
Engineering Technology (ENGR 101)

Basic Programming

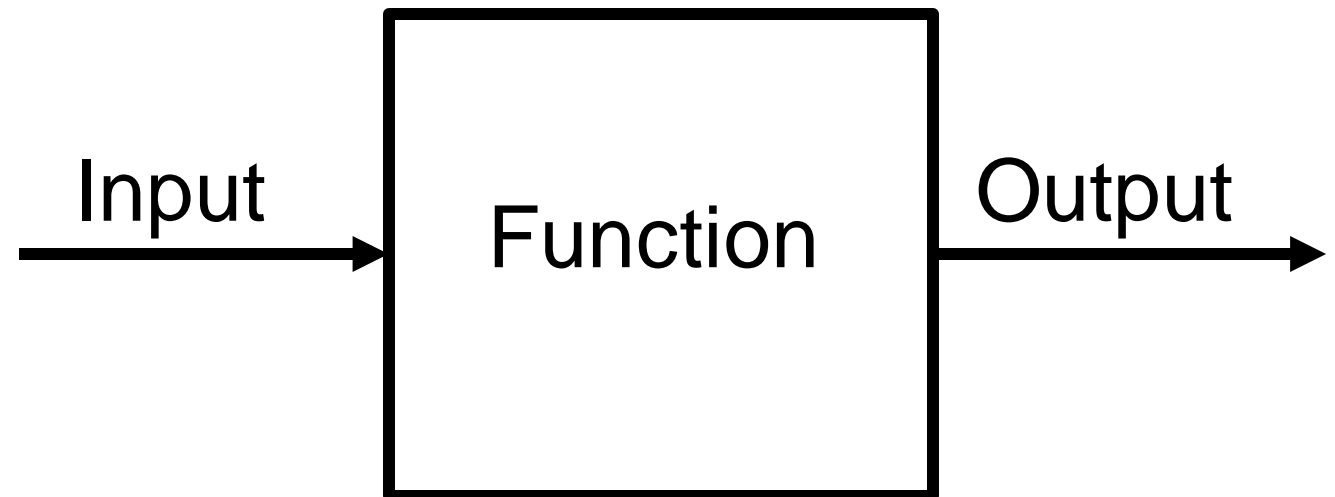
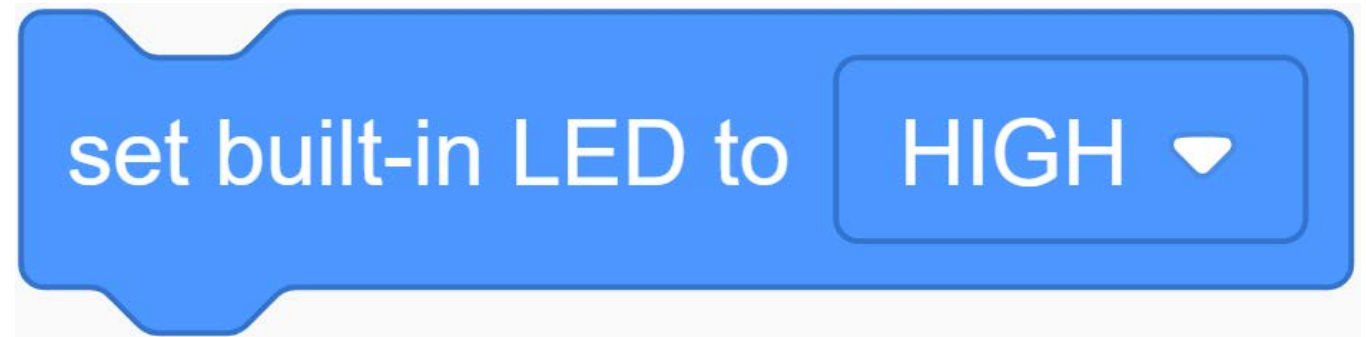
Mohammad Nekooei

School of Engineering and Computer Science

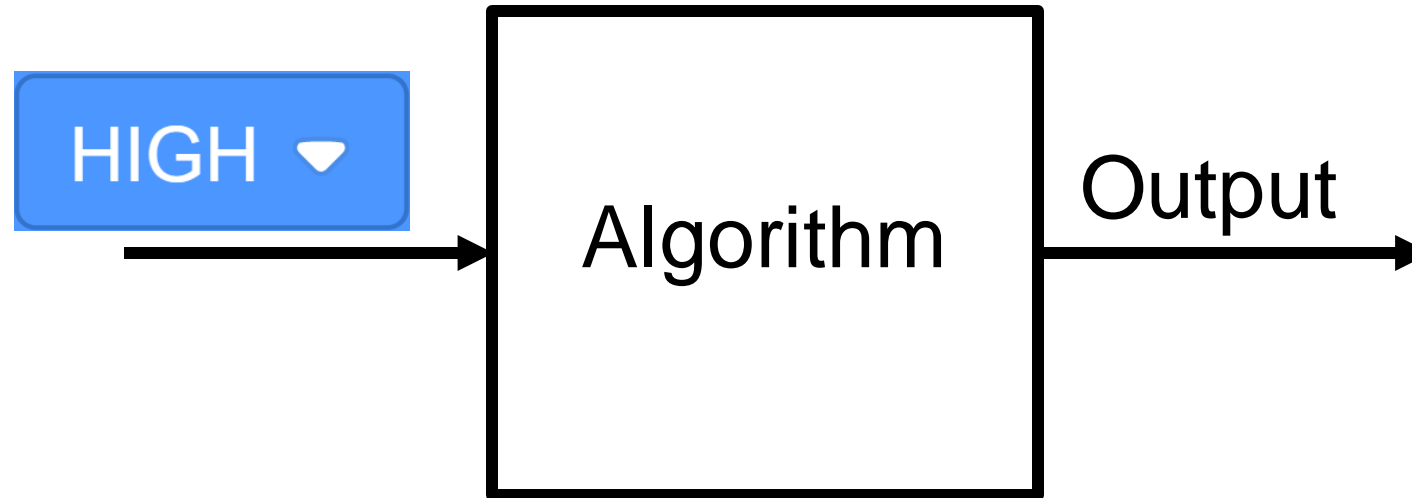
Victoria University of Wellington

Turn the built in LED on

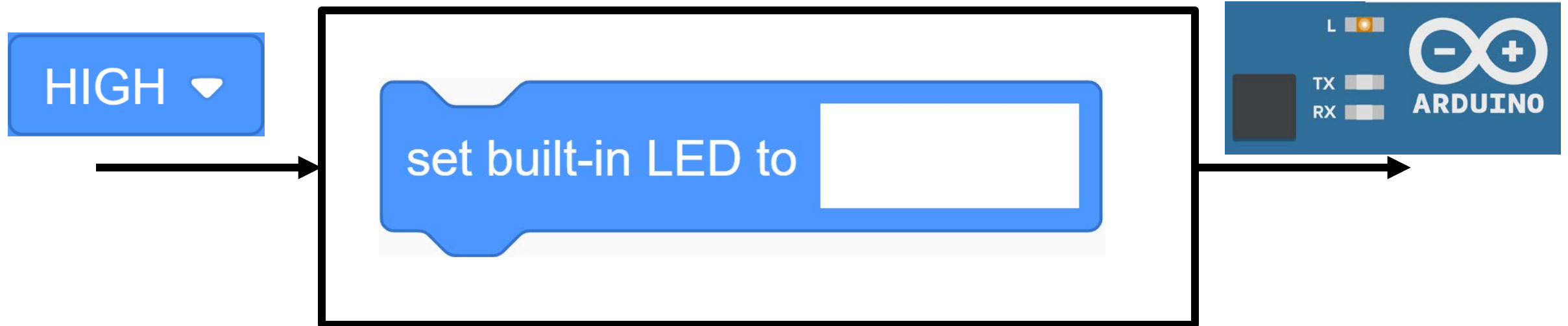
- This graphical block is a function
 - It takes an input (High voltage to turn the LED on)
 - The function uses instructions to solve the problem or to perform computation
 - The output is the result or side effect of the function



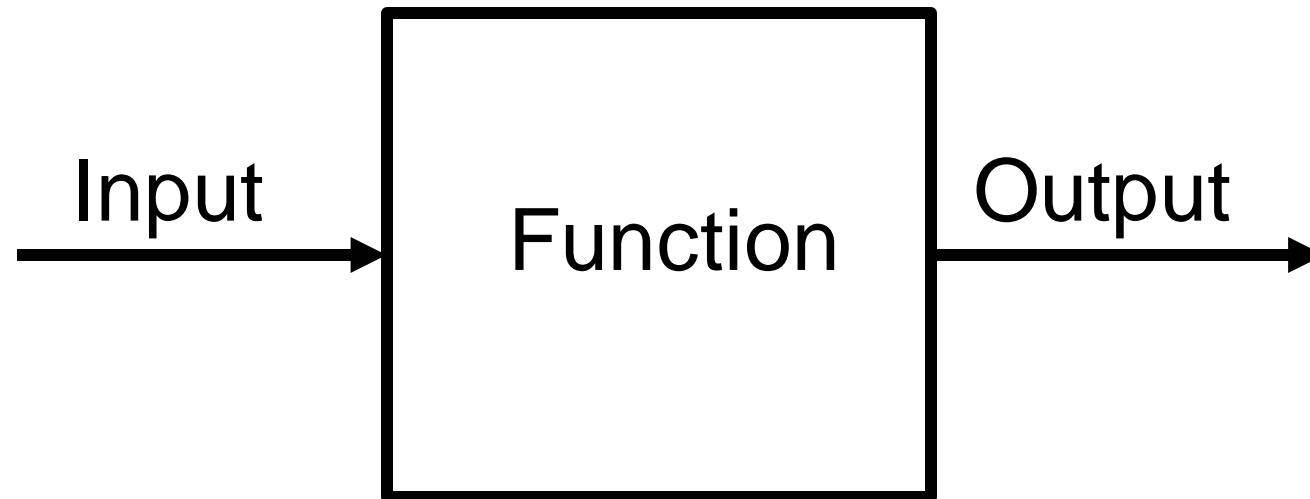
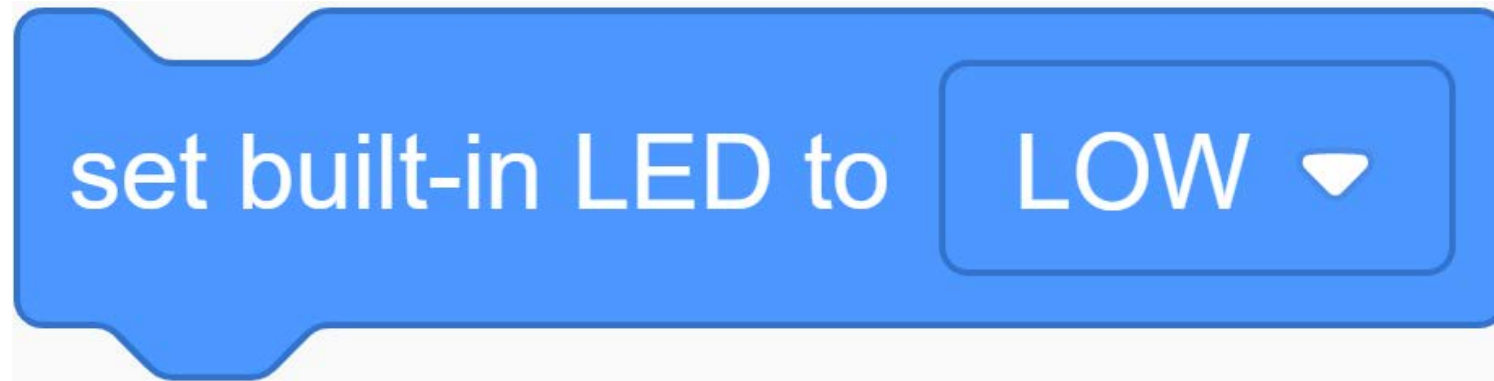
Turn the built in LED on



