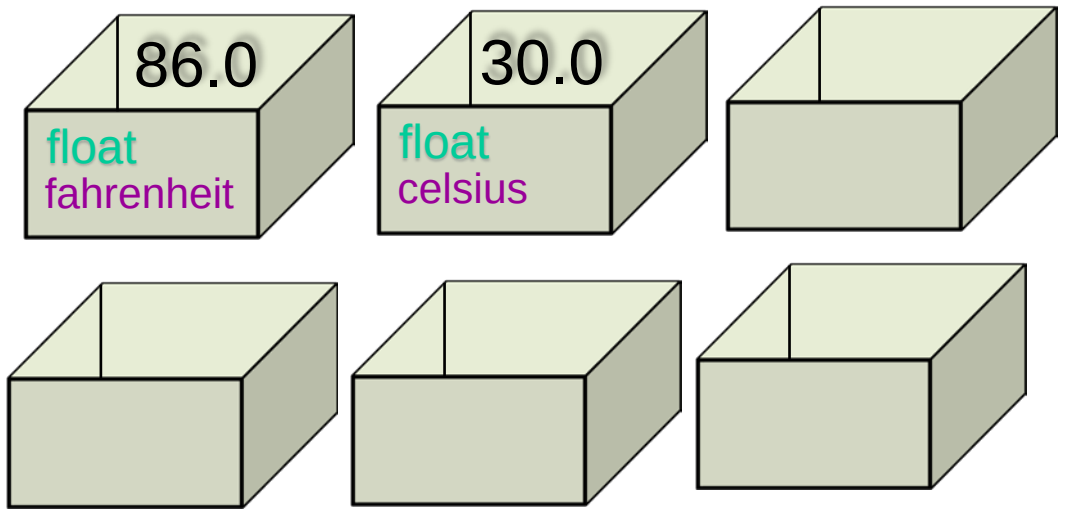
- You can think of memory as an actual physical box
  - Each box can contain a piece of information
  - If you want to use these boxes, we need to be able to refer to them.
    - Every box has a name

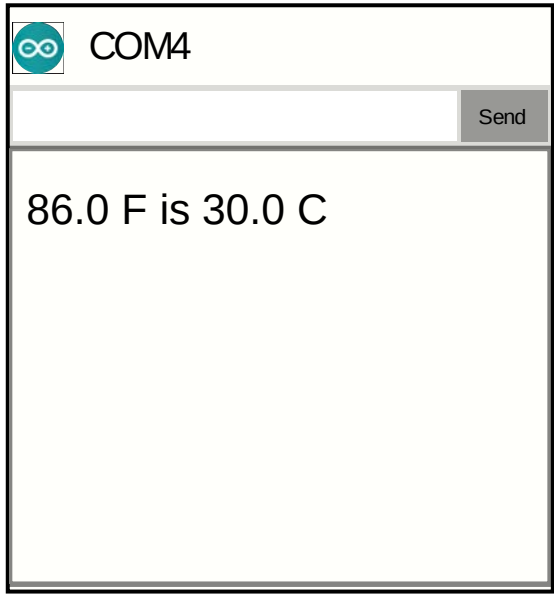


- A variable is a place in memory that can hold a value.
  - It has a name and type
  - I holds a piece of information

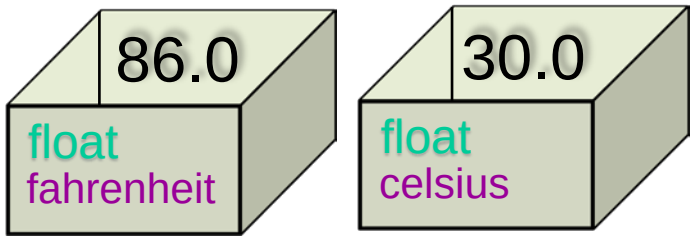
Variables

```
void setup() {
  float fahrenheit = 86.0;
  float celsius;
  Serial.begin(9600);
  celsius = (fahrenheit – 32.0)*5.0 / 9.0;
  Serial.print(fahrenheit);
  Serial.print(“ F is ”);
  Serial.print(celsius);
  Serial.println(“ C”);
}
void loop() {
}
```





Variables




```
void setup() {
  float fahrenheit = 86.0;
  float celsius;
  Serial.begin(9600);
  celsius = (fahrenheit – 32.0)*5.0 / 9.0;
  Serial.print(fahrenheit);
  Serial.print(“ F is ”);
  Serial.println(celsius);
  Serial.println(“ C”);
}
void loop() {
}
```

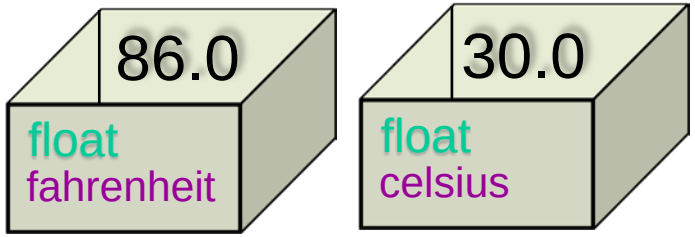
- A variable is a place in memory that can hold a value.
  - Must specify the **type** of value that can be put in the variable
    ⇒ “Declare” the variable first time it is mentioned.

    ⟨type⟩ → ⟨variable⟩
  - Must put a value into a variable before you can use it
    ⇒ “assign” to the variable
  - Can *use* the value by specifying the variable’s name
  - Can change the value in a variable (unlike mathematical variable)

Use a variable whenever you need the Arduino to remember something temporarily.

