

Programming Project - Unit 3

B# / Block#

Lewis Miller | Candidate Number: 2086

Computer Science | H446-03 Programming Project

Bournemouth School | Centre Number: 55119

# CONTENTS

CONTENTS 1

ANALYSIS 2

Identifying the problem 2

Identifying stakeholders and their needs <TBF> 2

How the stakeholders will use the product, and why it suits their needs <TBF> 2

Similar products and existing solutions 2

Python {IDLE} 2

Scratch {IDE} 4

Small Basic {IDE} 5

MicroBit {IDE} 8

Specifying a solution 11

Design Requirements <TBF> 11

Software Requirements <TBF> 12

Hardware Requirements <TBF> 13

Parameters of success <TBF> 13

DESIGN 14

The components of the problem 14

Programming Techniques 14

Designing a solution 15

DEVELOPMENT 16

REFERENCES 17

# ANALYSIS

## Identifying the problem

In a society that is rapidly advancing through the technological era, it is becoming more and more imperative for younger students to gain an understanding of not only how computers work, but also how they can be used and programmed. As such I will be developing a program designed to help younger students not only to become engaged in programming but also to develop thinking skills that can be helpful for completing tasks later in life.

Because the problem is primarily focused on how school students in lower years interact with and understand computers and computational methods, a suitable approach could prioritise a “hands-on” approach to the subject, aiming to encourage pupils to gain an understanding through trial and error. This would allow them to gain a deeper insight into how real computer programs run in real life. To do this, students could be provided with their own IDE or similar program, which itself would have to be programmed. These aspects make the program amenable to a computational approach as providing a rich learning environment integrated with the ability to produce a working example of software is often difficult to do with plain pen and paper.

## Identifying stakeholders and their needs <TBF>

From what I can tell, there are two primary groups that will use this software: Computing Teachers and their Students. As such the program will be an educational assistant and should aim to help teaching students about computing.

* Teachers
  + Effective teaching aid
  + Easy to show a class how to use
  + Engages students with a variety of features
* Students
  + Easy to learn, understand and use
  + Allows for developmental experimentation with computational methods

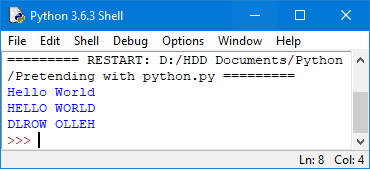
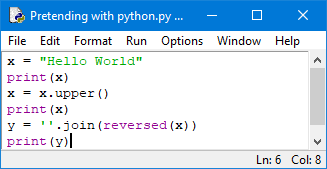
### How the stakeholders will use the product, and why it suits their needs <TBF>

Teachers could use a program like this to help young students who are trying to learn how to understand basic computational methods and algorithmic thinking. The

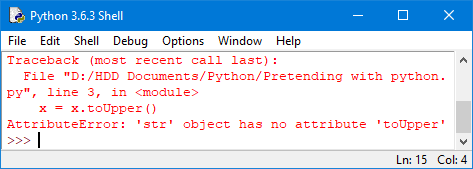
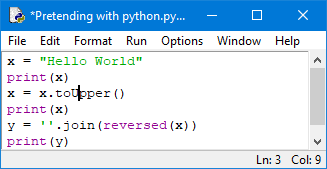
## Similar products and existing solutions

### Python {IDLE}

Python is a high-level programming language with simple whitespace- / colon-based syntax that uses an English-like command set. Version 0.9.0 was first released in 1991 and since then it has grown into one of the most popular programming languages for people new to programming. The Python IDLE is a very minimal IDE, highlighting keywords and automatically indenting your lines. It has no error checking or advanced features that IDEs like Visual Studio and WebStorm possess. The Shell will show the error when it occurs, but there is very little detail given, making it harder for novice programmers to understand what went wrong.



