Project overview
Investment Calculator

Compound interest is a wonderful thing, at least when you are in the plus! A lot of people don't even know what it is, so I decided to write this little program to make it visible. Also for a bit financial education for my children.

I am well aware that there are already plenty of savings and mortgage calculators online, but first of all replicating things is a great way to learn. In addition to the calculators I have seen, my program also accepts negative numbers for Start Balance, Monthly deposit and yearly interest, that opens the way to many different new examples.

For example:

What happens to a -1000$ credit card balance at 25% when you leave it for 5 years?

Or what is the buying power of 10,000$ cash at 4% inflation after 10 or 20 years? (set interest rate to -4%)

On the plus side if you give your child a $1000 at his birthday and put it into an investment funds at average return of 12%, how much would it be worth at 65? (Warren Buffets long-term return is actually 18%) Try it out.

Project overview

I started implementing this program as a GUI based application because it is just so much more user friendly. However following the conversation on discord I realized that this assignment is more about a terminal based application, so I wrote a terminal based user interface as well. I used the same set of functions for both versions, just the way I get the data from the user is different.

My program has the following input parameter:

- Start balance
- Interest rate
- Number of times interest is applied(compound frequency)
- number of years of investment

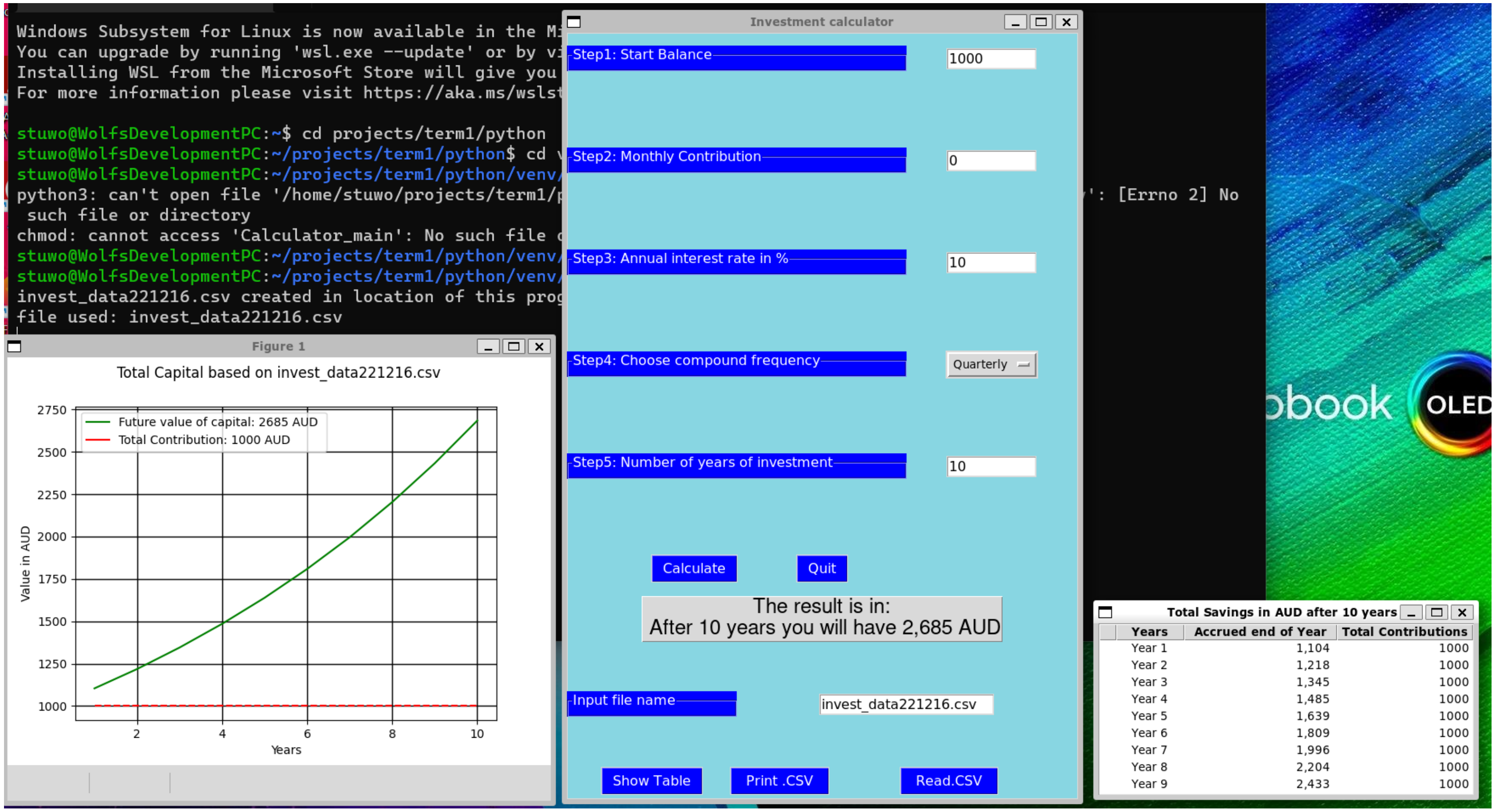
Project overview

It program has the following features:

- Calculate Capital after end of investment period
- Show table with growth of investment each year
- Show a chart with the data in the table
- Write the data into a .csv file
- Read the data of a .csv file previously generated with the program and show them in a chart

Screen shot

Input mask of the GUI based version



```
Windows Subsystem for Linux is now available in the Mi
You can upgrade by running 'wsl.exe --update' or by vi
Installing WSL from the Microsoft Store will give you
For more information please visit https://aka.ms/wslst

stuwo@WolfsDevelopmentPC:~$ cd projects/term1/python
stuwo@WolfsDevelopmentPC:~/projects/term1/python$ cd v
stuwo@WolfsDevelopmentPC:~/projects/term1/python/venv/
python3: can't open file '/home/stuwo/projects/term1/p                ': [Errno 2] No
 such file or directory
chmod: cannot access 'Calculator_main': No such file o
stuwo@WolfsDevelopmentPC:~/projects/term1/python/venv/
stuwo@WolfsDevelopmentPC:~/projects/term1/python/venv/
invest_data221216.csv created in location of this prog
file used: invest_data221216.csv
```

Step1: Start Balance	1000
Step2: Monthly Contribution	0
Step3: Annual interest rate in %	10
Step4: Choose compound frequency	Quarterly
Step5: Number of years of investment	10