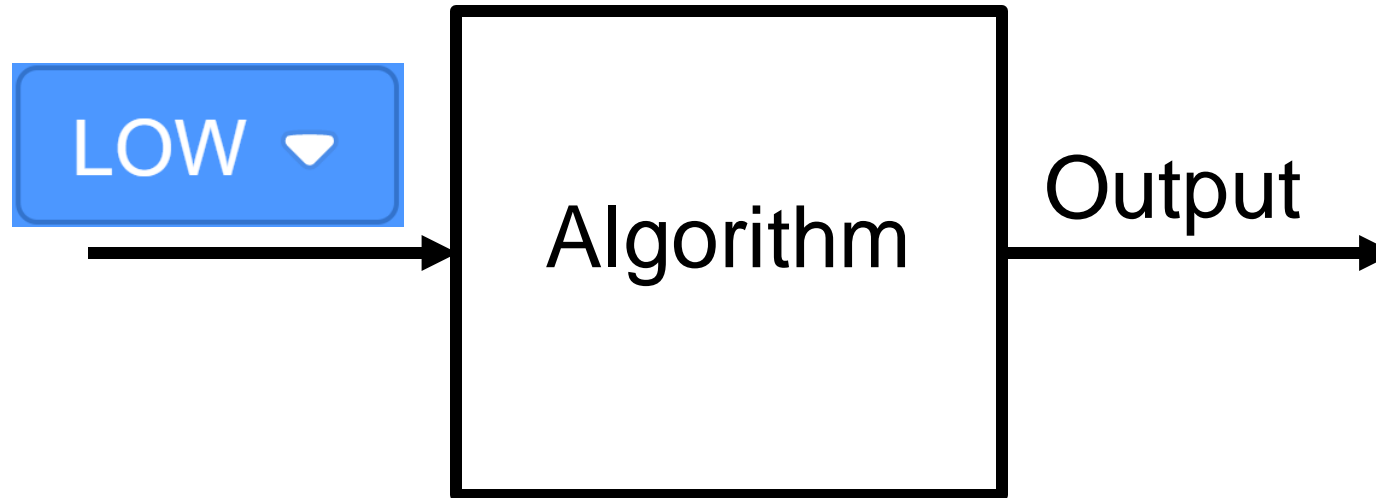
Turn the built in LED on



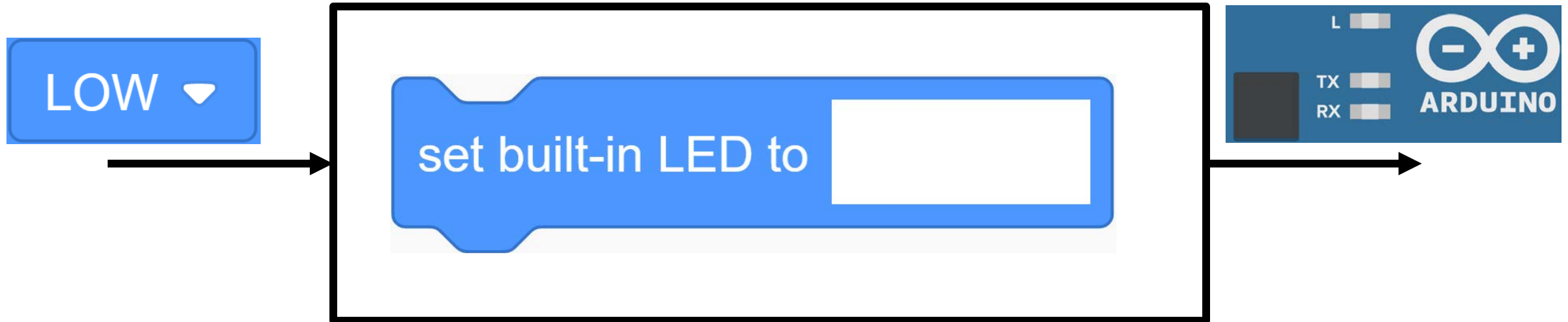
Turn the built in LED off



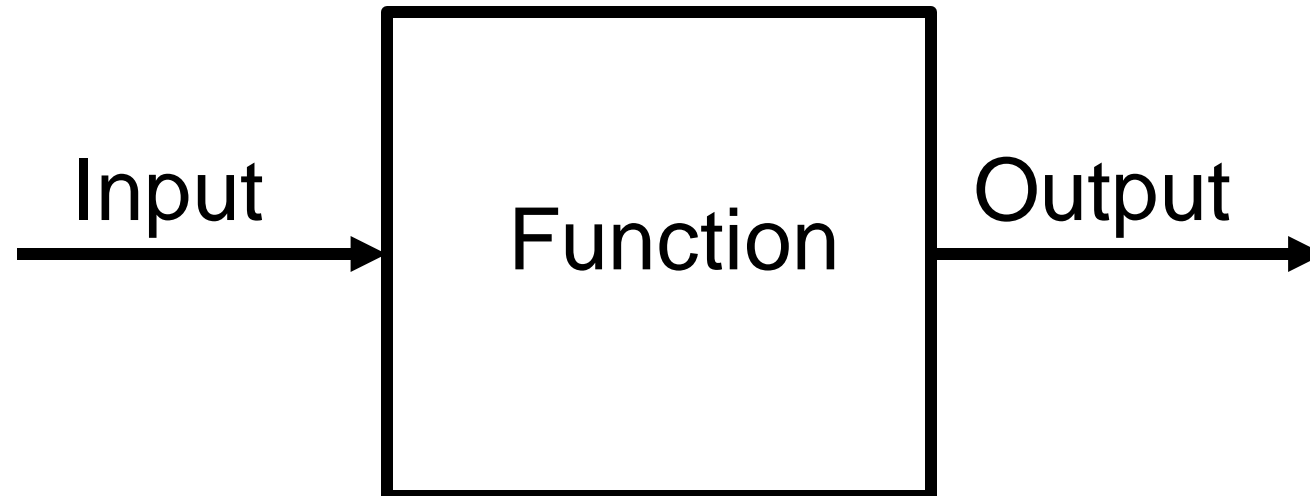
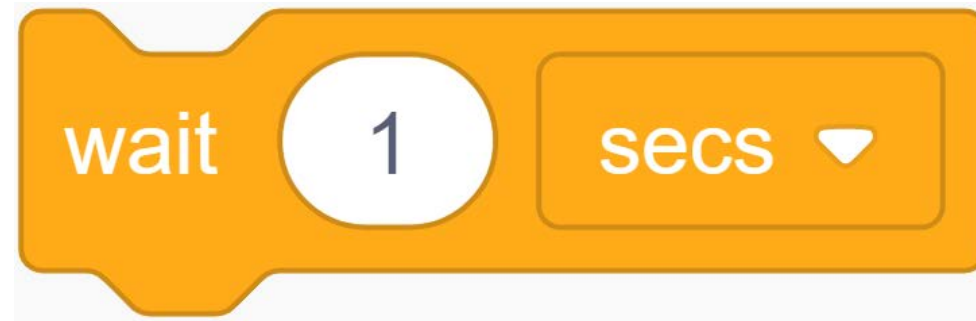
Turn the built in LED off



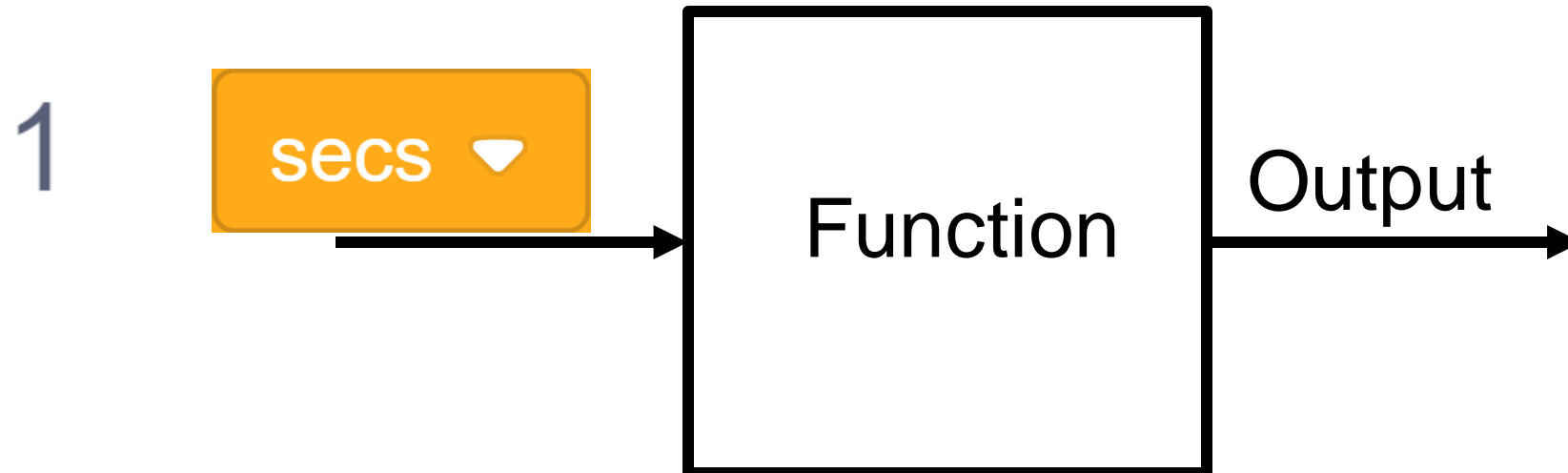
Turn the built in LED off



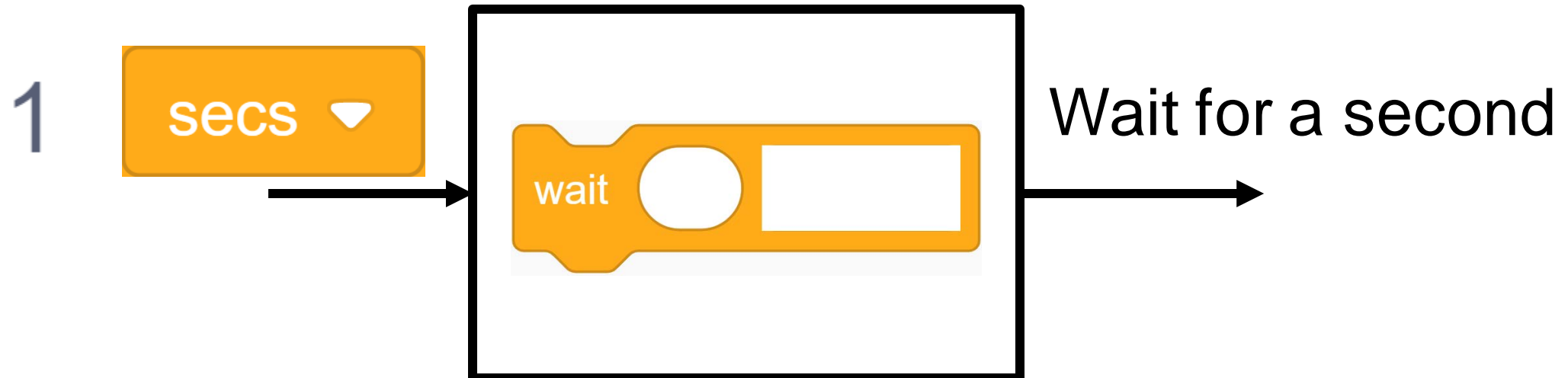
Wait for a second



Wait for a second



Wait for a second



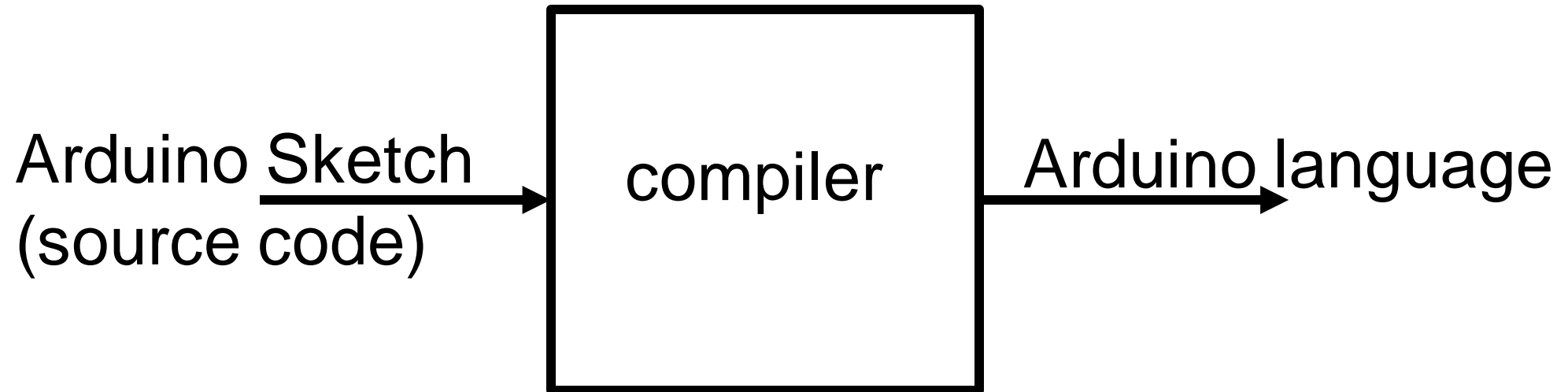
Arduino language

```
01111111 01000101 01001100 01000110 00000010 00000001 00000001 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000010 00000000 00111110 00000000 00000001 00000000 00000000 00000000
10110000 00000101 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
11010000 00010011 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 01000000 00000000 00111000 00000000
00001001 00000000 01000000 00000000 00100100 00000000 00100001 00000000
00000110 00000000 00000000 00000000 00000101 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
00111000 00000010 00000000 00000000 00000000 00000000 00000000 00000000
```

...

