Abstract
This paper describes a web-based interactive tutorial that enables computer science tutors and lecturers to practice applying the Bloom Taxonomy in classifying programming exam questions. The structure, design and content of the tutorial are described in detail. The results of an evaluation with 10 participants highlight important problem areas in the application of Bloom to programming assessments. The key contributions are the content and design of this tutorial and the insights derived from its evaluation. These are important results in continued work on methods of measuring learning progression in programming fundamentals.

Keywords: programming, Bloom, maturity, competence, learning progression, assessment

1 Introduction
A typical university Computer Science Bachelor degree is 3 to 5 years long (in the case of combined/double degrees). The degree consists of a series of semester-long subjects. The design of the curriculum for a degree must enable students to steadily progress in acquiring discipline skills, as reflected in the ACM Computer Science curriculum (ACM CS 2008). So, for example the first programming subject in this sequence may assume no computing pre-knowledge; students typically learn basic programming fundamentals (e.g. variables, loops, control structures, functions, syntax). A later programming subject in the sequence, such as Data Structures, typically assumes students have basic competence in using these programming fundamentals. This Data Structures subject would enable students to learn new concepts (e.g. lists, maps, sets, sorting algorithms). At the same time, students should increase their level of competence on the programming fundamentals introduced in the first programming subject.

This example highlights the progressive nature of skill development and maturity in a Computer Science degree. Students are not expected to immediately master all new concepts in a subject. Rather, their competence level should increase as they progress from one subject to the next. In order to support this progressive learning model, learning activities, assessment tasks and exams need to be appropriately structured to teach and assess students at the correct level of competence. That is, a final exam in the first programming fundamentals subject should be designed to assess whether students have reached the competence level that is appropriate for that stage. This design should also account for the range of learning achievements across the class. Any student who earns a passing grade should have basic levels of competence. The top-performing students should be able to demonstrate a more advanced level of competence. The second subject should then have assessed learning activities and exams that require a more advanced level of competence of programming fundamentals and a more basic level of competence of the new concepts.

The issue that arises out of this discussion is how to classify teaching activities and assessments at a particular level of competence. That is, how can a lecturer write an exam question that assesses a programming fundamentals concept at a novice level vs. a more advanced level? Additionally, given an existing exam paper, how can a lecturer judge what level of competence is required to correctly answer a particular exam question?

Several theories exist that can be used in this classification, the most prominent of which are Bloom’s Taxonomy (Bloom, B.S. 1956), the SOLO Taxonomy (Sheard, J. et. al. 2008) and Neo-Piagetian development theory (Lister, R. 2011). In this paper we focus on Bloom, as it is currently used in ACM CS curriculum (ACM 2008) and will be used in the revised 2013 curriculum (IEEE/ACM 2013). Bloom's taxonomy is also commonly recommended by university teaching support services as a guide for learning outcome specification (e.g. Centre for Learning and Professional Development, University of Adelaide, 2011; Learning & Teaching Centre, Macquarie University, 2008; Centre for the Advancement of
Teaching and Learning, University of Western Australia, 2005). Bloom offers the potential for more principled design of the curriculum as it helps classify the level of learning to be achieved in each subject and the curriculum design can ensure that there is progression. Additionally, Bloom should help a lecturer design examination questions so that they assess learning at the appropriate level of competence.

To do this though, a lecturer requires an understanding of the Bloom Taxonomy, how it applies in a computer science context, and how it can be used to classify programming assessment tasks. Our motivation is to enable lecturers to gain this understanding of Bloom so that they can apply it to their own exam questions. This will enable computer science lecturers to review existing exam papers and design future exam papers with a greater awareness of progression and maturity levels, thus avoiding either overly-high or under-ambitious expectations.

To this end, our contribution is a computer science contextualized web-based tutorial on the Bloom Taxonomy with interactive examples, user self-explanation and self-reflection. The tutorial is a useful resource in training participants on the application of Bloom in classifying programming assessment questions. The results from the evaluation of this tutorial are useful in identifying where Bloom is used inconsistently due to different assumptions about the learner, different interpretations of the Bloom categories, or a misunderstanding of the categories. The tutorial and Bloom insights are important inputs to future work on measuring learner progression in computer science and future ACM CS revisions.