The result is in:
After 10 years you will have 2,685 AUD

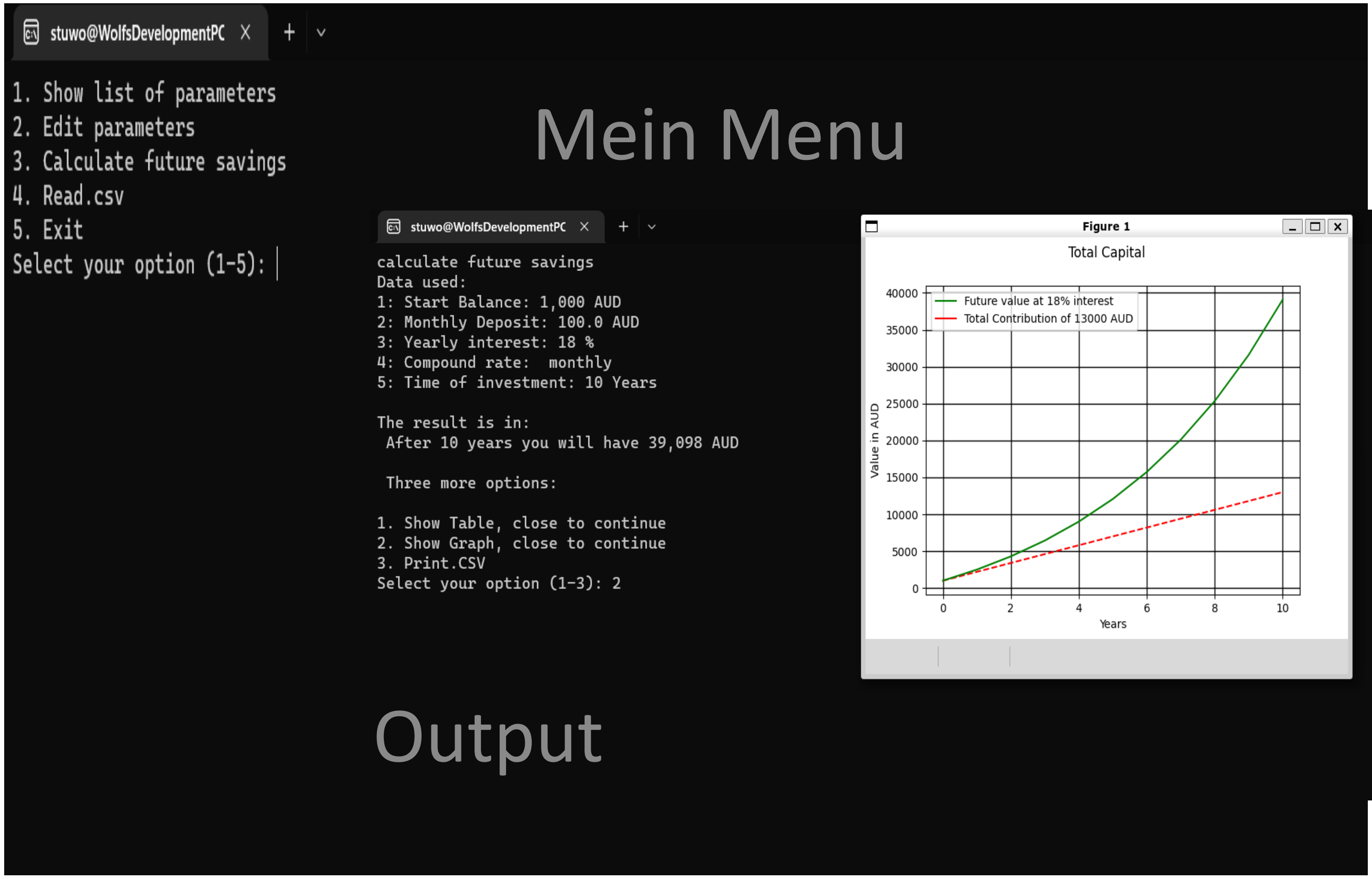
Input file name	invest_data221216.csv

Total Savings in AUD after 10 years

Years	Accrued end of Year	Total Contributions
Year 1	1,104	1000
Year 2	1,218	1000
Year 3	1,345	1000
Year 4	1,485	1000
Year 5	1,639	1000
Year 6	1,809	1000
Year 7	1,996	1000
Year 8	2,204	1000
Year 9	2,433	1000

Screen shots

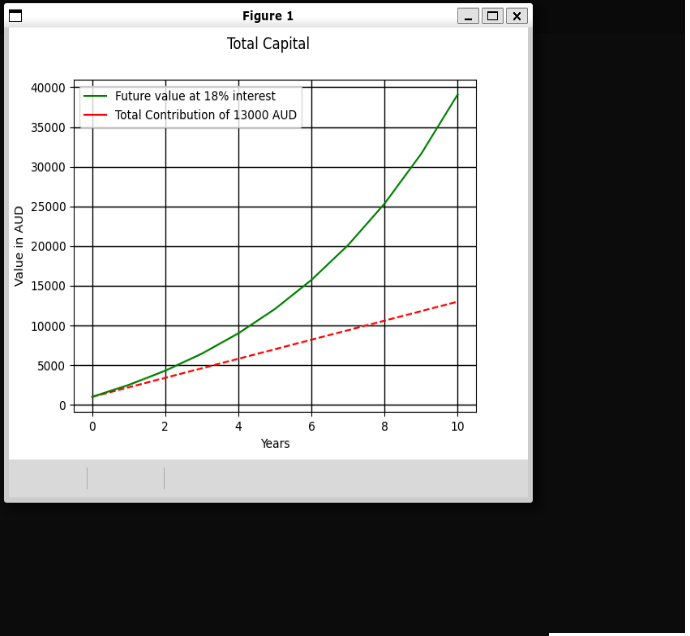
Input masks of the terminal based version



```
1. Show list of parameters
2. Edit parameters
3. Calculate future savings
4. Read.csv
5. Exit
Select your option (1-5):
```