The Python IDLE {left} and Shell {right}



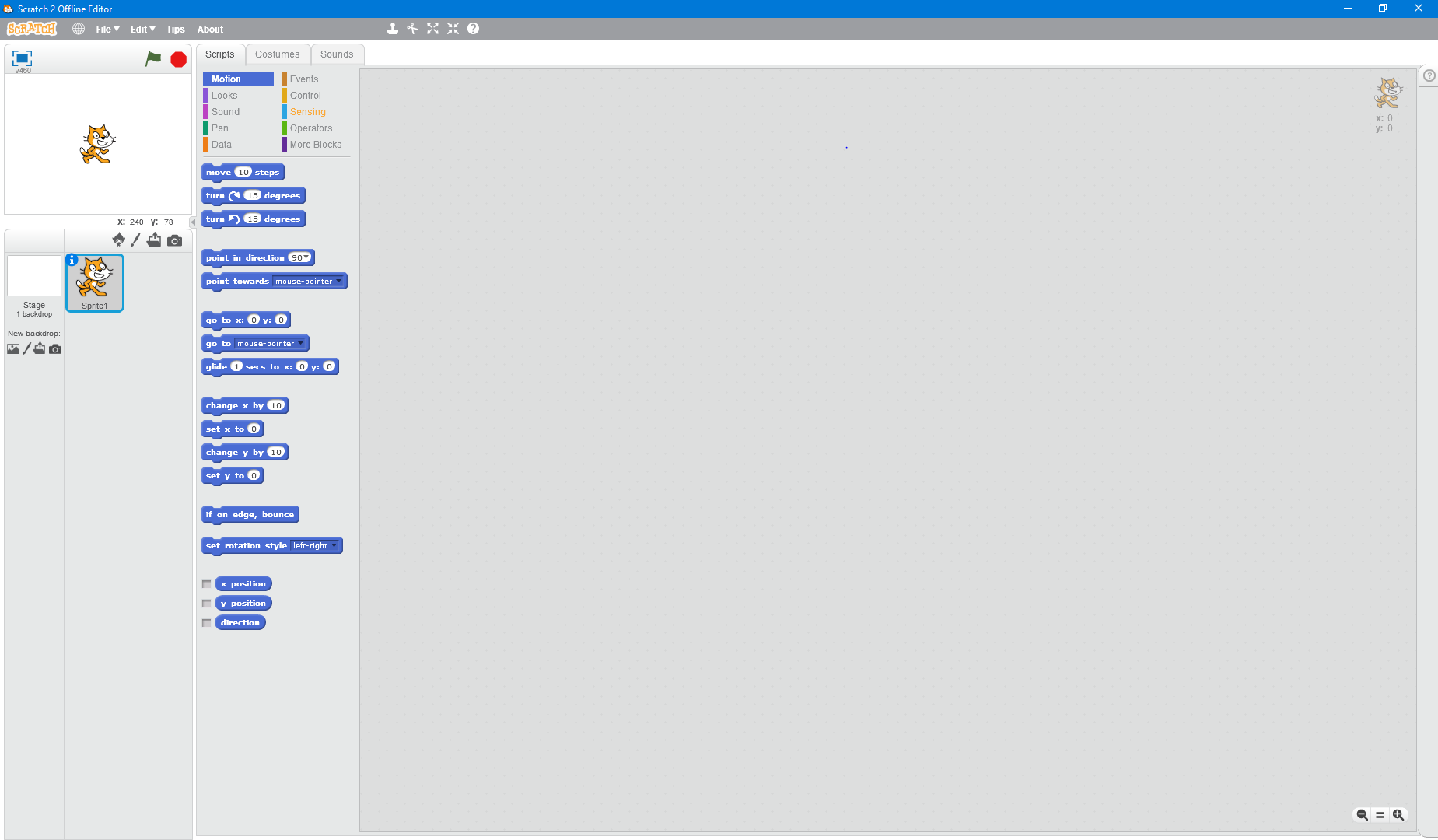
When an error occurs, the Shell tells you where it happened and what went wrong, but it is not a detailed description

Python is good for new programmers as it provides an easily understood programming language with a lot of extensibility and compatibility options. The primary drawback of Python isn’t the language, but the integrated IDLE. It lacks most forms of assistance that many other IDEs provide, requiring novices to refer to documentation repeatedly while starting out so they can understand what went wrong and why.

|  |  |
| --- | --- |
| Positive attributes | Negative attributes |
| Simple and easy to learn language | Poor built-in debugger |
| Rich and feature filled extensible environment | Does not allow for Types of objects in OOP programming |
| Lots of developer support available both on paper and internet | Dynamic typing can make debugging a program much more complicated |

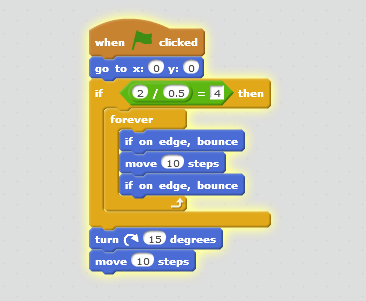
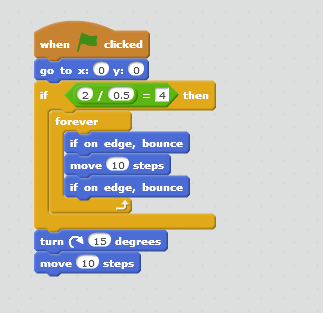
### Scratch {IDE}

Scratch is a visually orientated programming format produced by the Mississippi Institute of Technology. It provides a blank canvas and a “sprite” when first opened, along with a large number of “blocks” which each have their own individual functionality. As simple as Scratch is on the surface, the variety of tools available for use allow for an incredible range of design possibilities. The way Scratch treats “sprites” as individual objects makes programming their behaviours easy as you do not have to consider how they interact unless you want to while also allowing for an understanding of simple object orientation. The IDE is simple, but bright and colourful with a sans-serif font to allow for easy reading by younger students.



The Scratch IDE, with a sprite and background

Scratch does not provide any debugging tools aside from highlighting the currently executing blocks. This means that if the program stops working, the only description of the issue is where code was last executing.

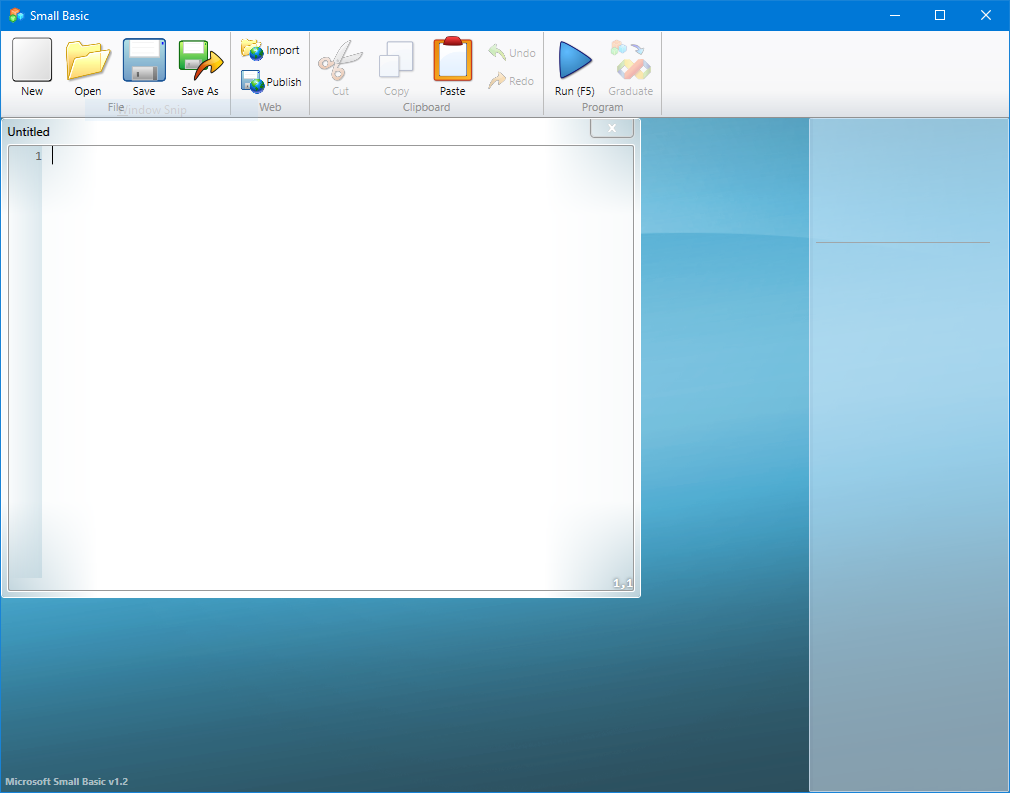


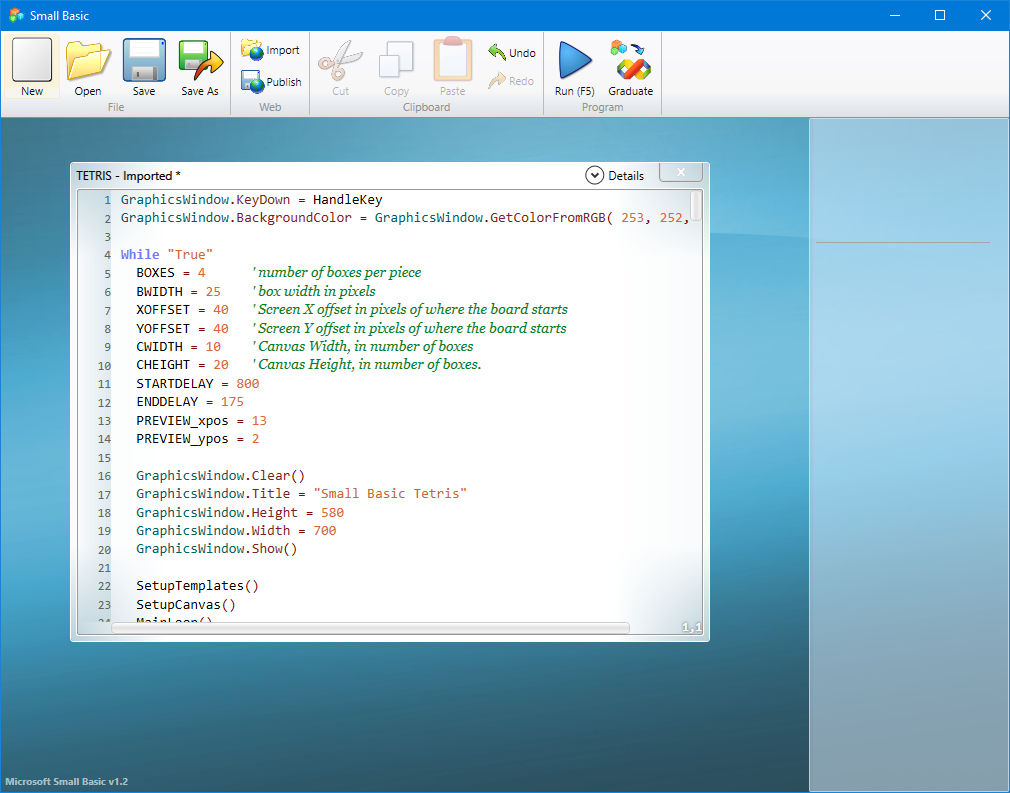
The editor highlights currently executing code.

|  |  |
| --- | --- |
| Positive attributes | Negative attributes |
| Bright colours are easy to understand, with each colour referencing a different type of block | Relies on the built-in graphics window to demonstrate code |
| Simple coding style is easy to follow, with blocks showing the order of execution | Lacks debugging assistance |
| Easy to use interface provides young users with a friendly environment | Relatively slow run speed |