Compiler

- Translates program to Arduino language



Source code produced by TinkerCad

The image shows the TinkerCad interface with a block-based sketch on the left and its generated C++ source code on the right. The source code is highlighted with a red border.

Block-based Sketch:

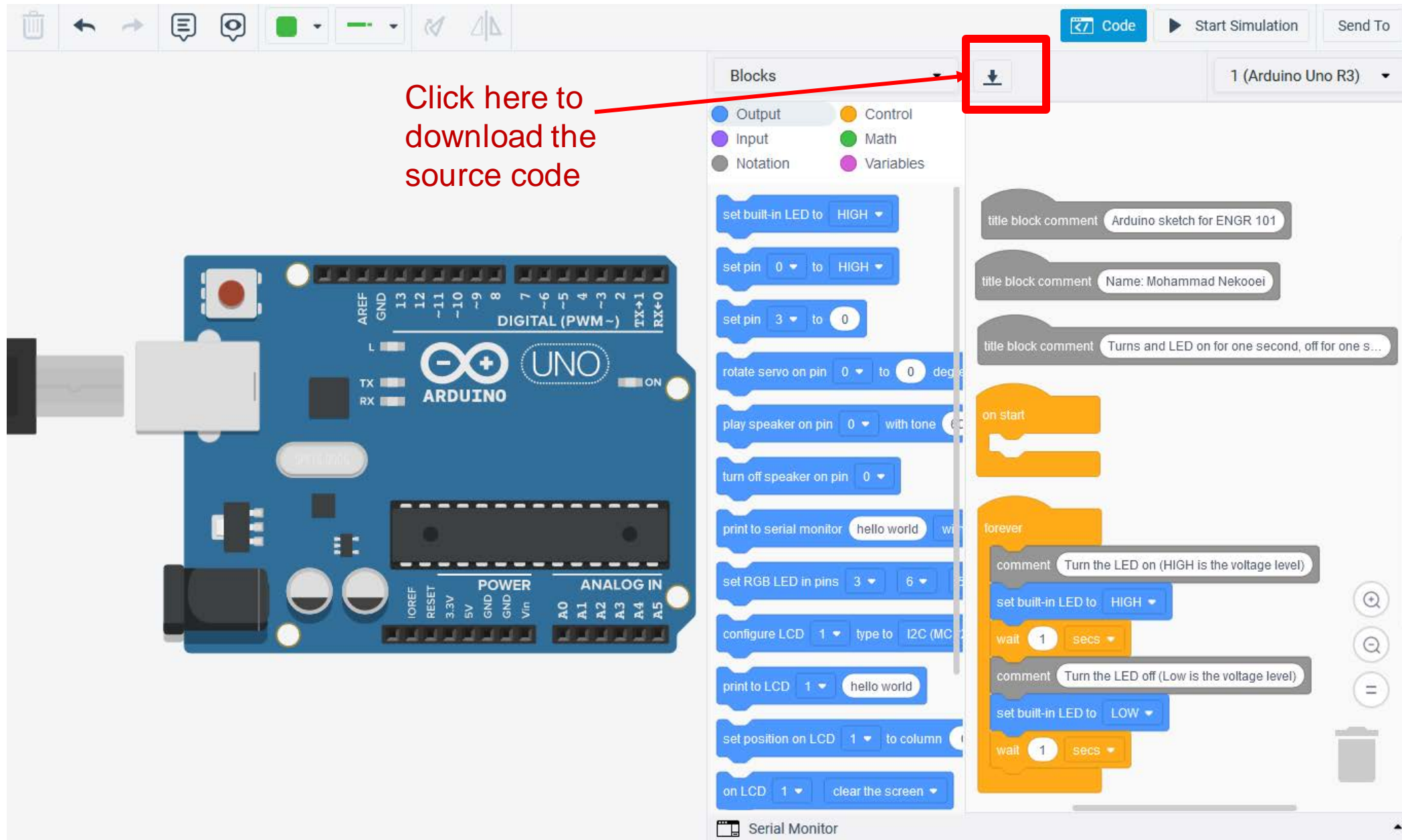
- title block comment: Arduino sketch for ENGR 101
- title block comment: Name: Mohammad Nekooei
- title block comment: Turns and LED on for one second, off for one s...
- on start
- forever loop:
 - comment: Turn the LED on (HIGH is the voltage level)
 - set built-in LED to HIGH
 - wait 1 secs
 - comment: Turn the LED off (Low is the voltage level)
 - set built-in LED to LOW
 - wait 1 secs

Generated C++ Source Code:

```

1 // C++ code
2 //
3 /*
4  Arduino sketch for ENGR 101
5
6  Name: Mohammad Nekooei
7
8  Turns and LED on for one second, off for one
9  second
10 */
11
12 void setup()
13 {
14   pinMode(LED_BUILTIN, OUTPUT);
15 }
16
17 void loop()
18 {
19   // Turn the LED on (HIGH is the voltage level)
20   digitalWrite(LED_BUILTIN, HIGH);
21   delay(1000); // Wait for 1000 millisecond(s)
22   // Turn the LED off (Low is the voltage level)
23   digitalWrite(LED_BUILTIN, LOW);
24   delay(1000); // Wait for 1000 millisecond(s)
25 }
  
```

Download source code from TinkerCad



Click here to download the source code

The screenshot displays the TinkerCad workspace. On the left is a 3D model of an Arduino Uno R3 board. The center panel shows a block-based code editor with various blocks for setting pins, controlling LEDs, and printing text. The right panel shows a code editor with the corresponding C++ code. A red box highlights the download icon (a downward arrow) in the top right corner of the workspace, with a red arrow pointing to it from the text instruction.

Blocks

- Output
- Input
- Notation
- Control
- Math
- Variables

Code

Start Simulation

Send To

1 (Arduino Uno R3)

title block comment Arduino sketch for ENGR 101

title block comment Name: Mohammad Nekooei

title block comment Turns and LED on for one second, off for one s...

on start

comment Turn the LED on (HIGH is the voltage level)

set built-in LED to HIGH

wait 1 secs

comment Turn the LED off (Low is the voltage level)

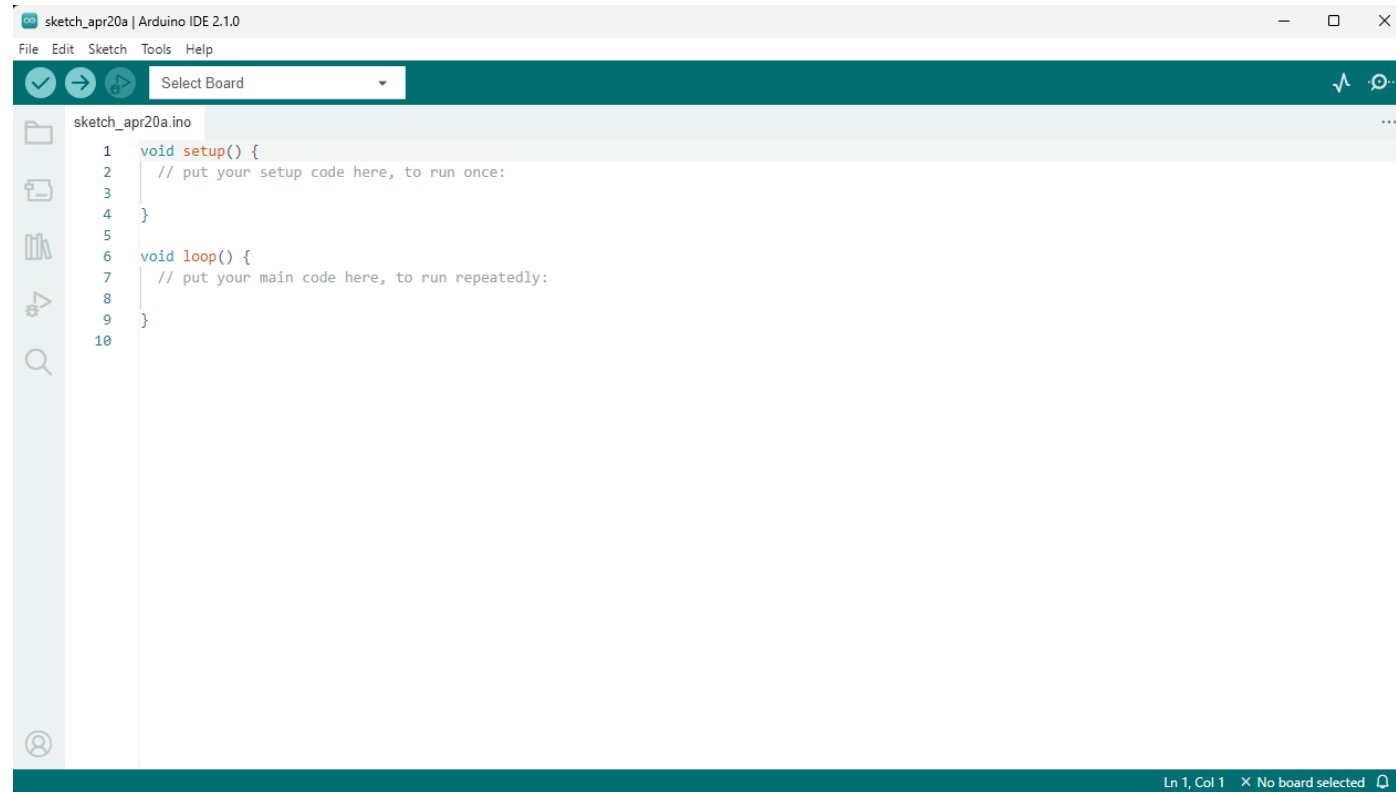
set built-in LED to LOW

wait 1 secs

Serial Monitor

Arduino Programming Environment

- A software environment
 - Program editor
(Programs written in the C language)
 - Compiler:
(Translates program to Arduino language)
 - Uploader
(Sends program to Arduino over USB)
 - Debugger
(Helps find errors in program)



The screenshot displays the Arduino IDE 2.1.0 interface. The window title is "sketch_apr20a | Arduino IDE 2.1.0". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checkmark, back, forward, and a "Select Board" dropdown menu. The main editor area shows a sketch named "sketch_apr20a.ino" with the following code:

```
1 void setup() {  
2   // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8  
9 }  
10
```

The status bar at the bottom right indicates "Ln 1, Col 1" and "No board selected".

Arduino Terminology

- “sketch” – a program you write to run on an Arduino board
- “pin” – an input or output connected to something. e.g., output to an LED, input from a knob.
- “digital pin” – value is either HIGH or LOW. (aka on/off, one/zero) e.g., switch state
- “analog pin” – value ranges, usually from 0-255. e.g., LED brightness, motor speed, etc.