Mein Menu

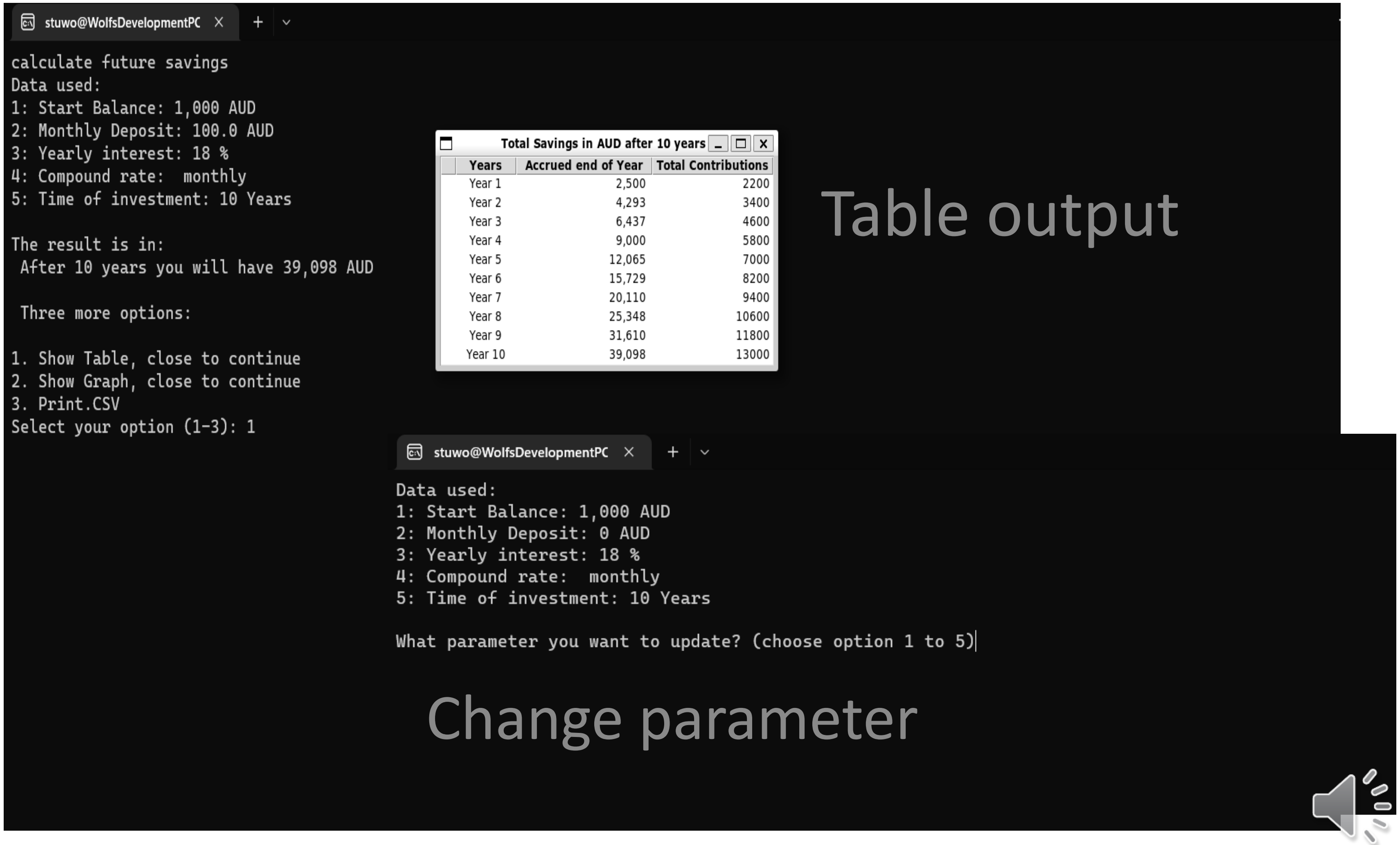
```
calculate future savings
Data used:
1: Start Balance: 1,000 AUD
2: Monthly Deposit: 100.0 AUD
3: Yearly interest: 18 %
4: Compound rate:  monthly
5: Time of investment: 10 Years

The result is in:
 After 10 years you will have 39,098 AUD

 Three more options:

1. Show Table, close to continue
2. Show Graph, close to continue
3. Print.CSV
Select your option (1-3): 2
```