### Small Basic {IDE}

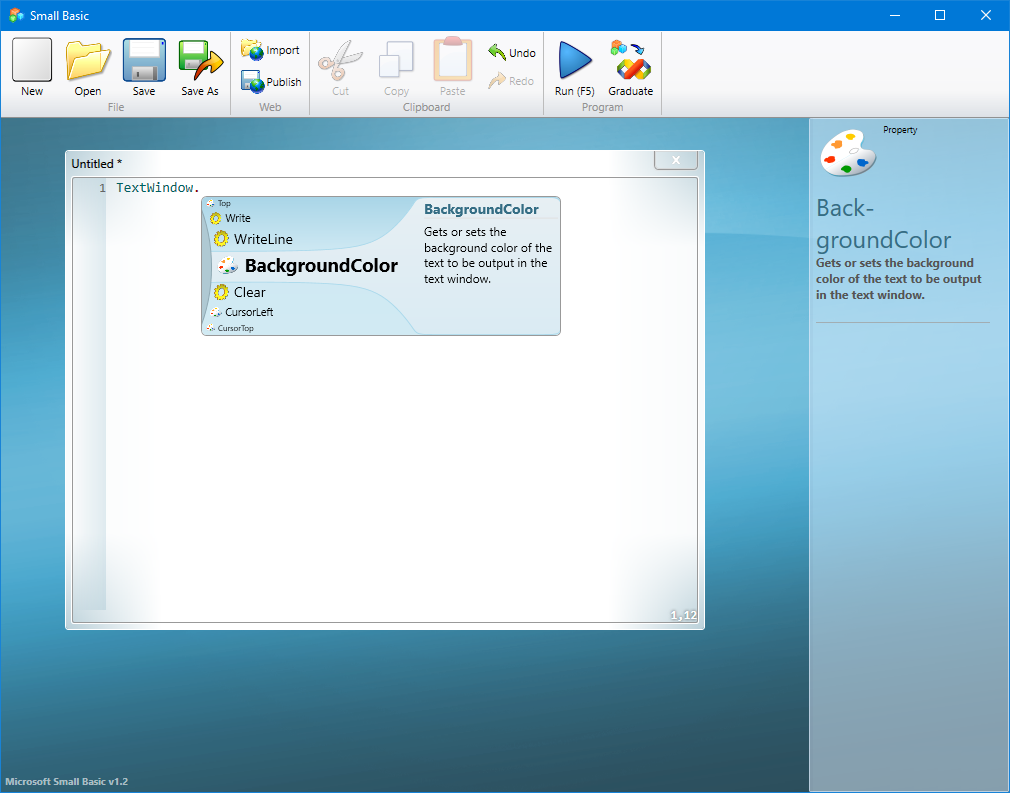
Small basic provides another IDE for young students similar to the Python IDLE, however the scripting language of choice is Small Basic itself. Small Basic provides a simplified version of the more advanced Visual Basic programming language developed by Microsoft. The IDE uses large, easy to understand icons to communicate with the user, and allows users to import code from the online library to help show them what they can achieve in Small Basic.





The top screen appears when you first launch Small Basic. The bottom screenshot shows the result of using the import feature with the ID “TETRIS”.

Small Basic also has a feature called IntelliSense, which is a professional-level tool incorporated into Microsoft’s other code editors such as Visual Studio and Visual Studio Code. IntelliSense allows the user to easily see what functions and variables are currently available to them, along with a brief description of what they do.



IntelliSense in action. The popup shows every command available, and can insert snippets using Tab. The details also appear in the pane on the right.

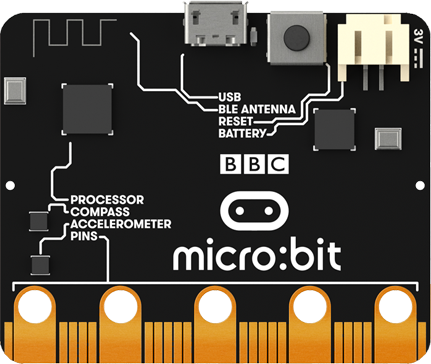
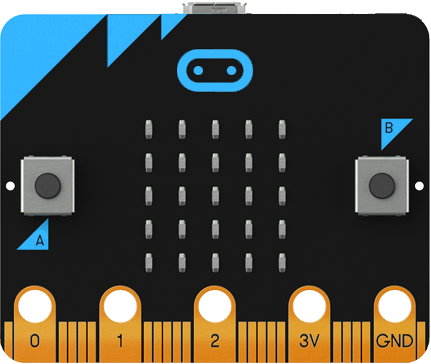
Like Python, Small Basic provides its own coding environment. It differs, however, not only in terms of coding assistance provided by IntelliSense, but also in that it is a compiled language. This means that all errors can be spotted upon compilation, instead of being encountered while the program runs. This is much more user friendly to a novice programmer as it allows them to understand the code they’re writing without having to look at a programmer’s reference as often.

Small Basic also provides two objects to assist students: The GraphicsWindow and TextWindow objects. The TextWindow is a simple command line interface that can receive text as inputs and outputs text as instructed. The GraphicsWindow has built in graphics methods that make it easy to use and interact with, and allows students to better see what their code is doing at any given time.

