# **School of Computing, Engineering and Mathematics**



**An investigation into creating a python based programming language and the educational implications of presenting it to year 8 Computing students**

Ionut Alin Neagu

Student ID: 5029475

BSc. Computer Science

Supervisor: Dr. Abdulrahman Altahhan

25th of April 2016

**300COM / 303COM Declaration of originality**  
I Declare that This project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc.) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

**Statement of copyright**

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to Coventry University. Support, including funding, is available to commercialise products and services developed by staff and students.  Any revenue that is generated is split with the inventor/s of the product or service.  For further information please see www.coventry.ac.uk/ipr or contact [ipr@coventry.ac.uk](mailto:ipr@coventry.ac.uk).

**Statement of ethical engagement**  
  
I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (https://ethics.coventry.ac.uk/) and that the application number is listed below (Note: Projects without an ethical application number will be rejected for marking)

Signed: Date: 25/04/2016

Please complete all fields.

|  |  |
| --- | --- |
| First Name: | Ionut Alin |
| Last Name: | Neagu |
| Student ID number | 5029475 |
| Ethics Application Number | **P40445** |
| 1st Supervisor Name | Dr. Abdulrahman Altahhan |
| 2nd Supervisor Name | Professor Anne James |

This form must be completed, scanned and included with your project submission to Turnitin. Failure to append these declarations may result in your project being rejected for marking.

**Contents**

|  |  |
| --- | --- |
| 1. Abstract |  |
| 1. Acknowledgements |  |
| 1. Introduction |  |
| 1. Literature Review |  |
| 1. Method    1. Choosing the syntax of the language    2. Pre-sets for the program to run    3. Lexical Analysis    4. Parsing and Semantic Analysis    5. The method of evaluating the improvement of knowledge in class |  |
|  |  |

**Abstract**

This paper describes the process of creating a python based programming language with a syntax chosen by the author. It is followed by an investigation into determining whether year 8 students have leant or not more knowledge regarding arithmetical operations in programming by contrasting the results from a test in Prog and SCRATCH2.

This project has mainly two objectives: First is to create a programming language from scratch using Python. Second is to evaluate whether year 8 students would benefit from learning it in addition to the the programming language they are studying at the moment (SCATCH2).

In order to achieve the goal, I had to create the language syntax, and then, in the lexical analysis phase, to create tokens that assign a type to every character or set of characters in the original text, followed by the parsing phase, which involves taking the tokens and converting them into intermediate code and then in the code generation phase, the intermediate code has been converted into executables. In regards to the applicable part of this project, the year 8 students at Cardinal Wiseman School in Coventry had to attend a lesson that I taught about Prog, which is the name of my programming language, then they completed a test, both in SCRATCH2 and Prog, followed up by a questionnaire highlighting the learning outcomes of the lesson.

There will be several references to the syntax of the new programming language and how these parts of it refer to the current goals that have to be met by students in order to pass their GCSEs.

By developing this study, the teacher and any other education officials will be able to assess whether teaching scratch as their first programming language creates a good programming background for the new generations of programmers, due to Scratch’s approach of teaching programming by creating and playing games.

I consider that the project has been a success as the students have gained more knowledge of how computational thinking works, and the project can easily be reproduced and even more importantly, studied in more in-depth.

**Acknowledgements**

Firstly, I want to thank Dr. Abdulrahman for providing so much support and inspiration into finalising this project. Without his help I might have ended up doing a project that I have never enjoyed and would have looked like: “just another project”. I also want to thank Professor Anne James for using her time to read this paper and also to listen to my presentation about what I have done. Also, I want to thank every teacher from Cardinal Wiseman School for being so understanding and believing in the fact that I could teach their pupils a programming language that I created. Last but not least, I want to thank my family for supporting me in all those years into achieving my dreams.

**Introduction**

As in the past few year, there has been a lot of reforms in the educational system in the UK, regarding IT studies converging to Computer Studies or Computer Science in the Secondary and Pre-University Sector, I considered that I could have a look at this issue. Followed by this interest, starting to teach at Cardinal Wiseman School, I realized that SCARTCH2 does not offer enough support into understanding how computational thinking and forward planning works. Therefore, I decided that I was going to write a programming language of my own, which will push students’ minds into thinking forward, planning their actions before they start writing the program.

Creating a programming language, you need to consider the syntax, looking at other educational programming languages (e.g. Pascal, which was one of the most successful programming languages for new learners a few years ago, SCRATCH, Python), and consider what concepts limit the students from understanding how computational thinking works. Then, after the syntax has been defined, you need to go through the lexical analysis phase, where you are creating tokens for each word that can appear in the program. The following phase is parsing; in this section, the compiler will take each token and create a parse tree, which will then be compiled into the final code in the code generation phase, in this case using python3.

This project can be easily reproduced by using the methods I have spoken about and by presenting it to a year 8, set 3 computing class. The test took place on Monday, April 18th, 2016, in period 3 (12:30-13:40). The whole class was consisting of 20 students who came back from the Easter holiday just a week before this test.

In the report:

* a.b will represent: all the elements from a to b, in the ASCII table.
* (a)\* will represent: a repeated 0 to infinity times e.g.: a or aaaaa or nothing
* aVb will represent: either a or b
* (a)+ will represent: a repeated at least one time e.g.: a or aa or aaaaa
* space will represent: a string formed of a single press of the space bar on the keyboard
* reset will represent: assign to a variable the value NULL, or empty string
* tab will represent: a string formed of a single press of the tab botton on the keyboard
* new line will represent: a string formed of a single press of the return button on the keyboard

In the next few chapters, I am going to present how each phase of the project has occurred and then I am going to analyse the output data from the primary method used.

**Literature review**

A base start for the research that I am going to do is to understand the basic concepts of programming languages because I want mine to have a clear, simple syntax. Therefore, the research starts with the chapter 2.5 of the article Usability of Programming Languages written by Dr. Alan Blackwell: Programming Languages in Education: The Search for an Easy Start. He is stating that the education has to be based on 4 basic pillars, and every skill acquired has to be in accordance with them:

* Develop new ways to handle knowledge
* Teach school curriculum
* Learn new computing concepts
* Develop cognitive skills

Based on these 4 concepts, the new programming language has to be concentrated on an innovative way of handling the knowledge, and by showing the stages in which a program is actually converted from plain text into a series of operations, the understanding of what computational thinking is would increase which would lead towards a higher understanding of how to write programs from scratch.

