INTERIM REPORT

AI Tutor for Programming Class (The Auto Grading Engine)

Li Angus Dicky  3035181920
Mak Yim Yung  3015124960
Siu King Fung  3035123083

Supervisor: Dr. Chui C. K.
COMP4801
Abstract

With the introduction of STEM education worldwide, the demand in learning programming has increased. This project aims to develop an AI supported auto-grading engine benefiting both teachers and students. The learning language selected is Python. Through data mining and analysis on programs, the AI tutor will give recommendations to students throughout the learning process. Teachers are also able to view analytic results to aid their teaching strategy. Ultimately, with a collaborative characteristic, the AI tutor is going to be more mature as more have used it.
Contents

Abstract ........................................................................................................................................... 1
List of Figures .................................................................................................................................... 4
1. Introduction .................................................................................................................................. 5
  1.1 Background ............................................................................................................................ 5
  1.2 Objectives ................................................................................................................................ 7
  1.3 Report Outline ....................................................................................................................... 7
2. Methodology ................................................................................................................................ 8
  2.1 The learning language—Python ............................................................................................. 8
  2.2 Analysis Tool—R ..................................................................................................................... 8
  2.3 Engine Design ........................................................................................................................ 9
    2.3.1 Checker ............................................................................................................................ 10
    2.3.2 Analysis ............................................................................................................................ 11
      2.3.2.1 Analysis Objectives ................................................................................................. 11
      2.3.2.2 Data collection ......................................................................................................... 11
      2.3.2.3 Data analysis .............................................................................................................. 11
      2.3.2.4 Estimation .................................................................................................................. 12
    2.3.3 Recommendation ............................................................................................................... 13
  2.4 Database Design .................................................................................................................... 14
  2.5 Interface Design .................................................................................................................... 15
3. Progress ...................................................................................................................................... 16
  3.1 Current Status ....................................................................................................................... 16
    3.1.1 Checker ............................................................................................................................ 16
      3.1.1.1 Input .......................................................................................................................... 16
      3.1.1.2 Observation list ........................................................................................................ 19
      3.1.1.3 Checking method ...................................................................................................... 20
    3.1.2 Analysis ............................................................................................................................ 23
    3.1.3 Recommendation ............................................................................................................. 28
    3.1.4 Database Design .............................................................................................................. 31
    3.1.5 Interface Design .............................................................................................................. 32
3.2 Work Remaining .................................................................................................................. 36

4. Limitations and Difficulties Encountered ........................................................................ 37
   4.1 Checker Limitations ........................................................................................................ 37
   4.2 Analysis Limitations ....................................................................................................... 37
   4.3 Interface Limitations ..................................................................................................... 37

5. Conclusion .......................................................................................................................... 38

References ................................................................................................................................ 39

Appendix .................................................................................................................................. 40
   Observation List Examples .................................................................................................... 40
List of Figures

Figure 1: Example of An Auto-grader ................................................................. 5
Figure 2: Extracted Chart From the HKEB report ............................................... 6
Figure 3: Flow Chart of the System ................................................................. 9
Figure 4: ER-Diagram of the Database ........................................................... 14
Figure 5: Example of JSON AST .................................................................. 17
Figure 6: Reading of JSON AST .................................................................. 18
Figure 7: Observation List ............................................................................. 19
Figure 8: Example of a nested for loop .......................................................... 21
Figure 9: Multi-syntax Example .................................................................. 22
Figure 10: Input Data ................................................................................ 23
Figure 11: Similarity Table .......................................................................... 23
Figure 12: Cluster History ........................................................................... 24
Figure 13: Distance Plot ................................................................................ 24
Figure 14: Dendrogram Illustrating the Cluster of Programs ......................... 25
Figure 15: Sample Result of Ward’s Method .................................................. 25
Figure 16: Sample Result of K-means method ............................................... 26
Figure 17: Estimation Table ........................................................................ 26
Figure 18: Estimation – Similarity Table ......................................................... 26
Figure 19: Estimation – Correlation Table ....................................................... 27
Figure 20: Estimation – Comparison ............................................................... 27
Figure 21: Pre-compiled data of Question 2 .................................................... 28
Figure 22: Result of Estimation .................................................................... 29
Figure 23: Observation List of Q2 Approach 2 ............................................... 30
Figure 24: Database Content in Cloud9 Server ............................................... 31
Figure 25: Cloud9 Workspace ...................................................................... 32
Figure 26: Login Page .................................................................................. 33
Figure 27: Front Page ................................................................................... 33
Figure 28: Question Page ............................................................................. 34
Figure 29: Answer Area and Result Area ....................................................... 35
1. Introduction

1.1 Background

With the introduction of STEM education introduced worldwide, learning programming has become an important component in contemporary education. According to a report from Education Bureau in 2016, learning programming is going to be a compulsory section under Technology Education, a S1-3 compulsory curriculum. In addition, the learning hours of the programming section under Information and Communication Technology (ICT) will gain a substantial increase.

Most programming learners are using a learn-by-doing approach. Therefore, the usage of auto-graders, combined with questions and test cases is essential. Nevertheless, most auto-graders do not provide extra information other than passes or fails.

![Example of an Auto-grader](image)

**Figure 1: Example of an Auto-grader**

On the other hand, an increasing supply of teaching power is required to cope with the demand of programming learners. In secondary schools, computer classes are often taught by Science or Mathematics teachers who may not have a firm Computer Science background. With reference to the same report mentioned above, only about 60% of respondents from schools “agree” or “strongly agree” that the computer teachers are capable of teaching programming. (Hong Kong Education Bureau, 2016)
To aid students and teachers, the AI tutor to be developed will provide recommendations to students on improvements and to teachers on analysis.
1.2 Objectives

The AI tutor is developed based on 3 major objectives.

Firstly, the grading of programs is automated online with a cloud system. The program is then stored in the database. By using the Internet, grading can be done without time or geographical constrains. This also provides a more flexible learning environment for both students and teachers.

Secondly, the system subsequently provides specified feedbacks to the students. Based on the guidelines, students may improve their programs without side-by-side help from a physical teacher.

Thirdly, when all students have completed an assignment, generated reports including performances of classes may be used in aiding teachers in modifying teaching strategies.

1.3 Report Outline

This report is going to discuss the methodologies in developing the AI tutor alongside the justifications. Next, it is going to report on the progress and the work to be done. Finally, it is going to conclude with the overall progress and future planning.
2. Methodology

In this chapter, it discusses Python and R as the learning language and analysis tool respectively. Next it introduces engine design, database design and interface design.

2.1 The learning language—Python

Python has been chosen as the learning language that the system provides. First of all, it is a widely used language. For instance, companies including Google, Dropbox and NASA use it on operating system and software development. Out of all widely used language, Python is chosen based on one major advantage—easier syntax for beginners. It is known that Python does not follow the C-style syntax, using semicolon (‘;’) and brackets (‘{}’) as C++ or Java. Combining with the compulsory indentation required by Python. Python is relatively easier to be written or read by beginners.

In addition, learning programming can be divided into two parts - learning logic and learning syntax. Since logic errors are more difficult to trace compared to syntax errors, we would like the system to help beginners to focus on the logic part of programming. As a result, new learners may spend less time in resolving syntactic confusion. Moreover, Python provides library that can be used to conveniently analyze Python programs. This is going to be further discussed below.

2.2 Analysis Tool — R

R is selected as AI Tutor analysis tool. It is the outstanding tool for statistics, data analysis and machine learning. The graphical capabilities are also excellent. Until now, more than 10,000 packages are available for download. Additionally, it is free, open source software and cross-platform and thus allows anyone to use and modify.
2.3 Engine Design

**Figure 3: Flow Chart of the System**

The system is divided into 3 major parts—a checker to determine features of programs, an analyzer to do analysis between programs, an AI tutor to give out recommendations based on the conclusion of analysis.
2.3.1 Checker

The checker is going to do ordinary auto-grading at first. Then, it proceeds to check and extract information from the source code.

To extract useful information from pure strings of source codes, a compiler approach is used. A simplified flow of a compiler is the following: source code, abstract syntax tree, control flow graph, object code.

The source code is the pure lines of codes written by humans. Abstract syntax tree (AST) is the tree form of the structure after the compiler parses the code. An example is that the parser will identify keywords such as “if” and “while”. Control flow graph (CFG) is a graph with directions in which the parsed instructions are expressed in a flow chart. In other words, the logic flow can be determined in the CFG of the program. Finally, object code is the code understood by the machine.

Although Python is not a compiled language but an interpreted language which uses an interpreter instead of a compiler, both constructs an AST throughout the process. With the “ast” library in Python, ASTs may be extracted out of the programs of students. Next, the logic flows in the programs may be determined in the light of compilers converting AST to CFG. Differing from compilers, an entire CFG is not going to be converted. Only partial key logic flows used by students in which we name them as “Observations” are going to be determined. For instance, it may deduce a student have used nested-looping if certain criteria are matched.

After gathering the observations, the program alongside with the information will be stored into the database for analysis.
2.3.2 Analysis

2.3.2.1 Analysis Objectives

The analysis’ s objectives are the following:

I. To find out the numbers of approaches for solving questions
II. To find out the characteristics of each approach
III. To estimate which approach is used after a new student has submitted a program
IV. To find out common mistakes in solving questions
V. To find out common breakthrough points when solving questions

2.3.2.2 Data collection

At the beginning, an assignment of three to four questions is designed. The questions cover the common and basic concepts of the programming language such as variables declaration and initialization, function definition, function call and data type. Next, about 100 students are invited to do the assignment. Finally, each student has to submit all compilable versions of programs. The reason is that students may have submitted more than one version before they could pass all test cases.

2.3.2.3 Data analysis

After collecting the data, the programs which passed all test cases will be selected for investigating the numbers of approaches for solving questions and the characteristics of each approach by using cluster analysis. It first measures the similarity by calculating squared Euclidean distance. It next applies Ward’s method which is a hierarchical cluster method based on observation value (OID). The number of clusters is determined by using elbow method\(^1\). Line graph will be plotted based on cluster history. It then will be further evaluated by using K-means method, which is a non-hierarchical cluster method, to check whether the result is consistent. It finally saves summary statistics of each cluster in the database.

---

\(^1\) It is to run on the dataset for a range of values of k and for each value of k calculate the sum of squared errors (SSE). A line graph of the SSE for each value of k is plotted. If the line looks like an arm, then the “elbow” on the arm is the value of k that is the best.
2.3.2.4 Estimation

It first extracts the summary statistics of each cluster from the database and a new student’s observation list generated by the checker. It measures similarity by calculating the squared Euclidean distance and calculating the correlation between the approaches and the new program. The highest positive value of correlation coefficient determines the new student has a higher possibility to use a certain approach. Finally, recommendations are given by comparing the summary statistics of that approach and the new student’s program.
2.3.3 Recommendation

The recommendation is a part that combines the result of checker and data analysis which is how the final product works. A website platform is built to combine these elements in order to perform the programming class with AI tutor.

The following is a scenario of the whole recommendation idea:

I. A new student is working on a question using the platform and submits his code.

II. After he submits, the website downloads the test cases of the question from the database and checks the correctness of his answer.

III. At the same time, the website runs the checker on the code to generate the observation list and stores it in the database.

IV. If his answer cannot pass all the test cases, that deduces that his answer needs improvements.

V. The precompiled data from data analysis part is downloaded from the database. Then, the website runs the script of data estimation to identify which approach the student is using.

VI. After that, the website compares the observation list of the student's answer and the observation lists of the same approach and gives the recommendation to the student.

VII. Suppose in this case, the student is a approach which is identified by the data analysis part. The website may find that his code is missing three observations.

VIII. The AI tutor will pop out and say that: “You may try to add more elements to your code such as If-else statement, recursion and print function to improve your code since most of the students who use the same approach with you also have these elements.”

Besides the idea mentioned above, there are other recommendation ways such as common mistakes and breakthroughs.

For common mistakes, as all versions of the answer will be collected and stored in database, the AI tutor can apply data analysis model to find out some common mistakes while student is coding. After that, the AI tutor can give some recommendations to remind the student in order to prevent those common mistakes.

For breakthroughs, the AI tutor can analyze the all versions of the answer in the same question to find out what is the critical changes in observation list between the fails and passes of some test cases. After that, the AI tutor can give some hints when the student cannot pass some test cases.
2.4 Database Design

phpMyAdmin is used for the database in the project since it can be used to manage the database easily when comparing to another database like MongoDB.

Two databases are built on different servers which is the HKU server and the Cloud9 server. The reason for one more database needed to be built in Cloud9 server is that it is easier to connect to the website. After developing the whole website in Cloud9, the website can be extracted to the HKU server by just changing the configuration of the database.

The following diagram is the structure of the database:

FIGURE 4: ER-DIAGRAM OF THE DATABASE
2.5 Interface Design

The AI Tutor is built in a website format for easier access by the users. If it is built as software, the users might need to download it before using the programming class. The website is built using Cloud9 because it is free and convenient for developing own website. Cloud9 also supports a lot of plugins such as python used for the checker program.

The interface has a login system for users. After login, users may choose the question they want to do. After choosing questions, users may see the question page which contains the question content, sample input and sample output.

Under the question page, there will be a text area for students to input their answer and submit. The steps what the website is doing will be displayed. Finally, the test cases result is going to be provided.

For recommendation interface, there will be an image of the AI tutor to give suggestions to the student. After the student has submitted the answer, the AI tutor will pop up on the right side of the page and give suggestions according to the result of data analysis such as improvement tips, common mistake and some breakthrough reminders.
3. Progress

3.1 Current Status

3.1.1 Checker

The checker is a function that takes the string of code as input and returns an observation list as output. In this section, input, observation list and the checking method of the checker are going to be discussed. The checker is also written in Python so as to utilize the Python library functions.

3.1.1.1 Input

By using a Python program named “astexport” by Federico Poli retrieved on GitHub, it is available to store the ASTs of the programs in a JSON format. Moreover, with the help of the “json” library in Python, transferring the JSON-type AST into a Dictionary type variable in Python to do observation checking can be further done. In addition, since “astexport” utilizes the “ast” library in Python, the extracted AST is a lossless representation of the code.
The following is an example of the AST of the simple program:

```python
print ("hello world")
```

```
{
    "ast_type": "Module",
    "body": [
        {
            "ast_type": "Expr",
            "col_offset": 0,
            "lineno": 1,
            "value": {
                "args": [
                    {
                        "ast_type": "Str",
                        "col_offset": 7,
                        "lineno": 1,
                        "s": "hello world"
                    }
                ],
                "ast_type": "Call",
                "col_offset": 0,
                "func": {
                    "ast_type": "Name",
                    "col_offset": 0,
                    "ctx": {
                        "ast_type": "Load"
                    },
                    "id": "print",
                    "lineno": 1
                }
            }
        }
    ]
}
```

**Figure 5: Example of JSON AST**

Through testing in different programs, information is gathered on how to read the ASTs of Python programs. Noticeably, the value of the key “ast_type” is important while the value of the key “col_offset” is not.
For instance, by removing unused or unimportant sections, the above program may be read as the following (removed parts are in red):

```json
{
    "ast_type": "Module",
    "body": [
        {
            "ast_type": "Expr",
            "col_offset": 0,
            "lineno": 1,
            "value": {
                "args": [
                    {
                        "ast_type": "Str",
                        "col_offset": 7,
                        "lineno": 1,
                        "s": "hello world"
                    }
                ],
                "ast_type": "Call",
                "col_offset": 0,
                "func": {
                    "ast_type": "Name",
                    "col_offset": 0,
                    "ctx": {
                        "ast_type": "Load"
                    },
                    "id": "print",
                    "lineno": 1
                },
                "keywords": [],
                "lineno": 1
            }
        }
    ]
}
```

**FIGURE 6: READING OF JSON AST**

The JSON file may then be translated as a program that has an expression (Expr) which calls (Call) a function (Func) namely (Name) “print” (id: print) which has an argument (args) of a string (“Str”) contented with “hello world” (s: hello world).
3.1.1.2 Observation list

The observation list is used to support the behavior of the checker. Since the checker is used to check for “observations” which are the characteristics of a program, the observation list is designed to include basic logic flows and programming techniques.

The following figure is the latest version of the observation list:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>Existence of for loop</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O2</td>
<td>Existence of consecutive for loop</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O3</td>
<td>Max occurrence of consecutive for loop</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O4</td>
<td>Existence of nested for loop</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O5</td>
<td>Max depth of nested for loop</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O6</td>
<td>Existence of while loop</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O7</td>
<td>Existence of consecutive while loop</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O8</td>
<td>Max occurrence of consecutive while loop</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O9</td>
<td>Existence of nested while loop</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O10</td>
<td>Max depth of nested while loop</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O11</td>
<td>Existence of loop</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O12</td>
<td>Existence of consecutive loop</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O13</td>
<td>Max occurrence of consecutive loop</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O14</td>
<td>Existence of nested loop</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O15</td>
<td>Max depth of nested loop</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O16</td>
<td>Existence of if-else</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O17</td>
<td>Existence of consecutive if-else</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O18</td>
<td>Max occurrence of consecutive if-else</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O19</td>
<td>Existence of nested if-else</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O20</td>
<td>Max depth of nested if-else</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O21</td>
<td>Existence of recursion</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O22</td>
<td>Number of recursive call</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O23</td>
<td>Number of self-defined function created</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
<td></td>
</tr>
<tr>
<td>O24</td>
<td>Existence of input</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
<tr>
<td>O25</td>
<td>Existence of print</td>
<td>1=True, 0=False</td>
<td>Give suggestion when program does not have this observation</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 7: OBSERVATION LIST**

For clarification of some ambiguous terms such as “consecutive”, “nested” and “max depth”, a table of examples is added in the appendix.
3.1.1.3 Checking method

The checking flow is the following:

I. Separating defined function and the main body of the program by checking the existence of “FunctionDef” under “ast_type”.

II. Determining the basic logic flows (if, for, while) of each defined function

III. Determining all of the functions called in each defined function.

IV. Determining whether a defined function is recursive by checking the functions called in it.

V. Determining the complex logic flows of each defined function if it has called other defined function.

VI. Determining all of the functions called in the main body

VII. Determining the complex logic flows of each main body sections by checking the functions called in it

VIII. Generating observation list.
3.1.1.3.1 Checking method - Nested Logic Flows

The following is an example of a nested logic flow (a nested for loop):

```
for i in range(3):
    for y in range(5):
        for z in range(7):
            print("hello world")
```

**Figure 8: Example of a nested for loop**

It may be observed that the dictionary of AST has the format of “body-for-body-for-body-for”. Therefore, to get the correct depth of nested logic flows, recursion is applied to check through the entire dictionary to get the correct value.
3.1.1.3.2 Checking method - Called Functions, Recursion, Complex Logic Flows

Since programming is flexible, it is possible to have several coding to express the same logic flow. For instance, the following three programs have the same logic:

```
for condition1:
    for condition2:
        statement1
```

```
def function2():
    function1()
def function1():
    for condition2:
        statement1
    for condition1:
        function1()
```

```
def function1():
    for condition2:
        statement1
    for condition1:
        function1()
```

![Figure 9: Multi-syntax Example](image)

However, if only the aforementioned recursive method is applied, it is not possible to determine the “true” depth of the logic flow for each statement. Therefore, it is essential to check the defined functions called in each section and section hereby means any of both defined functions and the statements in the main body.

To correctly check through all defined functions, the following method is applied:

I. Creating nodes of defined functions
II. Joining nodes with directions to create a graph of functional dependencies.
III. Applying Depth First Search for each function.
IV. Deducing loops in function dependencies as recursive functions
V. Considering all possible paths for non-recursive functions to determine the “true” depth of logic flows.
3.1.2 Analysis

For data collection, the assignment is designed and students are invited to finish it. For data analysis, random sample data is generated to test the analysis model. The procedures and results of random sample data are shown below.

![Figure 10: Input Data](image)

Using Excel to generate random sample data (see Figure 10), suppose 50 programs passing all cases are selected. The first column is the program number and from the second column to the last column is the observation value of each program.

![Figure 11: Similarity Table](image)

Next is to measure the similarity by calculating the squared Euclidean distance (see Figure 11) and applies Ward’s method based on OID.
After applying Ward’s method, cluster history (see Figure 12) is generated and a distance graph is plotted (see Figure 13). Elbow method determines the number of clusters. In Figure 13, the red circle shows the elbow point and thus, 3-cluster is suggested in this sample.
**Figure 14:** Dendrogram Illustrating the Cluster of Programs

**Figure 15:** Sample Result of Ward's Method
Figure 16: Sample Result of K-means Method

Figure 15 and Figure 16 are the summary statistics of each cluster by using Ward’s method and K-means method respectively. The results from both methods are similar. Based on these results, the number of clusters implies the number of approaches for solving questions. Those non-zero mean values of each cluster are the characteristics of each approach.

Figure 17: Estimation Table

<table>
<thead>
<tr>
<th></th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
<th>V51</th>
<th>V52</th>
<th>V53</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>0</td>
<td>13.778</td>
<td>10.778</td>
<td>1.111</td>
<td>2.111</td>
<td>10.778</td>
</tr>
<tr>
<td>G2</td>
<td>13.778</td>
<td>0</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>G3</td>
<td>10.778</td>
<td>9</td>
<td>0</td>
<td>7</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>V51</td>
<td>1.111</td>
<td>10</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>V52</td>
<td>2.111</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>V53</td>
<td>10.778</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 18: Estimation – Similarity Table
Table generated for estimation. The first three rows are the summary statistics of each cluster \((G1, G2, G3)\) extracted from the database and the last three rows are three new students’ observation lists \((V51, V52, V53)\) generated by the checker. It measures the similarity by calculating the squared Euclidean distance (see Figure 18). Then, it calculates the correlation between the approaches and the new observation lists (see Figure 19). In this example, \(V51\) and \(V52\) have a higher possibility using approach 1 \((G1)\) while \(V53\) has a higher possibility using approach 2 \((G2)\).

![Figure 19: Estimation – Correlation Table](image)

Finally, recommendations are given by comparing the summary statistics of that approach and the new student’s observation list. Referring to Figure 20, the red and blue rectangles are the examples. Considering column \(O5\), the value of approach 1 is 1.33 while the value of \(V51\) is 1. Since the values are not equal, \(O5\) is one of the suggestions to guide student in improving his code.

![Figure 20: Estimation – Comparison](image)
3.1.3 Recommendation

Recommendation is combining the checker, analysis parts and some of the configuration of the interface. The idea of recommendation has been designed and an example is shown below.

Scenario: A new student working on Question 2 using our platform and submitted his code. The question is an isosceles triangle problem.

Input: 2
Output:

Input: 3
Output:

\&
&
&
&&
&
&&
&&
&

After he submits his code, the website will analyze his code using checker and store the observations in the database. If his answer does not pass all test cases while checking the test cases result, that implies his code requires improvement to finish the question.

The website will extract the pre-compiled data of Question 2 and that of some test cases.

<table>
<thead>
<tr>
<th>#</th>
<th>group</th>
<th>n</th>
<th>mean</th>
<th>sd</th>
<th>se</th>
<th>group</th>
<th>n</th>
<th>mean</th>
<th>sd</th>
<th>se</th>
<th>group</th>
<th>n</th>
<th>mean</th>
<th>sd</th>
<th>se</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>02</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>03</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>04</td>
<td>1</td>
<td>27</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>05</td>
<td>1</td>
<td>27</td>
<td>1.33333</td>
<td>0.96077</td>
<td>0.18490</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>06</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>07</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>08</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>09</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>27</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>27</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>27</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>27</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>17</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 21: Pre-compiled data of Question 2**
Figure 21 shows that there are 3 approaches to finish the question 2 according to the analysis result.

Since the observation list of the student’s code has been generated by the checker, the website will use R to apply the data analysing module to identify which group the new answer belongs to.

<table>
<thead>
<tr>
<th></th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
<th>V51</th>
<th>V52</th>
<th>V53</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>1</td>
<td>-0.840</td>
<td>-0.437</td>
<td>0.985</td>
<td>0.945</td>
<td>-0.822</td>
</tr>
<tr>
<td>G2</td>
<td>-0.840</td>
<td>1</td>
<td>0.057</td>
<td>-0.812</td>
<td>-0.666</td>
<td>0.983</td>
</tr>
<tr>
<td>G3</td>
<td>-0.437</td>
<td>0.057</td>
<td>1</td>
<td>-0.330</td>
<td>-0.433</td>
<td>0.174</td>
</tr>
<tr>
<td>V51</td>
<td>0.985</td>
<td>-0.812</td>
<td>-0.330</td>
<td>1</td>
<td>0.966</td>
<td>-0.767</td>
</tr>
<tr>
<td>V52</td>
<td>0.945</td>
<td>-0.666</td>
<td>-0.433</td>
<td>0.966</td>
<td>1</td>
<td>-0.617</td>
</tr>
<tr>
<td>V53</td>
<td>-0.822</td>
<td>0.983</td>
<td>0.174</td>
<td>-0.767</td>
<td>-0.617</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 22: Result of Estimation**

Figure 22 shows that the newest student’s code belongs to approach 2.

After identifying the student’s code belongs to approach 2, the website will compare the observation list of the student’s answer and the approach 2.

Observation list of the student:

<table>
<thead>
<tr>
<th>O1</th>
<th>O11</th>
<th>O21</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>O2</td>
<td>O12</td>
<td>O22</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>O3</td>
<td>O13</td>
<td>O23</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>O4</td>
<td>O14</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O5</td>
<td>O15</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O6</td>
<td>O16</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O7</td>
<td>O17</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>O8</td>
<td>O18</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O9</td>
<td>O19</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O10</td>
<td>O20</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
After comparing the two lists, some differences will be found and some recommendations can be given to students like: “You may try to add more elements in your code such as If-else statement, recursion and print function to improve your code because most of the students who using the same approach with you also have these elements.”
3.1.4 Database Design

All the tables of the database have been created in both HKU server and Cloud9 server.

Some of the test data has been inserted in the Cloud9 server for developing the interface.

![Database Content in Cloud9 Server](Image)

**Figure 24: Database Content in Cloud9 Server**
3.1.5 Interface Design

The website is created using Cloud9 DLE. It contains a login page, a front page and a question page now. Users may answer the questions by typing the answer in the text area. All the steps the website is doing and the test cases results will be displayed in the result area. The Python compiler and test cases auto-grader have been developed.

The following figures are screen shots of the interface:

![Figure 25: Cloud9 Workspace]
Genius

You can log in here

User Name: 

Password: 

Submit

**Figure 26: Login Page**

```python
# Genius

Please choose one of the question on the left hand side!

Type your code below

1. \(x = 1 + 2\)
2. `print x`

Run
```

**Figure 27: Front Page**
You are required to write a Python program that will output a "square". The square will be composed of ampersand (&) characters. In response to a prompt from your program, the user will provide the size of the square, and you may assume that the user always enters a valid integer between 2 and 20, i.e., 2 ≤ size ≤ 20. Your solution should follow exactly the input/output format shown below in the sample runs.

Sample Input
2

Sample Output
&&
&&

Type your code below

```python
x = 1 + 2
print(x)
```
Figure 29 shows the answering area and the result area which will display the test cases result.
## 3.2 Work Remaining

<table>
<thead>
<tr>
<th>Action Items</th>
<th>Completion Percentage</th>
<th>Completed</th>
<th>Work Remaining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>90%</td>
<td>Database design and build the database</td>
<td>Fine tune the relationship when developing the interface</td>
</tr>
</tbody>
</table>
| Interface    | 70%                   | User interface, code compiler, test cases checking, Database connection | 1. Js for applying the checker to generate the observation list  
2. Js for applying the data analysis |
| Checker      | 80%                   | Check Functions for basic programming elements | 1. Refining code for extendible observation list  
2. Connection to the next 2 phases |
| Analysis     | 70%                   | Use sample data to find out number of approaches for different questions, characteristics of each approach and estimate which approach is used after a new student has submitted program | 1. Use real data to test model  
2. Classify test case by difficulty |
| Recommendation | 30%                | Data analysis for recommendation | Interface: the pop up image to give recommendation, JS for combining all elements |
4. Limitations and Difficulties Encountered

4.1 Checker Limitations

In the current development, the input code has the following assumptions:

Not using defined classes and the defined class functions

Must consist of at least one statement in the main body. i.e. cannot be function definitions only

For 1, to cope with defining classes and class functions, a rework in the current checker code is required. A full rework in Checker would hinder the development of the following phases. In addition, since class may or may not be considered as a topic in beginner programming, it shall be further discussed whether it is going to be added to the observation list or not. There are 2 possible solutions:

I. Rework on Checker

II. Not allowing students to use class. i.e. At the website, telling students not to use classes in questions. At the checker, if there is an existence of class, it will generate an error message to the website to ask the student to do the question again.

