Introduction to Financial Modeling

Objectives

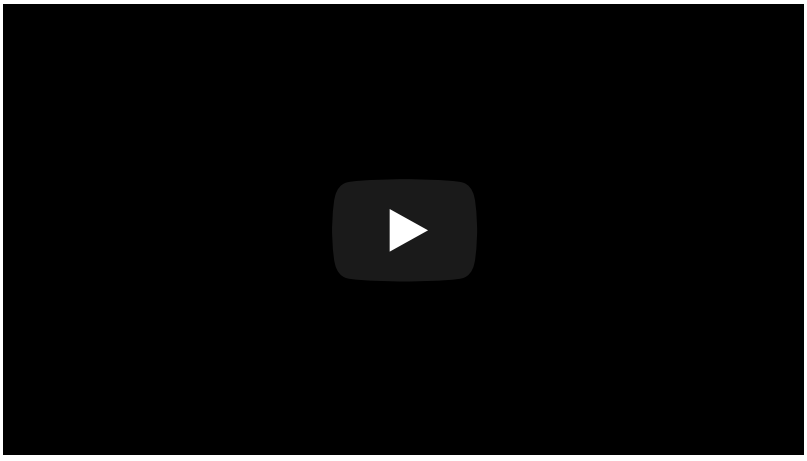
- Students will be able to explain what a financial model is.
- Students will be able to describe what the output of a financial model is used for.
- Students will be able to explain what financial models mean to executives.

A financial model is a tool that forecasts a company's financial success in the future. The income statement, balance sheet, cash flow statement, and supporting schedules are generally used to make the projection, which is based on the company's previous performance.

Whether inside or outside the firm, the output of a financial model is utilized for decision-making and financial analysis. Financial models will be used by executives inside a firm to make choices about:

- Capitalization (debt and/or equity)
- Buying (or selling) companies and/or assets
- Expanding the business (for example, by opening more stores or entering new markets).
- Assets and business units are being sold or divested.
- Planning and budgeting (for the years ahead)
- Allocation of capital (priority of which projects to invest in)
- Determining the worth of a company

Practice is the most effective approach to learn financial modeling. To become an expert at developing a financial model, you'll need years of expertise and a lot of practice. Reading equity study papers may be a good method to practice since you can compare your results to them. Using a mature company's past financials to create a flat-line model into the future and compute the net present value per share is one of the greatest methods to practice. This should be comparable to the current share price or stock research report target pricing.



