FINAL REPORT

AI Tutor for Programming Class (The Auto Grading Engine)

Li Angus Dicky         3035181920
Mak Yim Yung           3035124960
Siu King Fung          3035123083

Supervisor: Dr. Chui C. K.
COMP4801
Abstract

Nowadays, STEM education is introduced in Hong Kong and other countries, requiring primary and secondary students to learn programming compulsory. It aims to develop students to become lifelong learners of science, technology, engineering and mathematics, enabling them to meet the challenges in the 21st century. The demand of hiring Computer Science teachers and the burden of teachers are increasingly problematic. To solve this problem, an AI Tutor can benefit for both teachers and students.

The AI Tutor provides Python as the learning programming language and use R as the analysis tool. Its engine consists of three components, namely checker, analysis and recommendation. The programming data and the results of analysis will be stored in MySQL database management system. The AI Tutor is a web-based system and its interface is built by using Cloud9 IDE.

The system will start from a new student doing a question. After student submitting his/her programming, the checker will check the structure of the programming code and generate an observation list. The observation list will be compared with the results stored in database. If the student cannot pass all the test cases, the AI Tutor will give customized feedback to guide how to improve the code.

During construction of the AI Tutor, there are some limitations and difficulties encountered. For checker and interface problem, time spent on researching suitable packages. For question design, most of programming questions contains more than
one concepts. It is difficult to validate whether there are some underlying observations related to the questions. For test case analysis, students in general pass a group of test cases rather than pass one test case. It is challenging to classify the nature of each test cases.

“Learning Programming” is a new and advance trend. There are numerous online learning platforms such as Udacity, HackerRank and Coursera. By comparing these traditional auto-graders, the AI Tutor not only provides auto-grading but also provides personalized recommendation to users. The analysis will be more precise when more users use.
Table of Contents

Abstract ......................................................................................................................... 1

1. Introduction ............................................................................................................. 6
   1.1 Background ......................................................................................................... 6
   1.2 Objectives ........................................................................................................... 7
   1.3 Report Outline ................................................................................................... 8

2. Methodology ............................................................................................................ 9
   2.1 The learning language – Python ......................................................................... 9
   2.2 Analysis Tool – R ............................................................................................... 10
   2.3 Engine Design .................................................................................................... 11
      2.3.1 Checker ....................................................................................................... 12
      2.3.2 Analysis ....................................................................................................... 14
         2.3.2.1 Analysis Objectives .............................................................................. 14
         2.3.2.2 Data Collection .................................................................................... 14
         2.3.2.3 Data analysis ....................................................................................... 15
         2.3.2.4 Estimation ............................................................................................ 16
         2.3.2.5 Common Mistakes .............................................................................. 16
      2.3.3 Recommendation ......................................................................................... 17
      2.3.3 Recommendation ......................................................................................... 18
   2.4 Database Design .................................................................................................. 20
   2.5 Interface Design .................................................................................................. 23

3. Implementation ........................................................................................................ 25
   3.1 Checker ............................................................................................................... 25
      3.1.1 Input ............................................................................................................ 25
      3.1.2 Observation list ........................................................................................... 28
      3.1.3 Checking method ......................................................................................... 29
         3.1.3.1 Checking Method – Nested Logic Flows ............................................... 30
         3.1.3.2 Checking Method – Called Functions, Recursion, Complex Logic Flows .... 31
   3.2 Analysis ............................................................................................................... 32
   3.3 Recommendation ................................................................................................ 42
   3.4 Database Design .................................................................................................. 46
   3.5 Interface Design .................................................................................................. 47

4. Limitations and Difficulties Encountered ............................................................... 54
   4.1 Checker Limitations ............................................................................................ 54
   4.2 Analysis Limitations ............................................................................................ 55
   4.3 Interface Limitations ........................................................................................... 55
# Table of Contents

