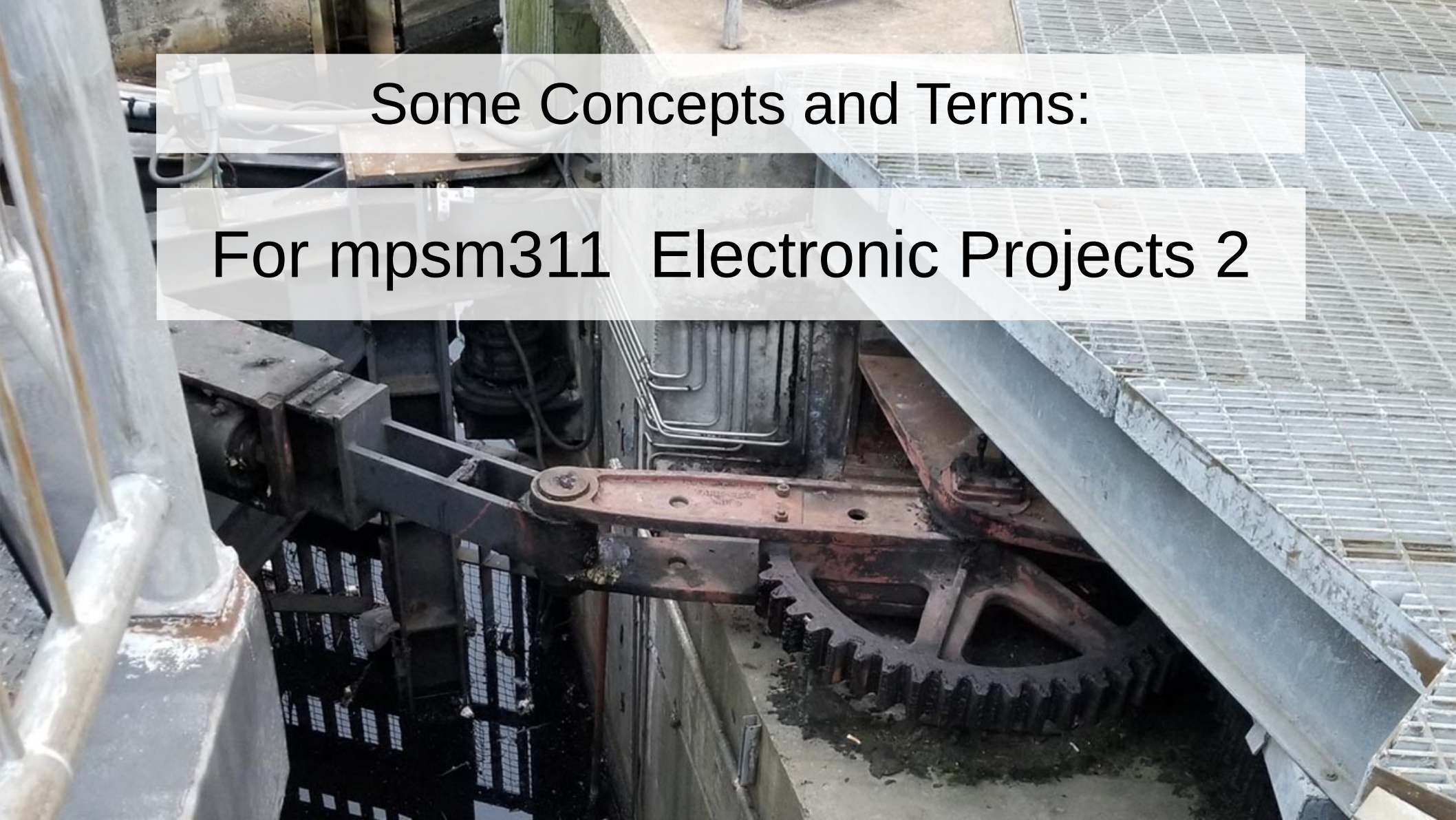
