General Syntax

What are general syntax elements in Arduino programming?

The ground rules that govern how the code is written. More similar to grammar in writing.

1.#define

used to assign a variable to a constant that need to be used somewhere in the program.

```
49  #define buttonUp 2
```

2.#include< >

Used to command the program to include libraries.

```
1  #include <LiquidCrystal.h>
```

3. /* ..... */

used to write single or multi line comments on the program. Starts with /* and ends with */. You can write your comment in between them.

```
15  /* This is a
16          multi line
17              comment */
```

4.//

Used to write a single line comment

```
15  // This is a single line comment
```

5.{ }

Used to segment the program. Starts with { and ends with }. In between you can write your programs.

```
73  void setup()
74  {
75    /* Write
76          your
77               code
78                   here*/
79  }
```

6.; (Semicolon)

Used at the end of a statement. Not used with #define and #include<>,

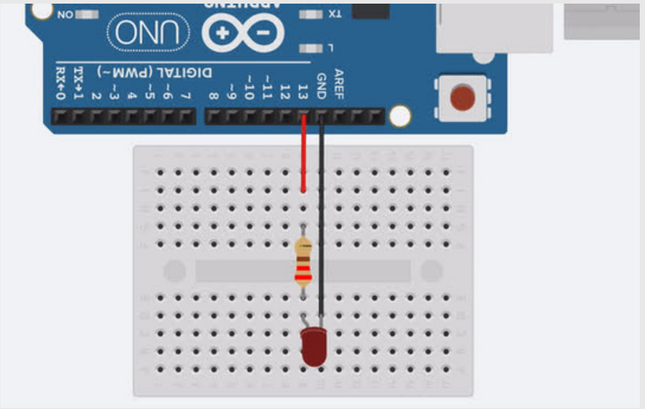
```
354          hours = hours + 1;
```

General Structure

void setup()

- The first function to be run when the sketch starts execution
- Most initialization functions are used inside this function (Eg: Pin declarations, variable initialization, etc.)
- Runs only ontime when the Arduino is powered and only runs again if the reset button is pressed.

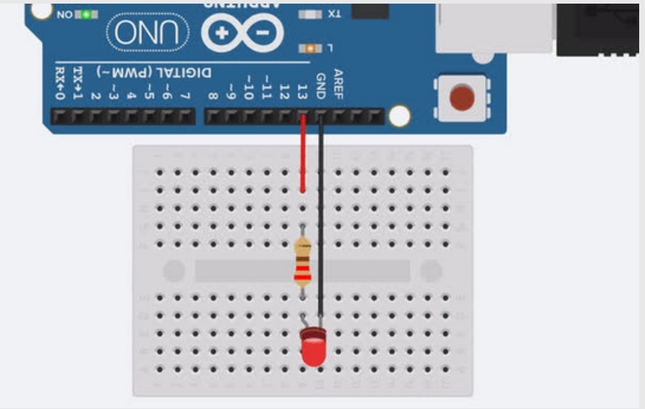
```
1  int LED = 13;
2  int time = 500;
3
4  void setup()
5  {
6    pinMode(LED, OUTPUT); // declare LED as an OUTPUT
7    // turn the LED on (HIGH is the voltage level)
8    digitalWrite(LED, HIGH);
9    delay(time); // Wait for 1000 millisecond(s)
10   // turn the LED off by making the voltage LOW
11   digitalWrite(LED, LOW);
12   delay(time); // Wait for 1000 millisecond(s)
13 }
```



void loop()

- Executes after void setup().
- Function that keep running until the Arduino board is powered off.

```
15 void loop()
16 {
17   // turn the LED on (HIGH is the voltage level)
18   digitalWrite(LED, HIGH);
19   delay(time); // Wait for 1000 millisecond(s)
20   // turn the LED off by making the voltage LOW
21   digitalWrite(LED, LOW);
22   delay(time); // Wait for 1000 millisecond(s)
23 }
```



CHEAT SHEET 5 ARDUINO

PinMode

What are PinModes?

Configures the specified pin to behave either as an input or an output. Commonly used Pin Modes are INPUT and OUTPUT . These functions are generally used in void setup().

1. INPUT

Used to get data from sensors to Arduino board for processing.

2. OUTPUT

Used to perform functions on specified pins.

Examples

```
79      // Pin Declarations
80      pinMode(buttonUp, INPUT);
81      pinMode(buttonDown, INPUT);
82      pinMode(buttonOk, INPUT);
83      pinMode(buttonCancel, INPUT);
84
85      pinMode(ledPin, OUTPUT);
86      pinMode(buzzerPin, OUTPUT);
```

CHEAT SHEET 3 ARDUINO

Variables

What are different types of variables in an Arduino program?

In Arduino, there are different data types exist to store different kind of data.

Data Type	Size	Description
int	16 bits	Data type used to store numbers. int has a range of -32,768 to 32,767.
char	8 bits	Used to store charactors.
float	32 bits	Used to store decimal numbers. float has a range of -3.4028235E+38 to 3.4028235E+38.
bool	8 bits	Used to store 2 values which are true and false.
long	32 bits	Extended version of int. Has a range of -2,147,483,648 to 2,147,483,647
double	32 bits	Extended version of float. But for Uno, this is same as float

- Arrays are used to store multiple variables of the same type as shown below.
- The string is a data type that stores texts. The string is an array of characters or a set of characters that consists of numbers, spaces, and special characters from the ASCII table.

