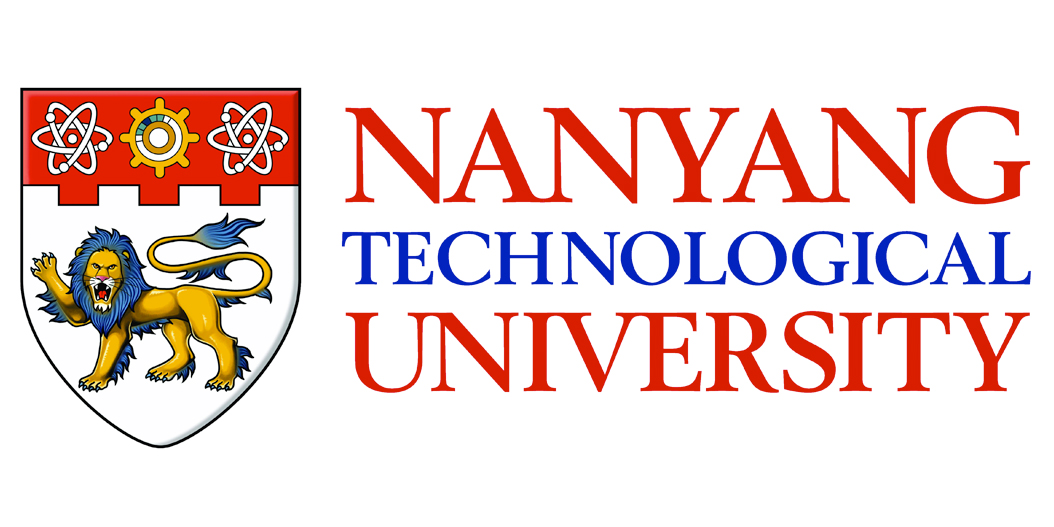
****

**MH8811 Python Programming**

**MSc in Financial Technology**

**School of Physical and Mathematical Science**

**Nanyang Technological University**

**Project Report Title:**

**Retirement Age Calculator for Pre-University Student based on Starting Salary of NTU course**

**Prepared by:**

**Group 5**

|  |  |
| --- | --- |
| **Marcus Chan Hua Rui** | **K2020190E** |
| **Ng Wan Jing** | **G2003003C** |
| **Tan Yan Ming** | **G2002094A** |
| **Wei Kai** | **G2001793D** |

**Number of words: 1196**

**exclude table of contents, flowchart, pseudo code, graphics, titles, appendix**

Percentage of contribution by Each Member

|  |  |  |
| --- | --- | --- |
| Member Name | % of contribution | Justification |
| Marcus Chan Hua Rui | 25 | * Came up with the initial outline and template for the code. * Combined the code from the various methods and functions together. * Helped in the debugging of the program to ensure that the program works as expected. * Wrote the calculations to calculate the retirement age and accumulated savings. |
| Ng Wan Jing | 25 | * Search for course and salary and clean up data * Dictionary for course salary, include years of study in calculation, add in assumptions in interface codes * Edit labels/comments for readability, error handling (prevent user from typing manually) with messages, formatting/add colors in interface, and consolidate/clean codes * Powerpoint on demo, improvement, conclusion and consolidate ppt * Flowchart & Pseudo Code in report * Wrote on program testing/output/improvements/conclusion & editing/formatting/add in graphics with captions in report with consolidation |
| Tan Yan Ming | 25 | * Came up with a list of interest and match the interest with the course * Create the button to get salary from the course selected * Powerpoint on Assumptions, Data Preparation and Overview of Application * Report on assumptions, reflection and challenges |
| Wei Kai | 25 | * Design the logic to choose interests and courses. * Wrote a input-checking mechanism. If the input is empty or incorrect, the program will remind the user and input again. * Adjusted the label names, instructions, and type setting of the interface to make it more user friendly. * Powerpoint on algorithm flowchart. * Algorithm design in report. |

Table of Contents

[1. Introduction 4](#_Toc50915574)