Asking for a place

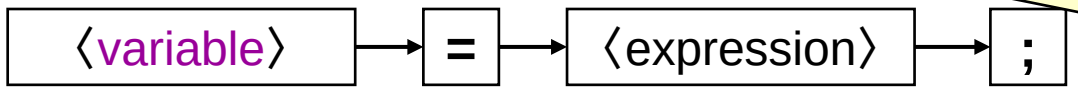
Assignment Statements




```
void setup() {
  float fahrenheit = 86.0;
  float celsius;
  Serial.begin(9600);
  celsius = (fahrenheit – 32.0)*5.0 / 9.0;
  Serial.print(fahrenheit);
  Serial.print(“ F is ”);
  Serial.println(celsius);
  Serial.println(“ C”);
}
void loop() {
}
```

- Assignment Statement:

where = *what* ;



Putting a value into a variable

name-of-place = *specification-of-value* ;

celsius = (fahrenheit – 32.0)*5.0 / 9.0;

Meaning: Compute the value and put it in the place

Expressions

combining declaration and assignment into a single line

```
void setup() {
  float fahrenheit = 86.0;
  float celsius;
  Serial.begin(9600);
  celsius = (fahrenheit – 32.0)*5.0 / 9.0;
  Serial.print(fahrenheit);
  Serial.print(“ F is ”);
  Serial.println(celsius);
  Serial.println(“ C”);
}
void loop() {
}
```

- Expressions describe how to compute a value.
- Expressions are constructed from
  - values
  - variables
  - operators (+, -, *, /, etc)
  - functions calls that return a value
  - sub-expressions,
  - …

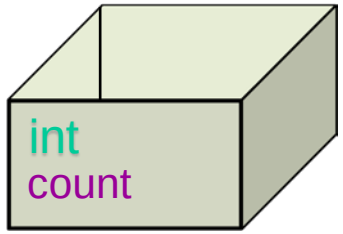
Values / Data

There are lots of different kinds ("Types") of values:

- Numbers
  - Integers **int** (or long) 42 -194573203
  - real numbers **double** (or float) 42.0 16.43
  - …
- Characters char 'X' '4'
- Text **String** " F -> "
- …

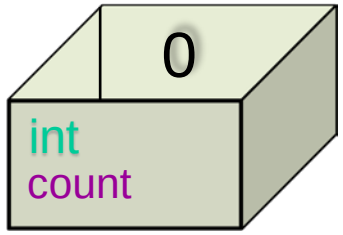
Using a Variable: int

```
int count;
void setup() {
  Serial.begin(9600);









}
void loop() {
}
```



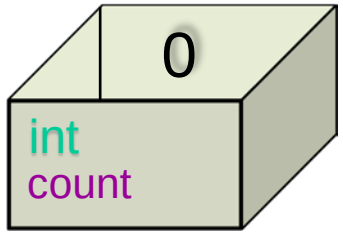
Using a Variable: int

```
int count;
void setup() {
  Serial.begin(9600);
  count = 0;




}
void loop() {
}
```



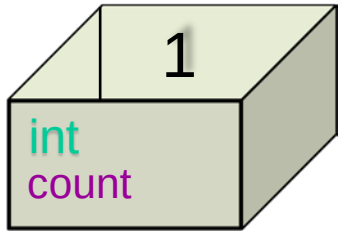
Using a Variable: int

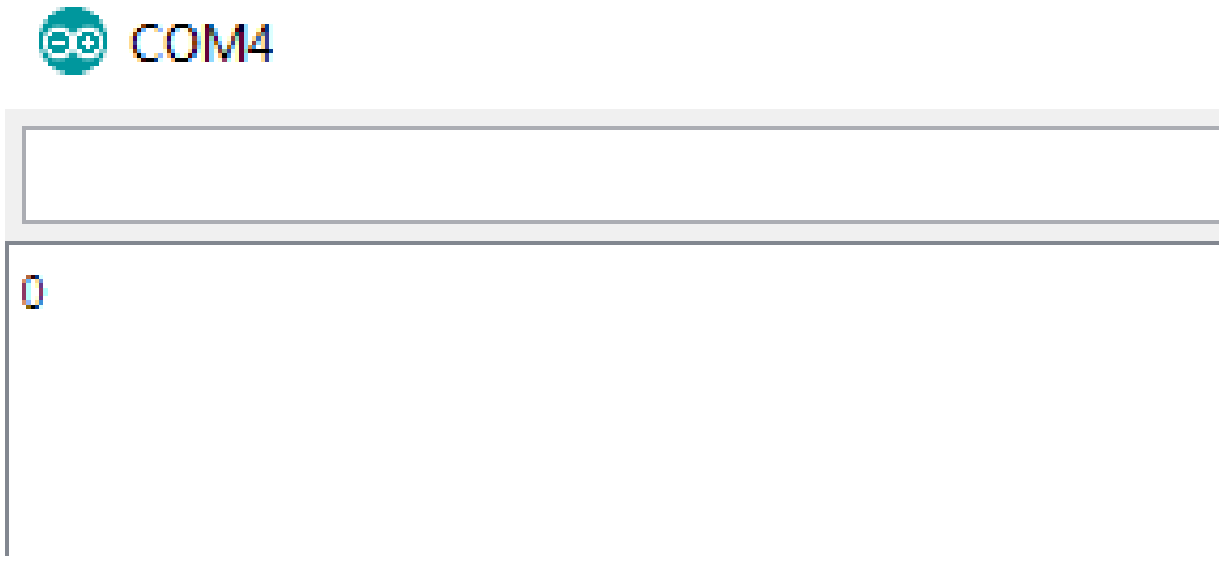
```
int count;
void setup() {
  Serial.begin(9600);
  count = 0;
  Serial.println(count);




}
void loop() {
}
```