Examples

```
1  // variables
2  int num1 = 35;
3  char c1 = 'a';
4  float num2 = 35.4;
5  bool isValid = false;
6  long num3 = 120000;
7  double num4 = 569030.5;
8  String s1 = "name";
9
10 // arrays
11 char c2[] = "age";
12 int a1[] = {1,2,3}
```

ALU Operations

What are Arithmetic and Logic operations?

- Arithmetic operations
  -Basic mathematical operations

+	(Addition)	-	Usedto add two numbers
-	(Substraction)	-	Used to substract two numbers
*	(Multiplication)	-	Used to multiply two numbers
/	(Division)	-	Used to divide two numbers
%	(Remainder)	-	Used to get the remainder after the division of two numbers. Only works with int datatype.

- Logic operations
  -Basic comparison operations

==	(equal to) -checks whether the two numbers on the left and right sides are the same or not
!=	(not equal to) -checks whether the two numbers on the left and right sides are not the same or not
<	(less than) -checks whether the number on left is less than the number on right
>	(greater than) -checks whether the number on left is greater than the number on right
<=	(less than or equal) -checks whether the number on left is greater than tor equal to the number on right
>=	(greater than or equal) -checks whether the number on left is greater than or equal to the number on right

Examples

```
1  int a1=5;
2  int a2=2;
3  float a3=3.2;
4
5  // arithmetic
6  print(a1+a2); //output - 7
7  print(a1-a2); //output - 3
8  print(a1*a2); //output - 10
9  print(a1/a2); //output - 2
10 print(a3/a2); //output - 1.6
11 print(a1%a2); //output - 1
12
13 // logical
14 print(a1==a2); //output - false
15 print(a1>a2); //output - true
16 print(a1!=a2); //output - true
```

Control & Conditional Statements

Conditional Statements

Helps you to make a decision based on certain conditions

- If statement

if statements are used when we need to run different things based on one condition or several conditions.

```
if ( first condition )
{
    //come here if the condition is true
}
else if (second condition)
{
    // come here only if the first condition
    //is false and the second condition is true
}
else
{
    //come here only if both the above
    //conditions are false
}
```

Control Statements

Helps you to control the flow of execution in a program

- for statement

for statements are used to repeat a block of statements if we know the exact number of repetitions it should run. Further, we can give the step size as well.

```
for (initialization; condition; stepsize)
{
  // block of statement
}
```

- while statement

while statements are used to repeat a block of statements if we don't know the exact number of iterations and rather want to adjust it using conditions inside the loop.

```
while(true){
  /*
      Add code to be repeated
  */
  if ({break condition})break;
}
```

- Branch and redirect

- break - quit from loop
- continue - move directly to the next iteration

CHEAT SHEET 7 ARDUINO

Serial Communication

What is Serial?

Serial communication allows micro controller to communicate with other devices such as Bluetooth, USB or with another micro controller.

Baud rate?

Baud rate is the speed of transmission over serial. There are several baud rates available such as 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200 bps.

Examples

```
1  void setup()
2  {
3      Serial.begin(9600); // baud rate
4      Serial.println("Welcome to Medi Box!"); // adds new line at the end
5      Serial.print("same ");
6      Serial.print("line");
7      Serial.print("\n"); // go to new line
8      Serial.print("new line");
9  }
```

Serial Monitor

```
Welcome to Medi Box!
same line
new line
```

- Serial communication is a good wired communication method for debugging purposes.

Functions

What are functions?

A reusable block of code that fulfill a specific task.

Why do we need functions?

To organize a complex program into smaller and manageable segments, while giving the ability to reuse wherever necessary.

How functions operate?

May or may not take something as an input, process/modify something and may or may not return something

What are local and global variables?

Variables that are defined inside functions are local variables and can only be accessed within the function. Global variables are outside the function and can be accessed within a function without redefining.

A function that take inputs and returns something

```
25 int local_addition(int a, int b)
26 {
27     int sum = a + b;
28
29     return sum;
30 }
```

A function that take inputs and not returns something

```
32 int summation = 0;
33
34 void global_addition(int a, int b)
35 {
36     summation = a + b;
37 }
```

A function that not take inputs and not returns something

```
4  void setup()
5  {
6    Serial.begin(9600);
7  }
```

Global variables ,local variables and usage of functions

```
1  //Creating a global variable
2  int summation = 0;
3
4  void setup()
5  {
6    Serial.begin(9600);
7  }
8
9  void loop()
10 {
11   summation = local_addition(9, 4);
12   Serial.print("Local summation : ");
13   Serial.println(summation);
14   delay(1000);
15
16   global_addition(8, 2);
17   Serial.print("Global summation : ");
18   Serial.println(summation);
19
20   //sum = 24;
21   //The above statement will rise an error
22   //Since the sum variable is declared locally
23   //inside a function. So we cannot access it
24 }
25
26 int local_addition(int a, int b)
27 {
28     //Creating a local variable
29     int sum = a + b;
30     return sum;
31 }
32
33 void global_addition(int a, int b)
34 {
35     //Accessing the global variable
36     summation = a + b;
37 }
```