For 2, the input of a function definition only code will cause a runtime error in the Checker. However, a function definition only code is compilable but will fail in most if not all generic beginner programming questions. The possible solution is to check if statements exist in the main body before the current checking. If statements do not exist, it would generate a message to the website to ask the student to alter the code.

4.2 Analysis Limitations

R script cannot be run on Cloud9 due to a version problem. There are two possible solutions:

I. To downgrade R version in Cloud9

II. To find other R packages to analyze data in order to fit the current version R in Cloud9.

4.3 Interface Limitations

Python compiler is difficult to be developed in the website. Therefore, skulpt is used in the interface for compiling the python code of the students.
5. Conclusion

This is a meaningful project. Due to increasing trend in learning programming and thus the increasing demand in programming teachers, our system can provide benefits to both students and teachers.

Since the system is using a collaborative concept in doing data mining and data analysis, the system will become more mature. When more students have used the AI tutor, the number of previously submitted programs are increasing to provide a more accurate analysis.

The team is optimistic about keeping the progress to develop a beneficial product.
References


Using the elbow method to determine the optimal number of clusters for k-means clustering. (n.d.). Retrieved from https://bl.ocks.org/rpgove/0060ff3b656618e9136b
Appendix

Observation List Examples

<table>
<thead>
<tr>
<th>Observation List</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Existence of for/while/loop/if-else</strong></td>
<td><strong>Existence of consecutive for/while/loop/if</strong></td>
</tr>
<tr>
<td>for <code>&lt;condition&gt;</code>: <code>&lt;statement&gt;</code></td>
<td>✓</td>
</tr>
<tr>
<td>while <code>&lt;condition&gt;</code>: <code>&lt;statement&gt;</code></td>
<td>✓</td>
</tr>
<tr>
<td>if <code>&lt;condition&gt;</code>: <code>&lt;statement&gt;</code></td>
<td>✓</td>
</tr>
<tr>
<td>print()</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Max occurrence of consecutive for/while/loop/if</strong></th>
<th><strong>Existence of nested for/while/loop/if</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>for <code>&lt;condition&gt;</code>: <code>&lt;statement&gt;</code> for <code>&lt;condition&gt;</code>: <code>&lt;statement&gt;</code></td>
<td>✓</td>
</tr>
<tr>
<td>for <code>&lt;condition&gt;</code>: <code>&lt;statement&gt;</code> for <code>&lt;condition&gt;</code>: <code>&lt;statement&gt;</code></td>
<td>✓</td>
</tr>
<tr>
<td>if <code>&lt;condition&gt;</code>: <code>&lt;statement&gt;</code></td>
<td>✓</td>
</tr>
<tr>
<td>if <code>&lt;condition&gt;</code>: <code>&lt;statement&gt;</code></td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Max depth of nested for/while/loop/if</strong></th>
<th><strong>If-else-if vs if elif</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>if <code>&lt;condition1&gt;</code>: <code>&lt;statement1&gt;</code></td>
<td>if <code>&lt;condition1&gt;</code>: <code>&lt;statement1&gt;</code></td>
</tr>
<tr>
<td>else: <code>&lt;statement2&gt;</code></td>
<td>if <code>&lt;condition1&gt;</code>: <code>&lt;statement1&gt;</code></td>
</tr>
<tr>
<td>if <code>&lt;condition2&gt;</code>: <code>&lt;statement2&gt;</code></td>
<td>else: <code>&lt;statement2&gt;</code></td>
</tr>
<tr>
<td>if <code>&lt;condition2&gt;</code>: <code>&lt;statement2&gt;</code></td>
<td>else: <code>&lt;statement2&gt;</code></td>
</tr>
<tr>
<td>if <code>&lt;not condition1&gt;</code>: <code>&lt;statement1&gt;</code></td>
<td>All of them have the same logic, thus they are all considered to be consecutive if-clause</td>
</tr>
</tbody>
</table>

Note that “loop” hereby includes both “for” and “while”