MATHEMATICAL PROBLEM SOLVING: ONE WAY TO PREVENT DEMENTIA

Jeremias Ivan¹, Rizky Nurdiansyah*¹, Arli A. Parikesit¹

¹ Department of Bioinformatics, School of Life Sciences, Indonesia International Institute for Life Sciences, Jakarta, Indonesia.

²

Article Info

Article type:
Research

Article History:

Received: 2019-01-03

Revised: 2019-01-27

Accepted: 2019-02-04

*** Corresponding author:**

Rizky Nurdiansyah

Department of Bioinformatics,

School of Life Sciences, Indonesia

International Institute for Life

Sciences, Jakarta, Indonesia.

E-mail: rizky.nurdiansyah@i3l.ac.id

ABSTRACT

Introduction:

Dementia is a common medical condition of older people which is marked by the decline of multiple cognitive abilities, such as memory and communication. Currently, there is no effective treatment for curing dementia, making prevention the most priority to this disease. Previous studies showed that cognitive ability training, such as mathematical problem solving, has a potential to slow down cognitive decline. The aim of this project is to create a simple yet interactive mathematical quiz as a way to train one's cognitive ability and reduce the risk of getting dementia.

Material and Methods:

The quiz was created by using tkinter module and its built-in functions in Python programming language.

Results:

The result showed that the quiz was simple but involved an active role of the user to input the answer. It also did not have certain time limit, preventing the user to feel rushed or burdened in doing the quiz. In addition, three different types of difficulty were provided to give a challenging atmosphere to the game.

Conclusion:

As a conclusion, this quiz provides a simple way for people to regularly train their cognitive skill, so the risk of getting dementia, especially in elderly stage, can be reduced.

Keywords:

Dementia, Prevention, Python, Tkinter

How to cite this paper

Ivan J, Nurdiansyah R, Parikesit AA. Mathematical Problem Solving: One Way to Prevent Dementia. Front Health Inform. 2019; 8(1): e10. DOI: [10.30699/fhi.v8i1.179](https://doi.org/10.30699/fhi.v8i1.179)

INTRODUCTION

Dementia is a general term of syndrome that is characterized by progressive deterioration in multiple cognitive domains which is severe enough to reduce someone's ability to perform daily activities [1]. It is commonly caused by Alzheimer's disease (AD), named as AD dementia [2]. According to Prince, Bryce, Albanese, Wimo, Ribeiro, and Ferri [3], the prevalence of dementia increases in parallel with age: it doubles every 5.5 – 6.7 years, depending on the regions. In 2010, there were 35.6 million people who lived with dementia worldwide, estimated to reach 65.7 million by 2030 and 115.4 million by 2050 [4]. Furthermore, Alzheimer's disease International [5] estimated that there were 1.2 million people in Indonesia who lived with dementia in 2015, placing Indonesia in the top ten country with the highest

dementia prevalence worldwide. As dementia is a common medical condition, especially in elderly population, studies about prevention and treatment of dementia are essential to be conducted.

Currently, there is no effective treatment for curing dementia. However, it was found that dementia has long asymptomatic latent period [6], meaning that prevention is the most feasible solution to this disease. Based on the previous studies, it was found that increased involvement in cognitive activities may slow down cognitive decline and reduce the risk for mild cognitive impairment [7]. In fact, there is a lot of cognitive activities that are easy to be conducted, such as solving simple mathematical problems. In this activity, the participant needs to solve several mathematical equations, such as addition, subtraction, and multiplication, between two numbers. In order to be interesting, the problems should be presented in a simple yet interactive way. In this case, tkinter module, which

is provided in Python programming language, becomes one of the best choices.

Tkinter is a standard Python interface to the Tk GUI (Graphical User Interface) toolkit [8]. In Python, tkinter module is not initially provided in the Python directory; it must be installed first before the user can use it. The module provides several functions for creating a simple GUI, such as button and entry. This project aimed to create a simple yet interactive GUI in Python programming language by using tkinter module to engage cognitive ability and reduce the risk of getting dementia. The program would give certain mathematical equation that must be answered by the user. If the given answer was right, the user could move to the next question; if not, he/she would stay in the same problem until the right answer was given. Then, the number of right and wrong answers and the calculation time of the user would be recorded at the very end of every round. As a result, cognitive activities can be done in interesting way without developing boredom or stress to the participants.