[2. Idea Concept 4](#_Toc50915575)

[3. Assumptions 5](#_Toc50915576)

[4. Algorithm design 6](#_Toc50915577)

[4.1 Flowchart 7](#_Toc50915578)

[4.2 Pseudo Code 8](#_Toc50915579)

[4.3 Output of Application 9](#_Toc50915580)

[5. Program Testing 10](#_Toc50915581)

[5.1 In our GUI 10](#_Toc50915582)

[5.2 Error message 11](#_Toc50915583)

[6. Reflection 12](#_Toc50915584)

[6.1 Challenges 12](#_Toc50915585)

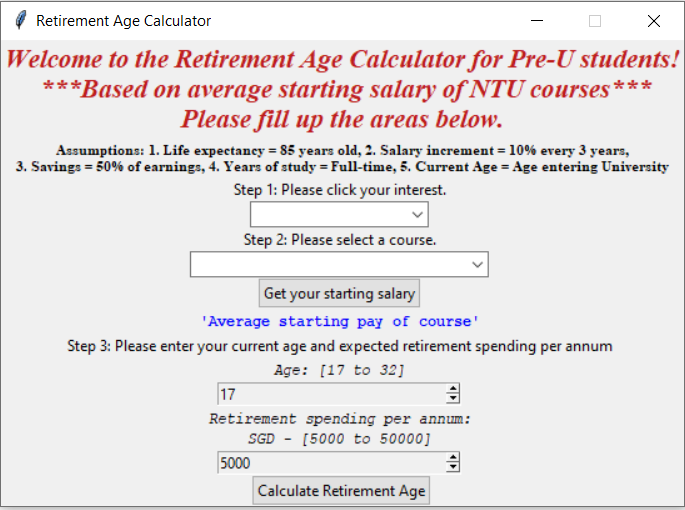
[6.2 Improvements 12](#_Toc50915586)

[7. Conclusion 12](#_Toc50915587)

[8. Appendix 13](#_Toc50915588)

# Introduction

In this project, we would like to make a retirement age calculator based on the users’ inputs. This calculator is aimed at prospective NTU students, helping them to make the huge financially significant choice; choosing a university course. This calculator is designed to take just 3 steps to output the user’s estimated retirement age and savings at the point of retirement. Firstly, it takes in the user’s interests, secondly the course of interest, and finally the age and expected spending per annum.



Step 1: Interest

Step 2: Course

Step 3: Age and Expected retirement spending per annum

Figure 1: Steps in GUI interface

# Idea Concept

For the idea concept for this project, we wanted to come up with an idea that would help prospective NTU students in making their choice for their university course. Many students take loans to pay off their university fees, and this is a cause for concern for many students. Many may also choose to undertake university courses for a higher starting salary then they would have if they had not done so. With that in mind, we wanted to come up with a retirement age calculator that would help them gauge their retirement age.

Granted, students cannot solely make their decision based on their starting salary but should choose a course based on their interests. Therefore, we decided to include an option that would help these students filter their results according to their interests.

We wanted to make a calculator that was easy to use and would only take a few seconds of the users’ time so that they could spend more time playing around with the variations rather than spending a lot of time filling in entries, hence, we decided to make a few assumptions that will be listed in a later section. This makes our calculator lightweight, easy to use, and very understandable.

# Assumptions

There are 5 assumptions that we took for own calculation. They are as follows:

1. Life expectancy = 85 years old
2. Salary increment of 10% every 3 years
3. Spent 50% of earnings and save 50% of it
4. The user is a Full-time student
5. Starting age of user = current age of user