Arduino Integrated Development Environment (IDE)

- Like a text editor
- View/write/edit sketches
- But then you program them into hardware
- **Verify:** Compiles code, checks for errors
- **Upload:** Compiles code, checks for errors, uploads to board
- **Serial Monitor:** Opens a window to communicate with the board
- More information:
<https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2>

Compile and Upload to Arduino (points to the Upload icon)

Verify sketch (points to the Verify icon)

Open sketch (points to the Open Sketch icon)

Board Manager (points to the Board Manager icon)

Library Manager (points to the Library Manager icon)

Debugger (points to the Debugger icon)

Search (points to the Search icon)

The Editor (points to the code editor area)

Error Console (points to the Output window)

Current Arduino model (points to the 'Arduino Uno' dropdown in the status bar)

Current USB port (points to the '[not connected]' dropdown in the status bar)

Serial Plotter (points to the Serial Plotter icon)

Open Serial Monitor (points to the Serial Monitor icon)

Tab menu (points to the tab menu icon)

```

1 void setup() {
2   // put your setup code here, to run once:
3   Serial.begin(9600);
4   Serial.println("Hello World!");
5 }
6
7 void loop() {
8   // put your main code here, to run repeatedly:
9
10 }
11

```

Compile Code

- Verify and Upload both compile
- Message window will show either completion message or error messages
- Error messages will show line numbers

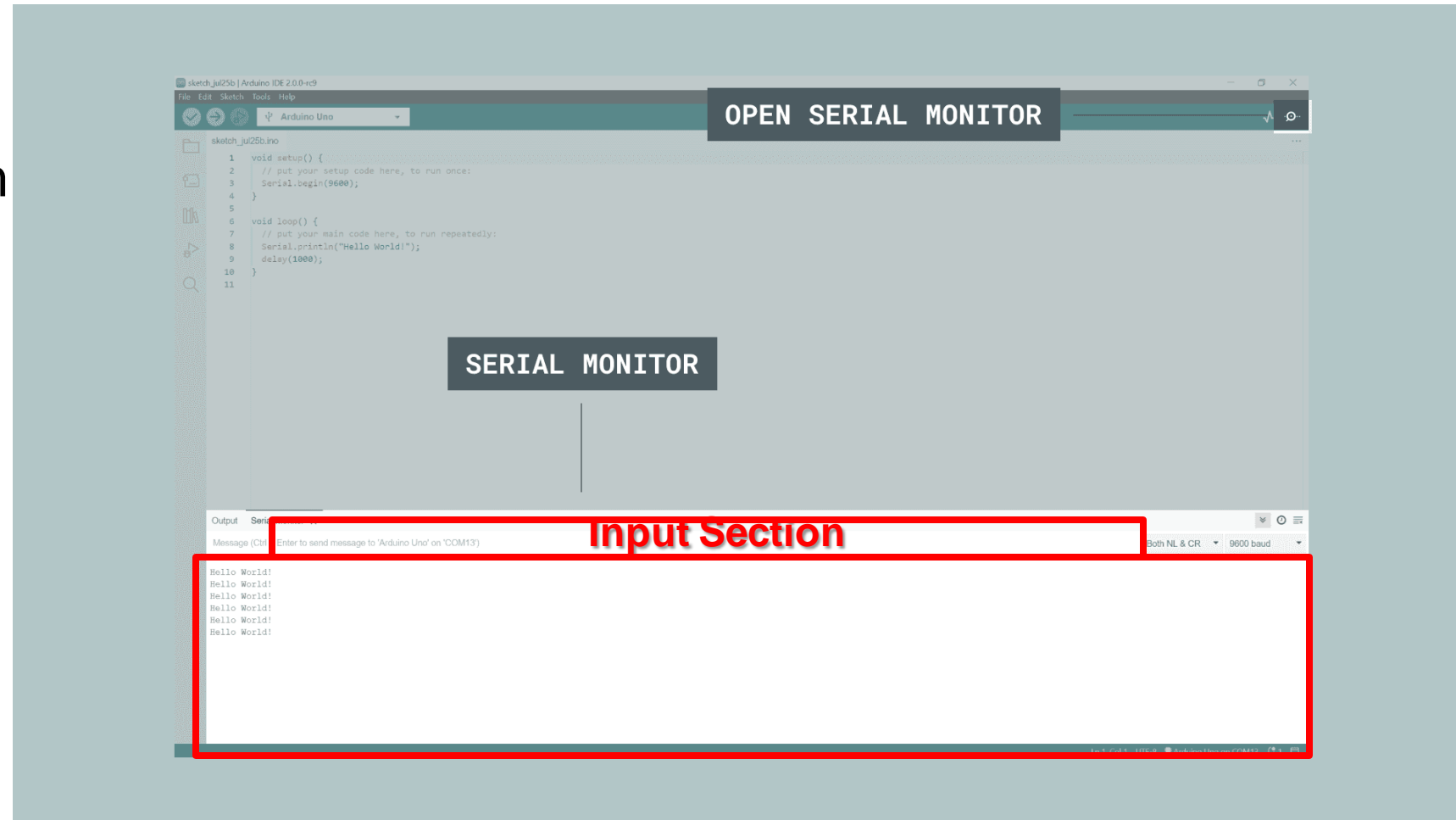
```
sketch_apr20a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Arduino Uno
sketch_apr20a.ino
1 void setup() {
2 // put your setup code here, to run once:
3 Serial.begin(9600);
4 Serial.println("Hello World!")
5 }
6
7 void loop() {
8 // put your main code here, to run repeatedly:
9
10 }
11
Output
C:\Users\nekoiei\AppData\Local\Temp\.arduinoIDE-unsave
}
^
exit status 1
Compilation error: expected ';' before '}' token
Ln 5, Col 1 Arduino Uno [not connected]
```

Error Message

Error line number

Serial Monitor

- Displays serial data sent from the Arduino
- Allows serial data to be sent to the Arduino from the keyboard
- Library functions in the serial library



Set Up the Arduino IDE

- Download the IDE

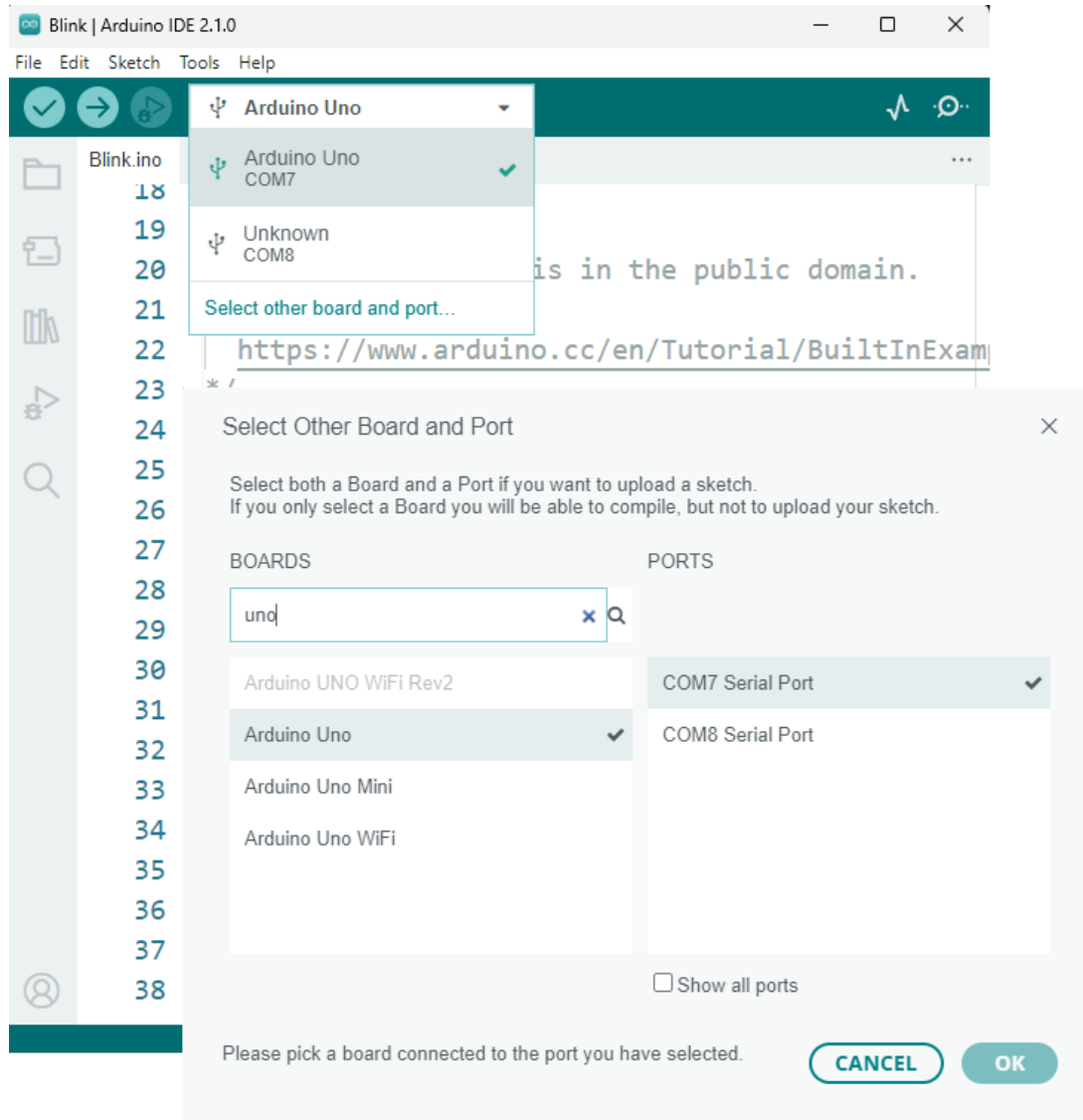
<https://www.arduino.cc/en/Main/Software>

- Easiest to run Windows Installer
- Also installs USB and other drivers
- Connect the board to your computer
 - Use USB cable
- Run the Arduino application
 - Starts the IDE
- Open the Blink example:
File>Examples>Basics>Blink

```
Blink | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Arduino Uno
Blink.ino
18 by COLBY NEWMAN
19
20 This example code is in the public domain.
21
22 https://www.arduino.cc/en/Tutorial/BuiltInExam
23 */
24
25 // the setup function runs once when you press r
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an ou
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again fo
32 void loop() {
33     digitalWrite(LED_BUILTIN, HIGH); // turn the
34     delay(1000); // wait for
35     digitalWrite(LED_BUILTIN, LOW); // turn the
36     delay(1000); // wait for
37 }
38
Ln 1, Col 1 Arduino Uno on COM7
```

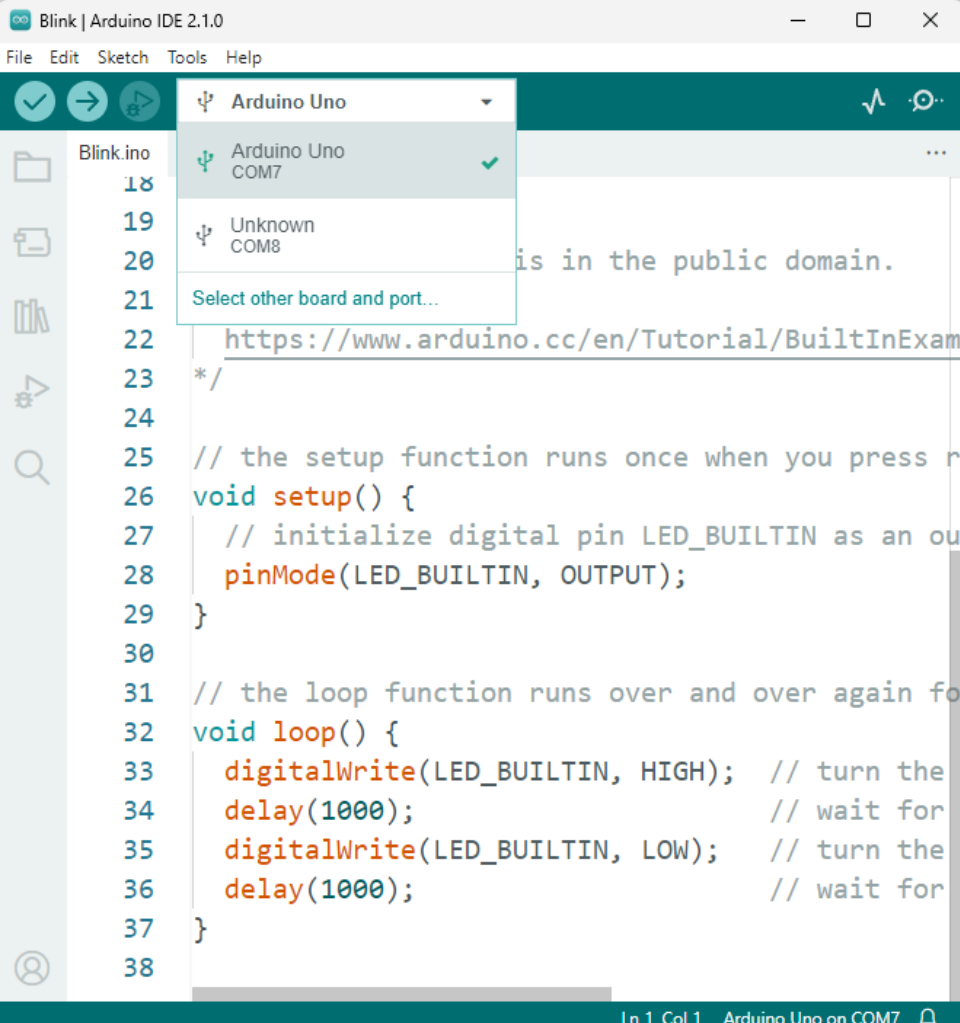
Set Up the Arduino IDE

- Select your Arduino in the tools>Board menu



Getting to Blink

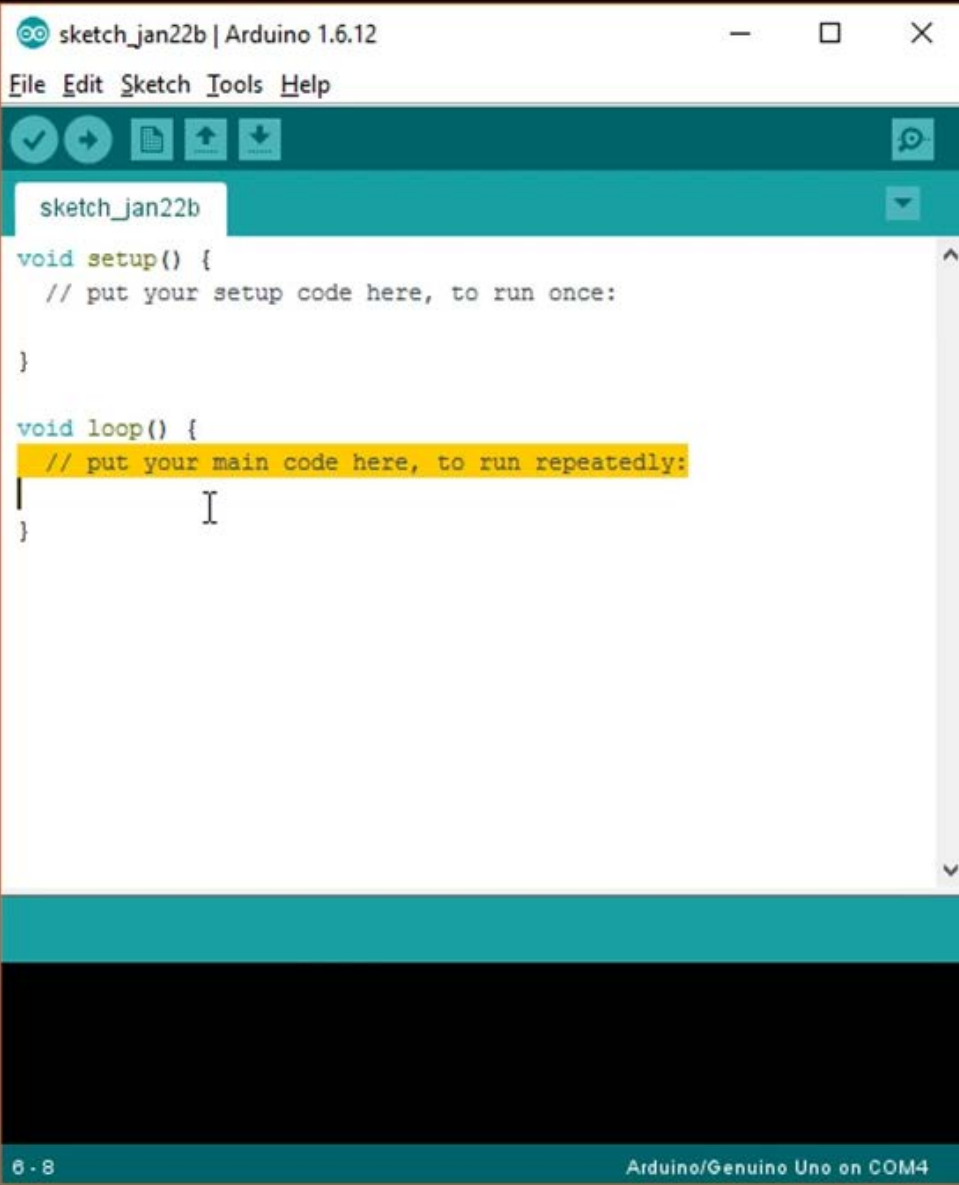
- Select your serial port in the tools>Port menu
 - There should be only one selection (COM3 or ..)
 - Check to make sure that the drivers are properly installed.
- Upload the program with the upload button
 - This writes the program onto the Flash of the Arduino
- The LED near pin 13 of the Arduino should blink



The screenshot shows the Arduino IDE 2.1.0 interface. The 'Tools' menu is open, and the 'Port' submenu is displayed. The 'Port' menu is currently set to 'Arduino Uno'. The 'Port' submenu shows 'Arduino Uno COM7' selected with a checkmark, and 'Unknown COM8' below it. A 'Select other board and port...' option is also visible. The main editor window shows the Blink.ino sketch with the following code:

```
18
19
20
21
22 https://www.arduino.cc/en/Tutorial/BuiltInExam
23 */
24
25 // the setup function runs once when you press r
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an ou
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again fo
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the
34   delay(1000); // wait for
35   digitalWrite(LED_BUILTIN, LOW); // turn the
36   delay(1000); // wait for
37 }
38
```

The status bar at the bottom indicates 'Ln 1, Col 1 Arduino Uno on COM7'.



The image shows a screenshot of the Arduino IDE interface. The window title is "sketch_jan22b | Arduino 1.6.12". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for a checkmark, a right arrow, a document, an upload arrow, a download arrow, and a refresh icon. The sketch name "sketch_jan22b" is displayed in the top bar. The code editor shows the following code:

```
void setup() {  
  // put your setup code here, to run once:  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  |  
  }  
}
```

The code in the `loop()` function is highlighted in yellow. The IDE status bar at the bottom shows "6 - 8" and "Arduino/Genuino Uno on COM4".

Run once ↓

Repeats forever ↻

Arduino IDE

- There are two special functions that are a part of every Arduino sketch
 - **setup()** is called once when the sketch starts. It's a good place to do setup tasks like setting pin modes or initialling libraries.
 - **loop()** function is called over and over and is the heart of most sketches.
- You need to include both functions in your sketch, even if you don't need them for anything.

Arduino IDE

A function is a block of reusable code that is used to perform an action.

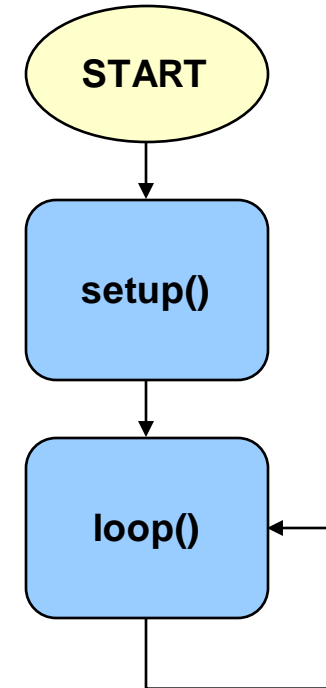
Indicates start of compound statement

```
void setup() {  
    // runs once
```

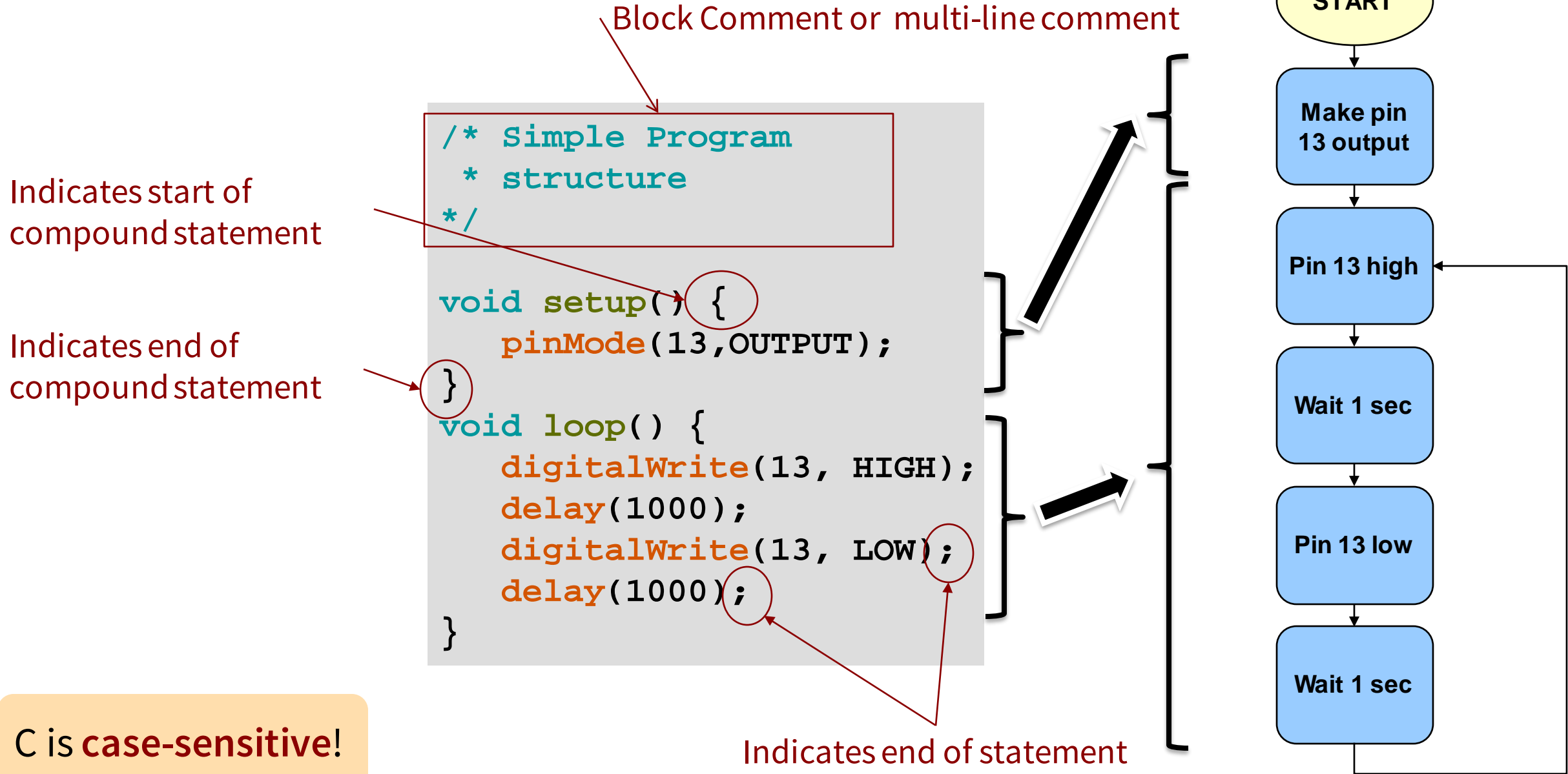
Indicates end of compound statement

```
}  
  
void loop() {  
    // repeats  
}
```

Single line comment



Arduino IDE



Program structure

```
/* Simple Program structure */  
int x;  
void setup()  
{  
  x = 1;  
}  
  
void loop()  
{  
  if(x == 1)  
  {  
    for( i = 0; i < 10; i++)  
    {  
      Serial.println(i);  
    }  
  }  
}
```

The diagram illustrates the program structure with nested blocks and arrows. The code is as follows:

```
/* Simple Program structure */  
int x;  
void setup()  
{  
  x = 1;  
}  
  
void loop()  
{  
  if(x == 1)  
  {  
    for( i = 0; i < 10; i++)  
    {  
      Serial.println(i);  
    }  
  }  
}
```

Red arrows indicate the flow of execution: from the start of the `setup()` function to the assignment `x = 1;`, from the start of the `loop()` function to the `if(x == 1)` condition, from the start of the `if` block to the `for` loop, and from the start of the `for` loop to the `Serial.println(i);` statement. Blue dashed boxes highlight the nested structure: the `setup()` block, the `loop()` block, the `if` block, and the `for` loop block.

