E11 - Autonomous Vehicles
Arduino Programming II
Create a program for the Mudduino that:

1. Prints to the serial monitor “Enter a Number”
2. Receives a number X from the user via serial monitor
3. Compares the number with a preset list of 5 integers
   \[ 3 \ 8 \ 4 \ 9 \ 1 \]
4. Prints to the serial monitor “Match with X” if the number is found in the list.
5. Else it prints “No Match with X”

Repeat this …
Outline

- Goal
- Review
- Variables & Data Types
- Inputs & Outputs
- Functions
- Conditionals
- Loops
Quick Review

- In our case...

- **Source**
  - Our desktop PC
  - We will “verify” the computer program

- **Target**
  - The Arduino Board
  - We will “run” the computer program
In our Arduino, there are two main “functions”

```c
void setup()
{
}
```

FIRST, this function runs ONCE.

```c
void loop()
{
}
```

THEN, this function runs REPEATEDLY.
Quick Review

- Lets kick start our solution to today’s problem

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

void loop()
{
  Serial.println("Enter a Number");
}
```
Outline

- Goal
- Review
- Variables & Data Types
- Inputs & Outputs
- Functions
- Conditionals
- Loops
Declaring and Using Variables

Often, we like to store and manipulate variables, e.g.

```plaintext
void setup()
{
    int x;
    int y;
    x = 4;
    y = x + 10;
}
```

This is a variable declaration of type `int`
Variables & Data Types

- **Variable Types**
  - Variables are first declared to be of some type.

**Global** variables are declared outside of all functions, and are accessible by all functions

**Local** variables are declared inside of a function, and is only accessible within that function
### Variables & Data Types

#### Type Example

```cpp
int y;
void setup()
{
    int x;
    x = 4;
}
```

- A Global variable being declared
- A Local variable being declared
Variable Initialization

Before they can be used, all variables must be “initialized”, i.e. set to an initial value.

```cpp
int y;
void setup()
{
    int x;
    y = x;
}
```
Variables & Data Types

- Variable Initialization
  - Before they can be used, all variables must be “initialized”, i.e. set to an initial value.

```cpp
int y;
void setup()
{
  int x;
  x = 1;
  y = x;
}
Yes!
```
Declaring Data Types

- A programmer can declare variables to be one of many data types (not just int’s).

- The data type determines
  - The type of data represented
  - The range of data
## Variables & Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Size (bits)</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>8</td>
<td>$-2^7 (-128)$</td>
<td>$2^7 - 1 (127)$</td>
</tr>
<tr>
<td>unsigned char</td>
<td>8</td>
<td>0</td>
<td>$2^8 - 1 (255)$</td>
</tr>
<tr>
<td>int</td>
<td>16</td>
<td>$-2^{15} (-32,767)$</td>
<td>$2^{15} - 1 (32,768)$</td>
</tr>
<tr>
<td>unsigned int</td>
<td>16</td>
<td>0</td>
<td>$2^{16} - 1 (65,535)$</td>
</tr>
<tr>
<td>long</td>
<td>32</td>
<td>$-2^{31} (-32,768)$</td>
<td>$2^{31} - 1 (32,767)$</td>
</tr>
<tr>
<td>unsigned long</td>
<td>32</td>
<td>0</td>
<td>$2^{32} - 1 (4,294,967,295)$</td>
</tr>
<tr>
<td>float</td>
<td>32</td>
<td>$\pm 2^{-126}$</td>
<td>$\pm 2^{128} * (2 - 2^{-15})$</td>
</tr>
<tr>
<td>boolean</td>
<td>8</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
Arrays
- A data type used to store lists of numbers.

We often have lists
- Phone numbers
- Student ID numbers
- Range measurements from multiple sensors
- ...

- Arrays are very powerful when combined with looping functions (more on this later)
Example of a array declaration which creates space for a list of 10 integers

```c
int listOfInts[10];
```

Example of retrieving the value of the 4th list member

```c
anotherInt = listOfInts[3];
```
Example of setting the values for the 3rd list member

```java
listOfInts[2] = 201;
```

Example of setting the values for the entire list

```java
int listOfInts[5] = {1, 5, 3, 2, 2};
```
Variables & Data Types

- Let's create a global variable integer list to help solve our lecture problem