Moreover, the school curriculum would be taught as this programming language would inherit some functionality from the C language, but will convert it into a simpler syntax which will lead to a higher understanding of all the basic concepts like: variable, operands and many other.

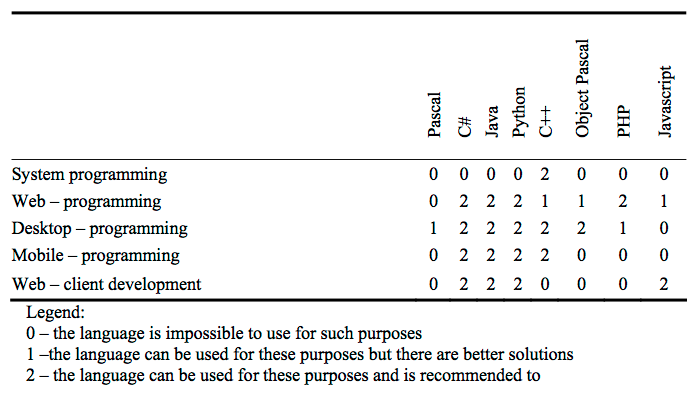
Then, the 3rd concept can expand the first idea even in more depth because at the moment, students are being taught in school how to use different functions, without being told why they should or should not be used. By overcoming this barrier, students would reduce the period of time necessary to understand computational thinking drastically, which may increase their grades.

The last concept: developing cognitive skills would be achieved by working on their practical skills because in programming, the more useful cognitive skill that can be learnt is debugging which is developed perfectly when the student is writing a multitude of programs and learn on the test-error method.

Now, emphasizing on the 3rd concept which is most relevant to this research, Mr. Blackwell is stating that the learning of content hypothesis which is formed of teaching school curriculum and learning computer concepts, opposes two concepts with various different goals. Firstly, there is the idea that programming only allows pupils to learn computer concepts but at the same time there’s also another idea that is showing that programming is helping students to learn other disciplines like: geometry, arithmetic and even grammar, due to the fact that programming is actually a set of rules that if they are being followed lead to a final product that provides the requirements desired, which perfectly applies to any grammar of any language.

Now, looking at a different programming language that is being taught in school as a second programming language, we stop at Python. Looking at the paper “Choosing the first educational programming language”, we can identify that the author is proposing Python as a suitable programming language because it is “powerful and simple”, and not as “old-fashioned” as its predecessor: Pascal. Moreover, looking at a graph in the same paper that is describing the use of different programming languages for educational purposes between 2002 and 2012, we can see that where C++ dropped from over 20% to under 10%, Python stayed constant between 10% and 15% which show a high level of efficiency. Some other programming languages listed are: C#, JAVA, JAVAScript, Pascal and PHP but none of them is actually very efficient for educational purposes, every one of them fluctuating under 5%.

Figure 1: Applicability of Programming Languages (Kruglyk and Lvov 2012)



In the same note, looking at Figure 1, we can see that the applicability of Python is broader that the one for C++, having 4-2’s whereas C++ only has 3-2’s and a 1. Now, looking at the applicability of these programming languages, Python can be used for Web-programming and Web-client development, Desktop and Mobile Development whereas C++ can be used for System programming, having in common with Python the Desktop and Mobile programming. However, at Key Stage 3, even in the next few tens of years, it is very unlikely that the System programming will be introduced, whereas the Web-client development can be easily introduced, either with C++ or, more commonly, JAVAScript. Of course C# and JAVA share the same applicability as Python, however, they are hardly used as initial programming languages to learn due to the complicate structure.

**Primary method**

# Choosing the syntax of the language

I have started by looking at both the syntax of Python3 and C++, at the same time considering SCRATCH2. I have started by considering how the program should start and end. Therefore, after considering it, I decided that it would be better if the program would only get on with what it has to do, without defining any kind of functions as you do in C++ (No need to call the main function to start the program).

Then, I needed to consider what functionalities I should have for the programming language. I started by considering that I will have numbers and strings, which means that the users will not be distracted by other data types like: integers, floats, chars, etc.

STRUCTURE:

NUMBER: ((+V-)(0.9)\*)V(0.9)\*

STRING: “(0.9Va.zVA.ZV.V,V:V;V/)\*”

Then, I considered that the data has to be stored. Therefore, I created variables, which are being define: $variable\_name. I this way, every single time the student will be seeing the dollar sign they will know that it is a variable that can store either a number or a string.

STRUCTURE:

VARIABLE: $(a.zVA.ZV0.9V\_)+

Following, I defined the input and print functions. To define the input function the user need to write either the input or INPUT followed by a message that they want to display to the screen, followed by the name of the variable they want to use. The print function will display either a number or a string or an evaluated expression or a variable. The expressions are automatically being evaluated using Python3, therefore, I did not have to consider the structure of the evaluated expression so I will just call it EXPR.

STRUCTURE:

INPUT\_STATEMENT: (input V INPUT) STRING VARIABLE

PRINT\_STATEMENT: (print V PRINT) (NUMBER V STRING V VARIABLE V EXPR)

The next step was to introduce the conditional statement if. It can evaluate if a string or number or variable or expression is equal or smaller or bigger or less-equal or bigger-equal or different to one of the data structure previously mentioned.

STRUCTURE:

IF\_STATEMENT: (if V IF) (NUMBER V STRING V VARIABLE V EXPR) (== V != V < V <= V > V >=) (NUMBER V STRING V VARIABLE V EXPR)

(INPUT\_STATEMENT V PRINT\_STATEMENT V FOR\_STATEMENT V IF\_STATEMENT)\*

Finally, as a repetitive statement I decided to implement the for statement. It will repeatedly run, a predefined number of steps.