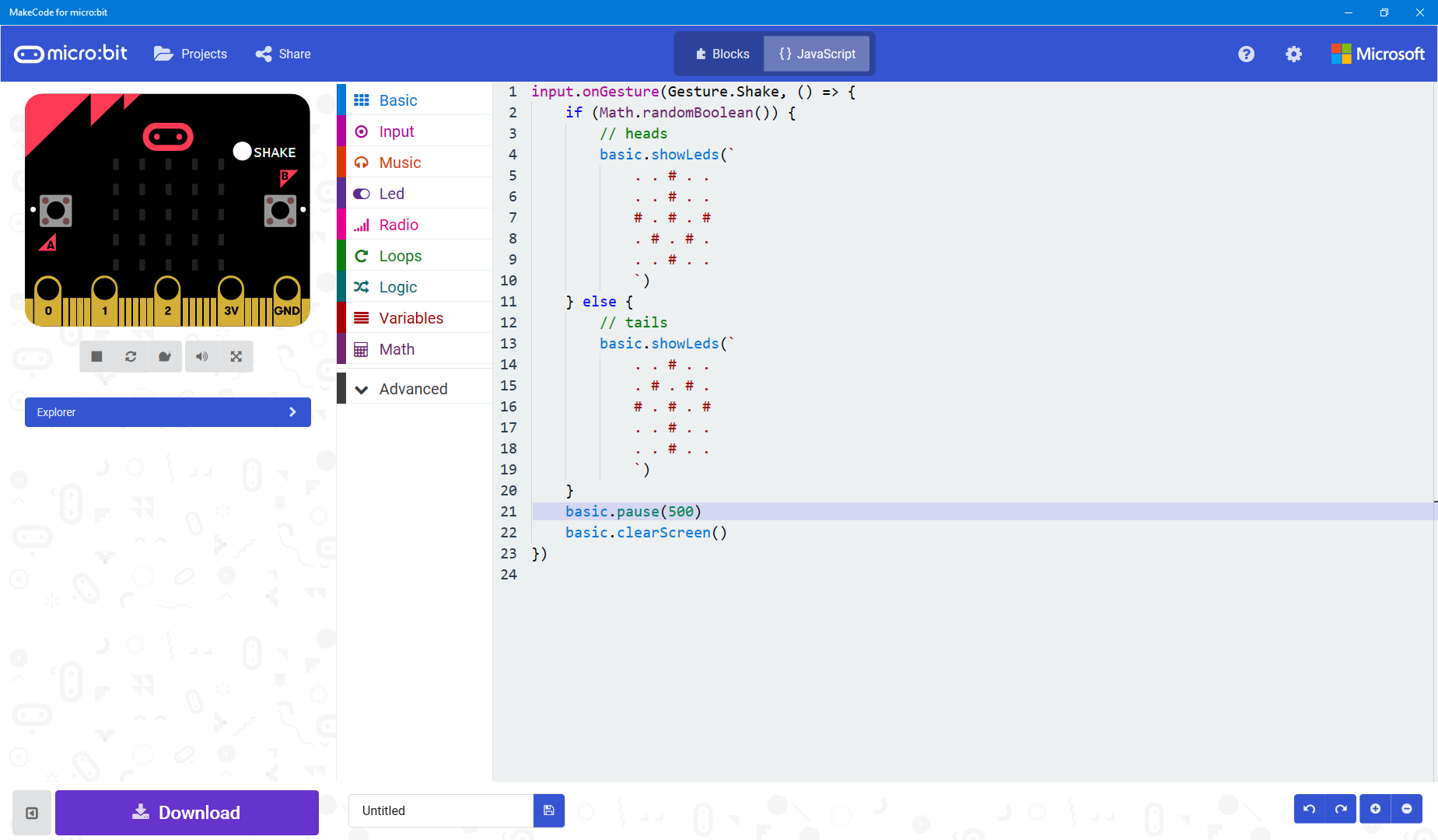
|  |  |
| --- | --- |
| Positive attributes | Negative attributes |
| Large icons and bright colours allow for an easy understanding of the features available | Takes a lot of time to learn the language to begin with |
| IntelliSense and compiler make debugging program much easier | IntelliSense may begin to make students complacent with coding |
| The built-in ability to migrate code to Visual Basic gives students a greater knowledge of how high-level programming languages work | Aside from migrating code, there is little room to build upon a finished project with Small Basic itself, preventing students from building on what they have already learned |