Using a Variable: int

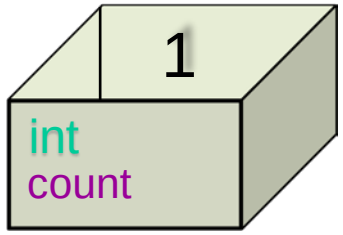
```
int count;
void setup() {
  Serial.begin(9600);
  count = 0;
  Serial.println(count);
  count = 1;




}
void loop() {
}
```

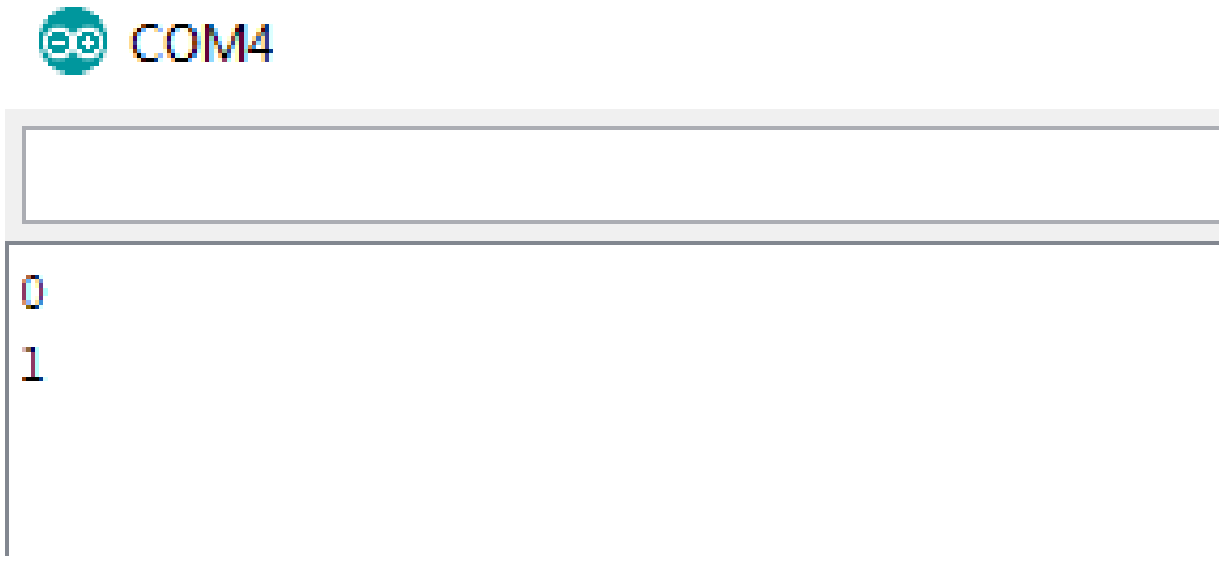




Using a Variable: int

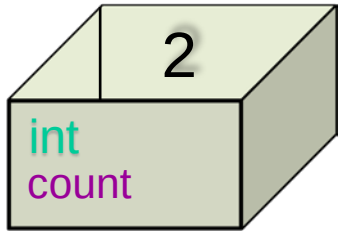
```
int count;
void setup() {
  Serial.begin(9600);
  count = 0;
  Serial.println(count);
  count = 1;
  Serial.println(count);



}
void loop() {
}
```

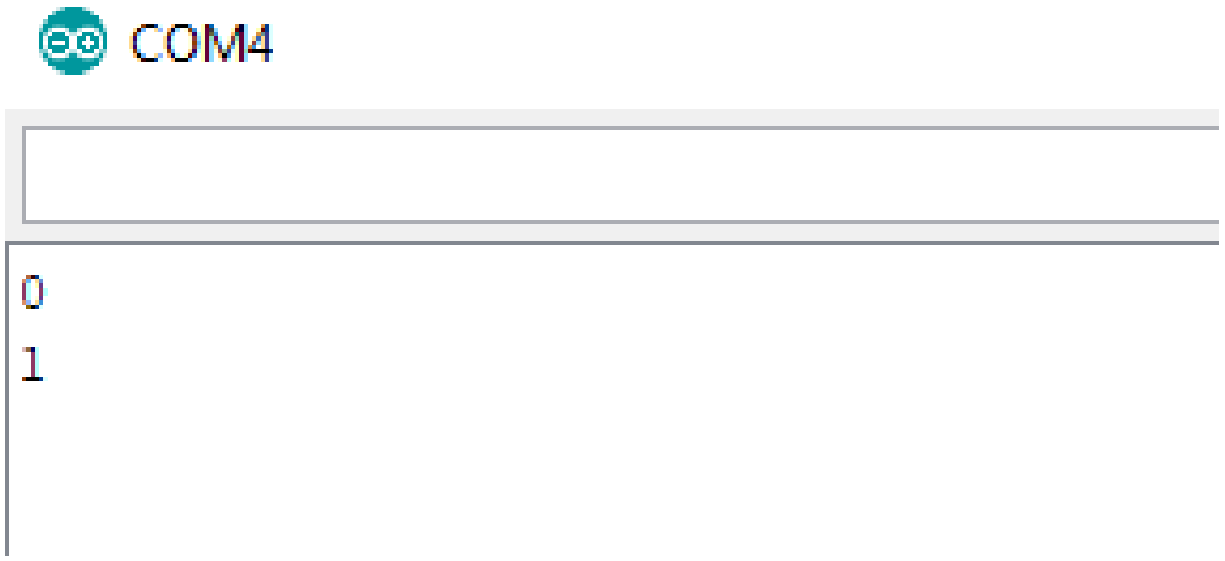




Using a Variable: int

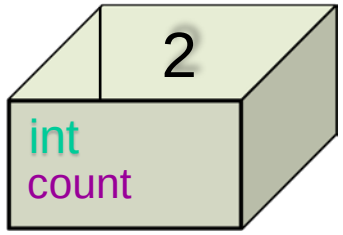
```
int count;
void setup() {
  Serial.begin(9600);
  count = 0;
  Serial.println(count);
  count = 1;
  Serial.println(count);
  count = 2;



}
void loop() {
}
```

