Year 9 Mathematics Investigation - Computational Algorithms for Modeling Population Growth

March 28, 2025

1 INTRODUCTION

"Five-second rule!" is an interjection you may have made after an unfortunate event involving your favourite snack. Scenarios involving population growth such as that of bacteria on said treat can be modelled mathematically using exponentiation. In this investigation you will explore various aspects of bacterial population growth by simulating this phenomemon using well thought-out computational algorithms. You can pretend that you are writing the program for a microbiologist to use in their research.

In order to simplify calculations, we will assume that the bacteria in question reproduce at fixed times, which we will call "fission events". Thus in a 24 hour period (one day), if there are 8 fission events it means that new bacteria are created every 3 hours.

As you go through the algorithm design process, make sure to document your progress in presentation slides. The designs in your slides should be in plain English using brief dot-points (with indentation to indicate blocks of instructions) or you can use diagrams, pseudocode or flowcharts if you prefer. It is important that you demonstrate a deep understanding of all formulae that you use – document these in your slides with an explanation of how they work.

You will likely need to make assumptions as you engage in design work. Make sure to document them in your slides (the slides must be saved in the same folder that contains your code).

1.1 Document any use of 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:

```
import population-modelling as pm
print(pm.pop_size(1000, 1.25, 'year', 5, 10))
```

The intent of this task is for you to write your own versions of the functions 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 online sources, including generative AI.

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, with the largest being a day which is subdivided into half-days, quarter-days, hours, minutes and seconds. You will therefore need to think of a way of telling the computer how many quarter-days in a day, hours in a half-day, seconds in a minute e.t.c.

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: NAIVE VS SOPHISTICATED MODEL COMPARISON

To get started, you are going to help the user of your program to quickly model bacteria population growth in a naive manner (the population grows by a the same fixed amount at each fission event) as well as creating a more realistic model where the larger the population is, the faster it grows.

You should ask for the initial population (I), the growth rate per time unit as a percentage and the amount of the time to project the population 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 fission events per growth rate time unit in the case of the sophisticated model (for example, if the growth rate is specified as 18% per day, and there are 6 fission events per day, then *you should interpret this to mean* that the growth rate is 3% every 4 hours). See the sample prompts below.

2.1 Prompt as specified and summarise the entered information (5 Marks)

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

```
MODULE 1: NAIVE AND SOPHISTICATED MODEL COMPARISON

Naive population model:

Enter the initial population: 1000

Enter the growth rate (enter 7% as 7): 12

Enter the growth rate time unit (day, half-day, quarter-day, hour, minute): day


Sophisticated population model:

Enter the initial population: 1000

Enter the growth rate (enter 7% as 7): 12
```

```
Enter the growth rate time unit (day, half-day, quarter-day, hour, minute): day

Enter the fission-event frequency time unit (day, half-day, quarter-day, hour,
minute, custom): hour


Future projection timeframe for both models:

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

Enter the projection time unit (day, half-day, quarter-day, hour, minute): day
```

.

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

```
Naive Model: I = 1000, g = 12% per day

Sophisticated Model: I = 1000, g = 12% per day, Fission-event frequency: hour

Projection timeframe: 4 day
```

2.1.1 Accept a custom fission-event frequency

If the user chose `custom` as the fission-event frequency time unit, your program should respond appropriately;

```
Enter the fission-event frequency time unit (day, half-day, quarter-day, hour,
minute, custom): custom

Enter the number of fission events per growth rate time unit: 6
```

The summary should include a line such as

```
Sophisticated Model: I = 1000, g = 12% per day, Fission-event frequency: 6
```

2.2 Output the correct projected population size and explain your design (10 Marks)

After the summary, you should output the following information

```
Naive model projected population size: _

Sophisticated model projected population size: _
```

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.

. . .

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.)")
```

3 PART II: TIME FOR A POPULATION TO REACH A TARGET SIZE

For this part you are to help the microbiologist ascertain the amount of time it would take for a bacteria population to reach a certain size that they are aiming for. *We will only use the sophisticated population model from now on.*

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

```
MODULE 2: TIME FOR A POPULATION TO REACH A TARGET SIZE

Enter the initial population: 1000

Enter the growth rate (enter 7% as 7): 20

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

Enter the fission-event frequency time unit (day, half-day, quarter-day, hour,
minute, custom): quarter-day

Enter the target amount: 1500
```

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

```
Sophisticated model: I = 1000, g = 20% per day, Fission-event frequency: 4

Target amount: 1500
```

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

Your program should output a projection of the population after each fission event as a list and the minimum number of fission events (in appropriate units) needed to *first* exceed the target population size.

```
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 quarter-days
```

The minimum requirement is to output the number of fission-event frequency time units needed. You can output the time taken in a more user-friendly manner (such as 2 days & 1 quarter-day for this example i.e. using the same time unit as that of the growth rate with the remainder being fission-event frequency time units).

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

4 PART III: COMPARE SOPHISTICATED MODELS (5 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 sophisticated population growth models specified in a variety of ways. For example, the user should be able to compare a 15% per day growth rate model with fission-event frequency of 4 with one at 10% per half-day with a fission-event frequency of 3 over a period of 5 days for both.

The outputs should take the form of projections showing how the numbers change over time. The 5 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: GENERATE MORE DETAILED PROJECTIONS

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

In this section, we continue modelling bacteria population growth and we want to keep track of the number of bacteria added to the population at each fission event. We will do this by creating projections (formatted as columns) showing the current (opening) population, the number of new bacteria added, and the closing population. 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: Keeping track of the of population increase at each fission event

Enter the initial population: 1000

Enter the growth rate (enter 7% as 7): 12

Enter the growth rate time unit (day, half-day, quarter-day, hour, minute): day

Enter the fission-event frequency time unit (day, half-day, quarter-day, hour,
minute, custom): hour

Enter the population size to project to (enter 0, to specify the amount of time
to project for): 0

In that case, enter the amount of time to project for: 3

Enter the projection time unit (day, half-day, quarter-day, hour, minute, custom)
day
```

5.1.1 Accept a length of time or a dollar amount as the projection target

Note that as shown above, the user should be able to enter the population size 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 population size, 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.