- Figure 1: Example of an Auto-Grader ................................................................. 6
- Figure 2: Extracted Chart from the HKEB Report .............................................. 7
- Figure 3: Engine Flow of AI Tutor .................................................................. 11
- Figure 4: ER-Diagram of the Database ............................................................. 20
- Figure 5: Example of JSON AST .................................................................... 26
- Figure 6: Reading of JSON AST ........................................................................ 27
- Figure 7: Observation List ................................................................................. 28
- Figure 8: Example of a Nested For Loop ........................................................... 30
- Figure 9: Multi-syntax Example ....................................................................... 31
- Figure 10: Input Data .......................................................................................... 32
- Figure 11: Similarity Table ............................................................................... 33
- Figure 12: Cluster History ................................................................................. 33
- Figure 13: Distance Plot in Elbow Method ......................................................... 34
- Figure 14: Plot the Average Silhouette Method .................................................. 35
- Figure 15: Plot the Calinski-Harabasz Method .................................................... 36
- Figure 16: Dendrogram Illustrating the cluster of programs ................................ 37
- Figure 17: Sample Result of Ward's Method ....................................................... 37
- Figure 18: Sample Result of K-Means Method .................................................... 38
- Figure 19: Estimation Table .............................................................................. 38
- Figure 20: Estimation – Euclidean Distance Table ............................................. 39
- Figure 21: Estimation - Correlation Table .......................................................... 39
- Figure 22: Estimation - Comparison .................................................................. 40
- Figure 23: Mistakes in Approach 3 .................................................................... 40
- Figure 24: Critical Version ............................................................................... 41
- Figure 25: Pre-Compiled Data of Question 2 ...................................................... 43
- Figure 26: Result of Estimation .......................................................................... 43
- Figure 27: Observation List of Q2 Approach 3 ..................................................... 45
- Figure 28: Database Content in Server ............................................................... 46
- Figure 29: Cloud9 Workspace .......................................................................... 47
- Figure 30: Login Page ....................................................................................... 48
- Figure 31: Front Page ......................................................................................... 48
- Figure 32: Question Page ................................................................................... 49
- Figure 33: Answer area and Result area ............................................................... 50
- Figure 34: Number of Approaches in Admin UI ............................................... 51
- Figure 35: Mistakes for each approach in Admin UI .......................................... 51
- Figure 36: Breakthrough Programs in Admin UI ............................................... 52
- Figure 37: Recommendations for each submission in Admin UI ....................... 52
- Figure 38: Observation List in Admin UI ............................................................ 53
- Figure 39: References in Admin UI ................................................................... 53
1. Introduction
1.1 Background

STEM education is being promoted in the renewal of the school educational programs in Hong Kong. One of the aims is to develop and enhance students’ computational thinking and thus programming will be introduced in school curriculum soon. In Hong Kong, Students studying Computer or Information and Communication Technology (ICT) may not be taught by teachers who are majoring in Computer Science. Most of these subjects are given by science teachers. Students may have insufficient learning and teaching resources in their schools. Moreover, although there are lots of online learning platforms such as Scratch and CodinGame to learn coding, they only provide questions and test cases with auto-grading. Users can know pass or fail after submission and there are no further suggestions and advice to them. In fact, the data can be used to do analysis and evaluation.

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

As coding will be introduced in primary and secondary school, all primary and secondary students learn programming compulsorily. The demand of hiring Computer Science teachers will become large. Furthermore, the duty of teachers is not only teaching students, but also have administration jobs and preparation, such as attending internal meeting, writing proposals and preference reviews, preparing lessons and teaching aids, setting examination papers and so on. Teachers are also
in charge of committees such as Discipline committee, Student Development Committee and so on. Accordingly, the burden of teachers becomes heavy as well.

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 programming tasks can be automated in the Cloud and next stored in the database. Teachers could reduce workload in increasingly large class size and give students meaningful feedback on their submission.

Secondly, students may encounter difficulties when doing exercises and would like to ask questions. The AI tutor provides customized feedback to teach and assist students so that students may improve their coding skills and enhance their logical thinking.
Thirdly, after completed an assignment, the system will generate a report showing general performances of classes. Teachers could make use of it to adjust strategies on teaching schedules and progress.