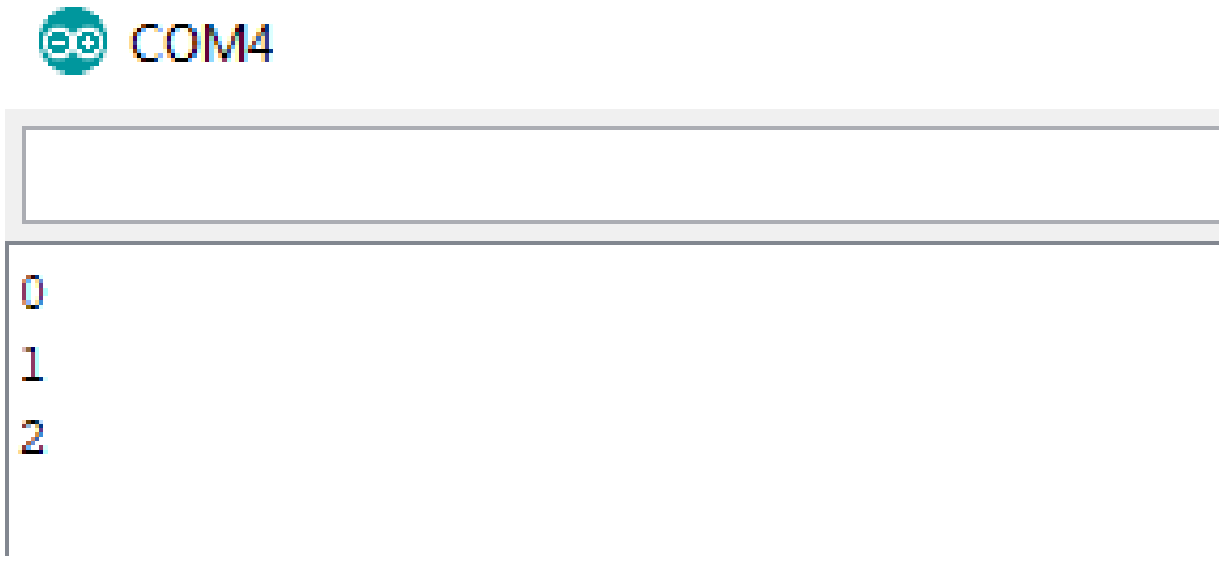




Using a Variable: int

```
int count;
void setup() {
  Serial.begin(9600);
  count = 0;
  Serial.println(count);
  count = 1;
  Serial.println(count);
  count = 2;
  Serial.println(count);

}
void loop() {
}
```

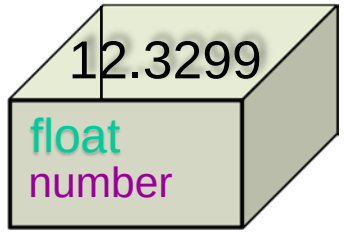


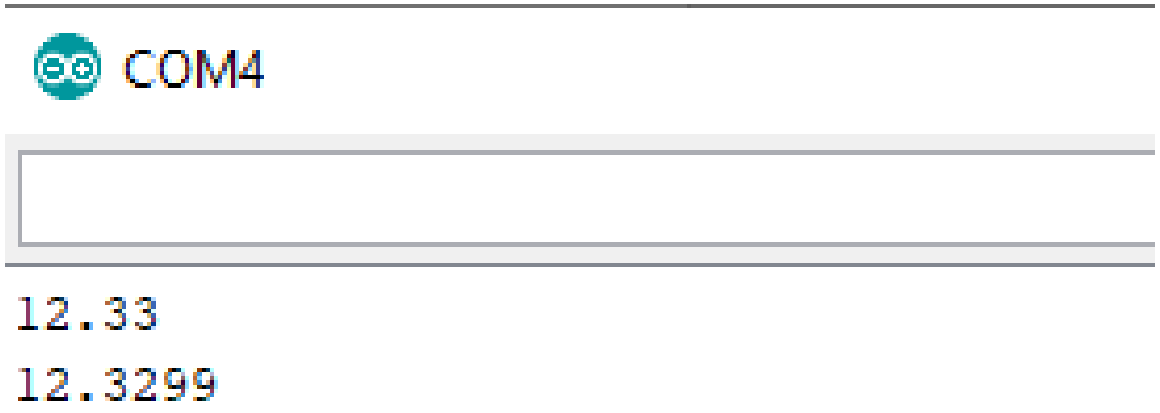


Using a Variable: float

- First println() automatically rounds the number off to two decimal places.
- In the second println(), the number of decimal places is specified as **4** by passing a second parameter value of **4** to the println() function.

```
float number = 12.3299;
void setup() {
   Serial.begin(9600);
   Serial.println(number);
   Serial.println(number, 4);
}

void loop() {
}
```

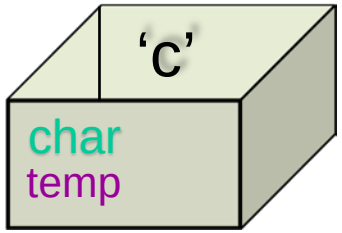




char

- char is meant to hold 1 ASCII character
  - see https://www.asciitable.com/

char temp = ‘c’;