### MicroBit {IDE}

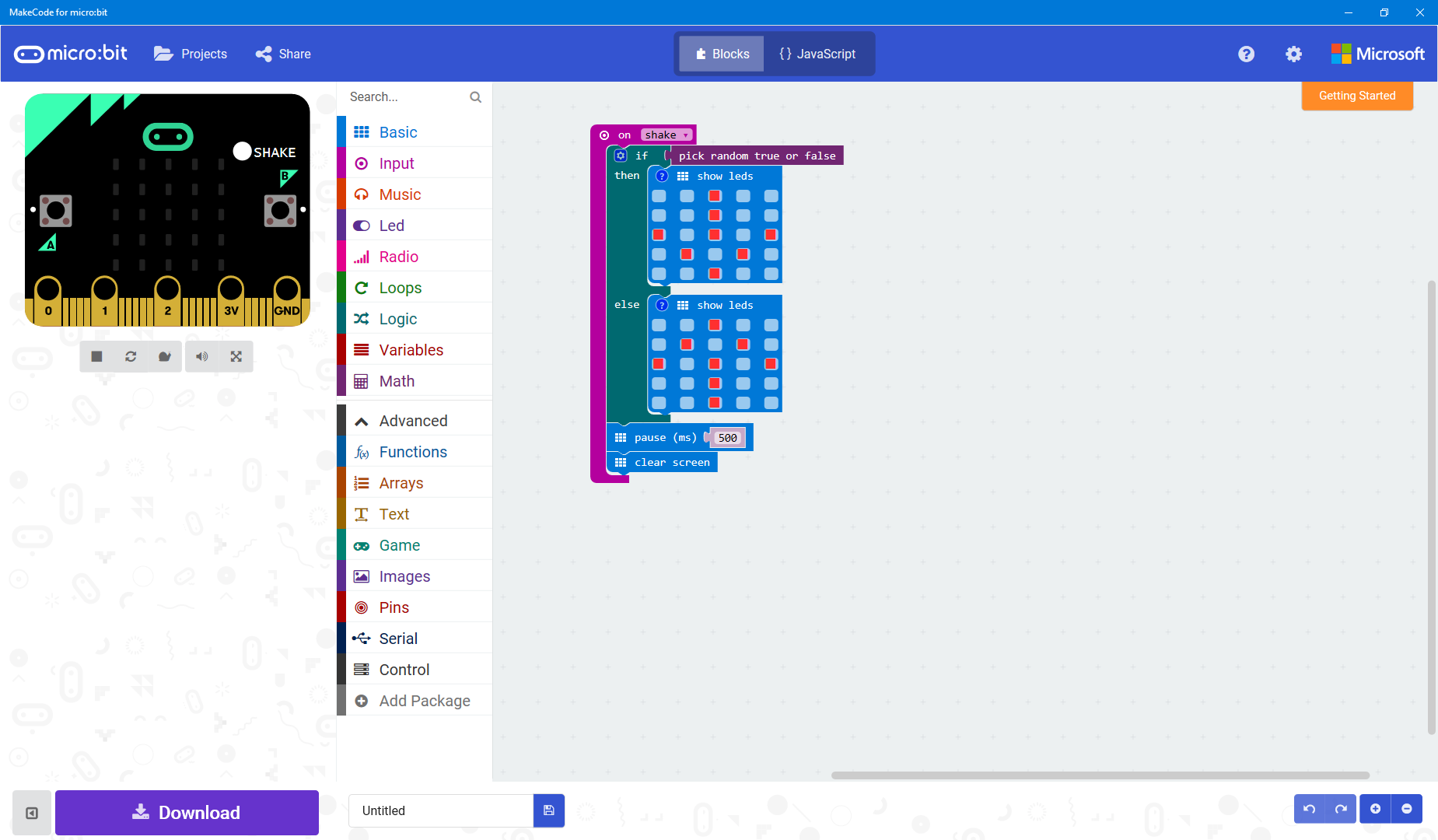
The MicroBit, produced by the MicroBit Educational Foundation {MEF}, is a piece of educational hardware designed to get young students to be interested in programming. The MicroBit itself is a handheld electronic device that can be programmed using Python or JavaScript {JS} using an online editor or an offline editor {I will be using the offline editor}.



The MicroBit hardware has two PTM switches and a 25-LED screen on the front, 20 pin connections at the bottom, a micro-USB port, a Bluetooth/Radio antenna, a compass, an accelerometer and a surprisingly powerful processor.