MATERIAL AND METHODS

The quiz was created by using tkinter module in Python programming language v. 3.6. In simple terms, there are three main procedures of the code: NUMBER, EQUATION, and MAIN. Each of them is used to generate a mathematical equation, calculate the right value of the equation, and run the quiz, respectively. When the MAIN procedure is executed, the program will ask the user to choose the difficulty. Then, the difficulty would be inputted into NUMBER procedure to determine the range of numbers of the equation: 0 – 10 (easy), 11 – 20 (medium), or 21 – 30 (hard). Then, the respective equation would automatically be calculated by the program to determine the true answer of the equation via EQUATION procedure. After that, the program will show the equation on the screen; the user should input the answer by clicking on the numerical buttons. The program will match the user's answer with the true answer. If the answer matches, the program will generate the next equation, together by adding the number of iteration and true answer by one. If the answer is wrong, the program will not move to the next equation; instead, it will add the number of wrong answers by one and ask for another answer. After the number of iterations reaches five, the program would stop generating equation and show the number of right and wrong answers, in addition to the user's calculation time. The pseudo code is shown below.

```
#set the initial number of repetition, right,
and wrong answers
Set i to be 1
Set right, wrong to be 0

#create procedure to generate the equation
procedure NUMBER(difficulty)
```

```
#randomize the mathematical equation
Set symbol by random between [plus, minus,
multiply]

If difficulty is easy then
    #randomize the numbers between
    1 until 10
    Set NUMBER.N1 by random
    between (1 to 10)
    Set NUMBER.N2 by random
    between (1 to 10)

If difficulty is medium then
    #randomize the numbers between
    11 until 20
    Set NUMBER.N1 by random
    between (11 to 20)
    Set NUMBER.N2 by random
    between (11 to 20)

If difficulty is hard then
    #randomize the numbers between
    21 until 30
    Set NUMBER.N1 by random
    between (21 to 30)
    Set NUMBER.N2 by random
    between (21 to 30)

End if
End procedure

#create procedure to calculate the equation
procedure EQUATION()

If symbol is plus then
    Set EQUATION.eq to be
    NUMBER.N1 + NUMBER.N2

If symbol is minus then
    Set EQUATION.eq to be
    NUMBER.N1 - NUMBER.N2

If symbol is multiply then
    Set EQUATION.eq to be
    NUMBER.N1 * NUMBER.N2

End if
End procedure

#create the main procedure of the quiz
procedure MAIN()

#ask the user to choose the difficulty
Set difficulty from difficulty buttons
Start stopwatch
    #start the stopwatch
Run NUMBER(difficulty) #generate 1st
equation
Run EQUATION()
    #calculate 1st equation
Print EQUATION.eq
    #print the equation
#create iteration to generate five
questions
While i is less than 6
    #ask the user to input the answer
    Set userInput from numerical buttons

#create condition if the answer is right
If userInput is the same with EQUATION.eq
    #add the number of iteration by one
    Add i by 1
    #add the number of right answer by one
    Add right by 1
```

```

Run NUMBER()
    #generate next equation
Run EQUATION()
    #calculate next equation
    Print EQUATION.eq
    #print the equation

#create condition if the answer is wrong
Else
    #add the number of wrong answer by one
    Add wrong by 1
End if

Stop stopwatch      #stop the stopwatch
Print right         #print number
of right answers
Print wrong         #print number
of wrong answers
Print stopwatch     #print the calculation
time
End procedure