Output

Screen shots

Input masks of the terminal based version



```
calculate future savings
Data used:
1: Start Balance: 1,000 AUD
2: Monthly Deposit: 100.0 AUD
3: Yearly interest: 18 %
4: Compound rate:  monthly
5: Time of investment: 10 Years

The result is in:
 After 10 years you will have 39,098 AUD

 Three more options:

1. Show Table, close to continue
2. Show Graph, close to continue
3. Print.CSV
Select your option (1-3): 1
```

Total Savings in AUD after 10 years

Years	Accrued end of Year	Total Contributions
Year 1	2,500	2200
Year 2	4,293	3400
Year 3	6,437	4600
Year 4	9,000	5800
Year 5	12,065	7000
Year 6	15,729	8200
Year 7	20,110	9400
Year 8	25,348	10600
Year 9	31,610	11800
Year 10	39,098	13000

Table output

```
Data used:
1: Start Balance: 1,000 AUD
2: Monthly Deposit: 0 AUD
3: Yearly interest: 18 %
4: Compound rate:  monthly
5: Time of investment: 10 Years

What parameter you want to update? (choose option 1 to 5)
```

Change parameter

Code overview

The core logic of the program is based on the compound interest formula:

$A = P(1 + r/n)^{(n*t)}$ with P: start balance, r: interest rate, n: number of times interest is applied(compound frequency) t: number of years of investment and A the future value of the investment.

For a one off start balance it is straight forward, however it gets a little more interesting if the user starts making monthly deposits. basically each deposit needs to be run through the formula with t the time from the moment of deposit to the end of the period of investment. Each iteration needs to be added to the result.

Code overview

1 Show the input mask or menu

get input data from user and check for plausibility, filter out

invalid inputs. In the terminal version the data is stored in 5 dictionaries, 1 set for each parameter. The fetch data function gets the actual values out of the dictionaries

2. Calculate the value of a future investment

As explained in the slide before

3. Show value of investment and monthly contribution in a table for each year of investment.

Opposite of point 2 that uses a one off value as a result, the total deposit and value of investment must be calculated for each row of the table before it is inserted. I used tkinter ttk.treeview to display the table. I did not see any value in converting it to a terminal output.

4. Store value of investment and monthly contribution in a .CSV file for each year. Basically the same strategy as in point 3; calculate the values for each row of the file and add the row for each iteration till the last year of the investment.

5. Plot chart with investment value and contribution over time.

To plot the chart I used matplotlib.pyplot and lines. It just looks so much needer and I would have no Idea how to display a chart in a terminal. Even to get pyplot doing what I want took me a lot of research. I needed to write the data for the plot function into 3 arrays: one for the time axis, one for start capital plus deposit each year and one for value of investment for each year.

Code overview

6. Read from previous generated .csv file and show data in a chart.

In fact similar to the last feature, except that I had to read the data from a file instead of calculating them for each year.

For that I used the csv.reader function which returns a list of lists, with each row in the file as a separate list.

Next I read the 2nd and the third element of each list and write it in a new list, so the result is a list for each column that can be processed by the plot function.

The last thing is to convert the list of string to a list of integer with the eval() function.

Design process and decision making

- I decided to use a graphical interface and the math plot lib to make the data visible.
- The input mask also allows the user to variate the input data without having to restart the program every time and visualise them in the chart.
- As per request from my tutor I also implemented a function to open a file that was previously generated by the program and print the data in a chart.

Development process review

- Originally I wanted to implement a program to automate the turtle game with the A* algorithm
- However given the limited time and the extensive requirements I didn’t feel confident to achieve that in 2 weeks
- The investment calculator looked simple, get the input values from the user and executing a formula with them, but creating all widgets for input and output as well as treating all the exceptions so the program doesn’t crash every time the user makes the wrong input brought me in a world of trouble.

Development process review

- I venture to say that all up id has cost me the same effort than it would have figuring out the A* algorithm and put that into a program.
- At least I learned a lot about the tkinter and matplotlib libraries. During my research I came across more professional platforms like PySide6 which allows the design by drag and drop the icons and than code generate the python part.
- In the end the app turned out as a great tool for financial planning and education and I would love to take it further to implement it as an app or make it available on my profile page. As I red, we will learn how to integrate python code into HTML next year.