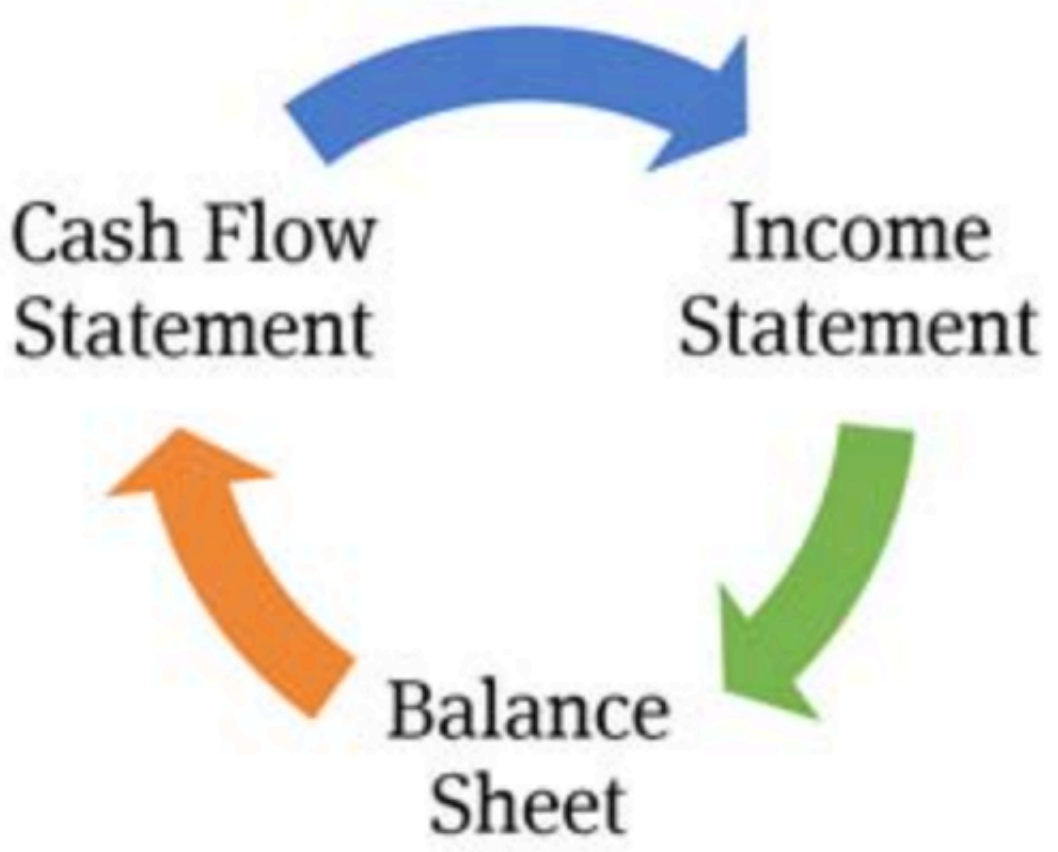
Types of Financial Models

Objectives:

- Students will be able to identify the different types of financial models.
- Students will be able to describe the different phases of building a three-statement model.
- Students will be able to detail the benefits of using a single worksheet model.
- Students will become familiar with forecasting from a financial model.
- Students will become familiar with a discounted cash flow model.
- Students will identify the various forecasting involved in a discounted cash flow model.

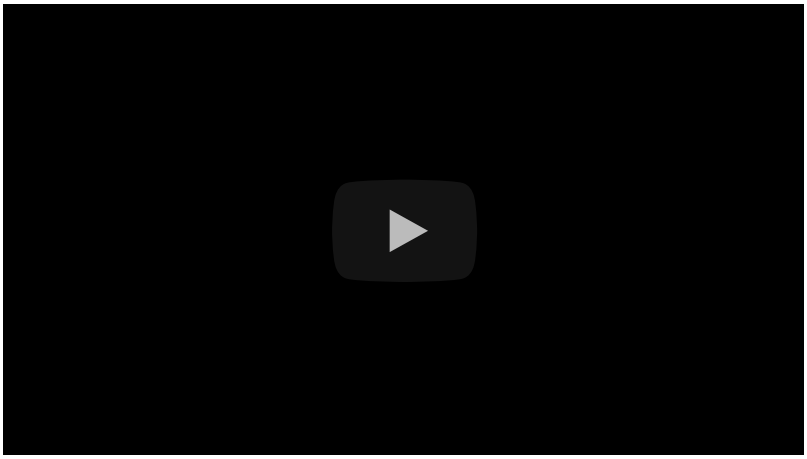
Financial models come in a variety of shapes and sizes. Financial modeling experts utilize a variety of models in corporate finance, and we'll go through some of the more prevalent ones.

3 Statement Model



An integrated 3-statement model

A three-statement model combines the income, balance, and cash flow statements into a single dynamically linked financial model. More complex financial models, such as discounted cash flow (DCF) models, merger models, leveraged buyout (LBO) models, and other forms of financial models, are developed on top of 3 statement models.



A **three-statement model** can be structured in one of two ways: single worksheet or multi-worksheet. While both techniques are valid, CFI strongly advises adopting a single worksheet structure (with grouping) for the following reasons.

The following are some of the benefits of using a single worksheet model:

- It's easier to navigate (you don't have to switch tabs)
- There's a lower chance of formulae being mixed up (all time periods are in the same column)
- The usage of grouping cells helps to keep things more orderly.
- Allow greater space for multi-business enterprises to be consolidated.