[2]:
```
# The zip() function useful for quickly putting together multiple lists
opening_population = [10,14,19,25,32]
num_of_new_bacteria = [1,2,3,4,5]
closing_population = [11,16,22,29,37]
# to arrange them into columns, we can 'zip' them together
zipped = zip(opening_population, num_of_new_bacteria, closing_population)

print('column headings: opening, number added, closing')
for row in list(zipped):
    print(row)
```

```
column headings: opening, number added, closing
(10, 1, 11)
(14, 2, 16)
(19, 3, 22)
(25, 4, 29)
(32, 5, 37)
```

6 PART V: SIMULATE INCREASES IN FISSION-EVENT FREQUENCY (10 MARKS)

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

Imagine that you are the microbiologist growing Lactobacillus (to make yoghurt of course) and that the particular strain of bacteria you have grows at the rate of 100% per day with four fission events. **You only have one day to work with**. Document how you can modify the functions you wrote for part IV to generate the projections (in columns) for the day and add these to your slides. Make sure to document how you have reused your previous designs in your slides.

Wanting to increase the population at the end of one day (with the same 100% per day growth rate), you find a way to make the bacteria reproduce once every two hours (i.e. 12 fission events). Model this scenario for a day and display the projections on the screen (and slides).

You are perhaps surprised that the population at the end of the day did not grow as much as you expected. You decide to use latest in CRISPR technology to increase the fission-event frequency to once every hour, then once per minute, then once every second. Produce projections for the population size at the end of the one day - you don't have to paste the last two sets of projections to your slides.

To help with visualisation, graph all projections for the various fission-event frequencies (quarter-day, 2-hour, hour, minute and second) to see how the population increases as the day 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 population size at the end of the day to grow beyond a certain limit? How many times the initial population 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 a 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 5 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 a naive and sophisticated model

(2) Time for a sophisticated model to reach the target population

(3) Compare two sophisticated population models

(4) Generate detailed projections formatted as columns

(5) Model increases in fission-event frequency

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). 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.

7.2 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_naive_with_sophisticated()
calculate_time_to_target()
compare_sophisticated_models()
display_projections_as_columns()
model_increase_in_fision_frequency()
```

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

8 PART VII: DOCUMENT YOUR CODING PROCESS WITH FREQUENT AND DETAILED COMMITS (30 MARKS)

Once you complete the design process, you are required to implement it in code using the Python programming language. You are expected to use code comments and functions extensively and there will be marks allocated for these. Consult the year 9 CAT resources in Compass under School Documentation | Curriculum and assessments | Mathematics (more in in Grok Academy's "Introduction to Programming 2" intermediate course, module 7).

Do NOT use Grok to write code for this investigation. You must use VS Code (or PyCharm) and you need to have installed Git so that you can document your coding process with frequent commits ('diary entries'). Writing a separate diary of your progress elsewhere other than by 'making commits' as instructed will not be awarded any marks. If any of these instructions are unclear, make sure to contact Mr. Kigodi during term 1 weeks 9 and 10.

Before you start coding, create a new folder called Y9 CAT Inv1 Name Surname (with your name and surname), open it in VS Code or PyCharm and initialize a new repository (follow the instructions in the year 9 Compass CAT resources). Create a main.py file to hold your code and save your design slides into the same folder that contains main.py. You should then immediately make your first commit with a diary entry saying you have just created a new project with your design slides and a blank file to hold your code. Do not move files or delete them once you have initialised a repository – if you are not careful, you might erase your commits (and lose the 30 marks).

As you code, debug and test what you have written, and once you have made a notable addition that works (such as a welcome message or a new function), commit it with a detailed description of what you've done. Continue adding code as per your design, and each time you write enough lines to make a notable change, make another commit with a detailed description. If you make a mistake or change your mind about some of your code or design, make the necessary changes and commit again with a description of what you've done. You will not be penalised for this – just make sure to adjust or refine and commit your design slides to reflect any new ideas you come up with as you code (this is normal). The code and the design in your slides must match.

The code at each commit checkpoint needs to be able to run and you should be able to describe the features that the code implements up to that point. Remember that frequent and detailed commits are required to earn the marks in this section (Git records the date and time of each commit). Inserting tens or hundreds of lines of code per commit will not meet the requirement. Essentially, your commit history should make it very clear that you have been gradually working on your investigation and that you understand intimately the code that you are committing. Remember that you will need to show your commit history when presenting to a random group of classmates.

9 PART VII: SUBMIT YOUR WORK BY THE DUE DATE

The due date for the slides and the code produced for this investigation is Friday 9 May 2025 (end of term 2 week 2). Make sure your presentation slides are in the same folder that contains your code and make the final commit by the due date. The code and slides in this final commit are what you will present to your classmates and the timestamp will be used to confirm that the submission was made on time.

In addition to making the final commit, you will also upload your code and slides to an online submission form that will be made available the week the investigation is due. Full submission instructions will be issued in week 2.

During week 3, you will do a 12-minute presentation of your computational and algorithmic thinking processes to a random group of peers with teacher supervision. You will use VS Code (or PyCharm) to present your code and projections. You will also need to showcase the slides saved during the final commit 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 and remember to 'import this' to get a refresher of the Zen of Python.