Extras for Week 1

Naming convention for any modern computer language

- There are three main naming conventions used in programming
  - Pick one and use it consistently
- Naming Convention1: (left over from the early days of programming (60s &70s))
  - Random sets of letters and numbers for things (not recommended)
  - Code becomes impossible to understand
  - e.g.: Let's call heart rate → xyzzy1 and,
  - Let's call body temperature → xyzzy2
- Naming convention 2: Microsoft Convention
  - Use fully descriptive names with Capitals and lower case letters
  - e.g.: Let's call heart rate → HeartRate and,
  - Let's call body temperature → BodyTemperature
- Naming Convention 3: (Joe's preferred approach)
  - Use fully descriptive names with lower case letters and underscores (_)
  - e.g.: Let's call heart rate → heart_rate and,
  - Let's call body temperature → body_temperature

Naming Conventions may be language dependent

- There are more rules than described on the previous pages
- For more information see:
  - https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/general-naming-conventions
  - https://en.wikipedia.org/wiki/Naming_convention_(programming)

- Pick an approach and use it consistently
- Strong Suggestion: Do not pick the random approach!
  - It leads to code impossible to debug
- Some companies and teams may have their own conventions
  - If you join that company/team, use its convention

Data Types – Integer Numbers

- int
  - Uno
  - Duo
- short
  - all arduinos
- unsigned int
  - Uno
  - Duo
- long
  - all arduinos
- unsigned long
  - all arduinos

- Integer
  - Range: -32,768 to 32,767
  - Range: -2,147,483,648 to 2,147,483,647
- Integer:
  - Range: -32,768 to 32,767
- Integer
  - Range: 0 to 65,535
  - Range: 0 to 4,294,967,295
- Integer:
  - Range: -2,147,483,648 to 2,147,483,647
- Integer
  - Range: 0 to 4,294,967,295

Data Types – Floating Point Numbers

- float
  - 32 bits:
  - -3.4028235E+38 to 3.4028235E+38
  - https://en.wikipedia.org/wiki/Single-precision_floating-point_format
- double
  - Uno same as float
  - Duo 64 bits
  - https://en.wikipedia.org/wiki/Double-precision_floating-point_format

- floats and doubles
  - Floats have only 6-7 decimal digits of precision
  - Floating point numbers are not exact
  - Thus floating point arithmetic is not exact.
  - For example 6.0 / 3.0 may not equal 2.0,
    - i.e. (6.0/3.0)-2.0 may not equal 0
  - When doing comparisons, we check that the absolute value of the difference between the numbers is less than some small number.

Use of int vs float

- Integer arithmetic will be processed much faster than floating point arithmetic
- But!
- Integer arithmetic can produce strange results
  - 1 / 2 = 0 in integer arithmetic
  - 3 / 2 = 1 in integer arithmetic
- Floating point arithmetic can also produce strange results
  - (2.0 / 1.0) -1.0 may not equal zero

- Which one you use will depend on what you need to do with your code

Data Types - Other

• bool	• Values of true or false
• boolean	• non-standard type alias for bool (don't use)
• byte	• 8-bit unsigned number, from 0 to 255
• char	• Character (a, b, A, 1, 2, etc.)
• unsigned char	• Number from 0-255 use byte instead
• word	• Unsigned integer from 0-65535 same as unsigned int
• array	• a collection of variables that are accessed with an index number; index starts at 0
• string	• An array of characters ending with 00 (\0)
• String()	• A better way of building a string (note capital S)
• void	• void is used in functions to show that the function does not return a value (we will discuss functions later)

Constants

- Constants are used when they do not change during the running of a program,
    - e.g. pi=3.141592. We do not normally change the value of pi
- Floating Point Constants
- Integer Constants
- Constants defined by the Arduino development environment
    - HIGH | LOW
    - INPUT | OUTPUT | INPUT_PULLUP
    - LED_BUILTIN
    - true | false

Arithmetic Operators and Comparison Operators

%	remainder
*	multiplication
+	addition
-	subtraction
/	division
=	assignment operator

!=	not equal to
<	less than
<=	less than or equal to
==	equal to
>	greater than
>=	greater than or equal to

Boolean Operators

- Boolean Operators are used to perform operations on variables
- If you do not understand how Boolean operators work, we will explain them when we get to the examples

• !	logical not
• &&	logical and
• ||	logical or

Conversion

- Sometimes we need to change the form of a variable
- For example, we might have an integer and need to change it to a float

- float()
- int()
- long()
- word()
- byte()
- char()