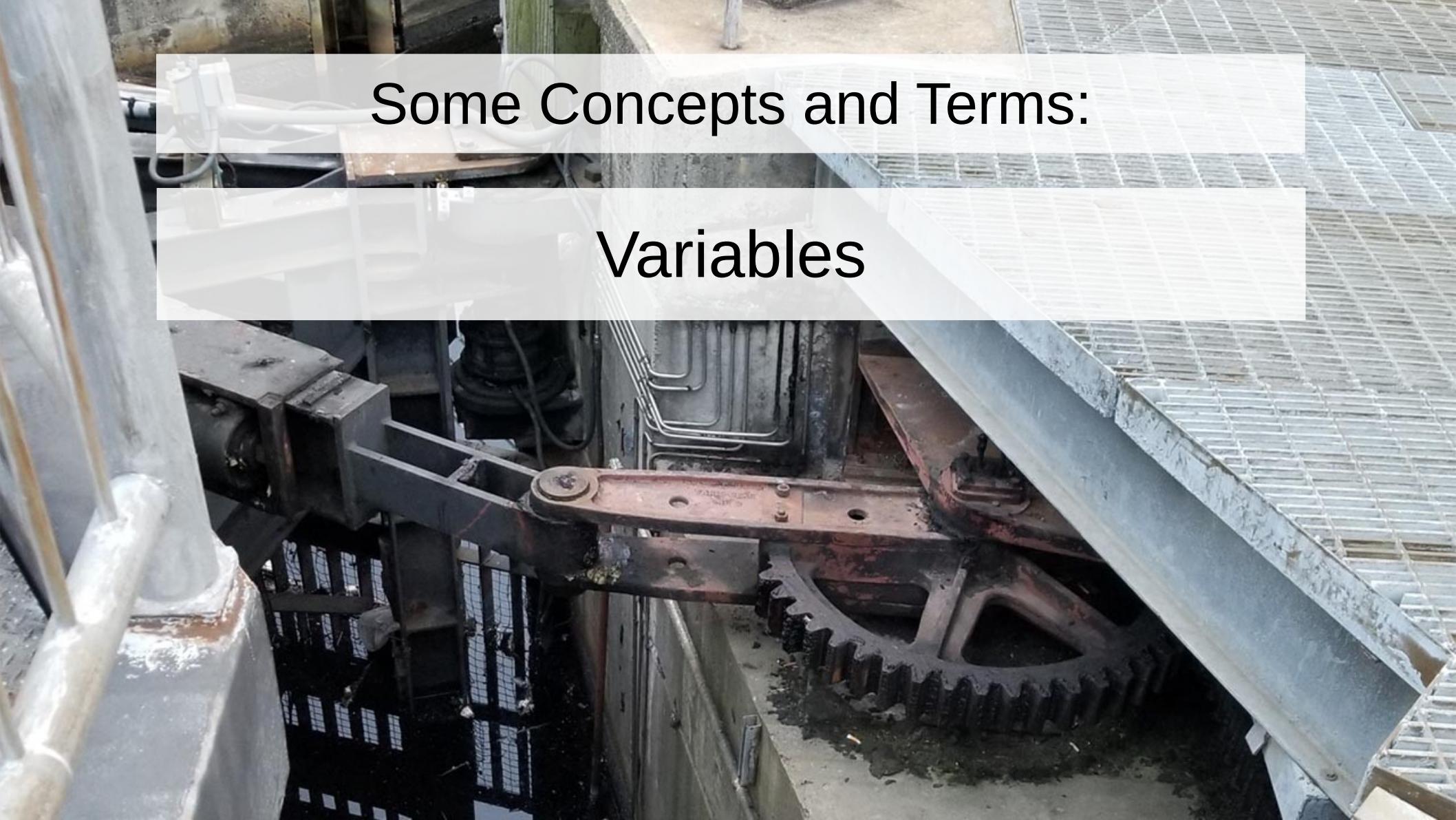