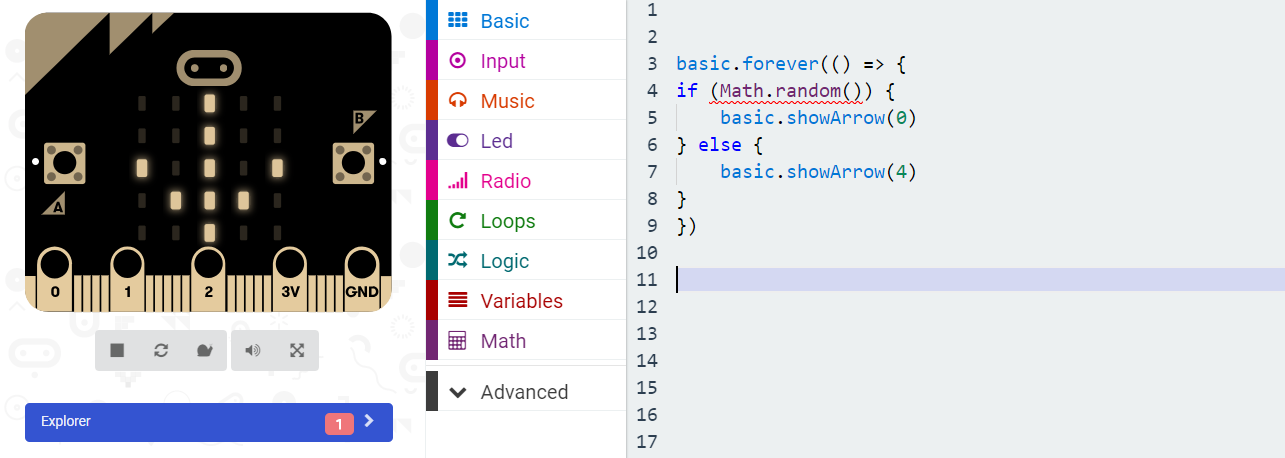


The MicroBit JS code editor {MakeCode for MicroBit Windows App}

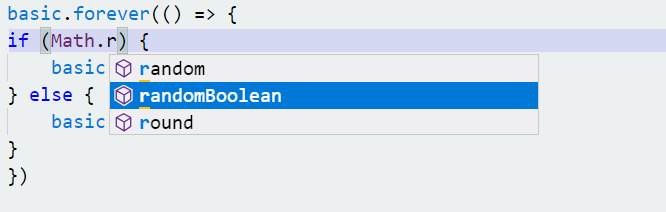


The MicroBit block editor for the same code

The JS editor allows you to edit MicroBit code in both a block-based format and a text-based format. This makes it possible to switch between the two, similar to how Small Basic allows its user to migrate to Visual Basic. Unlike the Small Basic migrator, however, the JS editor allows you to switch between text and blocks seamlessly. The JS editor also displays a MicroBit on the top left of the screen, allowing you to test your code without having to wait to download it to a MicroBit.



If your code contains an error, the MicroBit will appear greyed-out and a notification informing you where your code doesn’t work will be shown. The MicroBit also appears greyed-out when the code is compiling



This editor also makes use of IntelliSense, as Microsoft helped produce the software.

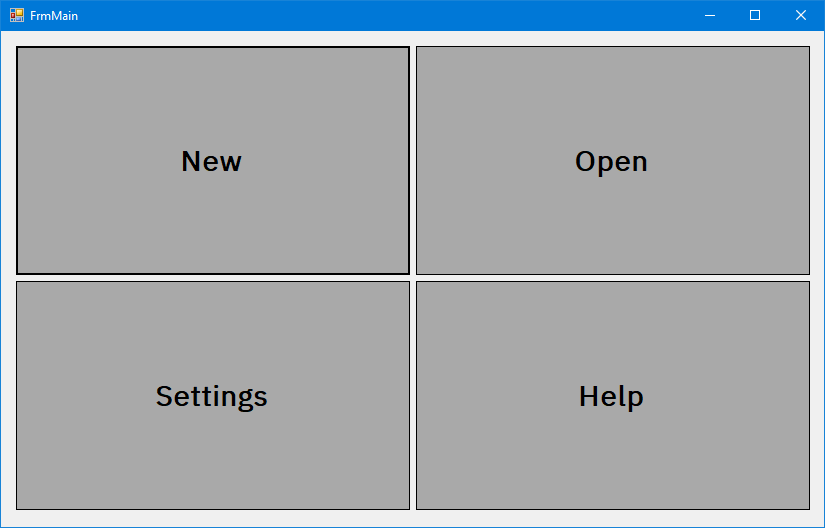
The MicroBit IDE is excellent for giving instant feedback about your code, as it can highlight errors and suggest code snippets as well as recompile your code as you work on it.

|  |  |
| --- | --- |
| Positive attributes | Negative attributes |
| Components display a tooltip over them to help detail their use | Like Small Basic, this IDE comes equipped with IntelliSense, which may cause some young programmers to become accustomed to it |
| Compilation occurs at the same time as editing, aiding troubleshooting | This IDE is specialised for a specific purpose, programming the MicroBit, however many IDEs now are suited to multiple tasks |
| Unlike Scratch and Python, the IDE has a debugger built into it, allowing for a simpler debugging process | The Windows App IDE is currently in development and unstable, however this is likely to change |
| The features used by the IDE are extensible, meaning that additional functionality can be created by pupils if they wish to improve their coding ability |  |