```

In the Python code, there were five functions that were created to support the GUI (other than the constructor): *level* (determines the difficulty of the quiz), *initnumber* (generates the first equation), *randomnumber* (generates the second until fifth equations), *validate* (updates the inputted value when minus and numerical buttons were clicked), and *update* (submits, deletes, or resets the inputted value when “Enter”, “Del”, and “Reset” buttons are clicked, respectively). As described in the pseudo code, the user first needs to choose the difficulty of the quiz. By clicking one of the difficulty buttons, the program would call *level* function to print the difficulty and set *self.h* variable into “easy”, “medium”, or “hard”. The value of *self.h* would be used as a basis to generate the problems when “Start” button is clicked. When it is clicked, the program would execute *initnumber* function, generating a mathematical equation between two numbers in the ranges of 0 – 10 (easy), 11 – 20 (medium), or 21 – 30 (hard). The user has to answer the problem correctly to move into the next problem. This is achieved by clicking on the number buttons in the GUI (keyboard input is not supported). The answer of the user would be recorded and validated via *validate* and *update* functions, respectively, where correct answer leads to the generation of the next question via *randomnumber* function. In addition, after each correct answer, the value of *self.x* would be added by one. When *self.x* value reaches five (i.e. after five problems are answered correctly), the program would show the numbers of correct and wrong answers that are inputted by the user and his/her calculation time (i.e. the time needed for the user to correctly answer all five problems). In addition, the program provides “Reset” button that can be used to reset the session (i.e. clear the equation and numbers of right and wrong answers). The complete flowchart of the program is shown in Fig. 1.

As it clearly seen in Fig 1, it contains different parts as

followings:

- ① The user needed to choose the quiz’s difficulty.
- ② After START button was clicked, the program would set the value of *self.x* to be zero.
- ③ The program would check the current value of *self.x*, whether it was smaller than five or not.
- ④ If the value of *self.x* was smaller than five, the program would generate a new problem, because the number of iteration had not reached the limit (maximum 5 problems in every round).
- ⑤ The user was asked to input the answer to the given problem.
- ⑥ The program would check the inputted answer, whether it was true or not.
- ⑦ If the answer was false, the user would be asked to input another answer.
- ⑧ If the answer was true, the program would add the value of *self.x* by one and
- ⑨ Return to the third step.
- ⑩ If the value of *self.x* was equal to or greater than five, the program would not generate a new problem; instead, it would show the calculation time
- ⑪ and the total amount of right and wrong answers.
- ⑫ The respective round was finished, because the number of iteration had reached the limit.

RESULTS

This project aimed to prevent dementia by training the cognitive skill. According to Foster [9], problem solving in mathematics was correlated with higher cognitive skills, including critical thinking and sound reasoning. It meant that simple mathematical quiz could be used to train our cognitive ability. One real-life example that shows the importance of doing mathematical problems was experienced by the author’s grandmother. When she was 80 years old, she got dementia; she easily forgot the name of other people, including her grandchildren. However, she was still able to do simple mathematical equations, such as multiplication between numbers from zero until ten, even though not all of them were correct. This proves that simple mathematical problem solving was relevant to be done by all people, including the elders.

In the quiz, there were three levels of difficult that were provided: easy, medium, and hard. Each difficulty was differentiated by the range of numbers that the program generated. The number ranged from zero to ten in easy difficulty, eleven to twenty in medium difficulty, and twenty-one to thirty in hard difficulty. These difficulty options were provided based on the assumption that different person had

different ability to solve mathematical equation. Therefore, the users could match the difficulty level based on their abilities and challenge themselves in higher difficulty level.

Fig 1: Flowchart of the GUI

In order to attract people, including the one who does not like math, the quiz should not be boring to start with. Therefore, it was set to be simple but interactive (Fig 2). It was simple because there was no excessive decoration, text, or color. On the other hand, it was interactive as the users were asked to type their own answer to the random mathematical equations by clicking on the numerical buttons. There were only five questions for each round without maximum time limit (Fig 3). As a result, the user might freely do the quiz in their free time; they would not be burdened nor stressed.

Fig 2: GUI Output

The general output consisted of four main parts: the upper-left column, upper-middle column, upper-right column, and bottom column. The upper-left column showed the introduction to the quiz and choices of difficulty. The upper-middle column showed the mathematical problem, number of right answers, and number of wrong answers during the quiz. The upper-right column provided numerical and operator buttons for the user to input the answer. The bottom column integrated specific output when certain action was conducted, e.g. when the "?" button was clicked.

Fig 3: Example of the Quiz (Easy Difficulty)

After the START button was clicked, the mathematical problem would appear on the upper-middle column. The difficulty level was shown on the bottom column. The clicked number would appear in the white box on the upper-right column.

Fig 4: Example Output of the Result

After the round was over, the total number of right and wrong answers were shown on the upper-middle column. The calculation time of the respective round was shown on the bottom column.

After every round, the calculation time of the user would be shown, together with the final amount of right and wrong answers (Fig 4). The calculation time was shown to motivate the user to beat his/her own high score, thus improving their cognitive ability. On the other hand, the amounts of right and wrong answers were initially shown as an indicator when the user inputted the wrong answer. These data could be used to compare the result between healthy people and those who suffered from dementia in further study.

CONCLUSION

This mathematical quiz provides a simple way for people to regularly train their cognitive skill, so the risk of getting dementia, especially in elderly stage, can be reduced. However, the significance level of the quiz to the development of dementia has not yet been studied. Therefore, the correlation between dementia development and the intensity of doing the quiz could not be determined.

ACKNOWLEDGMENT

The authors would like to thank Research and Community Engagements Institute (LPPM) of i3L for their heartfelt supports.

AUTHOR'S CONTRIBUTION

All the authors approved the final version of the manuscript.

CONFLICTS OF INTEREST

The authors declare no conflicts of interest regarding the publication of this study.

REFERENCES

1. Mangialasche F, Kivipelto M, Solomon A, Fratiglioni L. Dementia prevention: Current epidemiological evidence and future perspective. *Alzheimers Res Ther.* 2012; 4(1): 6-14. PMID: 22339927 DOI: 10.1186/alzrt104 [PubMed]
2. McKhann GM, Knopman DS, Chertkow H, Hyman BT, Jack CR, Kawas CH, et al. The diagnosis of dementia due to alzheimer's disease: Recommendations from the national institute on aging-alzheimer's association workgroups on diagnostic guidelines for alzheimer's disease. *Alzheimers Dement.* 2011; 7(3): 263-9. PMID: 21514250 DOI: 10.1016/j.jalz.2011.03.005 [PubMed]
3. Prince M, Bryce R, Albanese E, Wimo A, Ribeiro W, Ferri CP. The global prevalence of dementia: A systematic review and metaanalysis. *Alzheimers Dement.* 2013; 9(1): 63-75. PMID: 23305823 DOI: 10.1016/j.jalz.2012.11.007 [PubMed]
4. Wimo A, Prince M. World alzheimer Report 2010: The global economic impact of dementia. Alzheimer's Disease International. London, UK; 2010.
5. Prince M, Wimo A, Guerchet M, Ali G, Wu T, Prina M, et al. World alzheimer report 2015: The global impact of dementia. Alzheimer's Disease International. London, UK; 2015.
6. Sperling RA, Aisen PS, Beckett LA, Bennett DA, Craft S, Fagan AM, et al. Toward defining the preclinical stages of alzheimer's disease: Recommendations from the national institute on aging-alzheimer's association workgroups on diagnostic guidelines for alzheimer's disease. *Alzheimers Dement.* 2011; 7(3): 280-92. PMID: 21514248 DOI: 10.1016/j.jalz.2011.03.003 [PubMed]
7. Daviglus ML, Bell CC, Berrettini W, Bowen PE, Connolly ES, Cox NJ, et al. NIH state of the science conference statement: Preventing alzheimer disease and cognitive decline. *NIH Consens State Sci Statements.* 2010; 27(4): 1-30. PMID: 20445638 [PubMed]
8. Python Software Foundation. Tkinter; Python interface to Tcl/Tk [Internet]. 2019. [Cited: 19 Apr 2019; Updated: 13 May 2019]. Python 3.6.8 documentation 2019. Available from: <https://docs.python.org/3.6/library/tkinter.html>.
9. Foster A. Cognitive skills and mathematics problem-solving performance. Georgia Educational Research Association Conference; 2014. Available from: <https://digitalcommons.georgiasouthern.edu/cgi/viewcontent.cgi?article=1055&context=gera>