STRUCTURE:

FOR\_STATEMENT: (for V FOR) (NUMBER V VARIABLE) (to or TO) (NUMBER V VARIABLE)

(PRINT\_STATEMENT V FOR\_STATEMENT V IF\_STATEMENT)\*

# Pre-sets for the program to run

I will be writing pseudo code that represents the pre-sets of the program. The program will start from run().

import all the files from the current directory

tokens = empty list

symbols = empty dictionary

function open\_file:

input: file\_name

output: data from the file

data = all the text from the file file\_name

add <EOF> to the end of the document

return data

function run:

input: N/A

output: N/A

filename = empty string

filename = input from the user

data = call open\_file with parameter filename

toks = lex(data)

parse(toks)

# Lexical analysis

I consider that the best way to describe how I have completed this part, is by writing the pseudo code of the lexical analysis.

function lex:

input: text of the file (filecontents)

output: tokens

tok = empty string

state = false

expr = false

isexpr = false

var = empty string

varstarted = false

n = empty string

transform each character of filecontents in an element of a list filecontents

for each character char in filecontents:

add char to tok

if tok equals to space:

if state is false:

reset tok

else:

tok = space

elif tok equals to new-line or <EOF>:

if expr is not empty and isexpr is true:

append (“EXPR:” + expr) to tokens

reset expr

isexpr = false

elif expr is not empty and isexpr is true:

append (“NUM” + expr) to tokens

reset expr

elif var is not empty:

append (“VAR” + var) to tokens

reset var

varstarted = false

reset tok

elif tok equals to “=” and state is false:

if expr is not empty and isexpr is false:

append (“NUM” + expr) to tokens

reset expr

elif var is not empty:

append (“VAR” + var) to tokens

reset var

varstarted = false

elif previous element in tokens list is “EQUALS”:

replace previous element in tokens with “EQEQ”

elif previous element in tokens list is “LESS”:

replace previous element in tokens with “LESSEQUAL”

elif previous element in tokens list is “MORE”:

replace previous element in tokens with “MOREEQUAL”

else:

append (“EQUALS”) to tokens

reset tok

elif tok equals to “<” and state is false:

if expr is not empty and isexpr is false:

append (“NUM” + expr) to tokens

reset expr

elif var is not empty:

append (“VAR” + var) to tokens

reset var

varstarted = false

append (“LESS”) to tokens

reset tok

elif tok equals to “to” or “TO” and state is false:

if expr is not empty and isexpr is false:

append (“NUM” + expr) to tokens

reset expr

elif var is not empty:

append (“VAR” + var) to tokens

reset var

varstarted = false

append (“TO”) to tokens

reset tok

elif tok equals to “>” and state is false:

if expr is not empty and isexpr is false:

append (“NUM” + expr) to tokens

reset expr

elif var is not empty:

append (“VAR” + var) to tokens

reset var

varstarted = false

append (“MORE”) to tokens

reset tok

elif tok equals to “!=” and state is false:

if expr is not empty and isexpr is false:

append (“NUM” + expr) to tokens

reset expr

elif var is not empty:

append (“VAR” + var) to tokens

reset var

varstarted = false

append (“NOTEQUAL”) to tokens

reset tok

elif tok equals to “$” and state is false:

varstarted = true

add tok to var

reset tok

elif varstarted equals to true:

if tok equals to “<” or “>”:

if var is not empty:

append (“VAR” + var) to tokens

reset var

varstarted = false

add tok to var

reset tok

elif tok equals to “print” or “PRINT”:

append (“PRINT”) to tokens

reset tok

elif tok equals to “input” or “INPUT”:

append (“INPUT”) to tokens

reset tok

elif tok equals to “if” or “IF”:

append (“IF”) to tokens

reset tok

elif tok equals to “for” or “FOR”:

append (“FOR”) to tokens

reset tok

elif tok equals to any digit 0.9:

add tok to expr

reset tok

elif tok equals to “+” or “-“ or “\*” or “/” or “(“ or “)”:

isexpr = true

add tok to expr

reset tok

elif tok equals to tab:

reset tok

elif tok equals to (“):

if state is false:

state = true

elif state is true:

append (“STRING:” + string + “) to tokens

reset string

state = false

reset tok

elif state is true:

add tok to string

reset tok

return tokens

# Parsing and semantic analysis

I will first define a series of functions that will help me output the right result and then I will write the parse function. Both parts will be written in pseudo code.

function evalExpression:

input: expression expr

output: evaluated expression

return evaluate(expr)

function doPrint:

input: a string, number or expression “toPrint”

output: N/A

if the first 6 characters of toPrint is “STRING”:

toPrint = all characters of toPrint starting with the 8th character

toPrint = all characters of toPrint except the last one

if the first 3 characters of toPrint is “NUM”:

toPrint = all characters of toPrint starting with the 4th character

if the first 4 characters of toPrint is “EXPR”:

toPrint = all characters of toPrint starting with the 4th character

function doAssign:

input: 2 strings varname and varvalue

output: N/A

assign varvalue to the varname element of the symbols dictionary

function getVariable:

input: a string varname

output: element varname of symbols dictionary or error message

assign all characters of varname starting with the 4th character to varname

if varname can be found in the symbols dictionary:

return the varname element of symbols

else:

return error message

exit the program

function getInput:

input: 2 strings called string and varname

output: N/A

assign to i the return of the user input, while showing to the screen the string message

assign (“STRING:” + i + “) to the varname element of symbols

function parse:

input: list toks

output: N/A

i = 0

ifc = false

while i is less than the length of the toks array:

if element i of toks equals to “FOR”:

increment i by 1

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 elements of the i+2 element of toks equals to “NUM TO NUM”:

assign all characters starting with the 4th of the i element of toks to a

assign all characters starting with the 4th of the i+2 element of toks to b

elif the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 elements of the i+2 element of toks equals to “VAR TO VAR”:

assign the return of the call getVariable with the parameter i element of tok to a

assign the return of the call getVariable with the parameter i+2 element of tok to b

elif the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 elements of the i+2 element of toks equals to “NUM TO VAR”:

assign all characters starting with the 4th of the i element of toks to a

assign the return of the call getVariable with the parameter i+2 element of tok to b

elif the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 elements of the i+2 element of toks equals to “VAR TO NUM”:

assign the return of the call getVariable with the parameter i element of tok to a

assign all characters starting with the 4th of the i+2 element of toks to b

else:

print error

exit

increment i by 3

increment b by 1

if i element of tok + the first 6 characters of the i+1 element of toks equals to “PRINT STRING” or i element of tok + the first 4 characters of the i+1 element of toks equals to “PRINT EXPR” or i element of tok + the first 3 characters of the i+1 element of toks equals to “PRINT NUM” or i element of tok + the first 3 characters of the i+1 element of toks equals to “PRINT VAR”:

for x between a and b:

if the first 6 characters of the i+1 element of toks equals to “STRING” or the first 3 characters of the i+1 element of toks equals to “NUM” or the first 4 characters of the i+1 element of toks equals to “EXPR”:

call doPrint with parameter element i+1 of toks

elif the first 3 characters of the i+1 element of toks equals to “VAR”:

call do print with parameter call getVariable with parameter element i+1 of toks

increment i by 2

if length of array toks equals to i:

exit

if element i of toks equals to “IF”:

ifc = true

increment i by 1

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “NUM EQEQ NUM”:

if the i element of toks starting with the 4th character equals to the i+2 element of toks starting with the 4th character:

ifc = false

if the i element of toks starting with the 4th character not equals to the i+2 element of toks starting with the 4th character:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “VAR EQEQ VAR”:

if the return of the call getVariable with parameter i element of toks equals to the return of call getVariable with parameter i+2 element of toks:

ifc = false

if the return of the call getVariable with parameter i element of toks not equals to the return of call getVariable with parameter i+2 element of toks:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “NUM LESS NUM”:

if the i element of toks starting with the 4th character is less than the i+2 element of toks starting with the 4th character:

ifc = false

if the i element of toks starting with the 4th character is more than the i+2 element of toks starting with the 4th character:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “VAR LESS VAR”:

if the return of the call getVariable with parameter i element of toks is less than the return of call getVariable with parameter i+2 element of toks:

ifc = false

if the return of the call getVariable with parameter i element of toks is more than the return of call getVariable with parameter i+2 element of toks:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “NUM MORE NUM”:

if the i element of toks starting with the 4th character is more than the i+2 element of toks starting with the 4th character:

ifc = false

if the i element of toks starting with the 4th character is less than the i+2 element of toks starting with the 4th character:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “VAR MORE VAR”:

if the return of the call getVariable with parameter i element of toks is more than the return of call getVariable with parameter i+2 element of toks:

ifc = false

if the return of the call getVariable with parameter i element of toks is less than the return of call getVariable with parameter i+2 element of toks:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “NUM LESSEQUAL NUM”:

if the i element of toks starting with the 4th character is less or equal than the i+2 element of toks starting with the 4th character:

ifc = false

if the i element of toks starting with the 4th character is more than the i+2 element of toks starting with the 4th character:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “VAR LESSEQUAL VAR”:

if the return of the call getVariable with parameter i element of toks is less or equal than the return of call getVariable with parameter i+2 element of toks:

ifc = false

if the return of the call getVariable with parameter i element of toks is more than the return of call getVariable with parameter i+2 element of toks:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “NUM MOREEQUAL NUM”:

if the i element of toks starting with the 4th character is more or equal than the i+2 element of toks starting with the 4th character:

ifc = false

if the i element of toks starting with the 4th character is less than the i+2 element of toks starting with the 4th character:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “VAR MOREEQUAL VAR”:

if the return of the call getVariable with parameter i element of toks is more or equal than the return of call getVariable with parameter i+2 element of toks:

ifc = false

if the return of the call getVariable with parameter i element of toks is less than the return of call getVariable with parameter i+2 element of toks:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “NUM NOTEQUAL NUM”:

if the i element of toks starting with the 4th character not equals to the i+2 element of toks starting with the 4th character:

ifc = false

if the i element of toks starting with the 4th character equals to the i+2 element of toks starting with the 4th character:

ifc = true

if the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “VAR NOTEQUAL VAR”:

if the return of the call getVariable with parameter i element of toks not equals to the return of call getVariable with parameter i+2 element of toks:

ifc = false

if the return of the call getVariable with parameter i element of toks equals to the return of call getVariable with parameter i+2 element of toks:

ifc = true

increment i by 3

if ifc is false:

if i element of tok + the first 6 characters of the i+1 element of toks equals to “PRINT STRING” or i element of tok + the first 4 characters of the i+1 element of toks equals to “PRINT EXPR” or i element of tok + the first 3 characters of the i+1 element of toks equals to “PRINT NUM” or i element of tok + the first 3 characters of the i+1 element of toks equals to “PRINT VAR”:

if the first 6 characters of the i+1 element of toks equals to “STRING” or the first 3 characters of the i+1 element of toks equals to “NUM” or the first 4 characters of the i+1 element of toks equals to “EXPR”:

call doPrint with parameter element i+1 of toks

elif the first 3 characters of the i+1 element of toks equals to “VAR”:

call do print with parameter call getVariable with parameter element i+1 of toks

increment i by 2

elif the first 3 characters of the i element of toks + the i+1 element of toks + the first 6 characters of the i+2 element of toks equals to “VAR EQUALS STRING” or the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “VAR EQUALS NUM” or the first 3 characters of the i element of toks + the i+1 element of toks + the first 4 characters of the i+2 element of toks equals to “VAR EQUALS EXPR” or the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “VAR EQUALS VAR”:

if the first 6 characters of the i+2 element of toks equals to “STRING”:

call doAssign with parameters the i element of toks and the i+2 element of toks

if the first 3 characters of the i+2 element of toks equals to “NUM”:

call doAssign with parameters the i element of toks and the i+2 element of toks

if the first 4 characters of the i+2 element of toks equals to “EXPR”:

call doAssign with parameters the i element of toks and the return of the call evalExpression with parameter the characters of the i+2 element of toks starting with the 5th character

if the first 3 characters of the i+2 element of toks equals to “VAR”:

call doAssign with parameters the i element of i and the return of the call getVariable with parameter the i+2 element of toks

increment i by 3

elif the i element of toks + the first 6 characters of the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “INPUT STRING VAR”:

call getInput with parameters the characters of the i+1 element of toks starting with the 7th character and the characters of the i+2 element of toks starting with the 4th character

increment i by 3

if ifc equals true:

if i element of tok + the first 6 characters of the i+1 element of toks equals to “PRINT STRING” or i element of tok + the first 4 characters of the i+1 element of toks equals to “PRINT EXPR” or i element of tok + the first 3 characters of the i+1 element of toks equals to “PRINT NUM” or i element of tok + the first 3 characters of the i+1 element of toks equals to “PRINT VAR”:

increment i by 2

elif the first 3 characters of the i element of toks + the i+1 element of toks + the first 6 characters of the i+2 element of toks equals to “VAR EQUALS STRING” or the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “VAR EQUALS NUM” or the first 3 characters of the i element of toks + the i+1 element of toks + the first 4 characters of the i+2 element of toks equals to “VAR EQUALS EXPR” or the first 3 characters of the i element of toks + the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “VAR EQUALS VAR”:

increment i by 3

elif the i element of toks + the first 6 characters of the i+1 element of toks + the first 3 characters of the i+2 element of toks equals to “INPUT STRING VAR”:

increment i by 3

# The method of evaluating the improvement of knowledge in class

This task has been divided into a lesson plan. It started with an oral quiz, making the students respond to some questions, trying to test their knowledge of SCRATCH2, in regards to arithmetical operations. They were simple questions that the students managed to find answers for in a very short time. Followed by that, I delivered a lesson about Prog, and how the basic concepts of it work. Then I designed a test that the students have to take, that has been assessed in pairs (each student marked the other student’s paper), followed by my moderation of the marking done by sampling. Then I designed a questionnaire, illustrating the advantages and disadvantages of using Prog and SCRATCH2 in class. All of these results will be discussed in the next section.

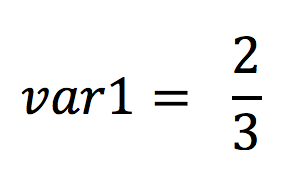
I had to do this twice because the first time, the results that I obtained could be considered a bit biased, according to the feedback that I have got for my presentation. You can find the questionnaire and the results in Appendix 1.

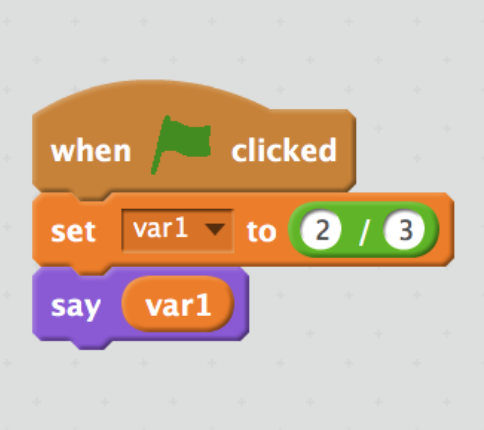
Appendixes

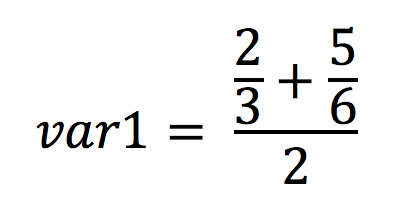
# Appendix 1

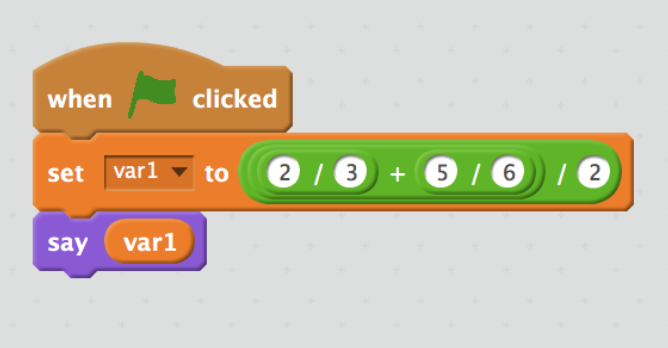
Questionnaire :

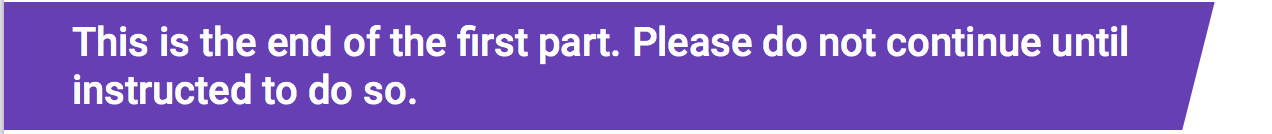
# ../Screen%20Shot%202016-04-21%20at%2018.47.52.png