```cpp
int listOfInts[5] = {3, 8, 4, 9, 1};
int response;

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

void loop()
{
    Serial.println(“Enter a Number”);
}
```
Outline

- Goal
- Review
- Variables & Data Types
- Inputs & Outputs
- Functions
- Conditionals
- Loops
Inputs & Outputs

- Console inputs/outputs
  - Output commands:
    
    ```
    Serial.print("Hello Me");
    Serial.println("Hello You");
    ```
  - Input command:
    
    ```
    Serial.read();
    ```
Example

```c
int incomingByte = 0;
void setup()
{
  Serial.begin(9600);
  Serial.println("Enter a value: ");
}
void loop()
{
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    Serial.print("I received: ");
    Serial.println(incomingByte, DEC);
  }
}
```
### Inputs & Outputs

#### ASCII

<table>
<thead>
<tr>
<th>Binary</th>
<th>Octal</th>
<th>Decimal</th>
<th>Hexadecimal</th>
<th>Glyph</th>
</tr>
</thead>
<tbody>
<tr>
<td>010 0000</td>
<td>040</td>
<td>32</td>
<td>20</td>
<td>space</td>
</tr>
<tr>
<td>010 0001</td>
<td>041</td>
<td>33</td>
<td>21</td>
<td>!</td>
</tr>
<tr>
<td>010 0010</td>
<td>042</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
</tr>
<tr>
<td>010 0011</td>
<td>043</td>
<td>35</td>
<td>23</td>
<td>#</td>
</tr>
<tr>
<td>010 0010</td>
<td>044</td>
<td>36</td>
<td>24</td>
<td>S</td>
</tr>
<tr>
<td>010 0100</td>
<td>045</td>
<td>37</td>
<td>25</td>
<td>%</td>
</tr>
<tr>
<td>010 0101</td>
<td>046</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
</tr>
<tr>
<td>010 0110</td>
<td>047</td>
<td>39</td>
<td>27</td>
<td>'</td>
</tr>
<tr>
<td>010 1000</td>
<td>050</td>
<td>40</td>
<td>28</td>
<td>(</td>
</tr>
<tr>
<td>010 1001</td>
<td>051</td>
<td>41</td>
<td>29</td>
<td>)</td>
</tr>
<tr>
<td>010 1010</td>
<td>052</td>
<td>42</td>
<td>2A</td>
<td>*</td>
</tr>
<tr>
<td>010 1011</td>
<td>053</td>
<td>43</td>
<td>2B</td>
<td>+</td>
</tr>
<tr>
<td>010 1100</td>
<td>054</td>
<td>44</td>
<td>2C</td>
<td>,</td>
</tr>
<tr>
<td>010 1101</td>
<td>055</td>
<td>45</td>
<td>2D</td>
<td>-</td>
</tr>
<tr>
<td>010 1110</td>
<td>056</td>
<td>46</td>
<td>2E</td>
<td>.</td>
</tr>
<tr>
<td>011 0000</td>
<td>057</td>
<td>47</td>
<td>2F</td>
<td>/</td>
</tr>
<tr>
<td>011 0001</td>
<td>058</td>
<td>48</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>011 0010</td>
<td>059</td>
<td>49</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>011 0011</td>
<td>060</td>
<td>50</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>011 0100</td>
<td>061</td>
<td>51</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>011 0101</td>
<td>062</td>
<td>52</td>
<td>34</td>
<td>4</td>
</tr>
<tr>
<td>011 0110</td>
<td>063</td>
<td>53</td>
<td>35</td>
<td>5</td>
