## Reserved Names:

* True
* False
* Default functions

## Data Types:

### Supported Data Types:

* Scalar Number
* Scalar String
* Scalar Bool
* 1D array of Number
* 1D array of String
* 1D array of Bool

### Data Type format:

* Number = numeric integers, float and any scientific representation
* String = Anything within the Quotes
* Bool = True and False (case insensitive)
* 1D = represented within the braces ‘[‘ and ‘]’

All the functions created should support the available data types or should have clear documentation on what would be the supported datatypes

### Data Type library

* Data Type class will be created
* This class will handle all the datatypes in its memory using the variant
* No class hierarchy will be developed for this as of now but they can be extended to a class based data type later

## Variables:

Variables is a Key Value pair concept where a Key is the name of the Variable and the Data is a data type object of one supported data type

### Syntax:

* ‘Key / Variable Name’ = ‘Data Type representation from the above’
* Key can be a string without any spaces and special characters. Variable cannot start with a numbers
* ‘\_’ is allowed

### Run Time Variable:

* Run time variable could be defined in the functions and its life time is only within the function
* Run time variable works based on the indentation of the scripts, the life time is for the inner indendation

### Global Variable:

* All the Variables defined in the Variable definition is Global Variable

## Variable Manager:

* Parses all the variables script
* Stores all the Variable information in it and it should be reference based
* List of all Key and Value Object Pairs
* Could use the Variable function to parse the details
* All the functions takes in the variable manager

## Functions:

Functions are a line of string that does anything

* A line could be parsed and associated to a function object
* A function line could be recursive and could handle many functions
* A function can take in list of Input Data Types or Variables and then return a data type
* Each function can take in the Global Variable Manager in and then create its own Run Time Variables whose life time is restricted to only that function

### Function Utils:

* Utils that helps load the function and supported functions
* Set the priority order to the functions
* Manage the functions in a specific order

### Smart Scan from String:

* Gets the LabVIEW Scan from string format identifier the variable count
* Functions can only ‘scan’ the variables and other syntax elements should be defined clearly
* One function can have several syntax’s

### Developer experience in creating a function:

* Unique Identifier for functions like for, while that can help in intellisense
* Simple way to define the function Variables along with how it is called
  + Define a Variable Cluster
  + Define the format Syntax based on the Variable cluster for your function
    - For Example, if your function is for(i = 10;i<=0;i++)
      * Then your variable is I,10,0
      * So the syntax should be ‘ for($Vairablei$ = $Init$;$Vairablei$<=$Last$;$Variablei$+$Increment$)’

### Function Types that are made possible:

* Simple standalone functions like Dialog (One button and Two button)
* Variables functions (that assigns value)
* Nested functions like for loop and while loop
* Nested functions that jumps like If and Switch
* Functions that return some values (conditional statement that returns true or false)
  + Return Values
  + Return break or continue functions
  + Abort

## Script Manager:

* Stores the Variable Manager and the list of functions
* Manages the sequence of functions
* Save, Load, Runs, Compiles the scripts

### Methods:

* Instantiate
* Destroy
* Compile
* Execute

## UI Containers:

These are the objects used in the UI for quickly putting together the scripting languages

### VariablesPane\_IndexedString:

### ScriptingContainer\_IndexedString:

### CompilationResults\_Listbox: