

Programming Project - Unit 3

[helping small children achieve their dreams since 2019™]

Lewis Miller | Candidate Number: 2086

Computer Science | H446-03 Programming Project

Bournemouth School | Centre Number: 55119

# Mark scheme

1. **Analysis**
   1. Problem identification
      1. Description of the problem
      2. Describing and justifying the features that make the problem solvable by computational methods
      3. Explaining why the problem is amenable to a computational approach
   2. Stakeholders
      1. Identifying parties that may have an interest in the solution
      2. Explaining how the solution is suited to the stakeholder’s needs
   3. Researching the problem
      1. Finding instances of the problem
      2. Researching pre-existing solutions
      3. Identifying and explaining essential features of a solution
      4. Identifying and explaining the limitations of the proposed solution
   4. Specification of a solution
      1. Specify and justify the solution requirements including hardware and software configurations where appropriate
      2. Specify and justify measurable success criteria for the proposed solution.
2. **Design**
   1. Breaking down the problem into its constituent components
   2. Describing a solution
      1. Describing the structure of a solution
      2. Describing and justifying the individual algorithms used in the complete solution
      3. Describing usability features used in the solution
      4. Identifying key variables / data structure / classes and justifying / validating my choices
      5. Identifying and justifying useable test data during development and post development
3. **Development**
   1. Iterative development
      1. Commented code listings for each stage
      2. Evidence of prototype solutions for each stage
   2. Testing to inform development
      1. Evidence of testing at each stage, justifying the reason for the test
      2. Evidence of remedial actions in later iterations
4. **Evaluation**
   1. Testing to inform evaluation
      1. Evidence of testing robustness of final solution
      2. Evidence of usability testing (User feedback)
   2. Success of solution
      1. Comparison of process and solution with the original success criteria
   3. Describing the final product
      1. Commenting on the effectiveness of the design and usability
   4. Maintenance and further development
      1. Discussion of maintainability of solution
      2. Discussion of potential further development

# Contents

Mark scheme 1

Contents 3

Analysis 4

Identifying the problem 4

Stakeholders 4

Similar products and existing solutions 4

Python (IDLE) 4

Scratch 5

Small Basic 6

BBC Micro:Bit 7

Specifying a solution 7

# 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.

## Stakeholders

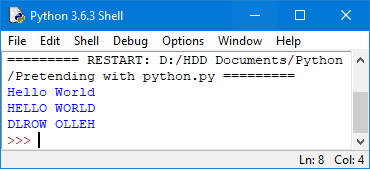
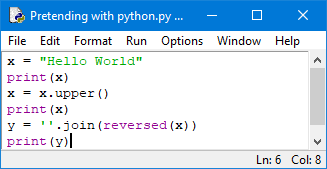
As the program is an educational assistant its primary stakeholders will be teachers and students.

* Teachers
  + Effective teaching aid
  + Easy to show a class to use
* Students
  + Easy to learn, understand and use
  + Allows for developmental experimentation

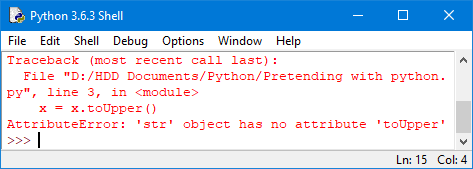
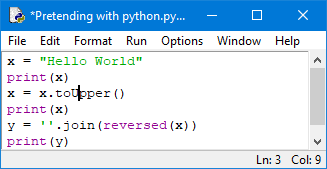
## Similar products and existing solutions

### 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)



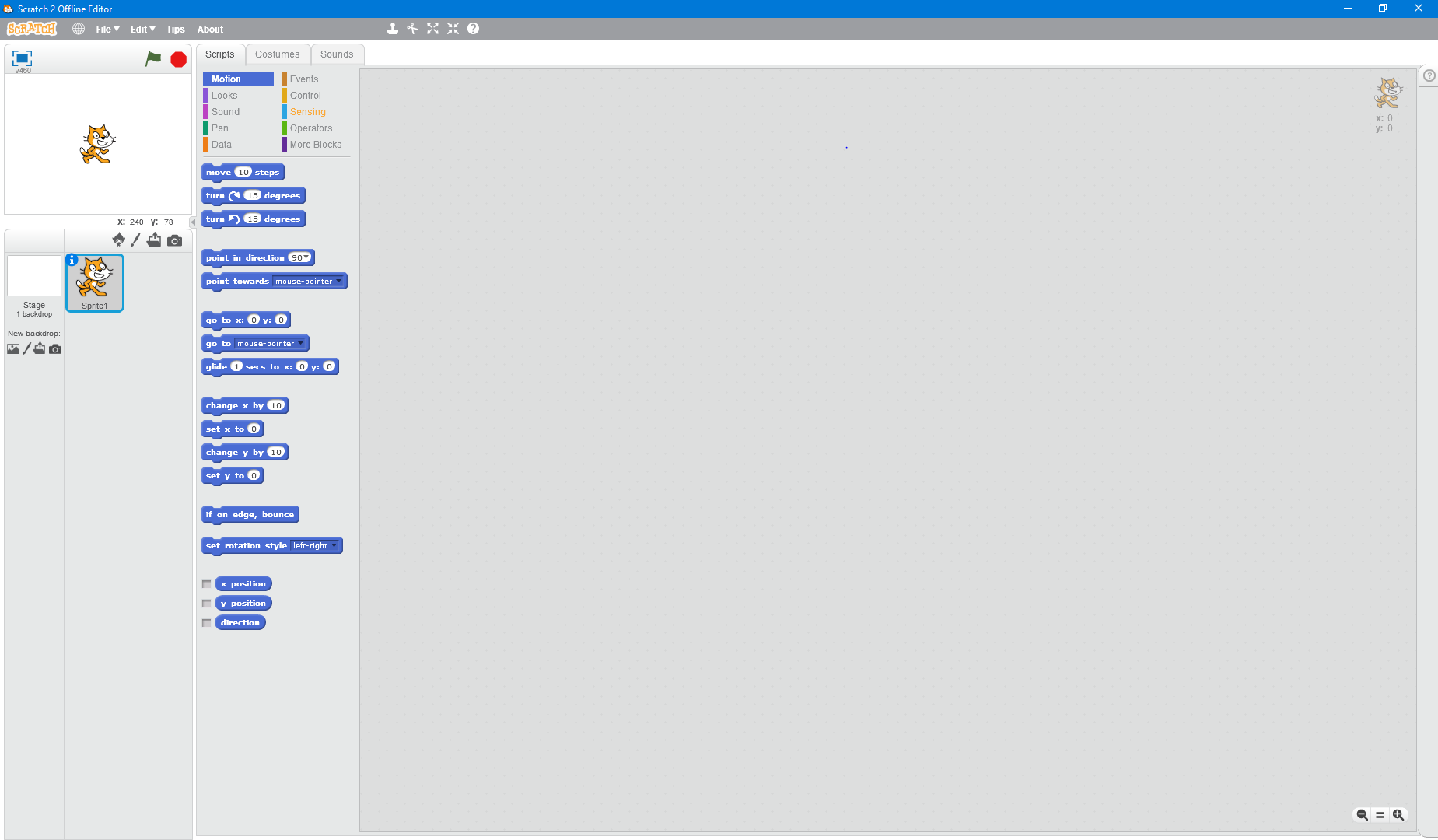
When an error occurs, the Shell tells you where it happened and what went wrong, but it is not a detailed description

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 by default |
| Lots of developer support available both on paper and internet |  |

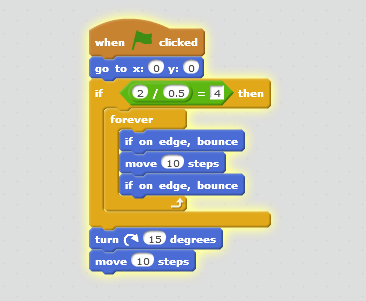
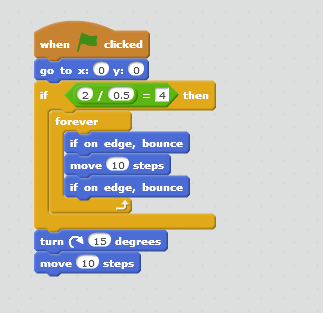
### Scratch (IDE)

Scratch is a visually orientated programming format. 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.

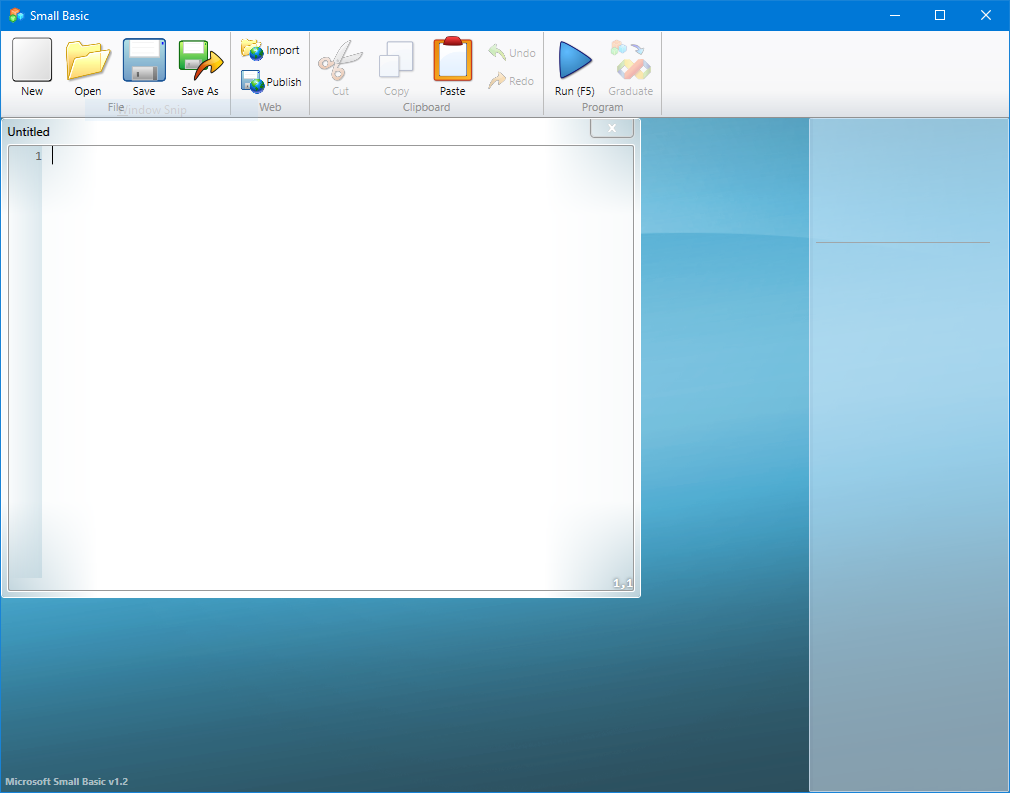


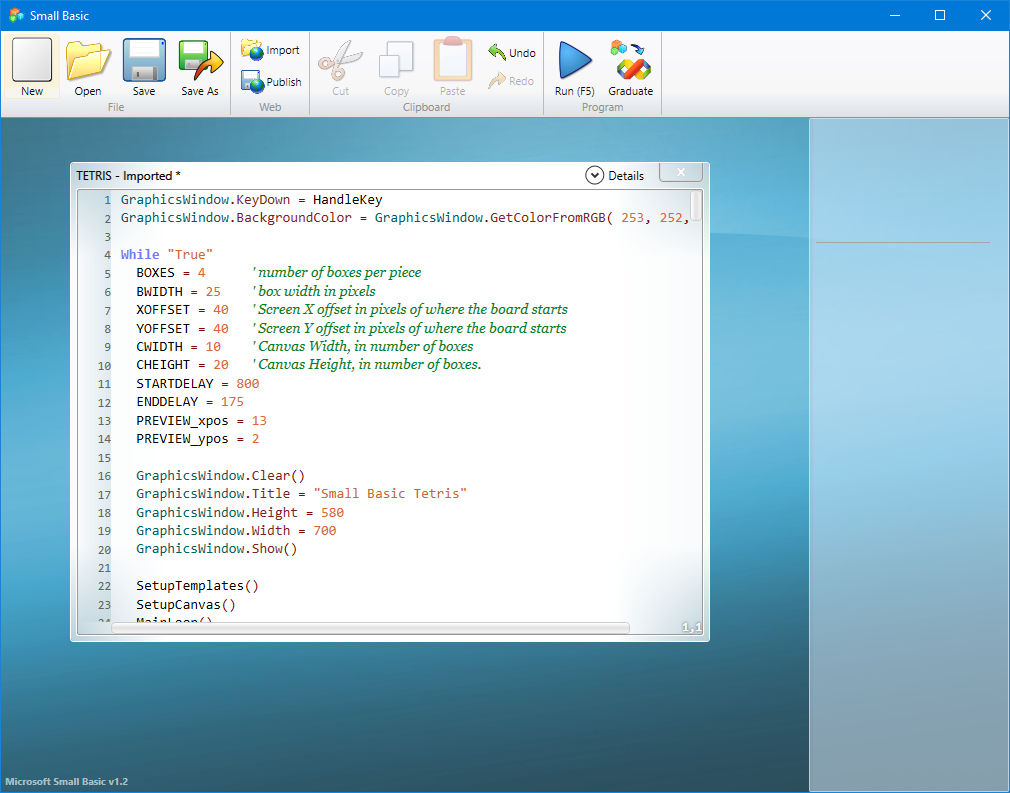
The editor highlights currently executing code.