Some Concepts and Terms:

For mpsm311 Electronic Projects 2



Some Concepts and Terms:

Variables

Variables:

C and Java are what's called a "strongly typed" languages.

Meaning, *you can't do this:*

```
x = 7;
```

if you want to declare a variable `x` and give it a value of 7.

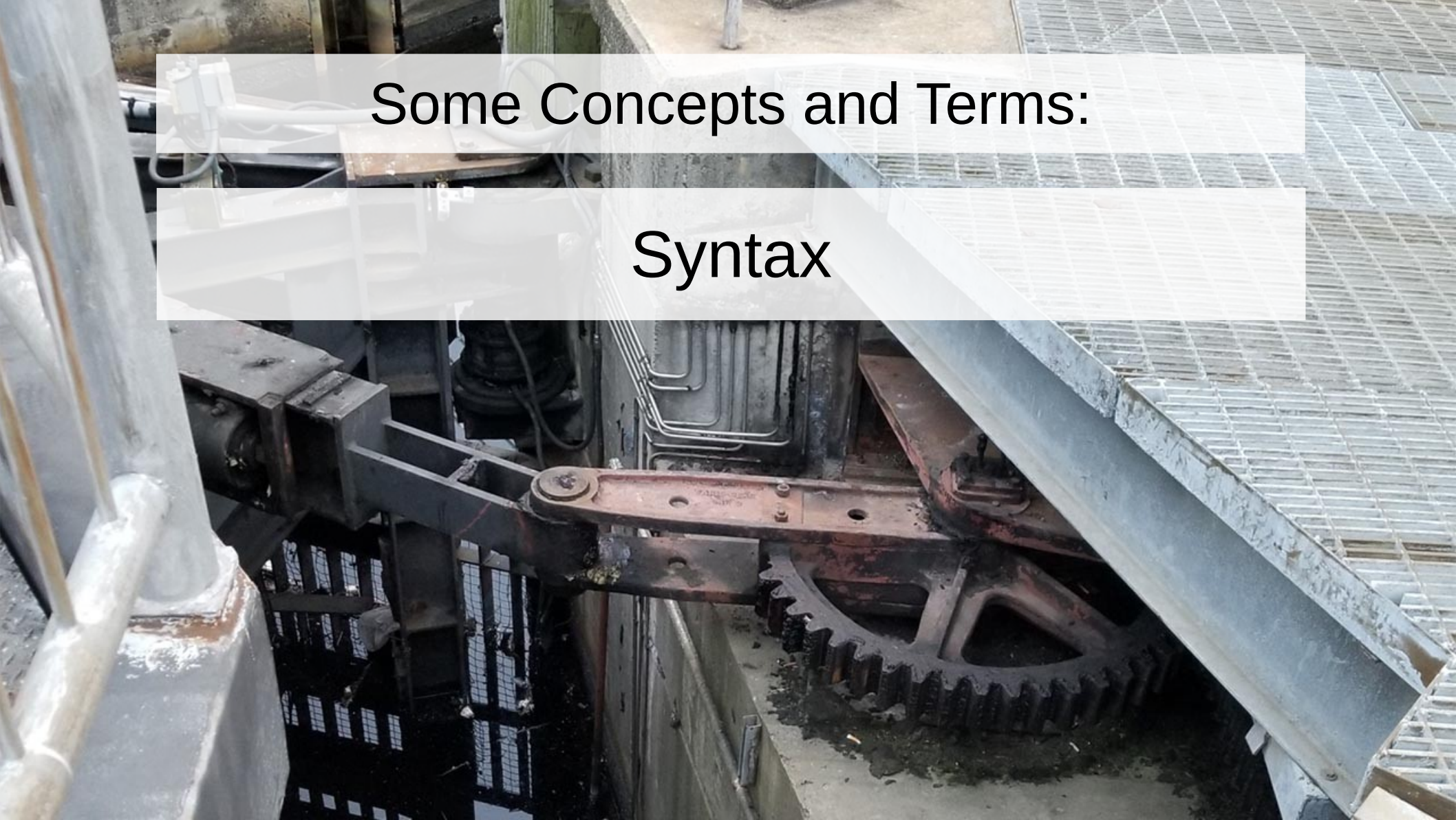
To make a variable, you first have to say *what type of variable it is*, (in this case, an *integer*.)

like this: `int x = 7;`

Variables:

Types of variables:

- - `void` : represents the absence of a variable.
- - `int` : integer (whole numbers)
- - `float` : floating point numbers (e.g. “2.5”, “1.618”, etc.)
- - `boolean` : a 1 bit number (either a “0” or a “1”)
- - `char` : a single character/letter
- - `String` : a sequence of chars (e.g. “Hello”)



Some Concepts and Terms:

Syntax

Syntax - statements:

A *declaration or statement* always ends with a semicolon (“;”) - like a period at the end of a sentence.