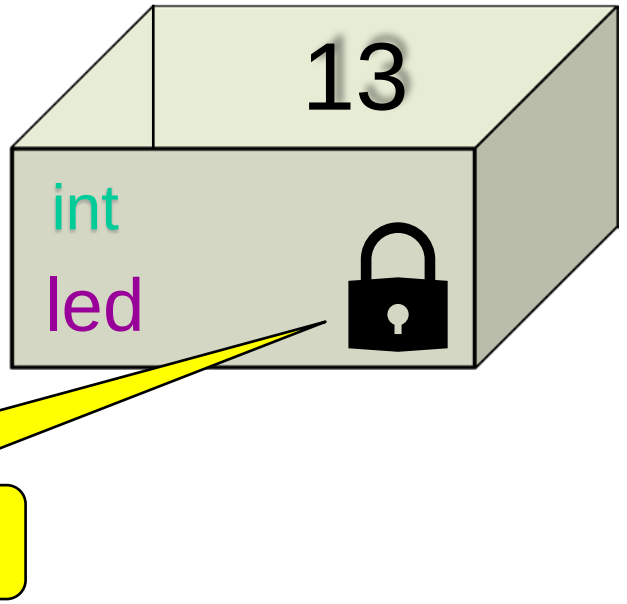
0	NUL	1	SOH	2	STX	3	ETX	4	EOT	5	ENQ	6	ACK	7	BEL
8	BS	9	HT	10	NL	11	VT	12	NP	13	CR	14	SO	15	SI
16	DLE	17	DC1	18	DC2	19	DC3	20	DC4	21	NAK	22	SYN	23	ETB
24	CAN	25	EM	26	SUB	27	ESC	28	FS	29	GS	30	RS	31	US
32	SP	33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	DEL

Constants

- Constants
  - integer constants
  - floating-point constants
  - character constants
  - string constants
  - enumeration constants
- Naming constants
  - Use the const qualifier

Use named constants for values that won't change while the program is running.

```
const int led = 13;
```



Naming Variables

- An identifier is a sequence of letters and digits
  - The first character must be a letter
    - The underscore character _ counts as a letter
    - Upper and lower case letters are different

- C reserved keywords cannot be used as identifiers!

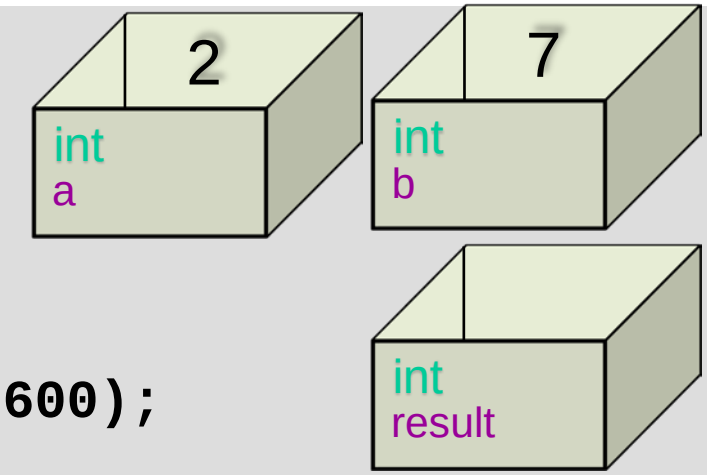
Use meaningful names when naming, *i.e.*, a name that describes the purpose of the entity

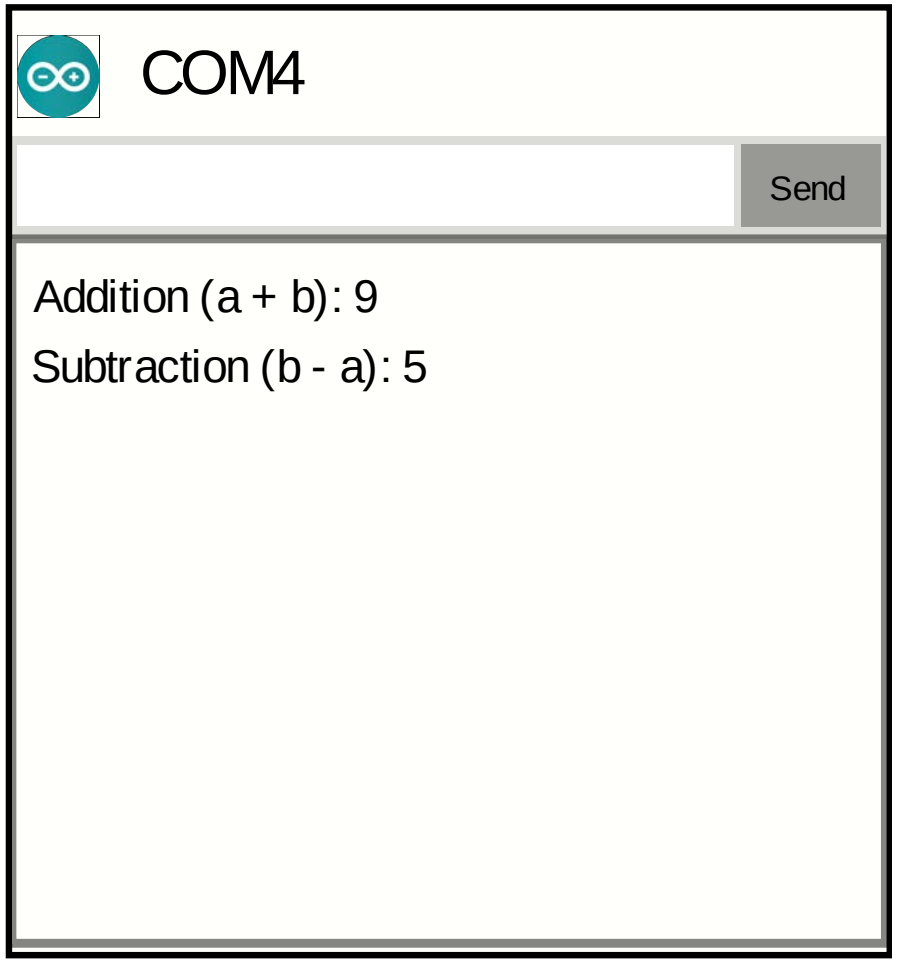
Examples

counter	Valid: consists of letters
_Temp_variable_2	Valid: consists of letters and digits
1myVariable	Invalid: first character is not a letter
steps{2}	Invalid: uses non-letter and non-digit characters
continue	Invalid: reserved word