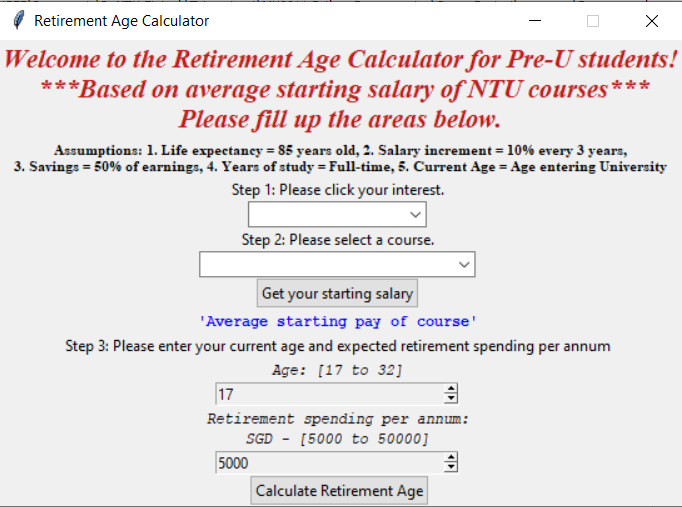


Figure 2: Assumptions listed in GUI interface

# Algorithm design

**Data:**

Data in the algorithm includes data of courses classified by interests and course salary, and is embedded in the code. Data of courses is stored in dictionary *Dictionary\_interests* in the form of *{interest1: {course1, course2, ...}, interest2: {course3, ...}, ...}*. Data of course salary is stored in dictionary *Dictionary\_StartingSalary*, in the form of {course1: salary1, course2: salary2, …}

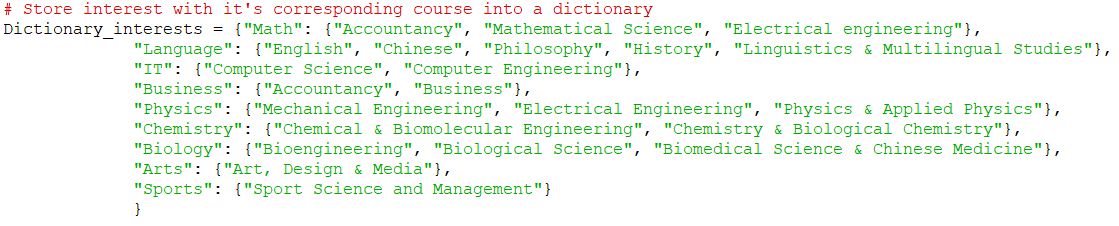


Figure 3: Dictionary for interests

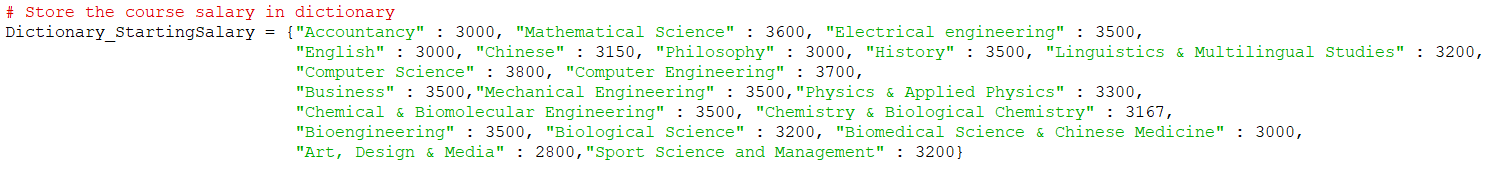


Figure 4: Dictionary for course starting salary

**Class: *Layout*.**

*Layout* is the GUI object to interact with user. It contains frame, label, combo box, button, and several methods to get inputs, do calculations and display the information.