Comments

```
/* Comment text */
```

- Compiler ignores everything from `/*` to `*/`

```
// Comment text
```

- Compiler ignores everything from `//` to the end of the line
- This commenting style originated from C++ and was adopted by C (C99 standard)

Hello world in Arduino IDE

Open Serial
Monitor Window

A function is a block of reusable code that is used to perform an action.

Initializes the speed of the serial port to 9600 baud rate

```
void setup() {
  Serial.begin(9600);
  Serial.println("Hello World!");
}

void loop() {
}
```

prints data to serial bus

- **setup()** function is run first and run only once, every time that the sketch is run.
- **loop()** function will run continuously from top to bottom and then back to the top.

The screenshot shows the Arduino IDE 2.1.0 interface. The sketch editor displays the following code:

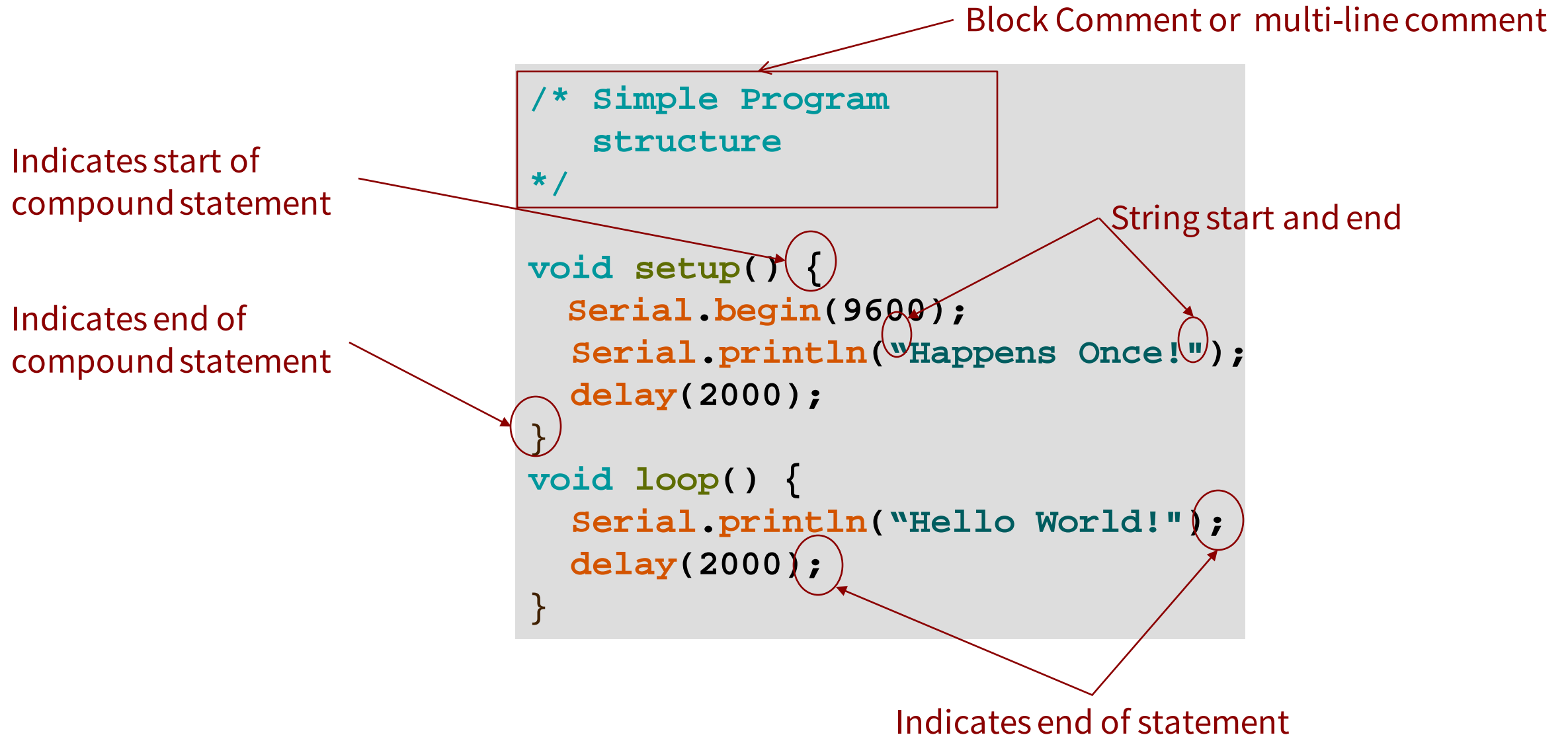
```
1 void setup()
2 {
3   //put your setup code here, to once:
4   Serial.begin(9600);
5   Serial.println("Hello World!");
6 }
7
8 void loop()
9 {
10  // put your main code here, to run repeatedly:
11 }
```

The Serial Monitor window is open, showing the output "Hello World!". The baud rate is set to 9600 baud. The status bar at the bottom indicates "Ln 10, Col 49 Arduino Uno on COM5".

Baud Rate

Program
Output Text

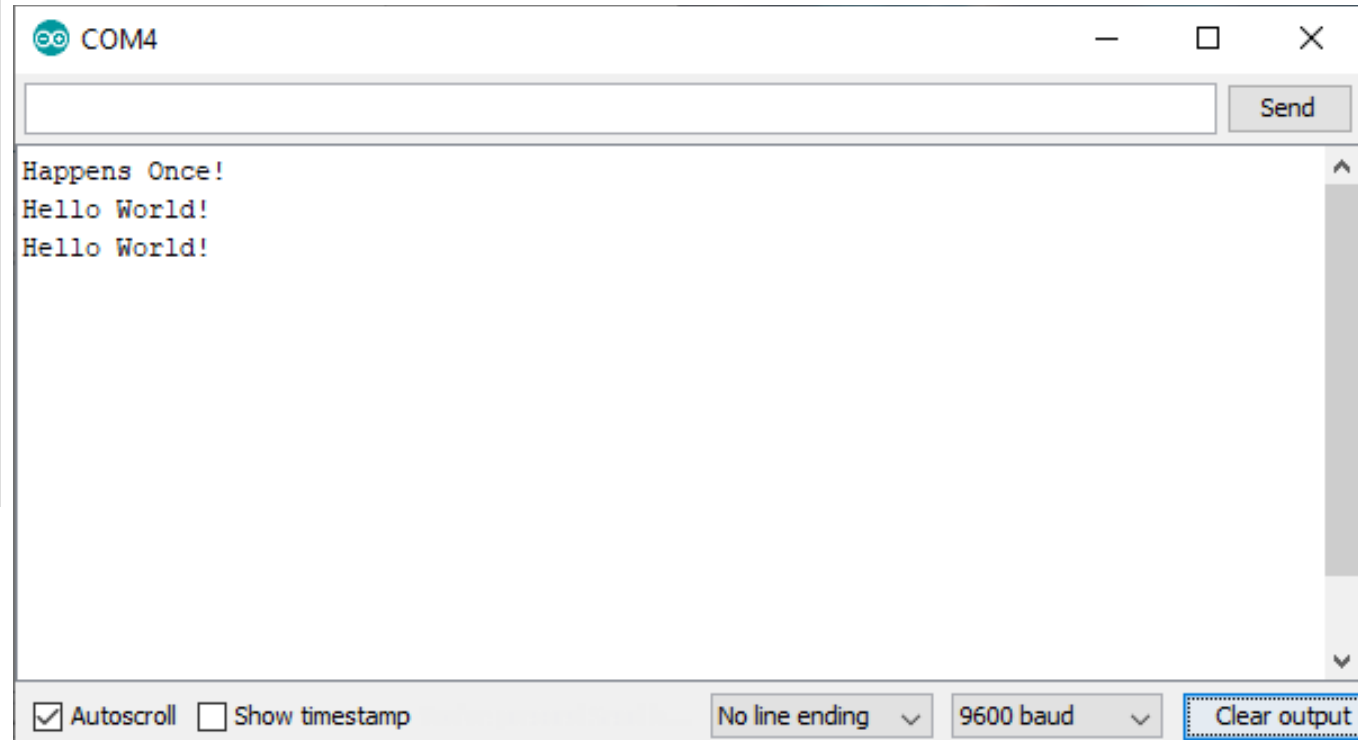
Program structure



Hello world in Arduino IDE

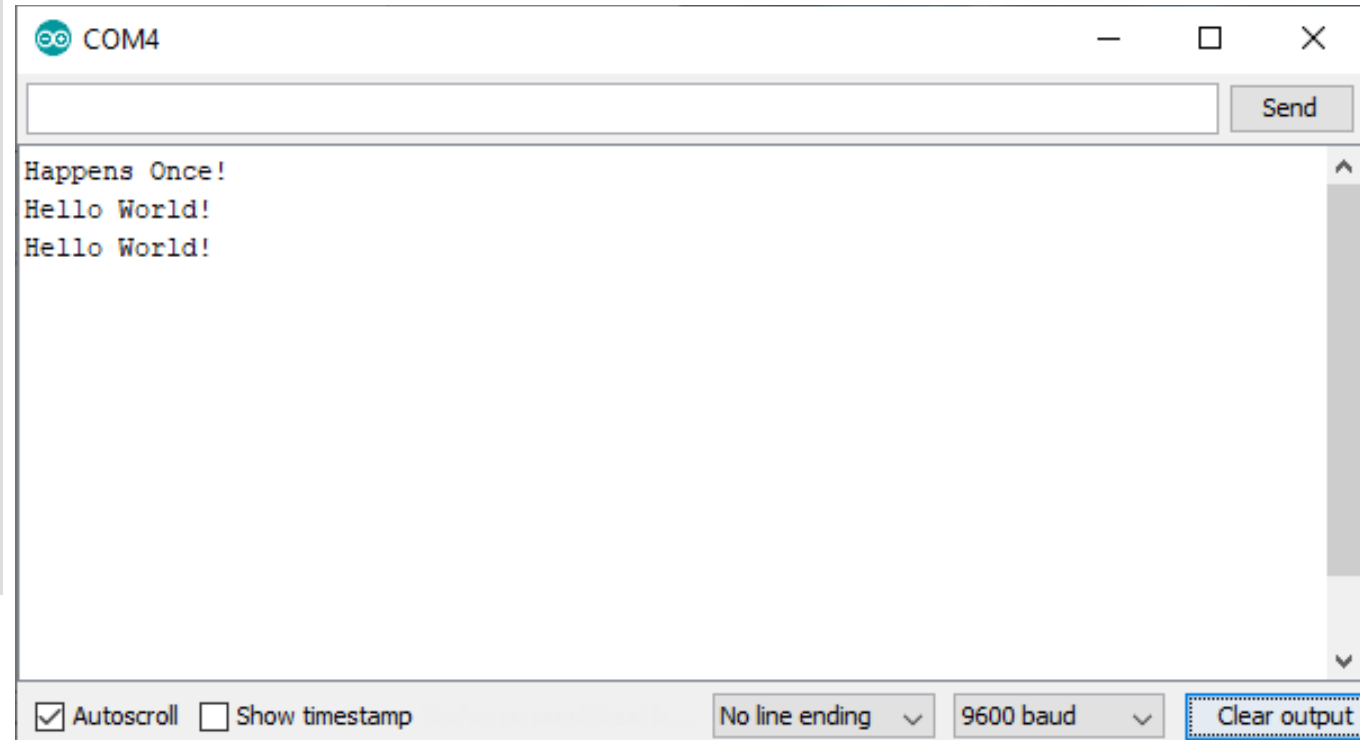
```
void setup() {  
  Serial.begin(9600);  
  Serial.println("Happens Once!");  
  delay(2000);  
}  
void loop() {  
  Serial.println("Hello World!");  
  delay(2000);  
}
```

- **delay(2000)** function causes a waiting period of 2 seconds (2000 milliseconds)



Hello world in Arduino IDE

```
void setup() {  
  Serial.begin(9600);  
  Serial.println("Happens Once!");  
  delay(2000);  
}  
void loop() {  
  Serial.print("Hello ");  
  Serial.println("World!");  
  delay(2000);  
}
```



Writing your own programs

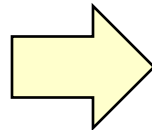
How?