Arithmetic Operators: + - * / %

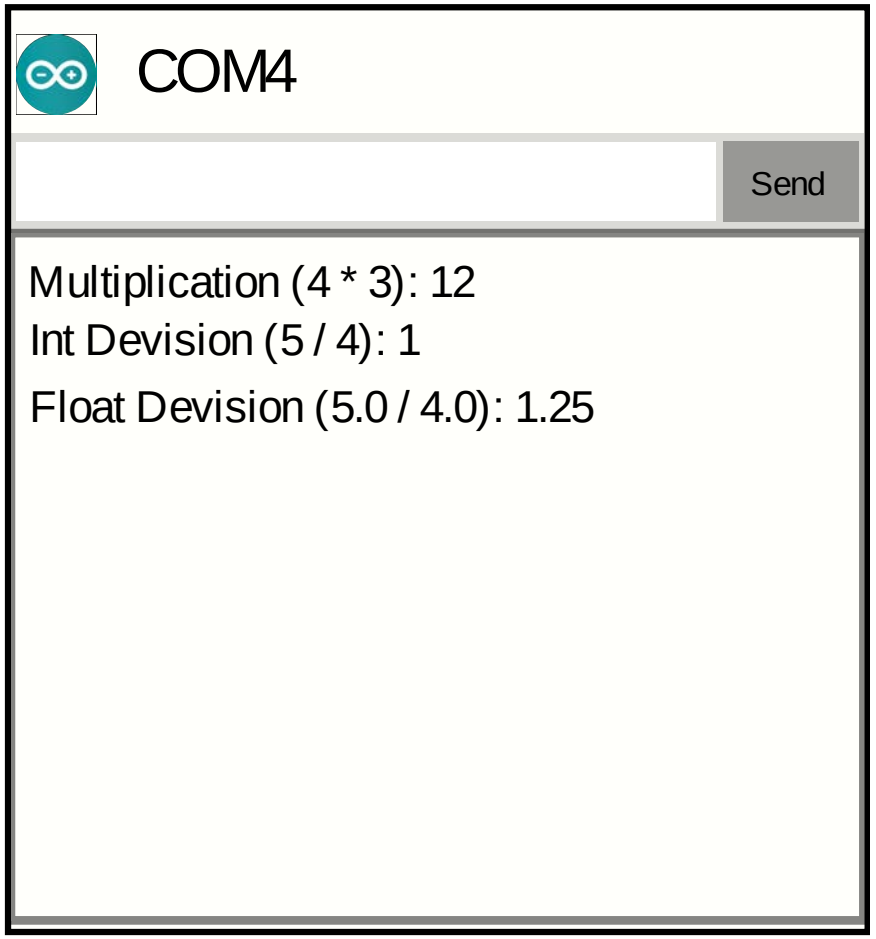
```
void setup() {
   int a = 2;
   int b = 7;
   int result;

   Serial.begin(9600);

   Serial.print("Addition (a + b): ");
   result = a + b;
   Serial.println(result);

   Serial.print("Subtraction (b - a): ");
   result = b - a;
   Serial.println(result);
}
```





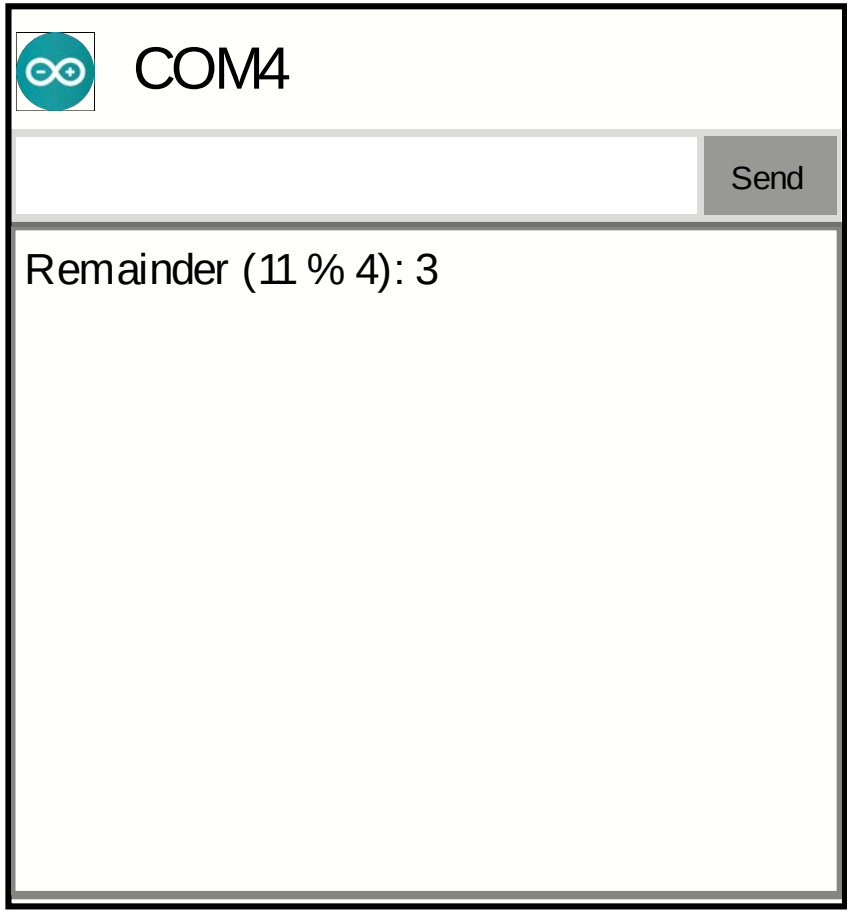
Arithmetic Operators: * /

```
void setup() {
   int result;
   float result_fl;
   Serial.begin(9600);

   Serial.print("Multiplication (4 * 3): ");
   result = 4 * 3;
   Serial.println(result);

   Serial.print("Int Division (5 / 4): ");
   result = 5 / 4;
   Serial.println(result);

   Serial.print("Float Division (5.0 / 4.0): ");
   result_fl = 5.0 / 4.0;
   Serial.println(result_fl);
}
```