1.3 Report Outline

The remainder of this paper proceeds as follows. Chapter 2 is methodology. It introduces the learning language, the analysis tool, the engine design and the database design. Chapter 3 is implementation. It discusses from data collecting to storing result in database and giving recommendations. Chapter 4 is limitations and difficulties encountered. It introduces analysis problem. The last chapter is conclusion 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. Nowadays, many large-scale companies including Google, Dropbox and NASA use it on operating system and software development. As a programming novice, it is easy to learn its syntax and easy to understand program code. For example, Python does not follow the C-style syntax, using semicolon (‘;’) and brackets (‘{‘) as C++ or Java. Thanks to the standard library, beginners can execute lots of complex functionalities. It also allows users to create data structures that can be re-use. More importantly, it is free, widely supported and cross-platform.

In addition, learning programming can be separated into two sections, namely learning logic and learning syntax. As logic errors are more challenging to trace than syntax error. The AI Tutor would like to help novice centering on the logic thinking of programming so that they could be faster and easier in fixing syntactic confusion. Furthermore, Python provides library that can be used to conveniently analyze Python programs. This will be talked about in checker phase.
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

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

The checker first does standard auto – grading and next proceeds to check and extract information from the source code.

A compiler approach is used to draw useful information from natural strings of source codes. The progress of a compiler is, (i) source code, (ii) abstract syntax, (iii) syntax tree, (iv) control flow graph and (v) object code.

Figure 4: Abstract Syntax Tree Example
The source code is the natural codes written by human beings. Abstract syntax tree (AST) is the tree structure after the compiler parses the code. An illustration is that the parser will recognize keywords such as “if” and “while”. Control flow graph (CFG) is a chart with directions in which the parsed instructions are presented in a flow chart. It means that CFG of the program determines the logic flow and eventually the object code is machine code.

Despite the fact that Python is not a compiled language but an interpreted language which uses an interpreter rather than a compiler, both of them construct an AST throughout the process. With the “ast” library in Python, ASTs could be separated out of the programs of students. On the basis of compilers, it converts AST to CFG and so the logic flows would be determined in the programs. Contrasting from compilers, the whole CFG would not be converted and merely partial crucial logic flows which is
named as “Observations” would be determined. For example, it may deduce a student have used nested-looping if certain criteria are matched.

After collecting 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 distance by calculating squared Euclidean distance and find the similarity by calculating correlation coefficient matrix. 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\(^1\), average silhouette method\(^2\) and Calinski - Harabasz Index\(^3\). 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.

\(^2\) It first runs clustering method and next calculate the average silhouette width, which measure how well each object lies within its cluster. The higher average silhouette width, the better clustering is.

\(^3\) It makes use of total within sum of squares for choosing the optimal k.
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 distance 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.

2.3.2.5 Common Mistakes

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 distance 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, comparing each OID of the certain approach with that of the new student’s observation list, if the values are not matched, it will mark the differences and its corresponding OID.
2.3.2.6 Breakthroughs

A teacher set a threshold value, say 5. A student may submit more than one version. If the difference of the number of passing test cases between two consecutive versions is greater or equal to the threshold value, the latter version is regarded as the breakthrough version. Next, its observation list will compare with the former version’s observation list. If the value of the breakthrough version’s observation list is greater than that of the former version’s observation list, it will mark the differences and its corresponding OID.
2.3.3 Recommendation

The recommendation combines the result of checker and data analysis to present how the final product works. A website platform is built to join these components so as to achieve 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.”

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

For common mistakes, AI Tutor applies data analysis model to discover common mistakes. The AI Tutor afterwards gives recommendations to assist the student so as to avoid those common mistakes.

For breakthroughs, the AI Tutor analyzes the all programs in the same question to reveal which program(s) is(are) regarded as critical version(s). Teachers would make use of these results to adjust their teaching scheme and students would improve their coding skills.
2.4 Database Design

phpMyAdmin is utilized for the database because it can be used to manage the database easily comparing to other databases like MongoDB.

Two databases are built on different servers which are the HKU server and the Cloud9 server. The reason for building one more database in Cloud0 server is that it is easier to connect to the website. After developing the entire website in Cloud9, the website can be moved to the HKU server by changing the configuration of the database.