Building a three-statement model entails numerous phases, including:

- Enter historical financial data

We take the company's historical financial data and either download, type, or paste it into e.g., Excel in this stage. Once you have the data in Excel, you'll need to conduct some basic formatting to make it easier to understand and follow the structure you want for your model. The historical information is inputted in a blue font color beneath the historical time periods, as shown in the picture below.

- Determine the forecast assumptions

We can start calculating some measures to analyze the company's past performance now that we have the historical financial data in Excel and in an easy-to-use format. We must compute revenue growth, margins, capital expenditures, and working capital terms, among other things (such as accounts payable, inventory, and accounts receivable). The assumptions section, which drives the prediction, is shown below as an example.

- Forecast the income statement

After you've established your assumptions, you can begin projecting the income statement, starting with sales and working your way down to EBITDA (Earnings Before Interest Taxes Depreciation and Amortization). We'll need to build support schedules for items like capital assets and finance activities at that time.

- Forecast capital assets

Before we can conclude the income statement in the model, we must predict capital assets such as Property, Plant, and Equipment (PP&E). To do so, we start with the previous period's closing balance, add any capital expenditures, subtract depreciation, and arrive at the current period's closing balance. Depreciation may be computed in a number of different methods, including straight line, decreasing balance, and percentage of income.

- Forecast financing activities

The next step is to create a debt schedule in order to calculate interest expenditure on the income statement. We take the previous period's closing balance, add any gains or losses in principle, and arrive at the closing balance, similar to the part before. The interest expenditure can be computed based on the outstanding debt's starting balance, closing balance, or average balance. Alternatively, if one is available, a comprehensive interest payment schedule might be followed.

- Forecast the balance sheet

Except for the cash balance, which will be the last step, it is easy to complete the balance sheet in our three-statement model at this point. The forecasting of working capital items is based on assumptions about typical days payable and receivable, as well as inventory turns. Capital assets (PP&E, for example) and debt amounts are derived from the timetable above.

- Fill in the blanks on the cash flow statement.

We can build the cash flow statement and finish our three-statement model in Excel now that the balance sheet is complete (except for cash). This section is essentially completed by simply linking to items that have already been calculated in the model above. Each of the three primary parts must be completed: cash from operations, cash from investment, and cash from borrowing.

Discounted Cash Flow (DCF) Model

The Weighted Average Cost of Capital (WACC) of a company indicates the needed rate of return that its investors anticipate. As a result, it's also known as a company's opportunity cost, implying that if they can't find a better rate of return elsewhere, they should buy back their own stock.

A firm is said to be “creating value” if it obtains rates of return that are higher than its cost of capital (their hurdle rate). They are “destroying value” if their rate of return is less than their cost of capital.

The needed rate of return for investors (as described above) is usually related to the investment's risk (using the Capital Asset Pricing Model). As a result, the greater the needed rate of return and the higher the cost of capital, the riskier the venture. The more distant the cash flows, the riskier they are, and hence the more they must be discounted.

Cash Flow Forecast in a DCF Model

This is a vast subject and predicting a company's performance is an art form. In simple terms, a financial analyst's duty is to provide the best educated prediction possible about how each of a company's drivers will affect its future performance. For additional information, see our guide to assumptions and forecasts.

Except in resource or long-life sectors like mining, oil and gas, and infrastructure, where engineering reports may be utilized to construct a long-term “life of resource” projection, a DCF model forecast typically goes out five years.

1. Revenue forecasting

There are various approaches to developing a revenue projection, but they may be divided into two groups: growth-based and driver-based. A growth-based projection is simpler and more appropriate for stable, established firms that can utilize a simple year-over-year growth rate. This is adequate for many DCF models. A prediction based on drivers is more comprehensive and difficult to create. It necessitates breaking down revenue into its different drivers, which include price, volume, goods, consumers, market share, and external variables. To identify the link between underlying drivers and top-line revenue growth, regression analysis is frequently employed as part of a driver-based prediction.