- Use other programs as models, and then modify
 - Very useful strategy
 - Lectures have examples that you can use as models for your assignment programs

A new program

- Change the “Hello world!” program
 - When you upload it to the Arduino, it prints your **name** to the Arduino Serial Monitor once every second.

```
void setup() {  
  Serial.begin(9600);  
  Serial.println("Happens Once!");  
  delay(2000);  
}  
void loop() {  
  Serial.println("Hello World!");  
  delay(2000);  
}
```



```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  
}
```

Writing your own programs

How?

- Use other programs as models, and then modify
 - Very useful strategy

BUT

- It can be hard to work out how to modify

Need to understand the language

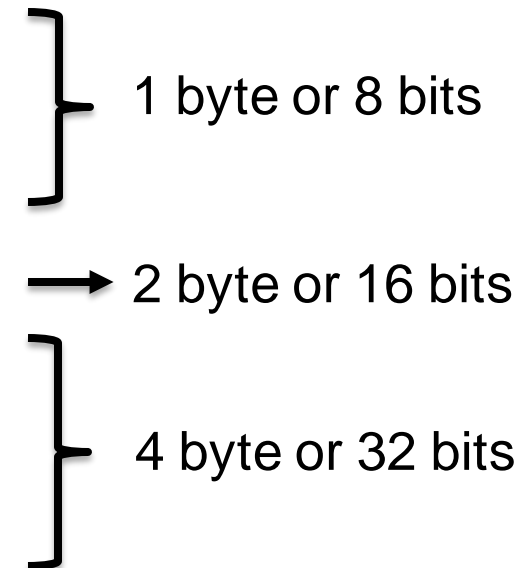
- ⇒ vocabulary
- ⇒ syntax rules
- ⇒ meaning (“semantics”)

Variables and Variable Types

- You can use variables in a similar way as they are used in math or physics
- All variables must be *declared* before they are used, and optionally,
- set an initial value (*initialising the variable*).
- <https://www.arduino.cc/reference/en/#variables>

- **Variable Types:**

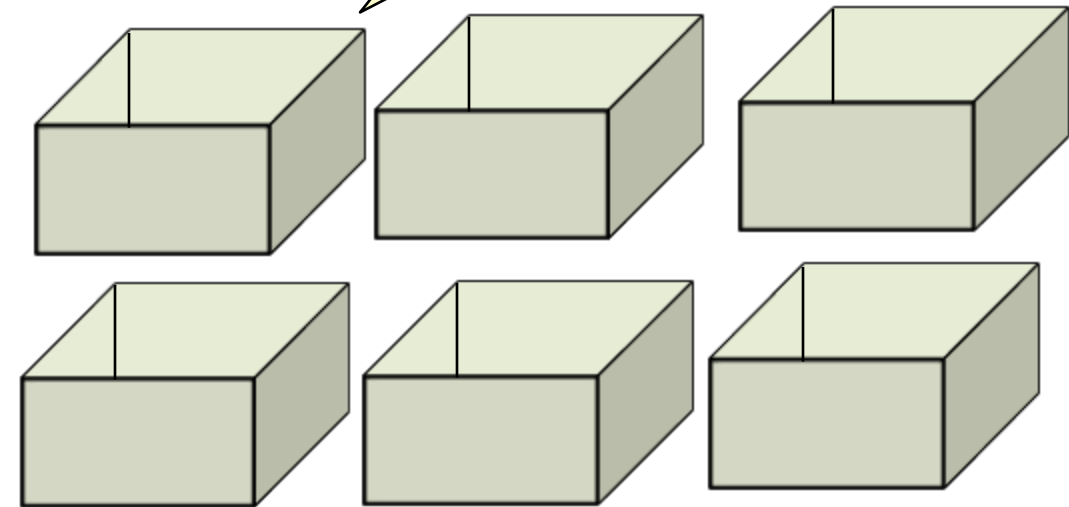
- **bool**, holds one of two values, *true* or *false*
- **byte**, holds a number from 0 to 255
- **char**, holds one character value
- **int**, in Arduino Uno holds a number from -32,768 to 32,767
- **long**, holds a number from -2,147,483,648 to 2,147,483,647
- **float**, for floating-point numbers. $-3.4 \cdot 10^{38}$ to $3.4 \cdot 10^{38}$
- **double**, for floating-point numbers.



Variables

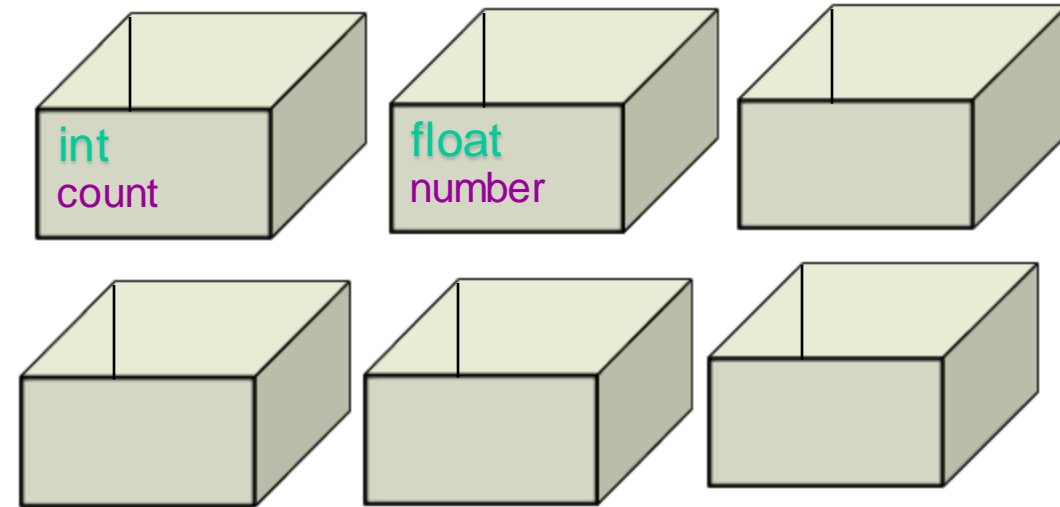
- The power of computer programming
 - Variables
 - Operations with variables
 - Input/output commands
- Variable is not the same as variables in math
- Variable in computer science is like a box that you can put different values into it

A variable is like a box. You can put values into!



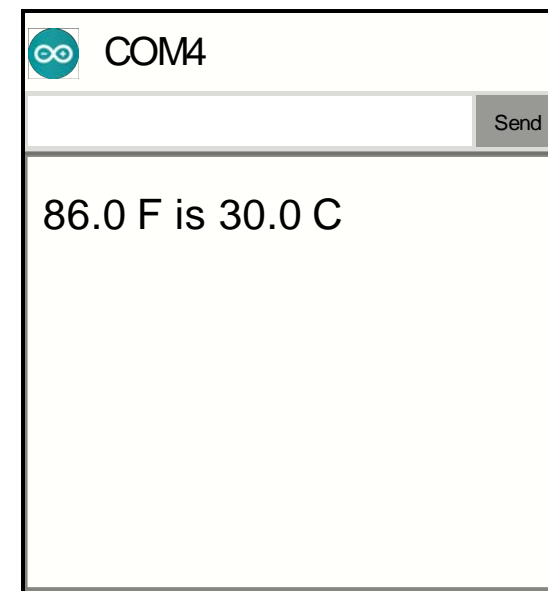
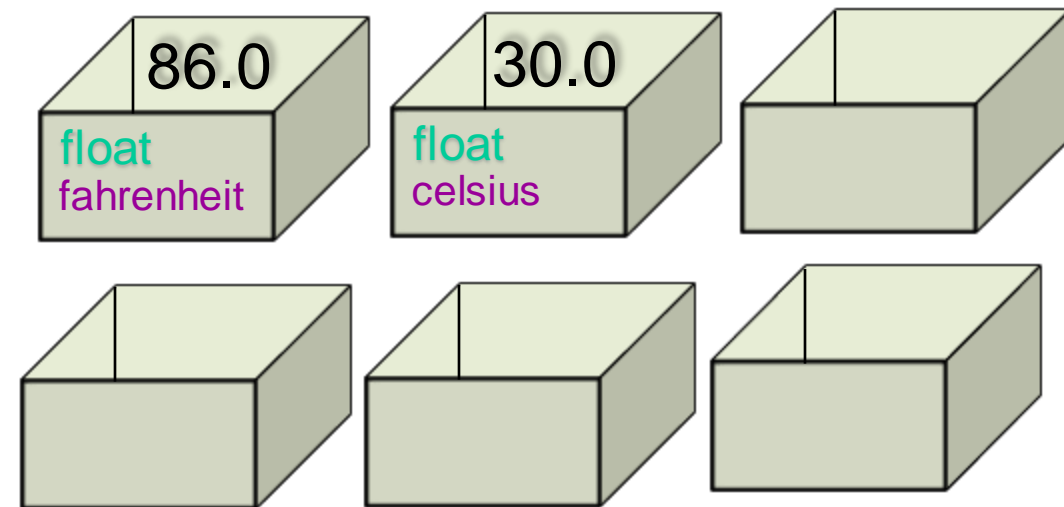
Arduino structure

- 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
 - It holds a piece of information

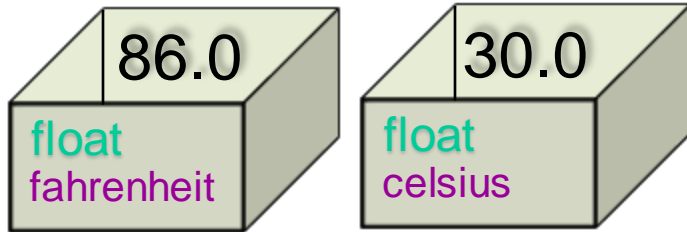


Variables

```
float fahrenheit = 86.0;
float celsius;
void setup() {
  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() {
}
```

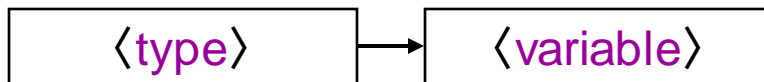


Variables



```
float fahrenheit;
float celsius;
void setup() {
  fahrenheit = 86.0;
  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.

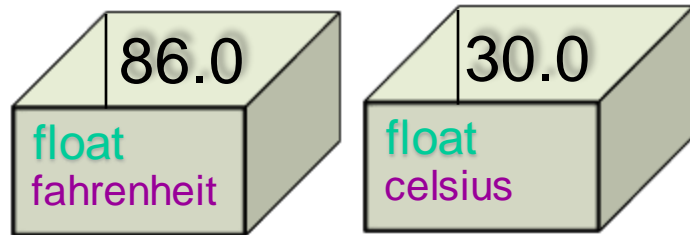


- Must put a value into a variable before you can use it
⇒ “assign” to the variable
- Can *use* the value by specifying the variable’s name
- Can change the value in a variable (unlike mathematical variable)

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

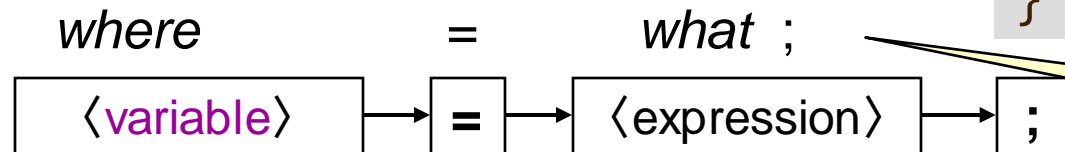
Asking for a place

Assignment Statements