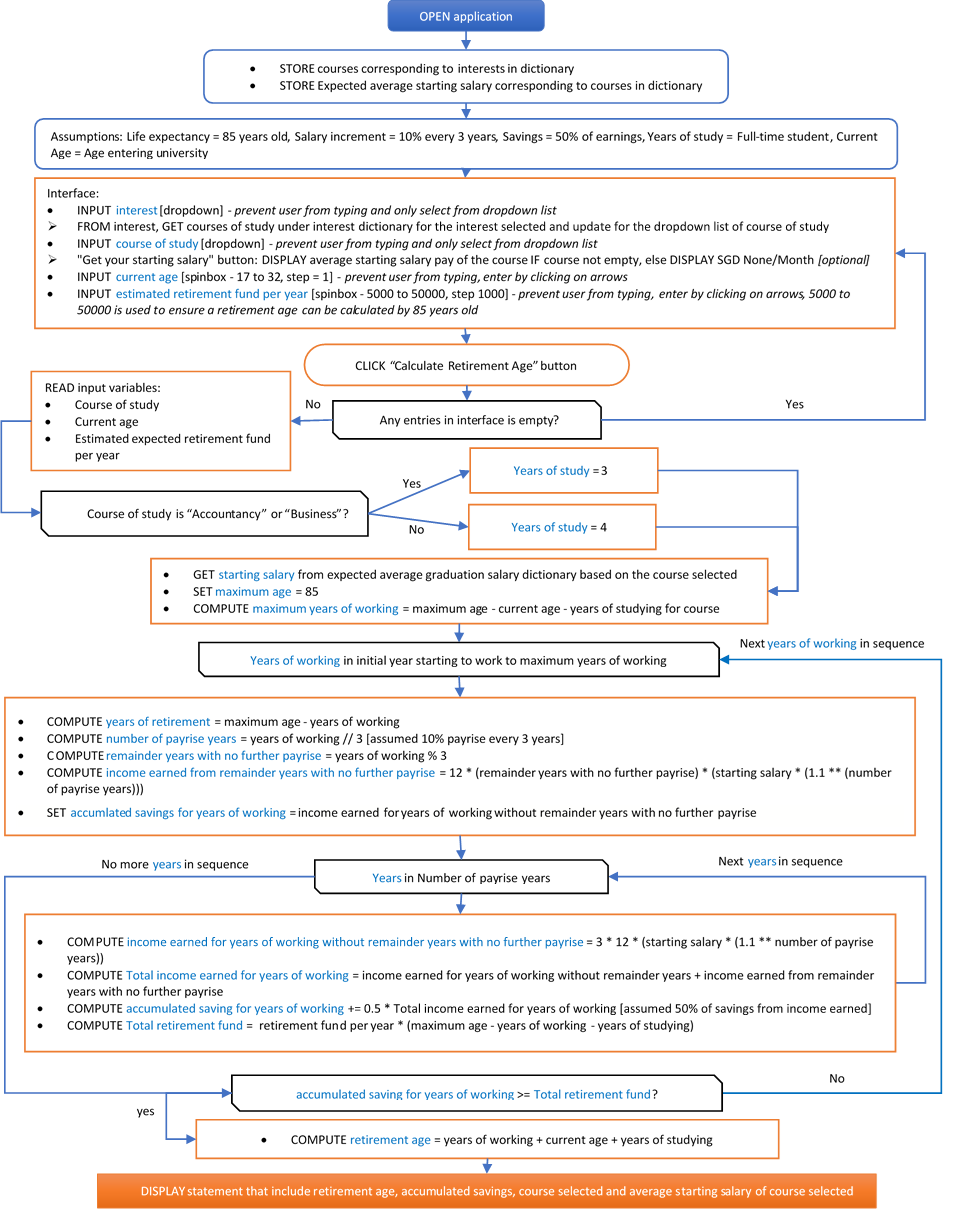
**Methods in *Layout*:**

* ***GetUpdateCourseData()***: Update dropdown box of course with interest chosen by user.
* ***DisplayStartingSalary()***: Display the starting salary in label according to the course chosen by user.
* ***GetCourseYears()***: Get the course years according to the course, if the course is Accountancy or Business, then return is 3, otherwise, it is 4.
* ***earned\_income()***: Calculate the accumulated income according to age, starting salary, pay rise instance, and remained years to retire, assuming the salary increment is 10% each time every 3 years. It will return a float.
* ***working\_years():*** This method aims to calculated number of working years after graduation based on the user’s age, retirement spending, starting salary and years of study based on their course. Once the amount of savings becomes more than the amount needed to retire, the for loop ends, and the method returns the retirement age and the accumulated savings.
* ***ExecuteCalculation\_UserInputs()*:** This method aims to check whether the requested inputs from the user is available, returning an error message if it is not. If all the inputs are available, then the method will proceed to calculate call the methods working\_years() and earned\_income() to generate the expected retirement age, as well as the accumulated savings at the time of retirement.

**Main function:**

Algorithm use *main()* to create a layout object and do the main loop.

## 4.1 Flowchart



## 4.2 Pseudo Code

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colour Specification** | | | | |
| Green: Comments on block of pseudo code | Blue: Variables | Purple: Keywords | Orange: Button in interface | Pink: Python mathematical Operation |



## 4.3 Output of Application

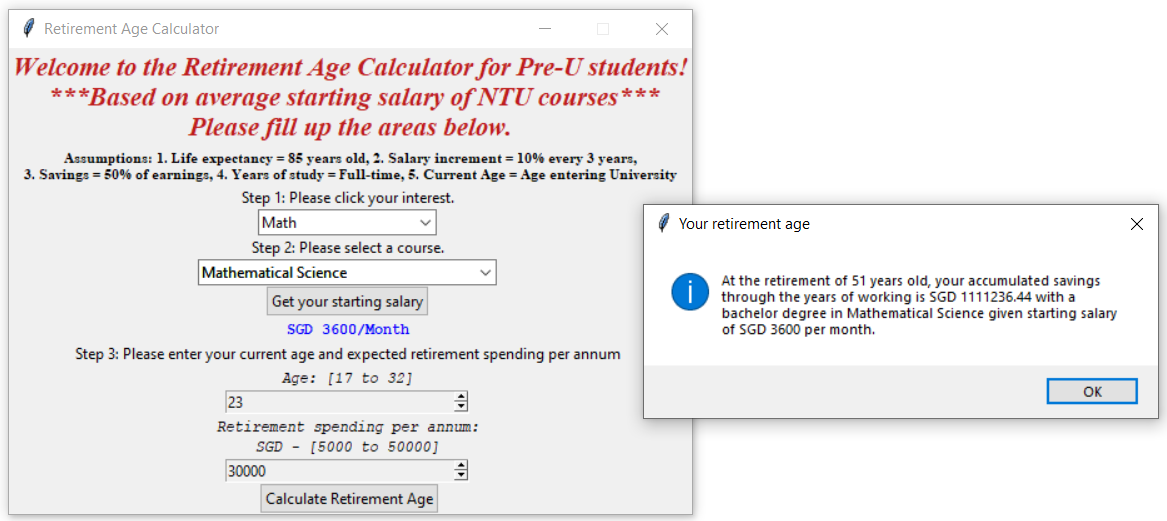


Figure 5: Output of Retirement Age application

From the figure above, with inputs of Math for interest, Mathematical Science for course, 23 for age and SGD30,000 for retirement spending per annum, it will output a statement of “At the retirement of 51 years old, your accumulated savings through the years of working is SGD 1111236.44 with a bachelor degree in Mathematical Science given starting salary of SGD 3600 per month.”.

# Program Testing

## In our GUI

* Dropdown box for interest and course selection would prevent user from entering manually and select solely based on the dropdown list

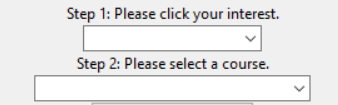


Figure 6: Dropdown box for interest and course

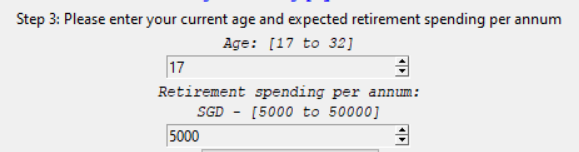
* Spinbox for current age and expected retirement spending per annum
  + Prevent user from entering any values and only enter through clicking on the arrows
    - Age: change of 1 for each click
    - Retirement spending per annum: change of SGD 1000 for each click 

Figure 7: Spinbox with clicking through arrows

* + Restricted value for current age and expected retirement spending per annum
    - Age from 17 to 32
      * Common age for pre-university students

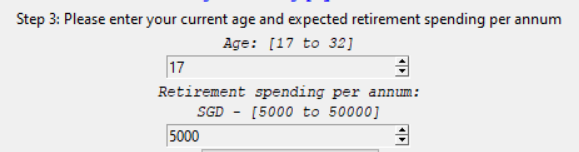


Figure 8: Range of age for Spinbox

* + - Retirement spending per annum from 5000 to 50000
      * We had do a trial and error test, anything that are too large would cause the person not being able to retire before 85 years old which will defeat the purpose of our application, hence the range of 5000 to 50000 per annum is a reasonable amount

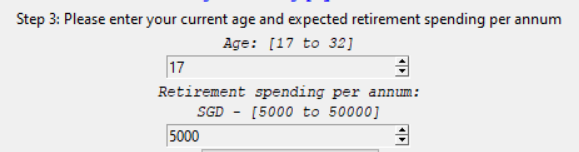


Figure 9: Range of expected retirement spending per annum for Spinbox

* If there is no course was selected, the “Get your starting salary” button would display SGD None/Month

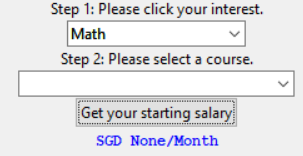
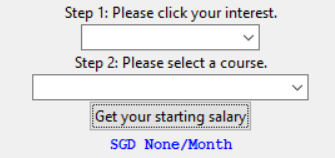


Figure 10: Average starting salary displaying SGD None/Month

## Error message

* When both interest and course were not selected, the program would show the error message “Please select your interest of study and subsequently select the course of study based on your interest.”

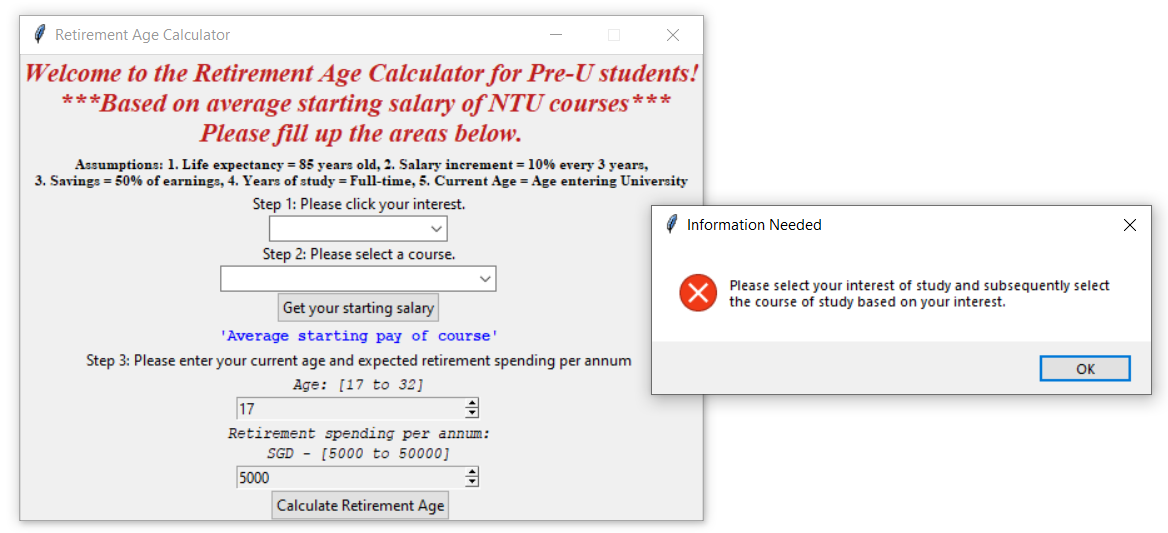


Figure 11: Error message when interest and course is blank

* When only interest was selected and course was not selected, the program would show error message “Please select a course based on your interest chosen.”

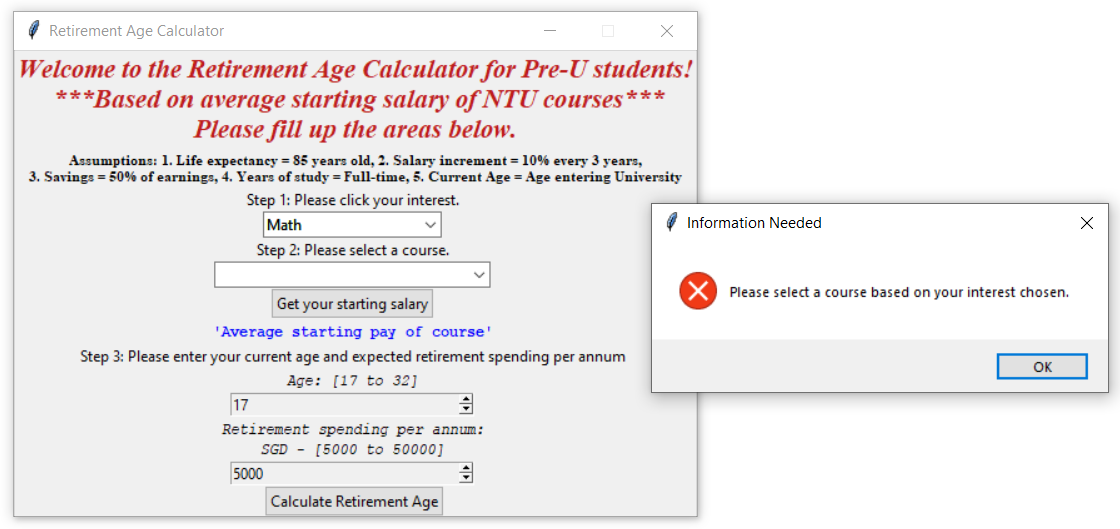


Figure 12: Error message when course is blank

# Reflection

This project gives us an opportunity to take what we learnt from the lectures and combine all the code together into a program. We use functions such as if-else statement, dictionary, classes, functions, GUI framework –Tkinter. We learnt the art of program design, development, integration, handling and testing.

## Challenges

Firstly, most of us are first time or beginner programmers, we have a lack of coding knowledge and experience. To overcome that we had to self-learn online through online forums and videos.

Secondly, our group is also new to the fintech industry, as we do not yet have experience working in the finance industry. Hence, we found it hard to come up with financial problems and relate our program to it. The presentation from other groups has helped to open our perspective on programming with finance applications. It has also inspired us to continue to learn and understand more about python and finance.

Thirdly, as the time frame is very short, we struggled to complete the program. As our group is made up of 2 full time student and 2 part time students, we had limited meeting periods because students and working adults have completely different schedules. The students are free in the day and busy at night for lectures whereas working adults are busy in the day working and free at night.

For the calculation part, we struggle at first when we need to decide how we are going to loop to determine the retirement age. We realized that some assumptions are needed to be put in place for the calculation to work. For example, there should be a maximum age that a person can live and percentage of saving per earning. We had to restrict the expected retirement spending per annum from 5000 to 50000 because if it is too large, the retirement age would exceed 85 years old which defeat the purpose of the application which is to get the retirement age.

## Improvements

Firstly, we can account for inflation because in the real world, inflation do exist. Secondly, we can update the course and average starting salary in real time. This is because new courses would be added, and some courses will be dropped. With course and average starting salary updated in real time, we can ensure that our calculation would be more accurate and up-to-date. Thirdly, we can make the savings more flexible so that it would be more personalize to individual such as a dropdown list of 10% to 70% of earnings for user to select. We can include more input options for the users to make the calculation more accurate and relatable to them. Our program could also have some graphic analysis for the user to understand more about their choices.

# Conclusion

This application is calculated solely based on the average starting salary of the individual without accounting for future commitments. Many factors are not accounted for in the future and the future is unpredictable. This application would serve only as a guide to the student.

# Appendix

**Methods in *Layout***