2. Expense forecasting

A thorough and granular expenditure projection may be created, or a basic year-over-year comparison can be performed. The most thorough technique is known as a Zero-Based Budget, and it entails starting from zero with no regard for what was spent the previous year. Every department in the firm is often required to justify every cost they have based on activity. This method is frequently utilized in cost-cutting situations or when financial controls are being implemented. It can only be done by corporate management, not by outsiders like investment bankers or stock research analysts, because it is only practicable to do it internally.

3. Capital asset forecasting and working capital fluctuations

After you've completed most of the income statement, it's time to predict capital assets. The most thorough technique is to build a distinct timetable for each of the key capital assets in the DCF model and then combine them into a total plan. There will be multiple lines on each capital asset schedule: opening balance, depreciation, dispositions, and closing balance. Working capital changes, which include accounts receivable, payables, and inventories, must be computed and added or removed based on their financial impact.

4. Capital structure forecasting

The method you construct this part will be mainly determined by the sort of DCF model you're creating. The most typical strategy is to simply maintain the company's present capital structure, assuming no significant changes other than known factors like debt maturity. This portion of the DCF model isn't really essential because we're utilizing unlevered free cash flow. However, if you're looking at things from the standpoint of a stock investor or equities research analyst, it's critical. Enterprise value is more significant for

M&A deals, when the complete business is acquired or sold, therefore investment bankers generally focus on it.

5. Terminal Value

A DCF model's terminal value is extremely significant. It frequently accounts for more than half of the business's net present value, especially if the projection term is shorter than five years. The perpetual growth rate technique and the exit multiple approach are two methods for calculating the terminal value.

6. Cash Flow Timing

In a DCF model, it's critical to pay particular attention to the timing of cash flows because not all time periods are created equal. At the start of the model, there is typically a "stub phase" where just a fraction of the year's cash flow is received. Furthermore, the cash outflow (making the real investment) takes a long time before the stub is received.

7. Enterprise value and DCF

The NPV that you arrive at when creating a DCF model using unlevered free cash flow is always the enterprise value (EV) of the firm. This is what you'll need if you want to evaluate the entire company or compare it to comparable firms without taking their capital structures into consideration (i.e., an apples-to-apples comparison). The focus will be on enterprise value in most investment banking deals.

8. Equity value calculated using DCF

Take the net present value (NPV) of the unlevered free cash flow and adjust it for cash and equivalents, debt, and any minority stake to get the equity value of the company. You'll get the equity value, which you'll divide by the

number of shares to get the share price. This method is more popular among institutional investors and equities research analysts, who are both interested in purchasing or selling stocks.

Building a Financial Model

Objectives:

- Students will be able to carry over their knowledge of a financial model to apply to this lesson.
- Students will become familiar with the steps necessary to build the financial model.
- Students will be able to run a stress test and an audit once they have built their financial model.

It's crucial to lay up a financial model in a logical, easy-to-understand manner. This usually entails creating the entire model on a single worksheet and then grouping it into parts. This makes it simple to extend and contract the model, as well as move it about. The process of financial modeling is iterative. You must chip away at various portions until you are ultimately able to connect them altogether.

1. Preliminary findings and assumptions

Every financial model begins with a company's past performance. You start by obtaining three years of financial information and entering them into Excel to create the financial model. Then, for the historical period, you reverse engineer the assumptions by calculating things like revenue growth rate, gross margins, variable expenses, fixed costs, AR days, inventory days, and AP days, to mention a few. From there, you may use hard-codes to fill up the forecast period's assumptions.