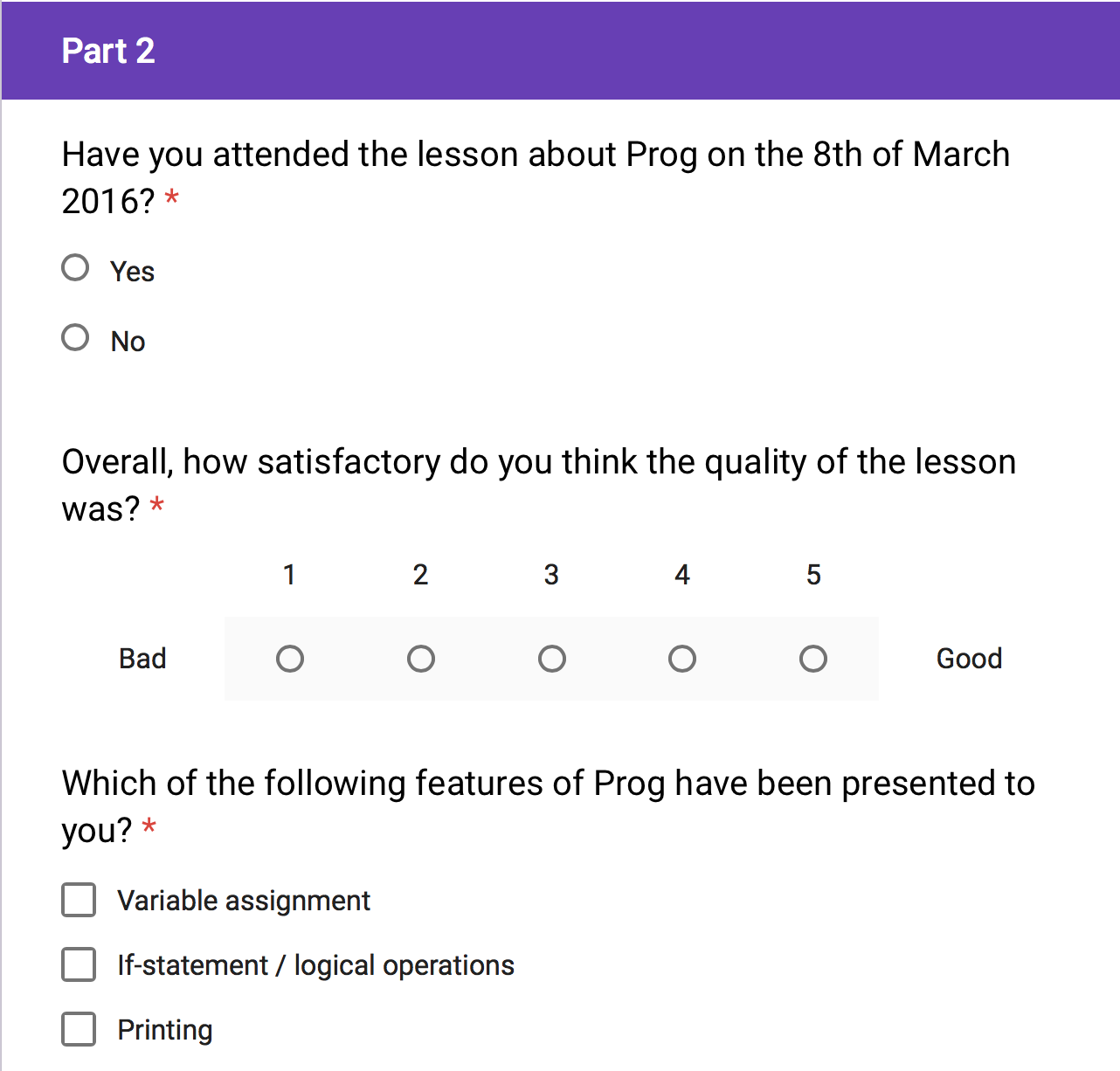


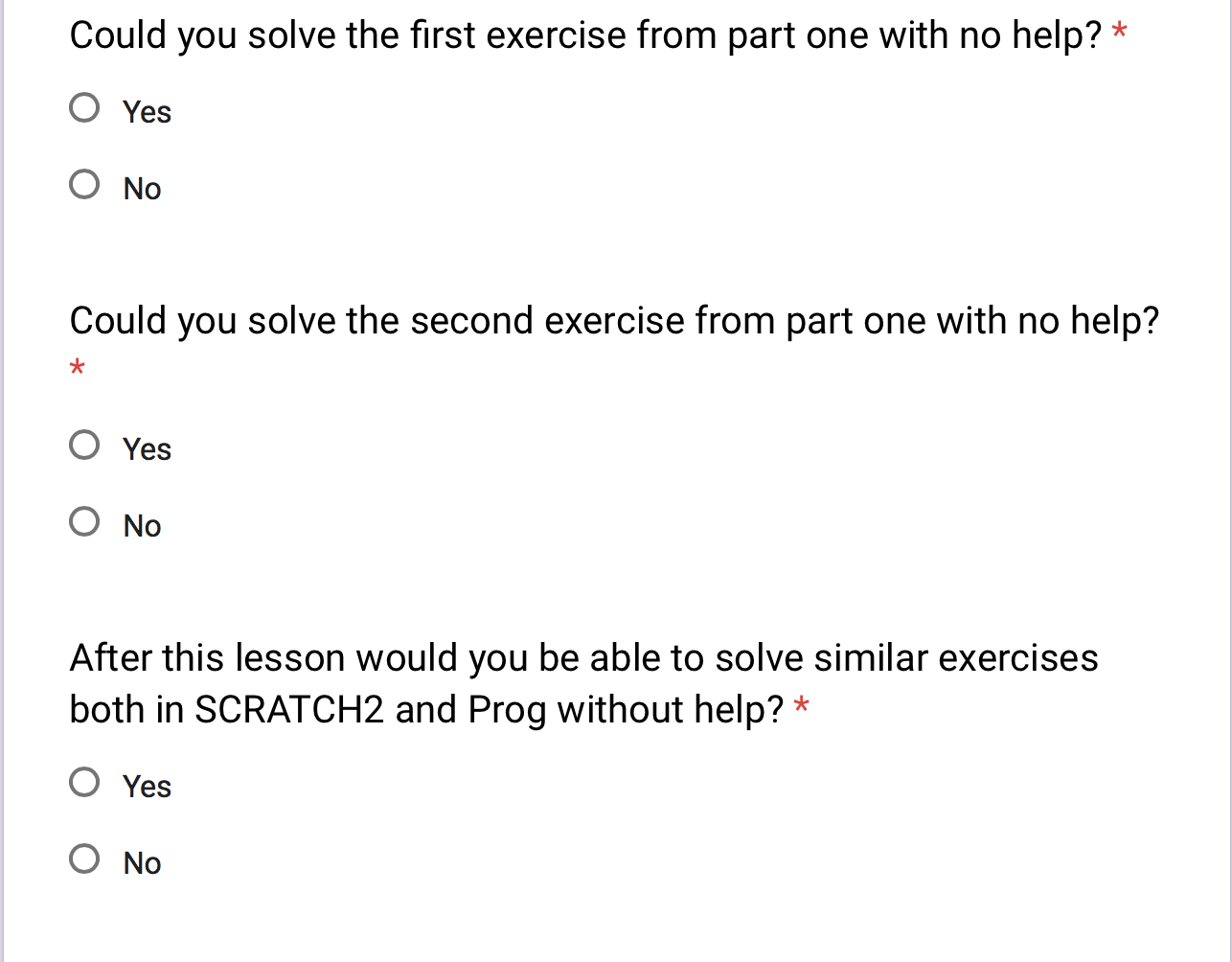


