Year 9 Mathematics Investigation - Computational Algorithms for Modeling Compound Interest

March 31, 2023

1 INTRODUCTION

This investigation has five parts which require you to think carefully and design a set of computational algorithms (which altogether make a program). You are to then implement your algorithms in code using the Python programming language. You can use Grok Academy's Python Playground to write your code (keep in mind that Python Playground 13 will be reserved for submission). It is, however, highly recommended that you use VS Code or PyCharm to develop your program. If you need help setting these up, email isaac.kigodi@education.wa.edu.au or come to the Maths office. If you have requested a mentor, you can contact them for help.

Be aware that most of the marks for this investigation will be allocated to the design work and explanation of your algorithms. It is therefore wise to create a slide presentation onto which you can start adding your designs. This investigation will frequently refer to the slides that you will be adding to as you go.

You must carefully document the process that you go through and include your designs and decisions so that you can present them to a small group of randomly-selected classmates. To help you with this, make sure to follow the process outlined below;

1. Read through each part of this document and take notes. Make sure to record and justify any assumptions you make.
2. Re-read each of the six parts and design your algorithms in English using brief dot-points (with indentation to indicate blocks of instructions) or you can use diagrams/flowcharts if you prefer. Add these to your presentation slides.
3. Make sure your overall design has separate parts that can be shared or reused and thus implementable in Python using functions. Refer to the Python Functions Tutorial posted in Connect earlier this year.
4. Once you have finished the first draft of your design, review your it and make modifications as needed. Document any changes you make and be sure to add them to your slides.
5. Code and test each part. If you find that your code starts to deviate from your design, just document the changes you make and provide reasons. You can make annotations to your original design.

The details of what your program needs to do for each section will be given below, together with the number of marks allocated out of 100. Additional informal guidelines can be found by typing (then running) `import this` into Grok Academy or VS Code/PyCharm.

Make sure you document your code using # comments. Code comments are like sunscreen – if in doubt use lots and lots.

1.1 Document exactly how you are using imported modules and libraries

Make sure you do not simply import a library from the internet and use it to do all of the work for you. For instance, doing the following will not earn any marks:

```python
import compoundinterest as ci
print(ci.calculate_amount(1000, 5, 'year', cpd_freq='month', cpd_periods=24))
```

The intent is for you to write your own versions of the functions (in this case, `calculate_amount()`) that perform the required operations so that you gain a deep understanding of how to think about, and design computational algorithms. This knowledge will also allow you to be able to understand and explain code that you obtain from sources on the internet as you work on this task.

If you do import some libraries (such as `numpy` or `scipy` or `pandas`), make sure to explain in your slides exactly how you are using them, making it clear that they only play a supporting role and that they do not implement entire parts of your investigation.

1.2 Create and explain a reusable means of converting time periods (5 Marks)

Throughout this investigation you will be converting between time periods, at a minimum, between days, weeks, months, quarters and years. You will therefore need to think of a way of telling the computer that there are 365 days in a year and 30 days in a month (to keep things simple) in addition to 7 days in a week, 3 months in a quarter and 52 weeks in a year.

As a hint for this section, a look-up table can come in handy and one way you can implement this is by using dictionaries inside a dictionary. See the Introduction to Programming 2 (Python) course on Grok Academy, module 5 under the headings "Scissors, Paper, Rock!" and "Dictionaries to the rescue!".

Make sure to document what you decide to do in your presentation slides. If you choose to implement this conversion table in another way, describe in your presentation why you chose to do it the way you did.

2 PART I: COMPARE SIMPLE AND COMPOUND INTEREST

To get started, you are going to help the user of your program to compare a Simple Interest (SI) savings account to a Compound Interest (CI) savings account. You should ask for the principal, the interest rate as a percentage and the amount of the time to project the accounts into the future. The user should be able to enter the amount of time as a positive integer and specify its unit. You should also ask for the number of compounding periods per interest rate time unit in the case of a CI account. See the sample prompts below.

2.1 Prompt as specified and summarise details of the accounts (5 Marks)

Examples are given below and note that for simplicity, you may specify time periods in singular form (`4 year` instead of `4 years`).

```
.
. see over the page
```

```
MODULE 1: SIMPLE AND COMPOUND INTEREST COMPARISON

Simple Interest Account:

Enter the principal amount in $: 1000

Enter the interest rate (enter 5% as 5): 3

Enter the interest rate time unit (year, quarter, month, week, day): year

Compound Interest Account:

Enter the principal amount in $: 1000

Enter the interest rate (enter 5% as 5): 3

Enter the interest rate time unit (year, quarter, month, week, day): year

Enter the compounding period time unit (year, quarter, month, week, day, custom):
month

Future projection timeframe for both accounts:

Enter the amount of time to project into the future: 4

Enter the projection time unit (year, quarter, month, week, day): year
```

.

You should then summarise the input data as shown below before performing calculations:

```
SI Account: P = 1000, r = 3% per year

CI Account: P = 1000, r = 3% per year, Compounding Frequency: month

Projection timeframe: 4 year
```

.

Hint: You might find it useful to use Python's f-strings as shown below when summarising.

```python
name = 'Leonhard'
num = 2.71828
print(f"{name}'s favourite number is {round(num,2)} (2 d.p.)")
```

2.1.1 Accept a custom compounding frequency

If the user chose `custom` as the compounding period time unit, your program should respond appropriately;

```
Enter the compounding period time unit (year, quarter, month, week, day, custom):
custom

Enter the number of compounding periods per interest rate time unit: 6
```

.

The summary should include a line such as

```
CI Account: P = 1000, r = 3% per year, Compounding Frequency: 6
```

2.2 Output the correct amounts and explain your design (15 Marks)

After the summary, you should output the following information

```
SI Account projected amount: $, Interest earned: $

CI Account projected amount: $, Interest earned: $
```

You will likely make assumptions as you complete this part - make sure to document and justify them in your slides. You must also include an explanation of how your design for this part works.

3 PART II: TIME FOR AN ACCOUNT TO REACH A TARGET

For this part you are to help the user ascertain the amount of time it would take for a compound interest savings account to reach a certain amount that the user is aiming for.

3.1 Prompt in the specified format and summarise (5 Marks)

```
MODULE 2: TIME FOR A CI ACCOUNT TO REACH A TARGET AMOUNT

Enter the principal amount in $: 1000

Enter the interest rate (enter 5% as 5): 10

Enter the interest rate time unit (year, quarter, month, week, day): year

Enter the compounding period time unit (year, quarter, month, week, day, custom):
quarter

Enter the target amount: 1500

.
```

You should then summarise the input data as shown below before performing calculations:

```
CI Account: P = 1000, r = 10% per year, Compounding Frequency: quarter

Target amount: 1500
```

3.2 Output projection and explain algorithm in your slides (10 Marks)

Your program should output a projection of the amounts at the end of each compounding period as a list and the minimum number of compounding periods (in appropriate units) needed to *first* exceed the target amount.

```
Forward projection: [1000.00, 1050.00, 1102.50, 1157.63, 1215.51, 1276.28,
1340.10, 1407.10, 1477.46, 1551.33]

Time taken: 9 quarters
```

The minimum requirement is to output the number of compounding periods needed. You can output the time taken in a more user-friendly manner (such as 2 years & 1 quarter for this example i.e. using the same time unit as the interest rate with the remainder being compounding period time units).

To earn full marks for this section, you need to include an explanation in your slides as to how you designed this part and how it works.

4 PART III: COMPARE TWO CI ACCOUNTS (10 Marks)

The intent of this part is for you to demonstrate how you can adapt the design (and thus functions) you created in Part II. As before, make sure to document your thoughts and ideas in your slides as you will need to talk about these during your presentation.

To that end, this module should allow the user to compare two compound interest accounts specified in a variety of ways. For example, the user should be able to compare a 6% interest per year account compounded quarterly for 3 years with one at 2% per quarter account compounded weekly for 2 years.

The outputs should take the form of projections showing how the amounts change over time. The 10 marks are for demonstrating that your projections work, and for explaining how your design for this section made use of ideas you have developed in part II.

5 PART IV: MODELING REGULAR DEPOSITS

Imagine that you are saving up for your first car and your maximum budget is $20,000. You intend to buy the car at the end of the calendar year that you finish year 12. You do not wish to borrow money and you are happy with either a new, or a used car (you can research and mention it during your presentation).

For example, for a 5% per year CI account compounded monthly, you can start the account with a $1000 gift from your caregivers and make a $250 per month regular deposit from your after-school job and allowance (pretend you started this at age 13 so that the process runs for 5 years). Will you have enough money at the end of the five years? What is the minimum you would need to regularly deposit into your account so that you just manage to reach your target?

5.1 Explain your design and output projections as columns (10 Marks)

Make sure to produce projections (formatted as columns) showing the opening amount, interest earned, regular deposit and closing amount. You should be able to adjust the inputs to generate projections and explore various scenarios.

Note that once again, you should not have to design this part from scratch. You should be able to reuse ideas and modules you have already designed, with some modifications – make sure to document these in your presentation slides.

Your prompts should capture information in the format specified below:

```
MODULE 5: Compound Interest account with regular deposits

Enter the principal amount in $: 1000

Enter the interest rate (enter 5% as 5): 5

Enter the interest rate time unit (year, quarter, month, week, day): year

Enter the compounding period time unit (year, quarter, month, week, day, custom):
month

Enter the regular deposit amount per compounding period: 250
Enter the dollar amount to project to (if you enter 0, you will be asked for the
amount of time to project for): 0
```

```
In that case, enter the amount of time to project for: 5
```

```
Enter the projection time unit (year, quarter, month, week, day, custom): year
```

5.1.1 Accept a length of time or a dollar amount as the projection target (5 Marks)

Note that as shown above, the user should be able to enter the dollar amount to project to. If this amount is non-zero, the program should not ask for the amount of time to project to. On the other hand, if zero is entered as the dollar amount, the program should ask for the length of time to run the projection for.

The example below is to get you started as to how to display your projections. You should research a better way to format the columns.

[1]:
```python
# As a hint for this section, you might find the zip() python function useful␣
↪for quickly putting together multiple lists


# the lists below represent projections that you can generate
opening_amount = [10,14,19,25,32]
interest = [1,2,3,4,5]
regular_deposit = [3,3,3,3,3]
closing_amount = [14,19,25,32,40]


# to arrange them into columns, we can 'zip' them together
zipped = zip(opening_amount, interest, regular_deposit, closing_amount)


print('column headings: opening, interest, deposit, closing')
for row in list(zipped):
    print(row)


# experiment with the zip function to get a better sense of how it works

# explore: can you use the zip() function to 'unzip' the zipped list to recover␣
↪the original lists?
```

```
column headings: opening, interest, deposit, closing
(10, 1, 3, 14)
(14, 2, 3, 19)
(19, 3, 3, 25)
(25, 4, 3, 32)
(32, 5, 3, 40)
```

6 PART V: A LOAN WITH MISSED INTEREST PAYMENTS

For this part, you are to reuse (with modification) the code you wrote in part IV relating to a CI account with regular deposits - you should not have to start from scratch.

Imagine a lender that charges 20% interest per year compounded weekly to someone who borrows $1000. Due to a series of unfortunate events, the borrower is unable to pay the interest due each compounding period for three years. When this happens, the lender simply adds the amounts to the outstanding loan.

6.1 Explain reuse of previous designs and generate projections (5 Marks)

Document how you can modify the functions you wrote for part IV to generate the projections for this part. Include the projections in your presentation slides.

6.2 Model and explain the effect of increasing the compounding frequency (15 Marks)

Another lender (going with the name Shark Loans) charges 100% interest per annum compounded quarterly with the total amount being due at the end of the year. Assume that a borrower is unable to make any payments into the loan and use your algorithm above to project how much the unfortunate borrower of $1000 will owe at the end of the year and include the projections in your presentation. Make sure to document how you have reused your previous designs in your slides.

Wanting to increase the amount of interest charged while keeping the same interest rate on paper (100% p.a.), Shark Loans decide to add to the fine-print the increase of the compounding frequency to weekly. Model this scenario for a year and include the projections in your presentation.

Frustrated that they are not collecting as much money at the end of the year as they had hoped, they decide to compound daily, then every hour, then every 10 minutes! Produce projections for the amounts owed at the end of the year in each scenario.

To help with visualisation, graph these projections with quarterly, weekly, daily, hourly and ten-minute-ly compounding periods to see how the amount owed increases as the year progresses. You can use Microsoft Excel, iWork Numbers or Google Sheets, or if you wish, you can use matplotlib (using matplotlib is optional and does not earn more marks). Paste your graphs onto your slides.

Will it ever be possible for Shark Loans to increase the amount collected at the end of the year beyond a certain limit? How many times the principal amount is this limit (give your answer to 3 decimal places)? Research this multiple and summarise what you find out in a slide of your presentation.

7 PART VI: CREATE A USER INTERFACE

Your program should allow the user to conveniently model the scenarios described above. This entails you creating a friendly way to interact with your program. You can choose to not implement a menu system and simply create five programs to meet the requirements of the investigation (this will forgo the 10 marks allocated to this section)

7.1 Provide a menu of choices to the user (5 marks)

```
This program has five modules. Choose a module by typing its number

(1) Compare simple and compound interest savings acccounts

(2) Calculate the time for a CI savings account to reach a target amount

(3) Compare two Compound Interest savings accounts

(4) Model a CI savings account with regular deposits

(5) Model a loan with missed interest payments

Enter 1 to 5, or 6 to quit:
```

Once the user enters their choice, your program should display the appropriate prompts (as specified in the individual sections of the investigation)

7.2 Return to the main menu (5 marks)

It is ideal that your program does not simply quit after the user interacts with the chosen module. You should ask the user to either start over with the same module, go back to the main menu, or quit. For each selection, your program should respond accordingly.

7.3 Use functions throughout your program (5 marks.)

You should ensure that you use functions whenever there is a need to repeatedly accomplish certain tasks. *Examples* of names of functions you could implement are shown below.

```
show_main_menu()
compare_SI_and_CI()
calculate_time_for_CI_target()
compare_CI_accounts()
model_CI_with_regular_deposits()
model_CI_loan()
```

Advanced: You are welcome to use object oriented programming if you are familiar with it.

8 PART VII: SUBMIT YOUR WORK

The due date for the slides and the code produced for this investigation is Friday 5 May 2023. Your presentation slides are the most important part of your investigation because they contain the planning and design work that you will be explaining to your classmates. These will need to be uploaded to a submission form that will be made available to you in the second week of next term.

Your code should be saved in Grok Academy's Python Playground 13 by the due date. If you use VS Code or PyCharm and imported a module that you wrote, remember to paste all of the code of the module in place of the `import` statement before saving into Grok Academy.

During week 3, you will do a 10-minute presentation of your computational and algorithmic thinking processes to a random group of peers with teacher supervision. You will need to showcase the slides you prepared documenting the process you went through and the designs you generated. Email isaac.kigodi@education.wa.edu.au or come to the Maths office if you need assistance.