</tr>
<tr>
<td>011 1000</td>
<td>064</td>
<td>54</td>
<td>36</td>
<td>6</td>
</tr>
<tr>
<td>011 1001</td>
<td>065</td>
<td>55</td>
<td>37</td>
<td>?</td>
</tr>
<tr>
<td>011 1010</td>
<td>066</td>
<td>56</td>
<td>38</td>
<td>$</td>
</tr>
<tr>
<td>011 1011</td>
<td>067</td>
<td>57</td>
<td>39</td>
<td>9</td>
</tr>
<tr>
<td>011 1100</td>
<td>070</td>
<td>58</td>
<td>3A</td>
<td>:</td>
</tr>
<tr>
<td>011 1101</td>
<td>071</td>
<td>59</td>
<td>3B</td>
<td>;</td>
</tr>
<tr>
<td>011 1110</td>
<td>072</td>
<td>60</td>
<td>3C</td>
<td>&lt;</td>
</tr>
<tr>
<td>011 1111</td>
<td>073</td>
<td>61</td>
<td>3D</td>
<td>=</td>
</tr>
<tr>
<td>100 0000</td>
<td>100</td>
<td>64</td>
<td>40</td>
<td>@</td>
</tr>
<tr>
<td>100 0001</td>
<td>101</td>
<td>65</td>
<td>41</td>
<td>A</td>
</tr>
<tr>
<td>100 0010</td>
<td>102</td>
<td>66</td>
<td>42</td>
<td>B</td>
</tr>
<tr>
<td>100 0011</td>
<td>103</td>
<td>67</td>
<td>43</td>
<td>C</td>
</tr>
<tr>
<td>100 0010</td>
<td>104</td>
<td>68</td>
<td>44</td>
<td>D</td>
</tr>
<tr>
<td>100 0100</td>
<td>105</td>
<td>69</td>
<td>45</td>
<td>E</td>
</tr>
<tr>
<td>100 0101</td>
<td>106</td>
<td>70</td>
<td>46</td>
<td>F</td>
</tr>
<tr>
<td>100 0110</td>
<td>107</td>
<td>71</td>
<td>47</td>
<td>G</td>
</tr>
<tr>
<td>100 1000</td>
<td>110</td>
<td>72</td>
<td>48</td>
<td>H</td>
</tr>
<tr>
<td>100 1001</td>
<td>111</td>
<td>73</td>
<td>49</td>
<td>I</td>
</tr>
<tr>
<td>100 1010</td>
<td>112</td>
<td>74</td>
<td>4A</td>
<td>J</td>
</tr>
<tr>
<td>100 1011</td>
<td>113</td>
<td>75</td>
<td>4B</td>
<td>K</td>
</tr>
<tr>
<td>100 1100</td>
<td>114</td>
<td>76</td>
<td>4C</td>
<td>L</td>
</tr>
<tr>
<td>100 1101</td>
<td>115</td>
<td>77</td>
<td>4D</td>
<td>M</td>
</tr>
<tr>
<td>100 1110</td>
<td>116</td>
<td>78</td>
<td>4E</td>
<td>N</td>
</tr>
<tr>
<td>100 1111</td>
<td>117</td>
<td>79</td>
<td>4F</td>
<td>O</td>
</tr>
<tr>
<td>101 0000</td>
<td>120</td>
<td>80</td>
<td>50</td>
<td>P</td>
</tr>
<tr>
<td>101 0001</td>
<td>121</td>
<td>81</td>
<td>51</td>
<td>Q</td>
</tr>
<tr>
<td>101 0010</td>
<td>122</td>
<td>82</td>
<td>52</td>
<td>R</td>
</tr>
<tr>
<td>101 0011</td>
<td>123</td>
<td>83</td>
<td>53</td>
<td>S</td>
</tr>
<tr>
<td>101 0100</td>
<td>124</td>
<td>84</td>
<td>54</td>
<td>T</td>
</tr>
<tr>
<td>101 0101</td>
<td>125</td>
<td>85</td>
<td>55</td>
<td>U</td>
</tr>
<tr>
<td>101 0110</td>
<td>126</td>
<td>86</td>
<td>56</td>
<td>V</td>
</tr>
<tr>
<td>101 0111</td>
<td>127</td>
<td>87</td>
<td>57</td>
<td>W</td>
</tr>
<tr>
<td>101 1000</td>
<td>130</td>
<td>88</td>
<td>58</td>
<td>X</td>
</tr>
<tr>
<td>101 1001</td>
<td>131</td>
<td>89</td>
<td>59</td>
<td>Y</td>
</tr>
<tr>
<td>101 1010</td>