2 Background

Benjamin Bloom himself once said that the original Bloom Handbook (Bloom B.S., 1956) was “one of the most widely cited yet least read books in American education” (Anderson, W.L., 1994). The taxonomy is a behavioral classification system of educational objectives. The framework specifies 6 categories, namely, Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation. Knowledge is the simplest behavior, with each category thereafter being more sophisticated. That is, a Knowledge level objective or assessment task requires a student to simply recall information from memory. In contrast, a Synthesis level task requires students to apply what they have learnt to create new and unique works. A very brief description of the categories follows (adapted from Anderson, W.L., 1994):

- **Knowledge** – recalling of information
- **Comprehension** – interpreting, translating or reordering of concepts, applying a given abstraction
- **Application** – identifying an appropriate abstraction to solve a problem without being prompted
- **Analysis** – breaking down a problem or communication into parts and identifying the relationships between the parts
- **Synthesis** – Identifying and putting together abstractions to create a new and unique artifact or solution to a non-trivial problem
- **Evaluation** – Commenting on the validity of a work with respect to implicit or explicit criteria

The problem of first year subjects being overly-ambitious and requiring students to very quickly answer assessment questions requiring competence at the Bloom Synthesis or Evaluation levels is discussed by Lister, R. (2000). This is an unrealistic expectation if students have not had the chance to steadily progress through the first 4 levels of mastery. The consequence of this is that first year CS subjects specify learning objectives that require a high-level of competence, but consequently have lax assessment marking schemes that allow students to pass under false pretences with very little competence in the specified objectives (Lister, R. 2001). Lister argues that the sequencing of material should be based explicitly on a model of learning progression, and proposes Bloom’s Taxonomy (Bloom, 1956) as a potential model.

Further, Lister, R. (2001) suggests CS1 students should be expected to primarily operate at the first two Bloom levels (Knowledge and Comprehension). “CS1 cannot produce accomplished programmers. That is the task of an entire sequence of programming subjects.” Students should thus not be expected to operate at the higher Bloom levels (Synthesis/Evaluation) by writing original code in a first semester subject, yet this appears to be a common occurrence.

This trend towards overly ambitious first year subjects is also noted by Oliver, D. and colleagues (2004). Here the authors took 6 subjects from a single Australian IT degree, and invited 4 lecturers to categorize the assessment questions on the original Bloom Taxonomy scale. The authors used the weighting of each question and the Bloom categorization to calculate a Bloom Rating for each subject as a whole. The results show a first year, first semester programming subject has a weighted Bloom Rating of 3.9, i.e. somewhere between Application and Analysis. The first year, second semester subject has a rating of 4.5. Students in this stream are thus expected to rapidly skip over the opportunity to progress through the lower levels of the scale.

Oliver, D. and colleagues (2004) also highlight the inconsistencies in applying Bloom to computer science exam questions. For the 1 example question presented in the paper, the 4 participating lecturers came up with 4 distinct Bloom classifications, ranging from Knowledge to Analysis. Whalley, J. et. al. (2006) find the use of Bloom’s taxonomy for rating the cognitive complexity of programming MCQ’s “challenging even to an experienced group of programming educators.” The difficulty is attributed to either some deficiencies in Bloom, or “the authors current level of understanding of how to apply the taxonomy.”

The ACM Computer Science Curriculum (ACM CS 2008) supports the notion of gradual student progression, although it does not give direction as to how this progression should be implemented. The curriculum specifies a collection of learning objectives and topics, organized by knowledge area. The learning objectives are
based on the revised Bloom Taxonomy (Anderson et al, 2001). As an example, the Programming Fundamentals / Data Structures knowledge area specifies the following 9 learning objectives (the italicized verbs are indicative of the Bloom levels):

- Describe the representation of numeric and character data.
- Understand how precision and round-off can affect numeric calculations.
- Discuss the use of primitive data types and built-in data structures.
- Describe common applications for each data structure in the topic list.
- Implement the user-defined data structures in a high-level language.
- Compare alternative implementations of data structures with respect to performance. Write programs that use each of the following data structures: arrays, strings, linked lists, stacks, queues, and hash tables.
- Compare and contrast the costs and benefits of dynamic and static data structure implementations.
- Choose the appropriate data structure for modeling a given problem.

These objectives show a spread of competence levels ranging from Bloom Knowledge (describe) to Bloom Synthesis and Evaluation (write, implement, compare & contrast). The ACM CS 2013 curriculum is expected to continue along similar lines, but will likely use a simplified Bloom Taxonomy consisting of only 3 categories to identify the depth of understanding: Knowledge, Application and Evaluation (IEEE/ACM 2013).

Thompson and colleagues (2008) attempted to contextualize the revised Bloom Taxonomy to computer science. They ran an experiment where 5 participants were asked to analyze 6 first-year computer science final exam papers and categorize each question on the Bloom scale. The results showed significant disagreement between the rankings performed by different participants. This was attributed to some having implicit knowledge of how the subject was taught, and hence had a better understanding of the cognitive processes of the students undertaking the exam papers.

Thompson and colleagues (2008) however did not discuss the participants’ prior knowledge of the Bloom Taxonomy, or its application in a computer science context. It was only after the participants had a chance to collaborate and discuss each classification that they reached consensus on each exam question.

3 Programming Bloom Tutorial

3.1 Introduction

ProGoSs (Program Goal Progression) is a research project on curriculum mapping and measuring learning progression in university degree programs. It is an online web-based system that allows university educators to tightly link the teaching activities and assessments in each subject to important curriculum learning objectives, such as those specified in the ACM CS (2008). This enables educators to optimize the sequence of subjects, topics and assessments as to provide maximum curriculum coverage. It also aims to distinguish between the bare-pass student and the top-performing student in terms of the learning objectives covered and level of competence for each.

In this particular experiment, we are evaluating the use of the Bloom Taxonomy for classifying computer science assessment/exam questions.

The experiment requires participants to complete our interactive Bloom Taxonomy tutorial. The intention of this tutorial is to quickly bring participants up to speed on how to apply Bloom in introductory programming. Participants will read about the Bloom Taxonomy and practice some classification examples. Participants are asked for self-explanations and self-reflections on their understanding during the tutorial. The experiment takes less than 60 minutes to complete.

3.2 Pre-Survey

The tutorial commences with a short pre-survey containing 3 questions:

1. When is the last time you have completed this tutorial?
2. Based on your existing understanding of the Bloom Taxonomy, how confident are you in being able to classify programming exam questions at the correct level out of the 6 possible levels?
3. What is your highest level of exposure to Computer Science programming concepts and methods?

A screenshot of this is shown in Figure 1.

Figure 1: Pre-Survey

3.3 Bloom Introduction

The Bloom Taxonomy is a framework for classifying learning objectives into different categories of cognitive behaviors. It describes 6 categories as follows (from the most sophisticated to the least sophisticated):

- Knowledge
- Comprehension
- Application
- Analysis
- Synthesis
- Evaluation

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. Evaluation</td>
<td>Most Sophisticated</td>
</tr>
<tr>
<td>5. Synthesis</td>
<td></td>
</tr>
<tr>
<td>4. Analysis</td>
<td>Moderately Sophisticated</td>
</tr>
<tr>
<td>3. Application</td>
<td></td>
</tr>
<tr>
<td>2. Comprehension</td>
<td>Least Sophisticated</td>
</tr>
<tr>
<td>1. Knowledge</td>
<td></td>
</tr>
</tbody>
</table>
Each category builds upon the cognitive behaviours found in the preceding categories. The six levels form 3 pairs where the lower element of a pair focuses upon providing an artifact and the second element in the pair is used to demonstrate understanding of such an artifact. Students are typically expected to climb up through the Bloom Taxonomy as their mastery of a discipline deepens. That is, novices would start at the knowledge level, whereas seniors would be expected to perform at the higher levels.

### 3.4 Category Descriptions

Participants were required to click on each tab (as seen in Figure 3) to read the category description and self-rate their understanding for each category before proceeding to the next tab. A progress bar filled up as they completed each tab. A note informed participants that they were not expected to become an expert in Bloom just by reading these descriptions, but hopefully they should improve their understanding as they progressed through the tutorial.

#### 3.4.1 Knowledge

Knowledge emphasizes the recall of information. The recall situation is very similar to the original learning situation. The knowledge category differs from the others in that remembering is the major psychological process involved, while in the others the remembering is only one part of a much more complex process of relating, judging and reorganizing.

**Key Verbs:** know, define, memorize, repeat, recall, record, list, name, relate, review, tell.

**Example:** What is the return data type of the following function?

```java
public int getMin(int[] data) {
    int min = 0;
    for(int i = 0; i < data.length; i++) {
        if(data[i] < min) {
            min = data[i];
        }
    }
    return min;
}
```

**Explanation:** This is a knowledge level question as the student needs to recall what a return type is to answer correctly.

#### 3.4.2 Comprehension

Comprehension requires an understanding of the literal message contained in a communication. In reaching such understanding, the student may change the communication in his mind to some parallel form more meaningful to him. That is, the student may translate the communication into another language or other terms. The student may also interpret the individual parts of a communication and re-order the ideas into a structure more meaningful to him.

Comprehension differs from Application in that the thinking is based on what is explicitly given, rather than on some abstraction the student brings from other experiences to the situation.

**Key Verbs:** restate, discuss, describe, recognise, explain, express, identify, locate, report, operate, schedule, shop, sketch.

**Example:** What is the return value of the function below when called with the following input data array: {3,7,2,9,4}.

```java
public int getMin(int[] data) {
    int min = 0;
    for(int i = 0; i < data.length; i++) {
        if(data[i] < min) {
            min = data[i];
        }
    }
    return min;
}
```

**Explanation:** This is a comprehension level question as the student is required to understand and trace through the code to derive at the correct answer.

#### 3.4.3 Application

Application requires the student to apply an appropriate abstraction without having to be prompted as to which abstraction is correct or without having to be shown how to use it in that situation. In a comprehension problem the student would be specifically told which abstraction he should use.

**Key Verbs:** translate, interpret, apply, employ, use, demonstrate, dramatise, practice, illustrate, criticise, diagram, inspect, debate, invent, question, relate, solve, examine.

**Example:** Write a function to return the minimum value from an integer array that is passed as a parameter.

**Explanation:** This is an application level question as the student is required to write the code using an...
abstraction that’s not already there (e.g. a loop, an if statement, a local variable).

3.4.4 Analysis

Analysis emphasizes the breakdown of the material into its constituent parts and detection of the relationships of the parts and of the way they are organized to form the whole. In comprehension, the emphasis is on the grasp of the meaning and intent of the material. In application it is on remembering and bringing to bear upon given material the appropriate generalizations or principles.

Analysis is the ability to identify unstated and/or incorrect assumptions in code, to recognize software design patterns and best practices.

Key Verbs: distinguish, analyse, differentiate, appraise, calculate, experiment, test, compare, contrast, create, design, setup, organise, manage, prepare.

Example: The following function will always return a correct result. True or False? Please justify your answer.

```java
public int getMin(int[] data) {
    int min = data[0];
    for(int i = 1; i < data.length; i++) {
        if(data[i] < min) {
            min = data[i];
        }
    }
    return min;
}
```

Explanation: This is an analysis level question as the student is required to break down the code and understand the relationships and assumptions between each part. In this case, the student must realize that a 0 length array or a null array will both cause an exception to be thrown.

3.4.5 Synthesis

Synthesis is defined as putting together of elements and parts so as to form a whole, in such a way as to create a program or a program design not clearly there before. This category recognizes creative behavior and student responses are expected to have a degree of variation and uniqueness.

Comprehension, application and analysis also involve the putting together of elements and the construction of meanings, but these tend to be more partial and less complete than synthesis in the magnitude of the task. Also there is less emphasis upon uniqueness and originality in these other classes than in synthesis.

Key Verbs: compose, plan, propose, design, formulate, arrange, assemble, collect, construct, choose, assess, estimate, measure.

Example: Write a program that will read in an arithmetic expression from the console and print out the result. For example, given the input 3*8/4+(6-(4/2+1)), your program should output the answer 9 on a new line. The program should gracefully handle all exceptions.

Explanation: This is a synthesis level question as students could come up with many different correct implementations (e.g. using different tokenizer methods, recursive descent trees and other design patterns). The answers are expected to include a level of creativity.

3.4.6 Evaluation

Evaluation is defined as the making of judgments about the value of a program or program design. It involves the use of criteria as well as standards for appraising the extent to which the program or program designs are accurate, effective, economical, or satisfying. The judgments may be either quantitative or qualitative, and the criteria may be either those determined by the student or those which are given to him. Only those evaluations which are or can be made with specific criteria in mind are considered. Such evaluations are highly conscious; require adequate comprehension and analysis of the program or program design; and are primarily based on considerations of efficiency, economy, utility or specific means for particular ends.

Key Verbs: judge, appraise, evaluate, rate, compare, value, revise, score, select.

Example: The function below is required to return the most frequently occurring character from a given input stream. You are a senior developer asked to review the implementation of this function as coded by a junior staff member. What comments would you make in regards to performance, correctness, assumptions, style and quality of the overall solution?

```java
public char getMostFrequentChar(InputStream in) {
    //code implementation omitted
}
```

3.5 Interactive Examples

After reading the 6 category descriptions above, and self-rating their understanding of each, participants were then asked to classify some examples of examination questions. Participants had to provide answers, explanations and ratings on each of the 12 examples, such as the example seen in Figure 4. Participants are encouraged to scroll back to refer to the category definitions if needed. A progress bar fills up as they complete each example.

### Figure 4: Tutorial Example Question

For each example, the participants classified the exam question on the Bloom scale. Participants were then required to self-rate their confidence in their classification, as well as to justify answers and comment
on any uncertainties in their confidence. This was done in accordance with work by Chi M.T.H and colleagues (1994) showing that “Eliciting self-explanations improves understanding”.

The 12 example questions, and earlier category descriptions, were created by our Bloom expert - a computer science academic with an active research interest in the application of Bloom, SOLO and Neo-Piagetian frameworks to programming. Out of the 12 example questions, 3 were targeted as Knowledge, 2 as Comprehension, 2 as Application, 2 as Analysis, 1 as Synthesis and 2 as Evaluation. The uneven numbers were used so that participants would not be able to guess the last few by discerning the pattern and counting answers. The order in which the questions were presented was randomized, but always in the same sequence for all participants. The following is a listing of these 12 examples and classification explanations (note that participants had to attempt classifying each example first before being shown the correct answer and explanation).

3.5.1 Example 1 - Application

Question: Write a function that will return a boolean indicating if the given integer array is sorted in ascending order. Use the following header as a starting point.

```java
public boolean isSorted(int[] data) {
    ...
}
```

Explanation: Here the student is required to identify the necessary abstractions for completing this task, namely a loop to go over the array elements, an if statement to compare the values, and a local variable or return statement to terminate the loop and return the correct result.

3.5.2 Example 2 – Knowledge

Question: Circle the primitive data types in the following code snippet.

```java
public boolean isSorted(int[] data) {
    boolean sorted = true;
    for(int i = 0; i < data.length; i++) {
        if(data[i] > data[i+1]) {
            sorted = false;
        }
    }
    return sorted;
}
```

Explanation: Here the student is required to simply recall the primitive data types as previously studied. This can be answered fully via memorization alone, without any need for understanding what a primitive data type is, or how it differs from other data types, or the differences between each of the 8 data types. That is, if the student knows an ’int’ is a primitive data type, he can circle it in the code.

3.5.3 Example 3 – Evaluation

Question: A video rental store has implemented an online system where customers can login, browse through movies, select available movies, and rent movies online using a credit card. The video store has recently become concerned about security due to high profile events in the media. They have hired you as a security consultant to analyze their authentication and payment code and identify any vulnerabilities. Go through the code below and comment on its security. If not secure, why, and how should it be re-written to make it more secure?

```java
String[] keys = {
    String[] keys = (String[] keys, keyVal->keyVal.getKey()}
    count = (KeyValuePair.get(keyVal));
    count = (KeyValuePair.get(keyVal));
    System.out.println(Arrays.toString(keys));
}
```

Explanation: Here the student must first comprehend and analyze the code by breaking it down into individual parts, then evaluate each part against security best practices. The student may identify buffer/integer overflows, SQL injection attacks, storing of plain-text passwords or weak encryption mechanisms, etc. The student can make a number of recommendations on how to fix these. Note that even though the question uses the word ‘analyze’, this is actually an evaluation level task.

3.5.4 Example 4 – Analysis

Question: The following function takes an array of strings as inputs, and prints out each string and the number of times it appears in the array, in descending order. The code however throws a runtime exception when executed. Explain why. How would you fix it?

```java
public static void printCounts(String[] items) {
    Map counts = new HashMap();
    for(int i = 0; i < items.length; i++) {
        int count = (KeyValuePair.get(keyVal));
        count = (KeyValuePair.get(keyVal));
        System.out.println(Arrays.toString(keys));
        System.out.println(Arrays.toString(keys));
    }
}
```

Explanation: Here the student is required to comprehend and analyze the code by breaking it down into individual parts, then evaluate each part against security best practices. The student may identify buffer/integer overflows, SQL injection attacks, storing of plain-text passwords or weak encryption mechanisms, etc. The student can make a number of recommendations on how to fix these. Note that even though the question uses the word ‘analyze’, this is actually an evaluation level task.

3.5.5 Example 5 – Comprehension

Question: What is the output of the following code?

```java
int[][] data = {{-3, 6, 9, 2, 4, 15, -7, 0}};
int result = data[0];
for(int i = 0; i < data.length; i++) {
    if(data[i] > result) {
        result = data[i];
    }
}
System.out.println(result);
```

Explanation: Here the student is required to read the code, interpret the individual parts, understand what it does, and trace the execution to derive the right answer. To do this the student requires knowledge of programming syntax, control structures and variable scope as a pre-requisite.
3.5.6 Example 6 – Comprehension
Question: What is the output of the following code snippet?
```java
int a = 3;
int b = 7;
int c = 0;
int[] data = {1,5,5,2,3,9,6,3,1,5,8,3,3};
for(int i = 0; i < data.length; i++) {
    if(data[i] > a || data[i] < b) {
        c++;
    }
}
System.out.println(c);
```
Explanation: Here the student is required to read the code, interpret the individual parts, understand what it does, and trace the execution to derive the right answer. To do this the student requires knowledge of programming syntax, control structures and variable scope as a pre-requisite.