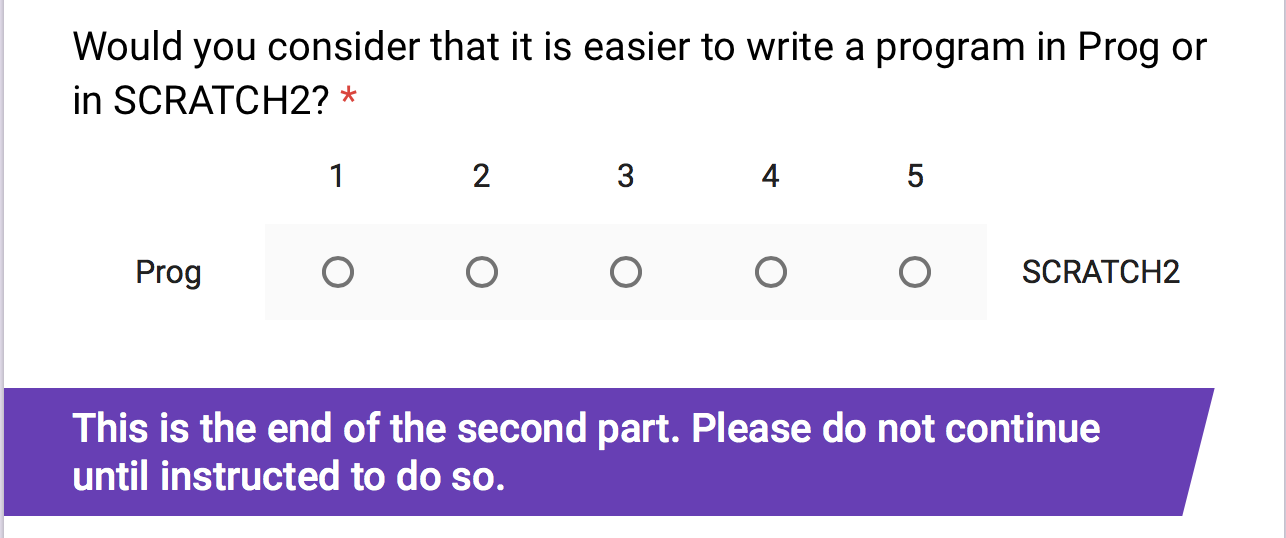


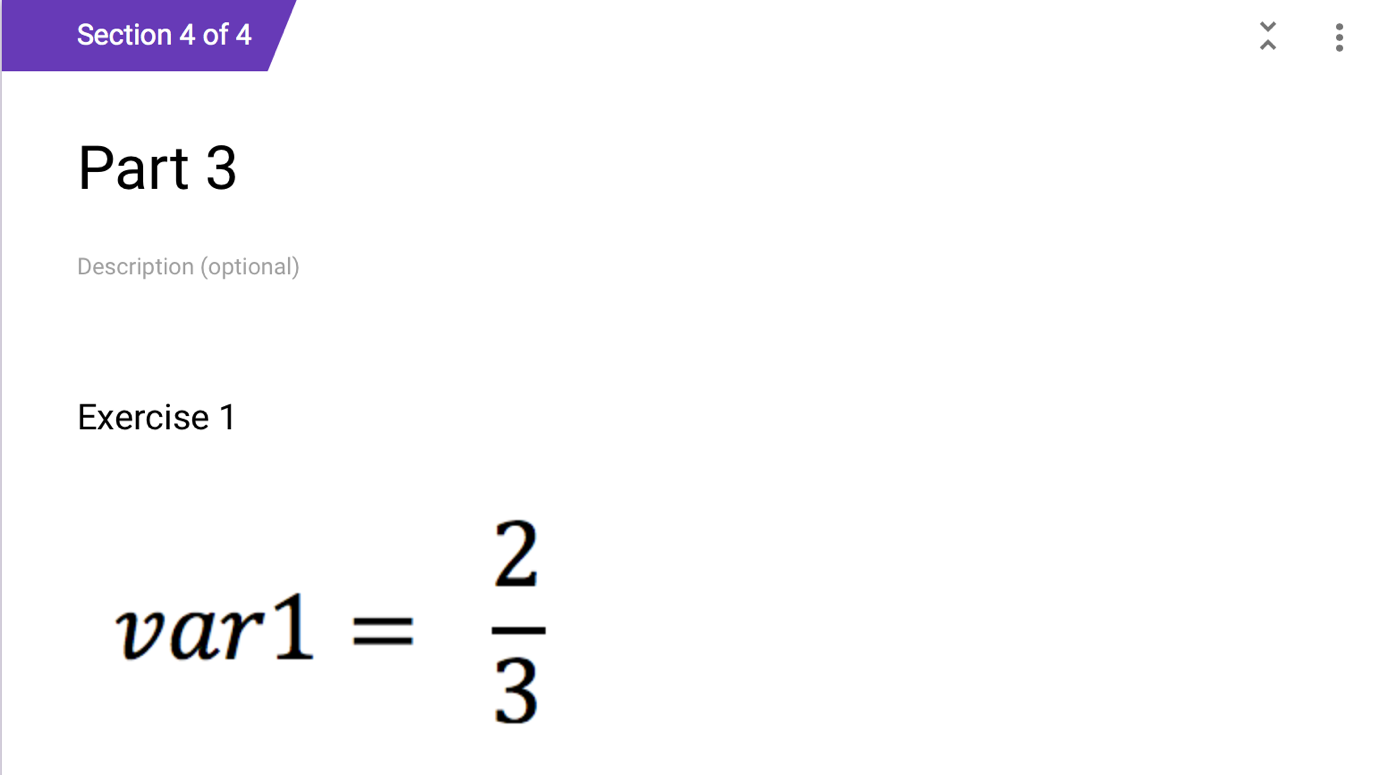