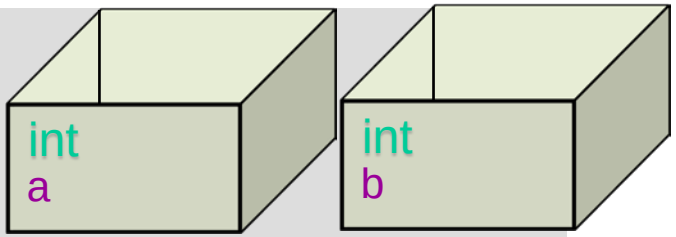
Arithmetic Operators: modulo (%)

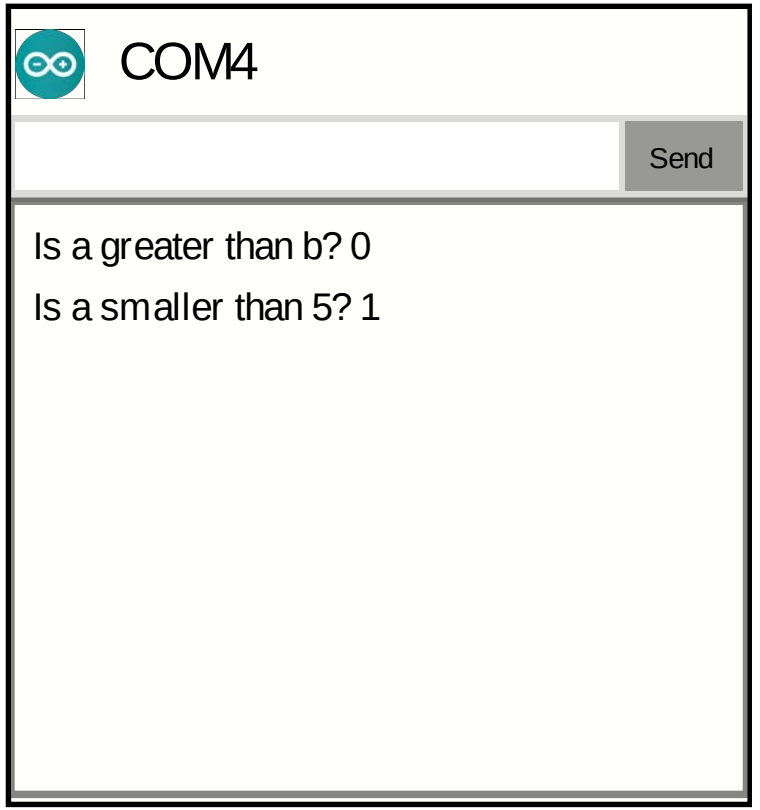
```
void setup() {
   int result;
   Serial.begin(9600);
   Serial.print("Remainder (11 % 4): ");
   result = 11 % 4;
   Serial.println(result);
}
```



Relational Operators: < > <= >= != ==

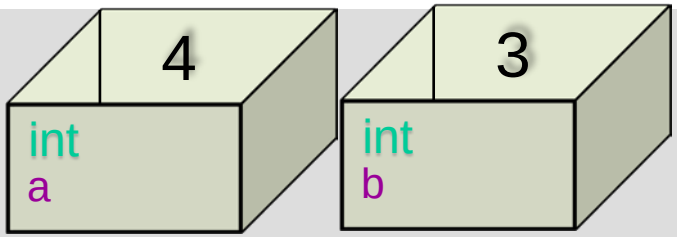
```
void setup() {
   int a = 2;
   int b = 3;
   Serial.begin(9600);
   Serial.print("Is a greater than b? ");
   Serial.println(a > b);

   Serial.print("Is a samller than 5? ");
   Serial.println(a < 5);
}
void loop() {
}
```