3.5.7 Example 7 – Knowledge
Question: The javac.exe command is used to compile java code. True or False?
Explanation: Here the student is required to simply recall the function of the javac.exe command, or which command is used to compile java code. This can be done by rote memorization without any further understanding of the compilation process or other java internals.

3.5.8 Example 8 – Synthesis
Question: A video store has a list of movies in a CSV file with the following header: "movie title", "year released", "genre", "main actor/s", "rating (1-5)". The main actor/s field can contain a single name, or multiple names separated by a comma. An example line is:

"The Social Network", "2010", "Jesse Eisenberg, Andrew Garfield, Justin Timberlake", "4.5"

The store has hired you to write a command-line program that will return the top 3 most popular actors in each genre (i.e. highest average ratings of all movies they appeared in in that genre). Assume the path to the CSV file is passed in as the first command-line argument.

Explanation: Here the student must first comprehend the scenario, then apply the correct programming abstractions to parse the CSV and process the data to derive the correct answer. There are multiple ways of implementing this correctly, and the task description leaves students open to use some creativity in coming up with unique solutions.

3.5.9 Example 9 – Knowledge
Question: Write a SortedHashMap implementation. As discussed in lectures and practised in tutorials, the sorted map should expose two iterators: one that loops through all key/value pairs sorted in ascending key order, the other that loops through all key/value pairs sorted in ascending value order. The Map should work with any object that implements Comparable.

Explanation: This would be a non-trivial problem which could be solved in many different ways if the student had never come across a SortedHashMap before. However, the question states "as discussed in lectures and tutorials", which implies the student has had sufficient practice at this exercise. So even though this could be a complex problem with unique solutions (i.e. Synthesis), since the students have had significant prior practice at this exact problem, it is actually a Knowledge question as it can be completed via rote memorization.

3.5.10 Example 10 – Analysis
Question: An employee at your company writes the following function, which takes a java InputStream as a parameter and returns the average word-count of sentences. The function sometimes returns incorrect results however. Why? How would you fix it?
```java
public double avgSentenceWordCount(java InputStream in)
{"code omitted from tutorial..."
}
```
Explanation: Here the student must first comprehend the question and intended behavior, then break down the function into logical parts, and identify if each part would operate as expected. This is not evaluation however, as the student is not asked to comment on the algorithm implemented, but rather to break down the algorithm to find the incorrect assumption that leads to the bug.

3.5.11 Example 11 – Application
Question: Fill in the missing code in the following function that calculates and returns n! (factorial).
```java
public long factorial(int n) {
    long result = 1;
    ...
    return result;
}
```
Explanation: Here the student is required to identify that the use of a loop is needed to compute the right answer, without being hinted of this. This is assuming that the student has not rote memorized the code for n!, in which case this would be a knowledge question.

3.5.12 Example 12 – Evaluation
Question: You are a senior developer in a company that creates iPhone games. A junior developer is tasked with creating a love score calculator which takes two names as parameters and returns a compatibility rating score based on the following rules [omitted]. The junior developer submits the following code as a solution. You are tasked with reviewing the code for quality, correctness, efficiency and style. What comments would you make and why?

Explanation: Here the student is required to comprehend and analyze the code by breaking it down into individual parts, then evaluate each part against a series of metrics. The student may identify a number of potential bugs (divide by zero, integer overflow), identify unspecified assumptions (treatment of space and other special characters, treatment of repeating characters), suggest graceful error handling, suggest better named variable names, etc.
3.6 Post-Survey

Upon completing all 12 example classifications, participants were asked to complete a 2-question post survey:

1. Based on your new understanding of the Bloom Taxonomy, how confident are you in being able to correctly classify programming exam questions at the correct level (out of the 6 possible levels)?
2. Did you find this tutorial useful and effective in increasing your understanding of the Bloom Taxonomy? Why or not?

4 Results

A total of 10 participants completed our interactive tutorial and example classifications. These consisted mostly of computer science tutors and 1 computer science professor. The average participant Initial Confidence (measured in the pre-survey) score was 30.8% (sd. 23.36) (only 2 participants self-rated at 70%, and one at 40%, while the rest self-rated at 30% or lower). Prediction Confidence (measured during the initial reading of the category descriptions) ranged between 67 and 71%. The results after participants had classified all of the 12 examples are summarized in Figure 5.

![Figure 5: Prediction Confidence, On-Task Confidence and Accuracy](image)

This chart shows the 6 Bloom categories along the horizontal axis. Each category is sub-divided into 3 columns. These are, from left to right, Prediction Confidence, On-Task Confidence and On-Task Accuracy. The Prediction Confidence and On-Task Confidence were discussed in the previous section. The On-Task Accuracy is the percentage of participants who agreed with our expert classification for each question. For the purposes of evaluating the tutorial system, we treated the expert’s classifications as being correct.

4.1 Participant Responses to the 12 Examples

The following is a listing of the participant classifications and comments for each of the 12 examples.

Example 1: Nine participants correctly classified this as Application. One participant however argued that this was a Synthesis task as “creating a solution requires putting lots of ideas together”. This participant expressed a 95% confidence in this answer, and disagreed with our explanation as to why our expert classified it as Application. The participant wrote “I think the rating as application would only apply if the student had been taught some recipes for problems like this, and had a good reason to think that the question would fit a recipe.”

Example 2: Nine participants correctly classified this as Knowledge. One participant classified it as Comprehension as “the student is required to understand and trace through the code to circle the primitive data types”, but subsequently agreed with our expert’s explanation.

Example 3: Eight participants correctly classified this as Evaluation. Two participants labelled it as Analysis as “It requires the student to look through the code snippet and comment on what security flaws it has”. These two participants did not specify if they subsequently agreed with our expert classification.

Example 4: Eight participants correctly classified this as Analysis, but two picked Comprehension. One of these 2 wrote “It could be in analysis, but it asks how it would be fixed which wouldn’t suggest it’s an analysis”. It seems this participant took a strict definition of the term analysis and disregarded this as a correct classification since the student had to fix the code. Both subsequently agreed with our expert classification.

Example 5: Nine participants correctly classified this as Comprehension. The other marked it as Knowledge, stating “This code seems to be absolutely standard, so the student should have seen it often, and know what it does. (find the max) They then just have to look at the input (I assume that knowing the max of numbers is trivial at this level)”. This participant assigned a 50% confidence score to this answer, and justified this as “If the student actually traces the code, that would be application in my view.” After being shown our expert’s classification and explanation, the participant commented “As noted, I think this code would be very familiar, and not need to be broken down and treated as parts.”

Example 6: Seven participants correctly classified this as Comprehension. Two marked it as Application and 1 as Analysis. One participant justified the choice of Application with “As noted, I see tracing as requiring more than knowledge; it requires complex abstractions of the machine model, and using them on the code that is given.” Another participant commented that “it could be comprehension” and the other participant believed that tracing had to be Application.

Example 7: Seven participants correctly classified this as Analysis. One participant classified it as Analysis and one as Application. The comments left by these two did not adequately describe their reasoning. The last participant labelled it as Comprehension, stating “the student is required to understand the how to compile java program”. This suggests a misreading of the question or incorrect assumption about the knowledge required to answer the question.

Example 8: Six participants correctly classified this as Synthesis. One picked Analysis and stated that “Could be anything from Application up to Synthesis but I’m guessing somewhere in between”. The other three participants all marked it as Application. One stated that “I assume that the student knows well the available commands, and just has to adjust the flags and then string them together in the right order ... if actual problem
solving was needed, i would rate it as synthesis.” Another participant stated “implementation questions most likely are application ... not an analysis, nor evaluation as it doesn’t ask for the students opinion”.

Example 9: No participants correctly classified this as Knowledge. Instead, 4 picked Synthesis, 5 Application and 1 Analysis. From inspecting the comments, the two main reasons for this were either that the participants “missed the clue that the student had seen this exact problem” or assumed that “this question is hard for students who are in knowledge level”. This second comment implies that a student in the Knowledge level cannot rote learn complex solutions. Five participants agreed with the expert classification after being shown the justification. The rest either disagreed on the basis mentioned above or did not elaborate further.

Example 10: Seven participants correctly classified this as Analysis. Three however marked it as Evaluation. One stated “I am evaluating another colleague’s code, which requires judgements in order to apply fixes to it”. Another commented “It could be just analysis, but it’s also working out what might be wrong with the code and suggesting alternatives.” The third incorrect participant stated “too much abstraction”.