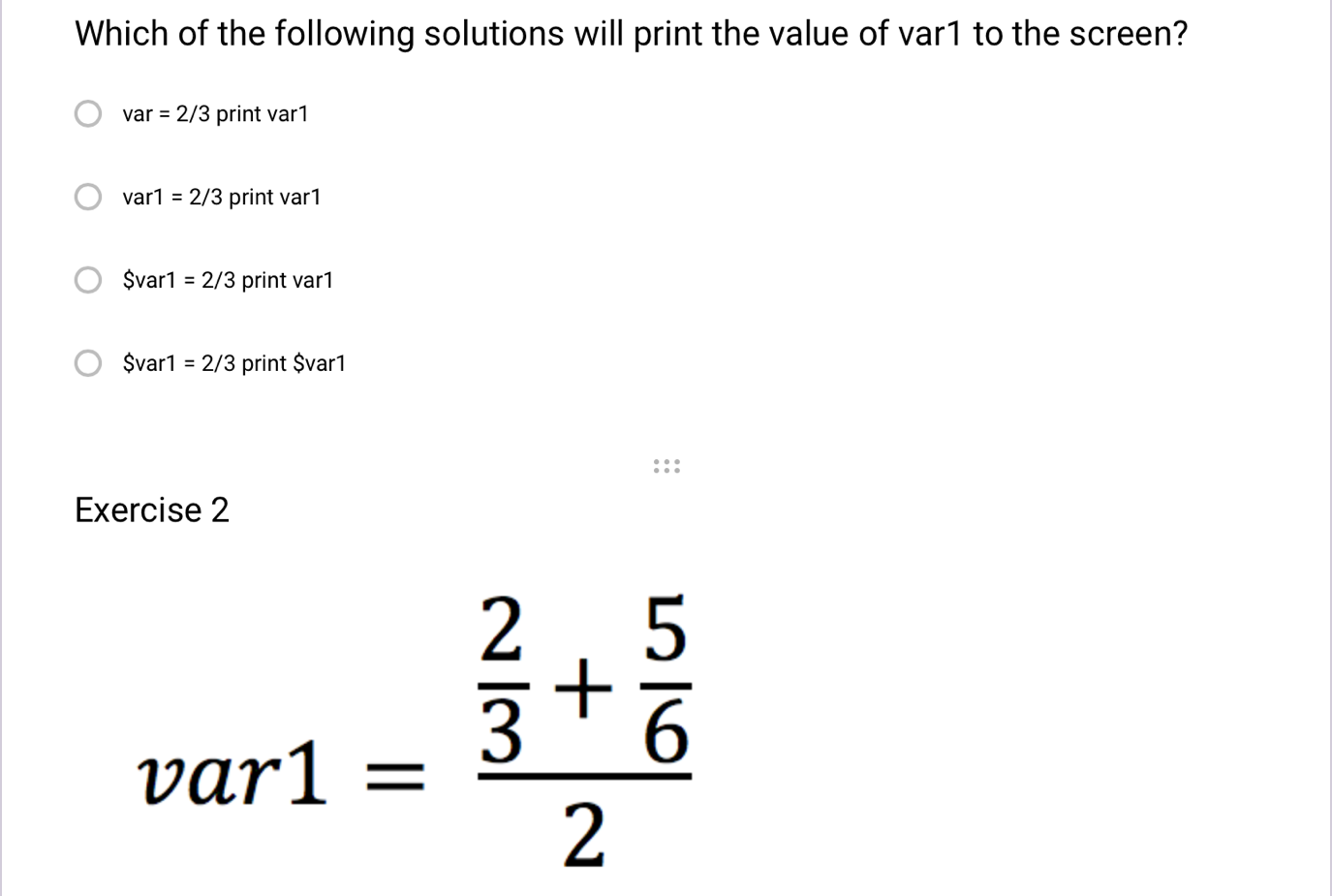


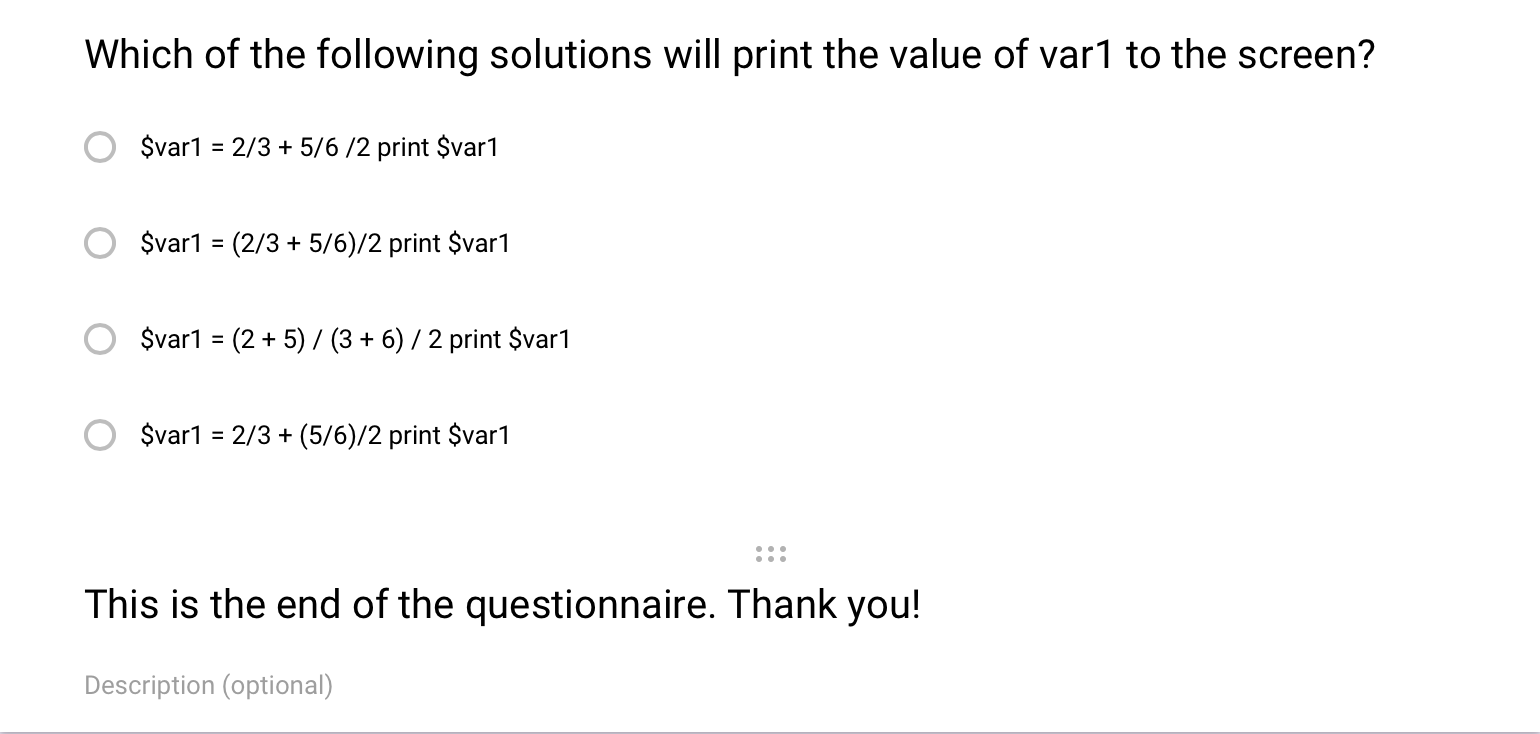












Results:

