# The Depth of a Financial Model, Continued

Nick DeRobertis

September 30, 2020

## 1 Using Jupyter to Structure a Python Model

- It can be a bit tricky in the beginning to structure Python models in Jupyter as you are dealing with two different layers of organization
- Jupyter gives us nicely formatted markdown cells which make it easy to organize sections of the model
- Markdown is actually a general markup language, not anything specific to Jupyter, and it supports a lot of features. Jupyter has their own extension to markdown which also adds LaTeX equation support
- Most often, you will just need section headers, bullets, and equations, and anything else you can look at a Markdown reference
- It is easy to add a table of contents for a Jupyter notebook and you should do this to increase the readability of your model
- When adding a TOC item, spaces get converted to dashes for the reference

## 2 Salaries in the Python Dynamic Salary Retirement Model

- For development purposes, create a new variable data which is set equal to model_data. When you are done with the model, you will remove this.
- Write the logic for a function in a cell and run it to ensure it works, then move it into a function
- Using data in the functions while the original variable is model_data ensures that you are not accidentally accessing the overall (global) model_data when it should be the specific instance of ModelInputs being passed
- This may be confusing and sound like extra unnecessary steps, but setting things up this way will enable your model to be easily extended
- Here we will create a function which can get the salary in any given year
- We will write also some example code to test the function and show its results
- Later we will use this function in the overall calculation

## 3 Wealth in the Python Dynamic Salary Retirement Model

- Here we will develop two functions which comprise the wealth sub-model
- First create a function which determines the amount of cash saved in a given year
- Then create a function which determines the amount of wealth in a given year
- We create some example code to show how the function works, but it will actually be applied in the Retirement sub-model

# 4 Retirement in the Python Dynamic Salary Retirement Model

- Now we will bring everything together to calculate the years to retirement
- The salary and cash saved functions are already getting called from within the wealth function, so we only need to call the wealth function in the final loop
- Here we are making use of a while loop to stop the loop once a certain condition is met, in this case once wealth exceeds desired cash
- We will use formatted strings and new lines to create a good display for the output

# 5 Lab Exercise

- Feel free to work from the example model though I would recommend you build that out yourself following the prior videos
- This exercise is exactly the same as the one we did for Excel to calculate the desired cash rather than taking it as an input
- Hint: You should add the new inputs to the ModelInputs dataclass and remove the desired cash input. Then you can create a function which calculates the desired cash based on the model inputs, and use that in place of where the desired cash was being accessed directly before