Example 11: Five participants correctly classified this as Application. One marked it as Knowledge and stated “If the student hasn't memorized this, I would see it as synthesis, unless they have a pattern that they know would be used to solve the problem. Only if they know which pattern to use is application the suitable level.” One participant labelled it as Analysis with 100% certainty, stating that “It requires that I break the problem down”. Two participants labelled it as Synthesis because “students could come up with many different correct implementations”. One participant marked it as Comprehension because “the student needs to comprehend the code before answering”.

Example 12: All 12 participants correctly classified this as Evaluation.

4.2 Final Confidence and Participant Feedback

After completing the 12 examples, the average participant Final Confidence score was 75% (sd. 11.55), an increase from the 30.8% Initial Confidence before starting the tutorial. All participants responded positively in the final feedback question. The common trend in these comments was that the category descriptions were good for gaining a basic grasp, but the interactive examples with justified answers were very useful in consolidating their understanding.

5 Discussion

Most of our participants had very little exposure to Bloom prior to taking the tutorial, hence the low average Initial Confidence of 30.8%. After completing our tutorial however, the Final Confidence average increased to 75%. Participants however still did not feel entirely confident in being able to do this consistently. An analysis of the self-reflection ratings is presented and discussed in greater detail in "removed citation to protect anonymity, will insert in camera-ready version if accepted".

Overall the tutorial was successful in training participants on how to apply Bloom to programming questions. Participants that had little confidence prior to the tutorial, came out with a much higher understanding of Bloom. However, the evaluation confirms ambiguity in the interpretation of the Bloom categories due to their dependence on the user’s familiarity with the learning context. Example 9 demonstrated this where a potential Synthesis exercise could be solved via rote memorization (i.e. Knowledge) in some cases, or via Application in other cases. The classification of such questions very heavily depends on the assumptions made by those doing the classification. Without quizzing the student on how a question was answered, multiple correct classifications are thus possible. Evidence showed that some participants also confused the literal definition of the Bloom category labels with the classification of some questions. More example exercises for each category type may have been useful to indicate whether these participants learnt to apply the categories with greater consistency.

6 Conclusion and Future Work

The aim of this paper was to explore ways to measure learning progression in the programming fundamentals sequence of subjects. The Bloom Taxonomy is a framework for specifying the sophistication of learning objectives, but it is often misunderstood. We thus created an interactive tutorial to train computer science educators on how to apply Bloom in classifying programming questions and evaluated the results.

The evaluation showed that the tutorial was highly effective in developing participants’ confidence in identifying the level of performance involved in programming exam questions. The evaluation also confirmed previously documented ambiguities in the application of Bloom to cases where knowledge about the learning context is required for accurate classification. Participant feedback comments at the same time revealed other reasons as to why different people make different classifications, namely due to pre-conceived misunderstandings of the categories, or different interpretations about the complexity of tasks and sophistication required to solve them.

The results suggest further work is required in methods to measure learning progression in programming. The experiment can be repeated with a greater participant pool and greater number of examples. This would give a clearer insight into the amount of classification inconsistencies arising from assumptions about the learning context, as opposed to a misunderstanding of Bloom. Additionally, the experiment can be repeated with a different cognitive development framework, e.g. Neod-Piagetian Theory, to see how this compares to Bloom in terms of classification consistency. What is also becoming apparent is that a seemingly Synthesis level task for one programmer, may be more like an Application or Knowledge level task for another. That is, writing a function to, say, return the minimum value in an array, may require a lot of thought and problem solving ability for a novice who derives a solution from first principles. For an experienced programmer however, the solution
may be strikingly obvious and require very little thought. A method to measure this hypothesis could be a useful tool in assessing the competence level of a programmer.

Our work was motivated by the wide use of Bloom in defining curriculum, particularly in the case of programming fundamentals (ACM CS, 2008). This suggests that people who teach subjects in these areas may be aided in the design of their own subjects and assessments if they can make use of Bloom. This paper makes a contribution in reporting the design of our online Bloom tutorial. A second key contribution is its evaluation which indicates that it is quite effective in enabling a teacher to quickly enhance their understanding of Bloom. We believe that our ProGoSs system's Bloom tutorial is the first such system that helps teachers of programming fundamentals have greater understanding of Bloom, as a foundation for more systematic design of teaching and learning materials and assessment of how well student learning meets the intended goals.

Acknowledgements
Will insert in camera-ready version if paper is accepted.

7 References


http://www.mq.edu.au/ltc/about_lt/assess_docs/writing_ppt


http://www.mq.edu.au/ltc/about_lt/assess_docs/writing_ppt