The following diagram is the structure of the database:

Figure 6: ER-Diagram of the Database
The following tables show what data is stored in the database.

<table>
<thead>
<tr>
<th>UID</th>
<th>Name</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ada</td>
<td>*****</td>
</tr>
<tr>
<td>1</td>
<td>Bill</td>
<td>*****</td>
</tr>
<tr>
<td>2</td>
<td>Calvin</td>
<td>*****</td>
</tr>
<tr>
<td>2</td>
<td>Davy</td>
<td>*****</td>
</tr>
</tbody>
</table>

Table 1: Student Table

Table 1 is a Student table and stores students’ UID and the account information.

<table>
<thead>
<tr>
<th>QID</th>
<th>Content</th>
<th>Sampleinput</th>
<th>Sampleoutput</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>You are required to write a Python program ...</td>
<td>2</td>
<td>&amp; &amp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&amp; &amp;</td>
</tr>
<tr>
<td>2</td>
<td>You are required to write a Python program ...</td>
<td>2</td>
<td>&amp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&amp; &amp;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&amp;</td>
</tr>
</tbody>
</table>

Table 2: Question Table

Table 2 is Question table and stores the question number, content, sample input and output.

<table>
<thead>
<tr>
<th>UID</th>
<th>QID</th>
<th>VID</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>x=1; if(x&gt;0) {...}</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>x=a+b; y=a*b; while {...}</td>
</tr>
</tbody>
</table>

Table 3: Version

Table 3 is Version table and stores the submission program codes.
<table>
<thead>
<tr>
<th>Unique</th>
<th>QID</th>
<th>TID</th>
<th>Input</th>
<th>Output</th>
<th>Nature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1+1</td>
<td>2</td>
<td>Addition</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1/1</td>
<td>1</td>
<td>Division</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1mod1</td>
<td>0</td>
<td>Modulo</td>
</tr>
</tbody>
</table>

Table 4: Test Case Table

Table 4 is Test Case table and stores test case contents, outputs, and natures.

<table>
<thead>
<tr>
<th>TRid</th>
<th>UID</th>
<th>VID</th>
<th>QID</th>
<th>TID</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>F</td>
</tr>
</tbody>
</table>

Table 5: Test Case Result Table

Table 5 is Test Case Result table and records the pass and fail results of each test case for each student.

<table>
<thead>
<tr>
<th>OrderID</th>
<th>OID</th>
<th>Observation Description</th>
<th>Result</th>
<th>Purpose</th>
<th>Remarks</th>
<th>Example</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O1</td>
<td>Existence of for loop</td>
<td>1 = True; 0 = False</td>
<td>Give suggestion when …</td>
<td>It seems that your …</td>
<td>It seems that your …</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>O2</td>
<td>Existence of consecutive for loop</td>
<td>1 = True; 0 = False</td>
<td>Give suggestion when …</td>
<td>It seems that your …</td>
<td>It seems that your …</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Observation List Table

Table 5 is Observation List table and stores observations’ contents, recommendation statement.
2.5 Interface Design

The AI Tutor is a web-based platform since the users can be easier to access. It is developed using Cloud9 since it is free and user-friendly for construction. Lots of plugins such as Python are supported by Cloud9 using in checker phase.

Figure 7: Interface
Users should login their own accounts. Next step is to select question and submit their program. In Figure 7, it contains question, sample input and output and a text area for students to write their code.

In recommendation interface, there is a figure of the AI Tutor to provide recommendations to the students. The AI Tutor will pop up on the right-hand-side of the page after submission and give feedback according to the result of data analysis such as improvement hints, mistakes or breakthrough reminders.
3. Implementation

3.1 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 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:

```
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
            },
            "keywords": [],
            "lineno": 1
         }
      }
   }
}
```

Figure 8: Example of JSON AST

Information is collected on how to read the ASTs of Python programs over testing in different programs. Obviously, the value of the key "ast_type" is significant while the value of the key "col_offset" is not.

For instance, by erasing unused or unimportant sections, the above program may be read as the following (removed parts are in red):

```
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
            },
            "keywords": [],
            "lineno": 1
         }
      }
   }
}
```
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).
3.1.2 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>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OID</td>
<td>Observation Description</td>
<td>Result</td>
<td>Purpose</td>
</tr>
<tr>
<td>2</td>
<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>
</tr>
<tr>
<td>3</td>
<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>
</tr>
<tr>
<td>4</td>
<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>
</tr>
<tr>
<td>5</td>
<td>O4</td>
<td>Existence of nested for loop</td>
<td>1=True; 0=False</td>
<td>Give suggestion when program does not have this observation</td>
</tr>
<tr>
<td>6</td>
<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>
</tr>
<tr>
<td>7</td>
<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>
</tr>
<tr>
<td>8</td>
<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>
</tr>
<tr>
<td>9</td>
<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>
</tr>
<tr>
<td>10</td>
<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>
</tr>
<tr>
<td>11</td>
<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>
</tr>
<tr>
<td>12</td>
<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>
</tr>
<tr>
<td>13</td>
<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>
</tr>
<tr>
<td>14</td>
<td>O13</td>
<td>Max occurrence of consecutive loop</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
</tr>
<tr>
<td>15</td>
<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>
</tr>
<tr>
<td>16</td>
<td>O15</td>
<td>Max depth of nested loop</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
</tr>
<tr>
<td>17</td>
<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>
</tr>
<tr>
<td>18</td>
<td>O17</td>
<td>Existence of consecutive if-else</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
</tr>
<tr>
<td>19</td>
<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>
</tr>
<tr>
<td>20</td>
<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>
</tr>
<tr>
<td>21</td>
<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>
</tr>
<tr>
<td>22</td>
<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>
</tr>
<tr>
<td>23</td>
<td>O22</td>
<td>Number of recursive call</td>
<td>Integer</td>
<td>Give suggestion when program does not match the result</td>
</tr>
<tr>
<td>24</td>
<td>O23</td>
<td>Existence of input</td>
<td>1=True; 0=False</td>
<td>Give suggestion when program does not have this observation</td>
</tr>
<tr>
<td>25</td>
<td>O24</td>
<td>Existence of print</td>
<td>1=True; 0=False</td>
<td>Give suggestion when program does not have this observation</td>
</tr>
</tbody>
</table>

*Figure 10: Observation List*
3.1.3 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 functions 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.
3.1.3.1 Checking Method – Nested Logic Flows