<td>132</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
</tr>
<tr>
<td>101 1011</td>
<td>133</td>
<td>91</td>
<td>5B</td>
<td>[</td>
</tr>
<tr>
<td>101 1100</td>
<td>134</td>
<td>92</td>
<td>5C</td>
<td>\</td>
</tr>
<tr>
<td>101 1101</td>
<td>135</td>
<td>93</td>
<td>5D</td>
<td>]</td>
</tr>
<tr>
<td>101 1110</td>
<td>136</td>
<td>94</td>
<td>5E</td>
<td>^</td>
</tr>
<tr>
<td>101 1111</td>
<td>137</td>
<td>95</td>
<td>5F</td>
<td>_</td>
</tr>
</tbody>
</table>

110 0000 | 140   | 96      | 60          | `     |

110 0001 | 141   | 97      | 61          | a     |

110 0010 | 142   | 98      | 62          | b     |

110 0011 | 143   | 99      | 63          | c     |

110 0100 | 144   | 100     | 64          | d     |

110 0101 | 145   | 101     | 65          | e     |

110 0110 | 146   | 102     | 66          | f     |

110 0111 | 147   | 103     | 67          | g     |

110 1000 | 150   | 104     | 68          | h     |

110 1001 | 151   | 105     | 69          | i     |

110 1010 | 152   | 106     | 6A          | j     |

110 1011 | 153   | 107     | 6B          | k     |

110 1100 | 154   | 108     | 6C          | l     |

110 1101 | 155   | 109     | 6D          | m     |

110 1110 | 156   | 110     | 6E          | n     |

110 1111 | 157   | 111     | 6F          | o     |

111 0000 | 160   | 112     | 70          | p     |

111 0001 | 161   | 113     | 71          | q     |

111 0010 | 162   | 114     | 72          | r     |

111 0011 | 163   | 115     | 73          | s     |

111 0100 | 164   | 116     | 74          | t     |

111 0101 | 165   | 117     | 75          | u     |

111 0110 | 166   | 118     | 76          | v     |

111 0111 | 167   | 119     | 77          | w     |

111 1000 | 170   | 120     | 78          | x     |

111 1001 | 171   | 121     | 79          | y     |

111 1010 | 172   | 122     | 7A          | z     |

111 1011 | 173   | 123     | 7B          | (     |

111 1100 | 174   | 124     | 7C          | )     |

111 1101 | 175   | 125     | 7D          | )     |

111 1110 | 176   | 126     | 7E          | ~     |
- Physical inputs/outputs
  - In `setup()`, you must tell the board how to use a physical pin
    
    ```
    pinMode(pin, mode);
    ```
  
  - The `pin` refers to the address of the physical pin
  - The `mode` can be set to be either `INPUT` or `OUTPUT`
Physical inputs/outputs

- For a pin set to **INPUT** mode, you can obtain data from the pin with
  
  ```
  digitalWrite(pin)
  ```

- For a pin set to **OUTPUT** mode, you can send data to the pin with
  
  ```
  digitalWrite(pin, value)
  ```

- The value can be set to be either or **HIGH** or **LOW**
**Inputs & Outputs**

- **Physical inputs/outputs**