|  |  |
| --- | --- |
| 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)

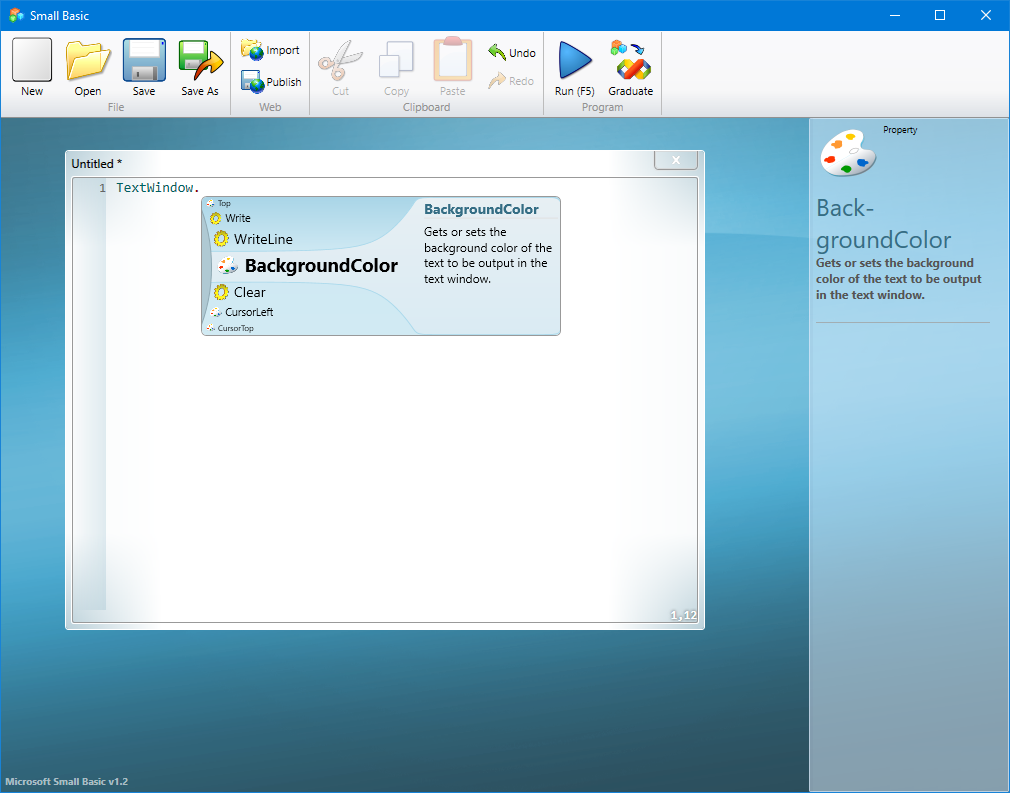
Small basic provides another IDE for young students similar to the Python IDLE, however the scripting language of choice is Small Basic itself. Small Basic provides a simplified version of the more advanced Visual Basic programming language developed by Microsoft. The IDE uses large, easy to understand 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.

|  |  |
| --- | --- |
| 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 |
|  |  |
|  |  |

### BBC Micro:Bit (IDE)

|  |  |
| --- | --- |
| Positive attributes | Negative attributes |
|  |  |
|  |  |
|  |  |

## Specifying a solution

namespace TestingBloks {

class EmptyNormalBlock : BaseBlock {

// yay

}

}

# 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 |
| BBC Micro:Bit Official Website | <http://www.microbit.org/> | 14/05/18 |
|  |  |  |