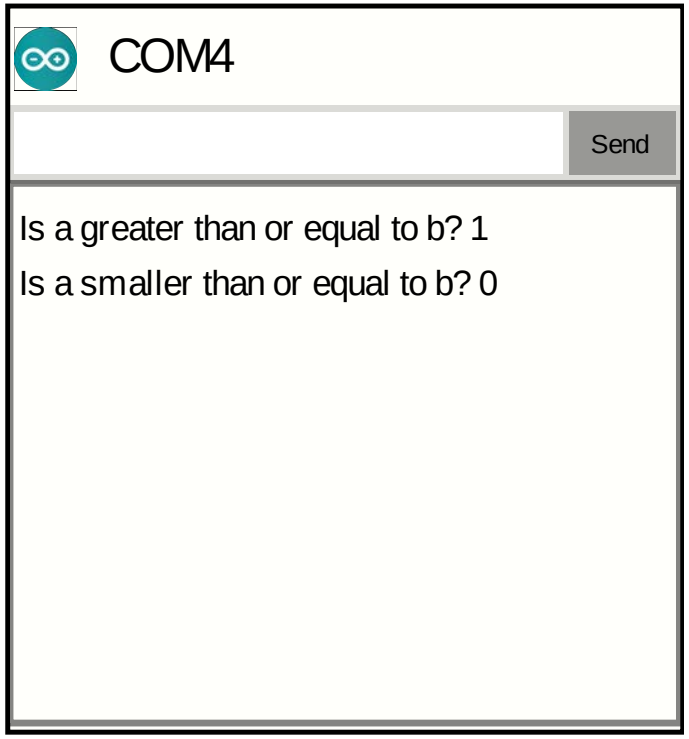




Relational Operators: <= >=

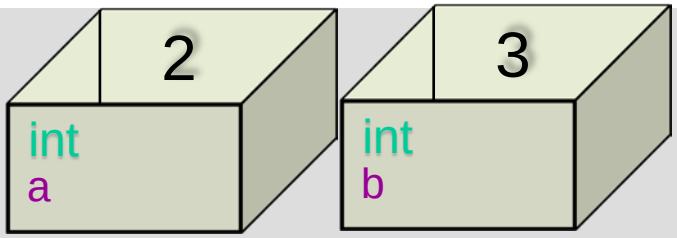
```
void setup() {
   int a = 4;
   int b = 3;

   Serial.print("Is a greater than or equal to b? ");
   Serial.println(a >= b);
   Serial.print("Is a smaller than or equal to b? ");
   Serial.println(a <= b);

}
void loop() {
}
```

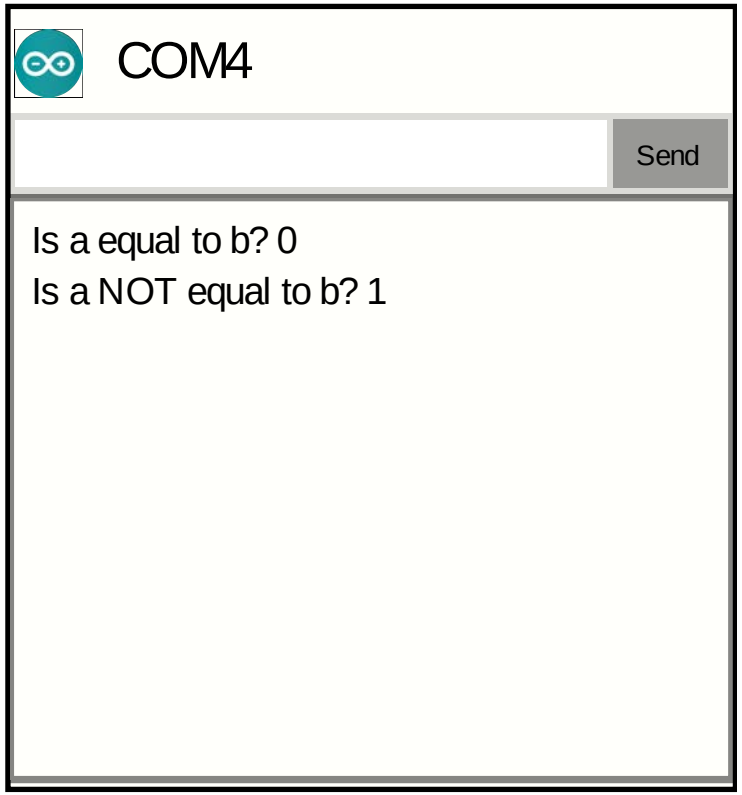




Relational Operators:!= ==

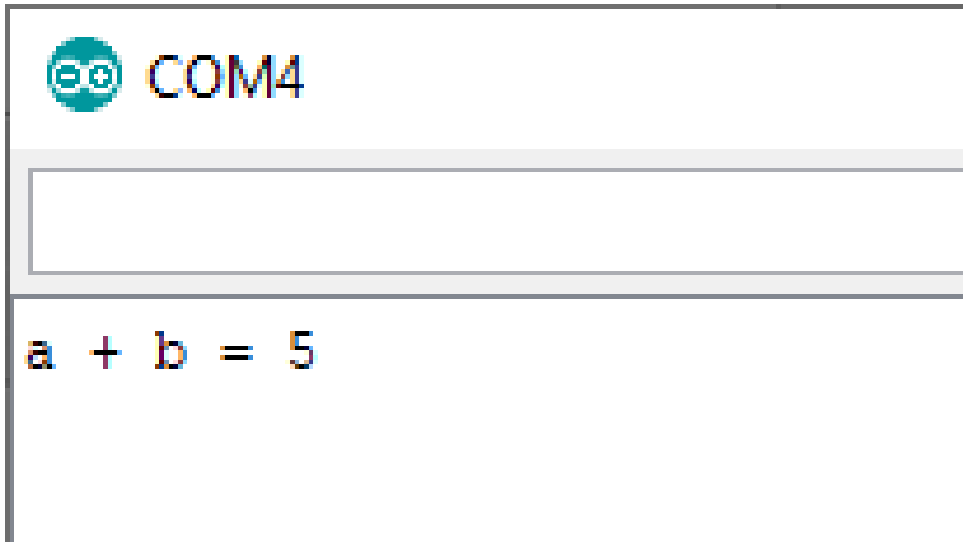
```
void setup() {
   int a = 2;
   int b = 3;

   Serial.print("Is a equal to b? ");
   Serial.println(a == b);

   Serial.print("Is a NOT equal to b? ");
   Serial.println(a != b);
}
void loop() {
}
```





Evaluating and Printing an Expression

```
void setup() {
    int a = 2;
    int b = 3;

    Serial.begin(9600);

    Serial.print("a + b = ");
    Serial.println(a + b);
}
void loop() {
}
```



Confusing = and ==

```
void setup() {
int a = 2;
int b = 3;

   Serial.begin(9600);

   Serial.print("a == b: ");
   Serial.println(a == b);

   Serial.print("a = b: ");
   Serial.println(a = b);
}
void loop() {
}
```