Team Members: Cory Belew, Nick Titone, Sevy Van Der Werf

Requirements Specification / Project Proposal:

Our client wants us to create a Python software and publish it on GitHub. The software should:

- Define
    - A main function
    - At least one additional function
- Include all 3 types of control structures:
    - Sequencing (default)
    - Selection (if/else)
    - Iteration (while)
- Utilize
    - Numbers
    - Text
    - Variables
    - Statements
    - Mathematical expressions with arithmetic operators

The client approved our project proposal 7/15. In the proposal, we explained that we will create a **compounding interest calculator**. This calculator will:

- Ask the user to input the principal, interest rate, rate of compounding, and time period size.
- Output the final amount.

This calculator could help people compare their financial options between different savings accounts, credit cards, and loans.

Sevy was the Analyst and Designer, Cory was the Programmer, and Nick was the Tester.

Our group agreed to develop the algorithm by the end of Wednesday 7/17, finish the first iteration and its testing by Saturday 7/20, and write the final code by Wednesday 7/24. We will release the project on GitHub by the end of the due date, Sunday 7/28.

When developing our algorithm, we first used the top-down approach to create an outline of an imperative algorithm. This version of our algorithm had a lot of redundancy–for example, doing similar calculations with slight changes repeatedly.

We decided that an object-oriented approach would be better, allowing us to create classes that could be slightly changed, instead of rewriting a function each time it needed to be tweaked. In this design document, we include both a representation of our original algorithm and the final algorithm.

Team Members: Cory Belew, Nick Titone, Sevy Van Der Werf

Original Algorithmic Solution:

Declare function compound_interest()

Print Greeting
Print Instructions

Ask for the principal (please do not use a dollar sign)
    Save principal as float variable

Ask for the interest rate
Check if the number is less than 1
    Else ask for the number to be entered as a decimal
Save interest rate as variable
Convert interest rate to float

Ask user to choose rate of compounding from list of options (monthly, quarterly, or yearly)
Save rate of compounding

Ask user to input their time period size (either in months, quarters, or years depending on previous selection)
Save elapsed time

Call function compound_interest()

Display output

Final Algorithmic Solution:

Develop 4 classes:

- Create a (static) class for enumerating compound interest type (enumerating means making it so you don't have to keep rewriting the same thing)
- Create a (static) class for enumerating interest length
- Create a wrapper for each class to make the "string" variable global
    - The 2 wrappers allow us to write the way the computer will ask the user for month/quarter/year input only once.
    - The software is then able to take that information and put it into all the other functions so we don't have to keep repeating ourselves.

Develop many functions.

- Create function to get user's choice
    - Interest
    - Length

Team Members: Cory Belew, Nick Titone, Sevy Van Der Werf

    - Compound frequency/rate
    - Get principal
    - EACH of these has a function to check that the user gave a valid integer or string for the given value
- Create function to do math
    - Runs the user's choices through the formula for compounding interest
- Make main() function that takes everything and brings it together as one.

Print instructions.

Call main().

Test Document

Test Details	Success or Failure?	Meets Project Requirements?	Design Document and Program Code Agree?
7/22	SUCCESS We input similar values and our code's output matched that of this government calculator: https://www.investor.gov/financial-tools-calculators/calculators/compound-interest-calculator	Yes	Yes, we based our final algorithm off of our new code.
7/24 Found that the user was able to enter negative values as their principal amount, added an additional check for x>=0 on line 33	SUCCESS Users are now forced to enter a positive value	Yes	Yes