```
float fahrenheit;
float celsius;
void setup() {
  fahrenheit = 86.0;
  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:



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

```
float fahrenheit = 86.0;
float celsius;
void setup() {
  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,
 - ...

Using variables

- To change the pin number from 13 to 14
 - Must find all the places where used pin 13 and changed them to 14

- To change the pin number from 13 to 14 using variable
 - Only need to change the value of the variable

```
void setup() {  
    pinMode(13, OUTPUT);  
}  
void loop() {  
    digitalWrite(13, HIGH);  
    delay(1000);  
    digitalWrite(13, LOW);  
    delay(1000);  
}
```

```
int led = 13;  
void setup() {  
    pinMode(led, OUTPUT);  
}  
void loop() {  
    digitalWrite(led, HIGH);  
    delay(1000);  
    digitalWrite(led, LOW);  
    delay(1000);  
}
```

Values / Data

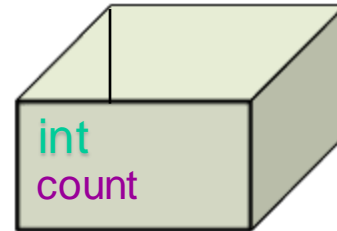
There are lots of different kinds ("Types") of values:

- 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

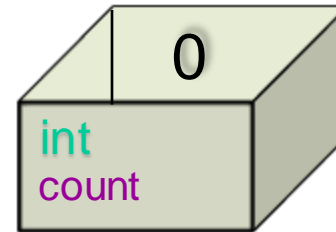
```
int count;
void setup() {
  Serial.begin(9600);

}
void loop() {
}
```



Using a Variable: int

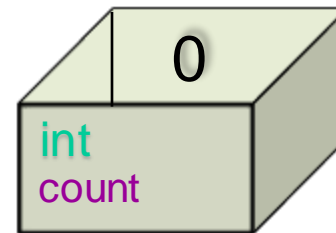
```
int count;
void setup() {
  Serial.begin(9600);
  count = 0;
}
void loop() {
}
```



COM4

Using a Variable: int

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

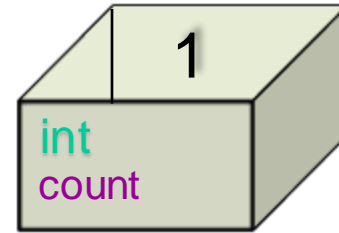


COM4

0

Using a Variable: int

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

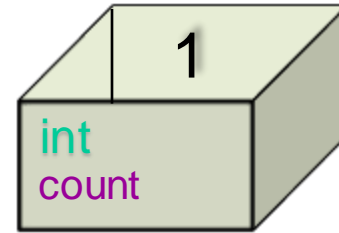


COM4

0

Using a Variable: int

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



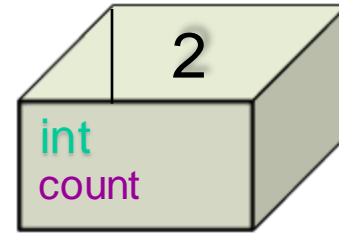
COM4

```
0
1
```

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() {
}
```

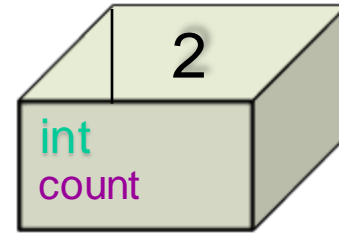


COM4

```
0
1
```

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() {
}
```



COM4

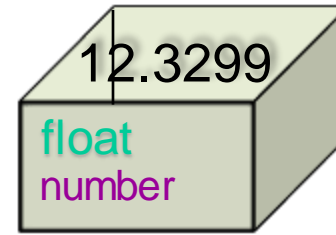
```
0
1
2
```

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() {
}
```



COM4

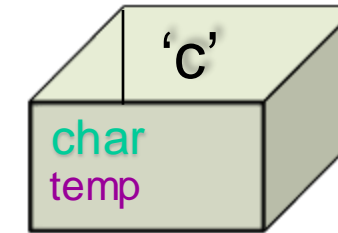
12.33

12.3299

char

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

char temp = 'c';



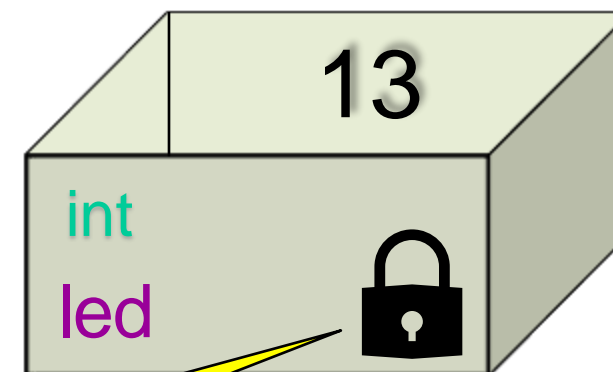
0	NUL	1	SOH	2	STX	3	ETX	4	EOT	5	ENQ	6	ACK	7	BEL
8	BS	9	HT	10	NL	11	VT	12	NP	13	CR	14	SO	15	SI
16	DLE	17	DC1	18	DC2	19	DC3	20	DC4	21	NAK	22	SYN	23	ETB
24	CAN	25	EM	26	SUB	27	ESC	28	FS	29	GS	30	RS	31	US
32	SP	33	!	34	"	35	#	36	\$	37	%	38	&	39	'
40	(41)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	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

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

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



The value can't be changed

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}
```

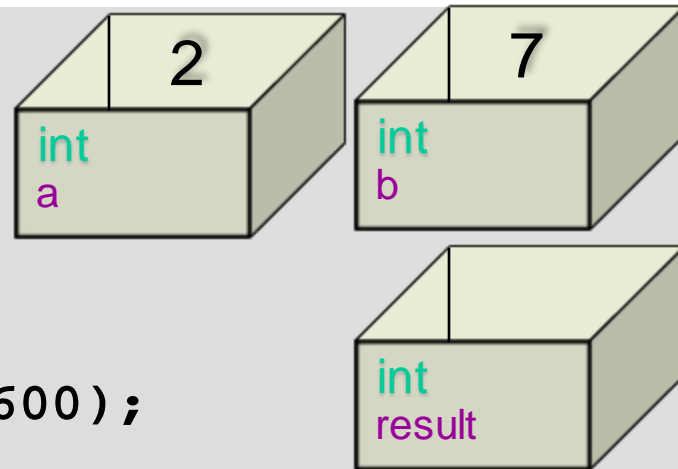
Invalid: uses non-letter and non-digit characters

```
continue
```

Invalid: reserved word

Arithmetic Operators: + - * / %

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



```
    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);
```

```
}
```



COM4

Send

Addition (a + b): 9

Subtraction (b - a): 5

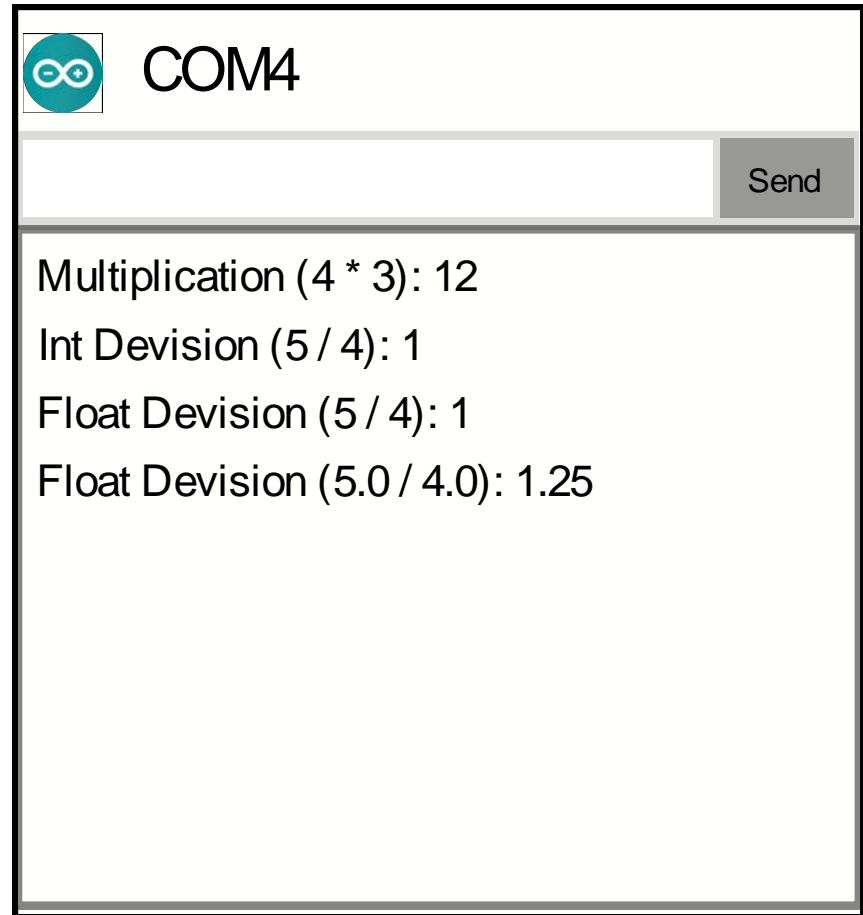
Arithmetic Operators: * /

```
int result;
float result_fl;
void setup() {
  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 / 4): ");
  result_fl = 5 / 4;
  Serial.println(result_fl);

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



Arithmetic Operators: modulo (%)

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



COM4

Send

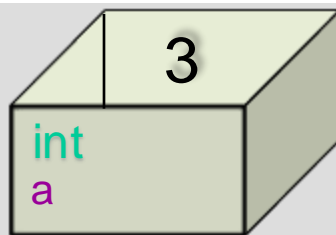
Remainder (11 % 4): 3

Using the if Statement

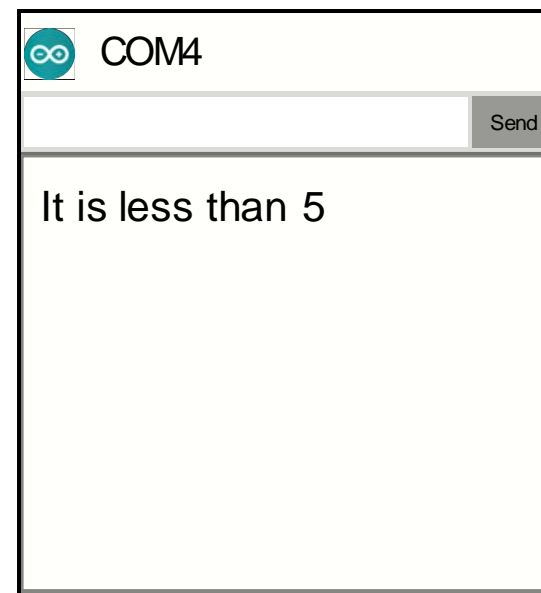
```
if (expression){  
    statement  
}
```

- If expression evaluates to true, statement is executed

```
int a = 3;  
void setup() {  
    Serial.begin(9600);  
    if( a < 5 ){  
        Serial.println("It is less than 5");  
    }  
}  
void loop() {  
}
```

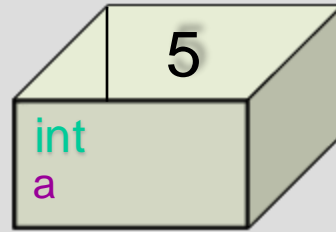


- Can do an action only in some circumstances



Using *if-else*

```
int a = 5;
void setup() {
  Serial.begin(9600);
  if( a < 5 ){
    Serial.println("It is less than 5");
  }
  else{
    Serial.print("It is greater than");
    Serial.println("or equal to 5");
  }
}
void loop() {
```



Can choose between different actions

```
if (expression){
    statement1
}
else{
    statement2
}
```

- The else part is optional
- If expression evaluates to true, statement₁ is executed
- Else, statement₂ is executed if the else part is present