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

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

The dictionary of AST has the format of "body-for-body-for-body-for". Therefore, recursion is applied to check through the entire dictionary to get the correct depth value of nested logic flows.
3.1.3.2 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:

```python
for condition1:
    for condition2:
        statement1

def function2():
    function1()

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

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

*Figure 12: Multi-syntax Example*

However, if only the aforementioned recursive method is applied, it is impossible to determine the “true” depth of the logic flow for each statement. Therefore, it is necessary 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.

3.2 Analysis

For data collection, the assignment was designed, and students were invited to finish it. For data analysis, random sample data was generated to test the analysis model.

The procedures and results of random sample data are shown below.

![Figure 13: Input Data](image-url)

Using Excel to generate random sample data (see Figure 10), suppose 150 programs passing all cases are selected. The first column is students’ UID. The second column is the program number and from the third column to the last column is the observation value of each program.
Next is to measure the distance 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). The number of clusters is determined by a majority vote of three methods finding the optimal value. The first method is Elbow method. In Figure 13, the dotted line shows the elbow point and thus, 4-cluster is suggested in this sample.
The second method is Average Silhouette method. The data have been classified using Ward’s method. It next calculates the average silhouette width for each cluster.

\[
s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}
\]

where \(a(i)\) is average distance between the data point \((i)\) and the points within its own cluster, \(b(i)\) is average distance between the data point \((i)\) and the points of neighboring cluster. The range of \(s(i)\) is \([-1,1]\).

In Figure 14, the maximum point is at 4 and thus, 4-cluster is suggested in this sample.
The third method is Calinski-Harabasz method. The data has been classified using Ward’s method and next calculate Calinski-Harabasz (CH) index.

\[
CH(k) = \frac{B_c(k)/(k-1)}{W_c(k)/(n-1)} = \frac{n-1 B_c(k)}{k-1 W_c(k)}
\]

\[
B_c = \sum_{k=1}^{K} |C_k| \| \bar{C}_k - \bar{x} \|^2
\]

\[
W_c = \sum_{k=1}^{K} \sum_{i=1}^{N} w_{k,i} \| x_i - \bar{C}_k \|^2
\]

where \( n \) is number of cluster and \( k \) is class, \( B_c \) and \( W_c \) is between and within cluster sums of squares respectively, \( w_{k,i} \) is the number of elements in the cluster \( C_k \).

In Figure 15, the maximum point is at 4 and thus, 4-cluster is suggested in this sample.

*Figure 18: Plot the Calinski-Harabasz Method*
Within these three methods, the majority vote is 4-cluster. Therefore, 4-cluster is suggested.

Figure 19: Dendrogram Illustrating the cluster of programs

Figure 20: Sample Result of Ward's Method
Figure 21: Sample Result of K-Means Method

Figure 17 and Figure 18 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 result, 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 23: Estimation – Euclidean Distance Table

<table>
<thead>
<tr>
<th></th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
<th>G4</th>
<th>V00101</th>
<th>V00102</th>
<th>V00103</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>0</td>
<td>10</td>
<td>11</td>
<td>16</td>
<td>8</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>G2</td>
<td>10</td>
<td>0</td>
<td>9</td>
<td>14</td>
<td>6</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>G3</td>
<td>11</td>
<td>9</td>
<td>0</td>
<td>17</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>G4</td>
<td>16</td>
<td>14</td>
<td>17</td>
<td>0</td>
<td>14</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>V00101</td>
<td>8</td>
<td>6</td>
<td>1</td>
<td>14</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>V00102</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>12</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>V00103</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>12</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 24: Estimation – Correlation Table

<table>
<thead>
<tr>
<th></th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
<th>G4</th>
<th>V00101</th>
<th>V00102</th>
<th>V00103</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>1.00000000</td>
<td>0.263100213</td>
<td>0.40395144</td>
<td>-0.3358583</td>
<td>0.49695261</td>
<td>0.60397184</td>
<td>0.60397184</td>
</tr>
<tr>
<td>G2</td>
<td>0.263100213</td>
<td>1.00000000</td>
<td>0.39437999</td>
<td>-0.3413029</td>
<td>0.51395231</td>
<td>0.67575583</td>
<td>0.67575583</td>
</tr>
<tr>
<td>G3</td>
<td>0.40395144</td>
<td>0.39437999</td>
<td>1.00000000</td>
<td>-0.4454516</td>
<td>0.97412216</td>
<td>0.75308265</td>
<td>0.75308265</td>
</tr>
<tr>
<td>G4</td>
<td>-0.3358583</td>
<td>-0.34130293</td>
<td>-0.4454516</td>
<td>1.00000000</td>
<td>-0.44645066</td>
<td>-0.38004776</td>
<td>-0.38004776</td>
</tr>
<tr>
<td>V00101</td>
<td>0.49695261</td>
<td>0.513952309</td>
<td>0.97412216</td>
<td>-0.4464507</td>
<td>1.00000000</td>
<td>0.88151866</td>
<td>0.88151866</td>
</tr>
<tr>
<td>V00102</td>
<td>0.60397184</td>
<td>0.67575583</td>
<td>0.75308265</td>
<td>-0.3800478</td>
<td>0.88151866</td>
<td>1.00000000</td>
<td>1.00000000</td>
</tr>
<tr>
<td>V00103</td>
<td>0.60397184</td>
<td>0.67575583</td>
<td>0.75308265</td>
<td>-0.3800478</td>
<td>0.88151866</td>
<td>1.00000000</td>
<td>1.00000000</td>
</tr>
</tbody>
</table>

Figure 19 is a table generated for estimation. The first four rows are the summary statistics of each cluster (G1, G2, G3, G4) extracted from the database and the last three rows are one new student’s observation lists (V00101, V00102, V00103) generated by the checker. It measures the distance between cluster and each version by calculating the squared Euclidean distance (see Figure 20). Then, it calculates the correlation coefficient between the approaches and the new observation lists (see Figure 21). In this example, V00101, V00102 and V00103 have a higher possibility using approach 3 (G3).
In common mistakes, it first selects the programs which do not pass all test cases. It then compares each OID of the certain approach with that of the new student’s observation list (see Figure 22). If the values are not matched, it will mark the differences and its corresponding OID. For example, in column O8, the value of V00101 is 1 while the value of G3 is 2. The value -1 and O8 will store in mistake table.

<table>
<thead>
<tr>
<th>OID</th>
<th>Error(%)</th>
<th>ObservationDescription</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>6</td>
<td>Existence of for loop</td>
</tr>
<tr>
<td>O6</td>
<td>6</td>
<td>Existence of while loop</td>
</tr>
<tr>
<td>O7</td>
<td>56</td>
<td>Existence of consecutive while loop</td>
</tr>
<tr>
<td>O8</td>
<td>81</td>
<td>Max occurrence of consecutive while loop</td>
</tr>
<tr>
<td>O16</td>
<td>12</td>
<td>Existence of if-else</td>
</tr>
<tr>
<td>O23</td>
<td>25</td>
<td>Existence of input</td>
</tr>
</tbody>
</table>

Considering all students submitted their programs, the system calculates the percentage of each OID by approaches in mistake table (see Figure 23).
The last objective in analysis phase is to find out the breakthrough programs. The breakthrough program is defined as a program which the number of passing test cases is greater than that of the previous submission and the difference of the number of passing test cases is greater or equal to the threshold value which is set by a teacher. The default threshold value is 5. In every submission, there are ten test cases to test whether a program is correct or not. The pass mark is 50%.

In the example, five programs are regarded as the breakthrough programs (see Figure 24). The breakthrough programs’ observation lists will compare with the previous version’s observation list correspondingly. If the value of the breakthrough version’s observation list is greater than that of the former version’s observation list, it will mark the differences and its corresponding OID.
3.3 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.

<table>
<thead>
<tr>
<th>Input: 2</th>
<th>Input: 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>Output:</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;&amp;&amp;</td>
</tr>
<tr>
<td></td>
<td>&amp;&amp;</td>
</tr>
</tbody>
</table>

After submission, the system analyzes his code using checker and next stores the observations in the database. If his program 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 retrieve the pre-compiled data of Question 2 and that of some test cases.
Figure 25 shows that there are 4 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 analyzing module to identify which group the new answer belongs to.

Figure 26 shows that the newest student’s code belongs to approach 3.
After identifying the student’s code belongs to approach 3, the website will compare the observation list of the student’s answer and the approach 3.

Observation list of the student:

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

![Figure 30: Observation List of Q2 Approach 3](image)
3.4 Database Design

Both HKU server and Cloud9 server have all the tables of the database. Some of the test data has been inserted in the Cloud9 server for developing the interface.

Figure 31: Database Content in Server
3.5 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:

![Cloud9 Workspace](image-url)
**Figure 33: Login Page**

**Figure 34: 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., \( 2 \leq \text{size} \leq 20 \). Your solution should follow exactly the input/output format shown below in the sample runs.

**Sample Input**

```
2
```

**Sample Output**

```
&
&
```

Type your code below

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

Figure 32 is the question page which contains the question content and the sample input output.
Figure 33 shows the answering area and the result area which will display of the test cases result.
### Figure 37: Number of Approaches in Admin UI

<table>
<thead>
<tr>
<th>Approach</th>
<th>Approach 1</th>
<th>Approach 2</th>
<th>Approach 3</th>
<th>Approach 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD</td>
<td>Number</td>
<td>Observation/Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>1</td>
<td>Existence of consecutive for loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>2</td>
<td>Max occurrence of consecutive for loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>1</td>
<td>Existence of if-else</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 38: Mistakes for each approach in Admin UI

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Observation/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Existence of if-loop</td>
</tr>
<tr>
<td>02</td>
<td>Existence of consecutive for-loop</td>
</tr>
<tr>
<td>03</td>
<td>Max occurrence of consecutive for-loop</td>
</tr>
<tr>
<td>04</td>
<td>Existence of nested for-loop</td>
</tr>
<tr>
<td>05</td>
<td>Existence of while-loop</td>
</tr>
<tr>
<td>06</td>
<td>Existence of if-else</td>
</tr>
<tr>
<td>07</td>
<td>Existence of consecutive if-else</td>
</tr>
<tr>
<td>08</td>
<td>Max occurrence of consecutive if-else</td>
</tr>
<tr>
<td>09</td>
<td>Existence of input</td>
</tr>
</tbody>
</table>
Figure 39: Breakthrough Programs in Admin UI

Figure 40: Recommendations for each submission in Admin UI
From Figure 34 to Figure 39, it is an Admin page to give an overview of the students’ performance.
4. Limitations and Difficulties Encountered

4.1 Checker Limitations

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

I. Not using defined classes and the defined class functions
II. 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
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

It is challenging to develop a new console in the website for compiling the code. The reason is console is a complicated tool. It spends much time to research what packages or add-on tools is suitable for the system. Finally, SKULPT is selected to install in the website for compilation.

In addition, any new program submitted are in the form of a string; however, the checker is not able to accept a long string as its input. Therefore, a PHP script on the server side is implemented to create temporary “.py” files as the input of the checker.
5. Conclusion

In summary, there are two purposes of the AI Tutor. The first one is to reduce teachers’ workload by grading of programming code in the Cloud automatically. The second one is to give guidelines to students when they encounter challenges. In the system, the learning programming language is Python and the analysis tool is R. The engine consists of three components, namely checker, analysis and recommendation. In addition, the system uses MySQL as the database management system and the interface of the system is built by using Cloud 9 IDE.

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 AI Tutor now targets novice and assumes that the programming questions are very simple. Some observations such as the usage of libraries, self-defined class and generator expressions are not included in the final version. In future, observation list keeps updating and that will help AI Tutor more intelligent. In the checker phase, more situations will be considered to avoid students cheat the AI Tutor. In the analysis phase, more information will be explored and implement other statistic models and filtering to hit higher accuracy in analyzing students’ program. In the recommendation phase, the students’ ranking and other functions like hints will be provided to students.

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

Observation List

Examples

Note that "loop" hereby includes Both "for" and "while"
Existence of recursion

<table>
<thead>
<tr>
<th>Function Call</th>
<th>Recursion Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>func1(5)</td>
<td>1</td>
</tr>
<tr>
<td>func1(5)</td>
<td>2</td>
</tr>
<tr>
<td>func1(x)</td>
<td>3</td>
</tr>
</tbody>
</table>

Number of recursive call

<table>
<thead>
<tr>
<th>Function Call</th>
<th>Recursive Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>func1(5)</td>
<td>1</td>
</tr>
<tr>
<td>func1(5)</td>
<td>2</td>
</tr>
<tr>
<td>func1(x)</td>
<td>3</td>
</tr>
</tbody>
</table>

Number of self-defined functions created

<table>
<thead>
<tr>
<th>Function Call</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>func1(5)</td>
<td>1</td>
</tr>
<tr>
<td>func1(x)</td>
<td>3</td>
</tr>
</tbody>
</table>

Existence of print/input

<table>
<thead>
<tr>
<th>Function Call</th>
<th>Print/Print</th>
</tr>
</thead>
<tbody>
<tr>
<td>print(x)</td>
<td>2</td>
</tr>
<tr>
<td>print(y)</td>
<td>3</td>
</tr>
</tbody>
</table>

For conditions:

<table>
<thead>
<tr>
<th>Condition</th>
<th>print(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 5</td>
<td>1</td>
</tr>
<tr>
<td>y = 100</td>
<td>2</td>
</tr>
<tr>
<td>z = 3</td>
<td>3</td>
</tr>
</tbody>
</table>