Examples:

(declaring a variable)

```
int x = 7;
```

(statement that calls a function)

```
Serial.print(“Hello world!”);
```

Syntax - expressions:

An *expression* in Java or C involves operators like addition (+), assignment (=) or function calls.

Examples:

```
n = n + 2;
Serial.print(“Hello world!”);
```

Syntax - logical expressions:

There is a special type of expression that uses operators called relational operators that describe a relationship that is either “true or “false” (or mathematically can only be a “1” or a “0”.)

These are called *logical expressions* or *“boolean expressions”* -or sometimes *“boolean tests”*.

Example:

```
a > b
```

Syntax - boolean operators:

Types of boolean tests:

operator:	**meaning:**
`a > b`	True (1) if a is greater than b, false (0) otherwise
`a < b`	True (1) if a is less than b, false (0) otherwise
`a >= b`	True (1) if a is greater than or equal to b, false (0) otherwise
`a <= b`	True (1) if a is less than or equal to b, false (0) otherwise
`a == b`	True (1) if a is exactly equal to b, false (0) otherwise
`a != b`	True (1) if a is not equal to b, false (0) otherwise

Syntax - loops:

A “*while loop*” repeatedly executes the statements within the curly braces as long as the statement in parentheses is true:

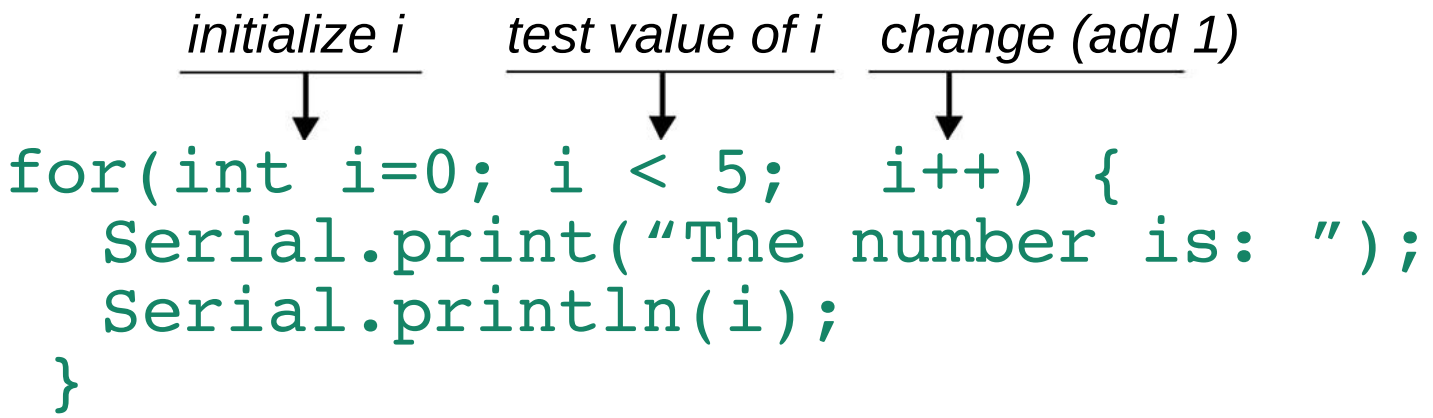
```
int i = 1;
while( i < 5 ) {
   Serial.print(“The number is: ”);
   Serial.println(i);
   i = i + 1;
}
```

Syntax - loops:

A “*for loop*” is used for counting. It repeatedly executes the statements within the curly braces as long as the statement in the test is true:

example:

”loop for as long as i is less than 5.”



Syntax - conditional statements:

A "*conditional statement*" *changes the flow of the execution of the program based on the result of a boolean condition.*

example:

```
n = n + 1;
if(n > 100){
   message = "it's too big.";
}
```

Syntax - conditional statements:

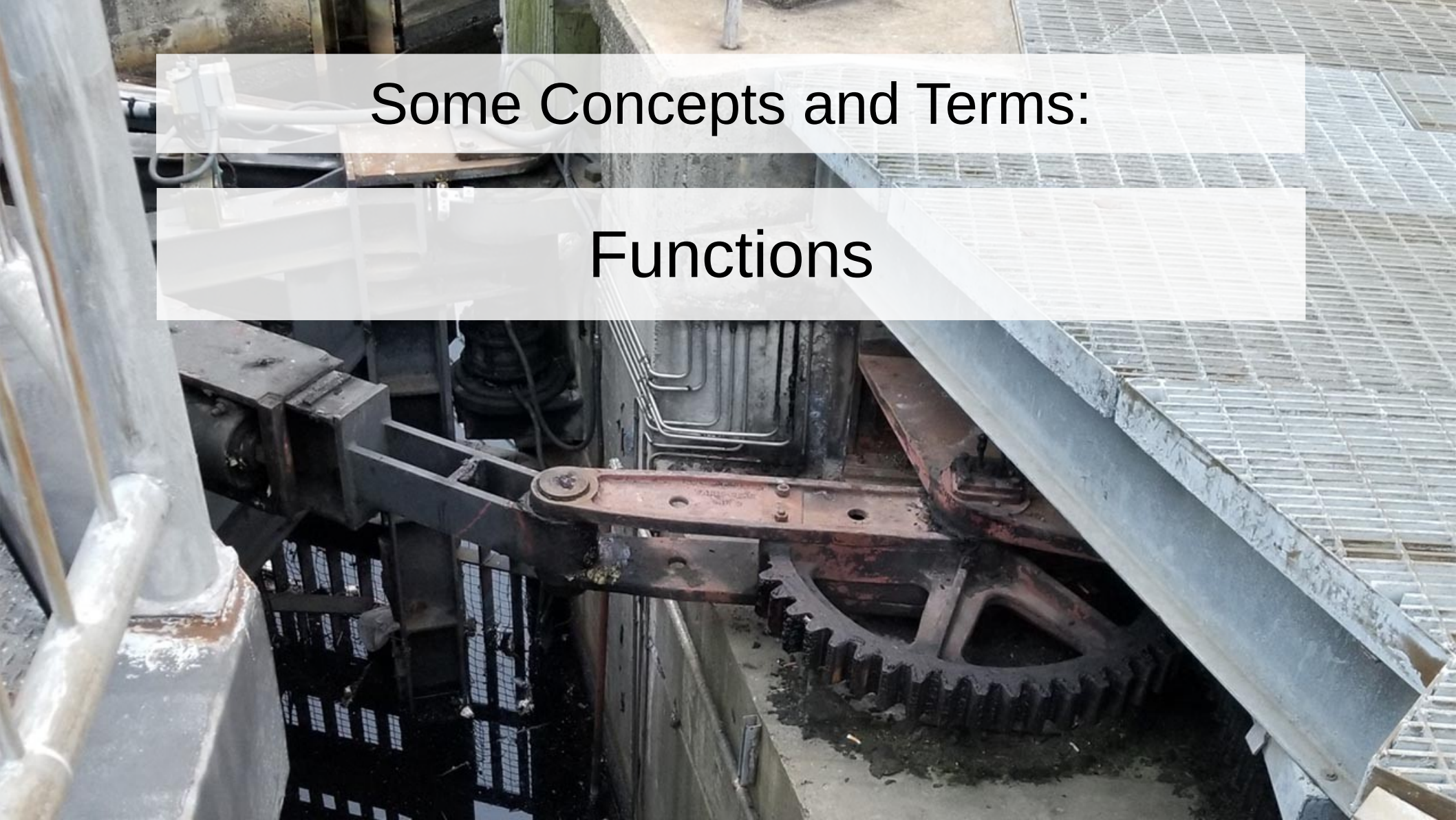
more examples:

simple else-if :

```
n = n + 1;
if(n > 100){
   message = “It’s too big.”;
} else {
  message = “It’s ok.”;
 }
```

else.. if.. else :

```
n = n + 1;
if(n > 100){
   message = “It’s too big.”;
} else if(n < 0) {
   message = “It’s too small.”;
 }
```



Some Concepts and Terms:

Functions

Syntax - functions:

Functions are the building blocks of your programs.


```
int blinkAdder(int num){
      num = num + 1;
      return num;
}
```

How it would be used: (We say, “calling a function”.)

```
int newNumber = blinkAdder(2);
```

Syntax - functions:

Another example:

```
void  printButtonState() {
   buttonState = digitalRead(buttonPin);
   if(buttonState == LOW) {
     Serial.print("The button is closed.");
     } else {
     Serial.print("The switch is open.");
     }
   delay(1000);
 }
```