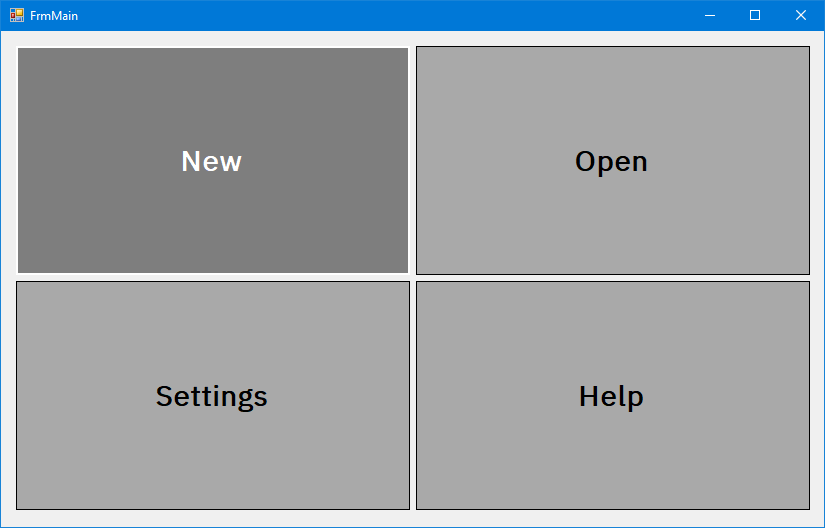
## Specifying a solution

### Design Requirements <TBF>

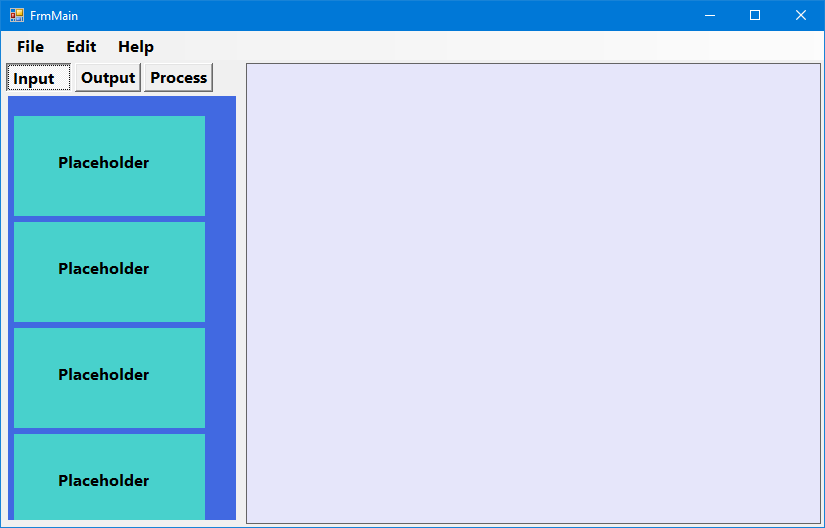
|  |  |  |
| --- | --- | --- |
| Requirement | Justification | Description / Implementation |
| Easy to read and understand | Young students are not always able to read as competently as their elder counterparts, as such the design should reflect this | Large, colourful interfaces with contrasting text colours to maximise readability. Sans-Serif fonts could also be used |
| Intuitive usage | Explaining new concepts can be quite difficult, so the program should be easy to use, with help available if necessary | Tooltips could pop up after the mouse is hovered over the block, or a button could be clicked to show a help box. A full help page should also exist |
| Stability | A program that crashes often is not usable in any environment, and as such the program should be as stable as possible. | Because students will be encouraged to code their own applications, errors could be handled within the program instead of crashing the entire program on a block-by-block basis |
|  |  |  |
|  |  |  |



An idea for the starting screen {No colours present, however the individual tiles would have their own colour}



The colours would change when the user moves their mouse over the button



Alternatively, the user could be taken directly to the IDE window

### Software Requirements <TBF>

|  |  |  |
| --- | --- | --- |
| Requirement | Description | Justification |
| Windows Vista or later | The OS should be a modern, Windows-based OS | My chosen language, C#, compiles directly into a “\*.exe” file, which can only be run by Windows OSs |
| <Look up software requirements for C#> |  |  |
|  |  |  |

### Hardware Requirements <TBF>

|  |  |  |
| --- | --- | --- |
| Requirement | Description | Justification |
| 4GB of RAM or more | The system running the program should possess at least 4GB RAM | This allows the system to run the program with a lower chance of running out of RAM |
| <TBC> |  |  |
|  |  |  |

## Parameters of success <TBF>

* Does the program run without crashing?
  + Even when it encounters an unknown or new error?
* Can a teacher use it to help assist students learning?
  + Do the students feel that it is an effective learning aid?
  + Does the teacher find it easy to teach with?

# DESIGN

## The components of the problem

The problem can be split into several sections:

* Assists students in the development of programming techniques
* Is easy to use
* Has a clear output
* Allows for basic debugging

### Programming Techniques

This is by far the largest of all the sections, and can be split down again into several subsections

* Inputs and outputs
* Logic and logical operations
* Functions and subroutines
* Data types and structures
* Development techniques

#### Inputs and Outputs

By providing a clear way of inputting data and outputting results, students are given the ability to begin to understand how programming works, as simple command-line programs can be developed and built on easily.

#### Logic and Logical Operations

A fundamental part of computing is the logical operations that allow computers to run. An understanding of how these work is essential, and as such the program will contain references to these.

#### Functions and Subroutines

Programmers use functions and subroutines to break a program down into individual elements. These allow them to make more progress and to make code easily re-useable, and are essential in modern programming.

#### Data Types and Structures

Types are used to allow data to be represented, manipulated and stored. They are incredibly useful in programming as they allow data to always be used in a given way. Building on from types are structures, which provide extensibility to standard types as well as added data security, which are important concepts to understand.

#### Development Techniques

## Designing a solution

To begin with, I will develop a Command-Line Interface {CLI} based Input-Output {IO} system that allows users to both take inputs from the CLI and output results to it. If I get the time I would like to also develop a Graphics-Based interface as well, but given the timeframe I have to work with, along with inherent difficulties that come with an approach using C#, it is unlikely that this will be completed.

The logical operators, data types and functions will all be focused on

# DEVELOPMENT

# REFERENCES

|  |  |  |
| --- | --- | --- |
| Website Name | Web Address | Date First Accessed |
| Python Official Website | <https://www.python.org/> | 14/05/18 |
| Scratch Official Website | <https://scratch.mit.edu/> | 14/05/18 |
| Small Basic Official Website | <http://wwwsmallbasic.com/> | 14/05/18 |
| MicroBit Official Website | <http://www.microbit.org/> | 14/05/18 |
|  |  |  |