Programming with Alice

Variables and Functions
So far with Alice, we have learned:

- How to set up our world / scene
  - How to create OBJECTS from CLASSES
  - How to pose and move our OBJECTS (and using the SET POSE property)
  - How to use the VEHICLE PROPERTY to move OBJECTS together
- How to use PRIMITIVE and CUSTOM METHODS to animate our OBJECTS
  - Including creating our own METHODS
- The importance of thinking and planning out our program ahead of time
  - i.e. Writing out our program in pseudocode
Our next step in programming is the idea of **CONSTANTS** and **VARIABLES**
- both will become more important as our understanding of programming increases

From mathematics, we know that
- **CONSTANTS** are values (numbers) that never change
  - Hence, they are **constant**
- **VARIABLES** are values (numbers) that can change, depending on the use of the equation
  - Hence, they are **variable**
Programming with Alice

Variables

From a programming point of view, both CONSTANTS and VARIABLES represent stored memory locations in the computer (usually stored in the RAM of the computer)

- A **CONSTANT** is a space that the computer uses to store data in the program that will not change during the program
  - E.g. the value of pi (π) is approximately 3.14159, so we can save it as a constant called *pi*

- A **VARIABLE** is a space the computer uses to store a data that will change during the program
  - E.g. if we create a circle object, we could store that circle’s diameter as a variable called *diameter*
    - The diameter will change if we re-size the circle
Using variables and constants, we can store information that will be useful as part of our program later on.

For example, we can then use the CONSTANTS and VARIABLES to perform mathematic calculations:
- E.g. to find the CIRCUMFERENCE of the circle, we multiply the value stored in $\pi$ by the value stored in $\text{diameter}$.

\[
\text{Circumference} = \pi \times \text{diameter}
\]
Programming with Alice

Variables

In programming, we really only create variables
  – A constant is just a variable that does not change during the program

When creating variables, the programmer must specify the:
  – NAME of the variable
  – TYPE OF DATA that the variable will hold
  – SCOPE of the variable
NAME of the Variable

- Should be meaningful and represent what is being stored in the memory location
  
  Remember: you may not be the only one who has to work with this program, so use variable names that other people would understand and would make sense
  
- Should be written in camelCase
  
  (1st word lowercase, 2nd word uppercase, only letters and numbers, no spaces)
Creating Variables

DATA TYPE

- Specifies to the computer what kind of data is to be stored
  - This determines how much space in the memory that your variable will need
- Examples of data types
  - Boolean (a true or false value, yes/no)
  - Numbers
  - Text (written words)
Programming with Alice
Creating Variables

**SCOPE**

– Determines what parts of the program will have access to this variable
  
  **Is the variable available to the entire program?**
  OR
  **Is the variable only used in part of the program, therefore it is only needed in one part?**

– Keeping variables organized makes your program easier to understand
It is proper programming practice to give your new variable an **INITIAL VALUE**
- This way we are sure the variable at least has a value that we know when the program starts.
Programming with Alice
Creating Variables in Alice

In Alice, we have to specify:
- **Name of Variable**
- **The Type of Variable**
  - In Alice we have the following options:
    - **Number** – can be any *real* number (integer, fraction, etc.)
    - **Boolean** – holds either the value TRUE or the value FALSE
    - **Object** – holds an Object in Alice
    - **Other** – can be many different options (string, colour, etc.)
- **The Initial Value**
Programming with Alice
Creating Variables in Alice

The SCOPE (category) of the variable is determined by where we created the variable

1. Local Variables
   - belong to a specific method.
   - can only be used inside the method where it is defined. Outside the method, it is like the variable does not exist.
   - When the end of the method is reached, the variable ceases to exist.

2. Parameter Variables
   - used to hold an argument that is being passed to a method.
   - e.g. the move to method allows you to specify another object as the destination. This target object is specified using the parameter variable of the move to method.

3. Class-Level Variables
   - Are like the properties that belong to specific classes (and objects).
   - Can be set or modified before or during program execution.

4. World-Level Variables
   - properties that belong to the entire world.
   - All objects in the world can see their values, and any method in the world can use or modify the variable.
Programming with Alice
Creating Variables

ACTIVITY
– Work on
  - Exercise 2.1: Creating and Using Variables in Alice
  then
  - Exercise 2.2: Changing a Variable Using the SET Instruction
FUNCTIONS

– Are special subprograms that are used to return a value back to the original program
– In Alice, these functions are special methods that return values back to the instruction that called the function

The VALUE that is returned is very similar to a VARIABLE

– and like a VARIABLE, a function can be used as a PARAMETER for a method or another function
In Alice, there are PRIMITIVE FUNCTIONS associated with all OBJECTS, including the WORLD OBJECT.

To explore the available functions, select the object and select “functions” from the DETAILS panel.

The functions are separated into the following categories:

- Boolean Logic
- Math
- Random
- String
- Ask User
- Mouse
- Time
- Advanced Math
- Other
ACTIVITY

– Work on

Exercise 2.3: Asking the User to Input a Number in Alice
Programming with Alice
Functions

**Primitive Object Functions**

– In general fall into these categories:
  - Proximity – returns info about how close the Object is to other objects
  - Size – returns info about the size of the Object
  - Spatial Relation – returns info about where the Object is relative to other objects
  - Point of View – returns info related to the Object’s point-of-view
  - Other
ACTIVITY

- Work on

  Exercise 2.4: Using the Proximity Function in Alice
Programming with Alice
Creating Variables

**ACTIVITY**

– Begin working on Activities 1 to 3 in the ALICE VARIABLE AND FUNCTIONS ACTIVITY (on my WIKI page)