```cpp
void setup()
{
    Serial.begin(9600);
    pinMode(13, OUTPUT);
}

void loop()
{
    Serial.println("Testing LED");
    digitalWrite(13, HIGH);
    delay(200);
    digitalWrite(13, LOW);
    delay(200);
}
```
Outline

- Goal
- Review
- Variables & Data Types
- Inputs & Outputs
- Functions
- Conditionals
- Loops
Functions & Function Calls

- Many functions are pre-written and provided in programming language library. E.g.

```
Serial.println("Hello World");
```

- Many functions are written by programmers like you

```
s = getChrisClarksShoeSize();
```
A function is a named, self-contained, section of a program that is called to perform a specific task.

```java
void setup()
{
    ...
}
```
Functions & Function Calls

```
output setup(inputs)
{
    // operation
}
```
Functions & Function Calls

- Example Functions

```cpp
void sayHello(int n)
{
    Serial.print("lucky number is");
    Serial.println(n);
}

int getFive()
{
    return 5;
}
```
When creating a function, we write both
  - A function definition, e.g.

    ```cpp
    void sayHello()
    {
        Serial.println("Hello");
    }
    ```

  - One or many calls to that function, e.g.

    ```cpp
    void loop()
    {
        sayHello();
    }
    ```
For our goal of the lecture, let us

- *Define* an empty function called `getUserResponse()` that waits for an integer number from the serial Monitor and returns that integer

- *Call* that function every iteration of `loop()`
int listOfInts[5] = {3, 8, 4, 9, 1};
int response;

void setup()
{
    Serial.begin(9600);
}
void loop()
{
    Serial.println("Enter a Number");
    response = getUserResponse();
}

int getUserResponse()
{
    return 0;
}
Outline

- Goal
- Review
- Variables & Data Types
- Inputs & Outputs
- Functions
- Conditionals
- Loops
We want our program to make decisions depending on particular states of the system (e.g. values of variables.)

The most common function is...

```
if
```
Conditionals

- Syntax for if

```java
if (boolean statement)
{
    ...
}
```

- Read this as...
  - If the `boolean statement` is true, then do all operations within the `{} brackets` that follow.
For example

```java
if (speed > 10)
{
    speed = 10;
}
```
Another use of if

```java
if (speed > 10)
{
    speed = 20;
}
else
{
    speed = 0;
}
```
Goal
Review
Variables & Data Types
Inputs & Outputs
Functions
Conditionals
Loops
Loops

- We may repeat an operation many times, but we don’t want to write the code many times.

- The most common ways to loop are
  
  while loops
  
  for loops
Loops

- The syntax for while loops are

  ```java
  while (boolean statement) {
    ...
  }
  ```

- Read this as...
  - If the boolean statement is true, then keep doing the operations within the {} brackets that follow.
Loops

- For example

```c
int speed = 0;
while (speed < 10)
{
    speed = speed + 1;
}
```
Loops

- For loops iterate over an integer variable whose value changes with each iteration

```cpp
int speed = 0;
for (int i=0; i<10; i++)
{
    speed = speed + 1;
}
```
Loops

Read this as,

1. Let's declare an integer $i$ and set it to 0
2. If $i$ is less than 10, then run all operations in the {} brackets that follow.
3. After running the operations, let $i = i + 1$ and goto 2.

```java
for (int i=0; i<10; i++)
{
    speed = speed + 1;
}
```
Loops

- Lets combine *for loops* with *arrays*

```c
int speeds[10];
for (int i=0; i<10; i++)
{
    speeds[i] = 2*i;
}
```
Recall our lecture problem

```c
int listOfInts[5] = {3, 8, 4, 9, 1};
int response;

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

void loop()
{
    Serial.println("Enter a Number");
    response = getUserResponse();
}

int getUserResponse()
{
    return 0;
}
```
Loops

- We want to check our response with all 5 numbers on the listOfInts

```java
for (int i=0; i<5; i++) {
}
```
We want to check our response with all 5 numbers on the listOfInts

for (int i=0; i<5; i++)
{
  if (listOfInts[i] == response)
  {
  }
}
Loops

- We want to check our response with all 5 numbers on the listOfInts

```java
for (int i=0; i<5; i++)
{
    if (listOfInts[i] == response)
    {
        Serial.print("Match with ");
        Serial.println(response);
    }
}
```
Loops

- We want to check our response with all 5 numbers on the listOfInts

```cpp
for (int i=0; i<5; i++)
{
    if (listOfInts[i] == response)
    {
        Serial.print("Match with ");
        Serial.println(response);
        i = 10;
    }
}
```
for (int i=0; i<5; i++)
{
    if (listOfInts[i] == response)
    {
        Serial.print("Match with ");
        Serial.println(response);
        i = 10;
    }
    else if (i==4)
    {
        Serial.print("No Match with ");
        Serial.println(response);
    }
}
void loop()
{
    Serial.println("Enter a Number");

    response = getUserResponse();

    for (int i=0; i<5; i++)
    {
        if (listOfInts[i] == response)
        {
            Serial.print("Match with ");
            Serial.println(response);
            i = 10;
        }
        else if (i==4)
        {
            Serial.print("No Match with ");
            Serial.println(response);
        }
    }
    Serial.println(" ");
}
Loops

- We still need to write the function `getUserResponse`

```cpp
int getUserResponse()
{
    while (true)
    {
        if(Serial.available() > 0)
        {
            int pressed = Serial.read() - 48;
            return pressed;
        }
    }
}
```