2. Begin income statement

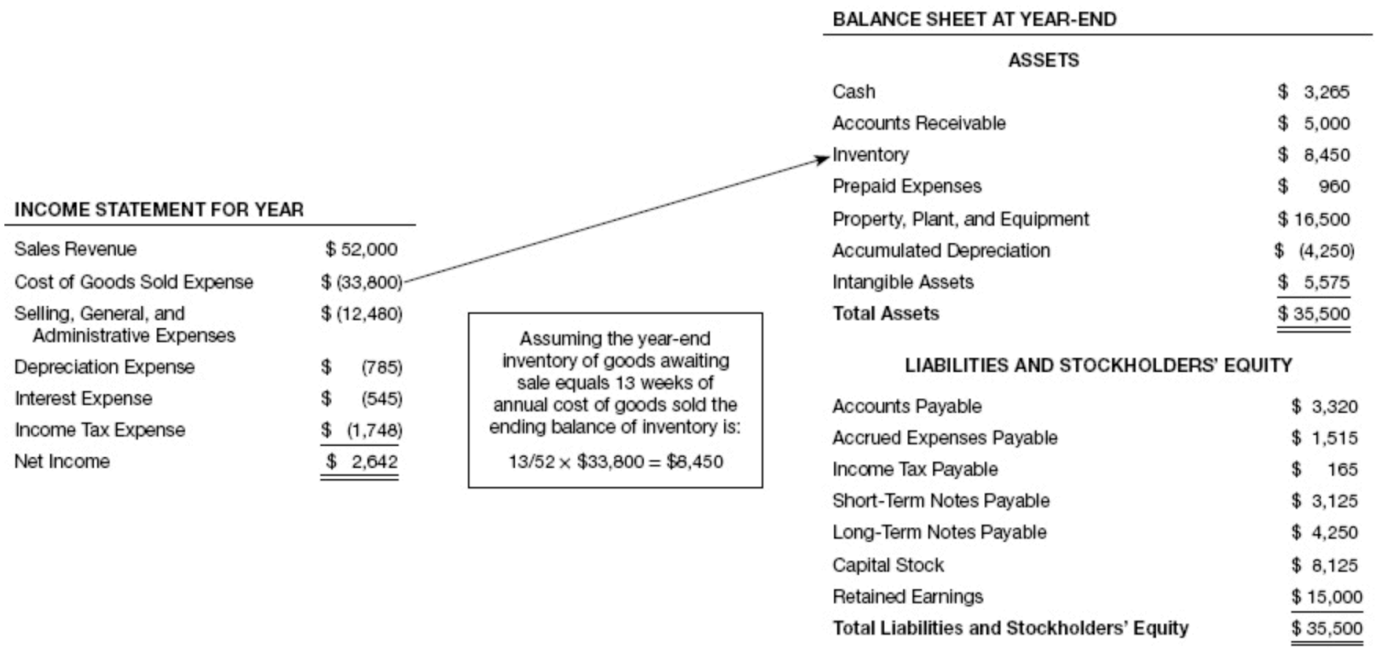
With the forecast assumptions in place, you can compute sales, COGS, gross profit, and operating expenditures all the way down to EBITDA at the

top of the income statement. Calculating depreciation, amortization, interest, and taxes will have to wait.

XYZ Corporation			
Income Statement			
For the year ended December 31,2012			
Revenues			
	Net Sales	$65,000.00	
	Dividend Revenue	$ 5,000.00	
	Total Revenues		$70,000.00
Expenses			
	Cost of Goods Sold	$15,000.00	
	Selling Expenses	$ 4,500.00	
	Administrative Expenses	$ 3,500.00	
	Income Tax Expense	$ 4,500.00	
	Total Expenses		$27,500.00
	Net Income		$42,500.00
	Earnings Per Share		$ 1.74

3. Begin balance sheet

You may begin filling up the balance sheet now that the top of the income statement is complete. Begin by computing the revenue and COGS functions of accounts receivable and inventory, as well as the AR days and inventory days assumptions. Fill in the accounts payable section, which is based on COGS and AP days.



4. Create the ancillary schedules

You must establish a schedule for capital assets, such as PP&E, as well as debt and interest, before completing the income statement and balance sheet. The PP&E schedule will increase capital expenditures and remove depreciation from the historical period. In terms of the debt schedule, it will also use past data to add debt growth and remove repayments. The average debt balance will be used to calculate interest.

Capital Assets - PP&E Schedule

	2016	2017	2018	2019	2020	2021	2022
Opening Balance							
Technology	0	240,000	220,000	190,000	510,000	370,000	1,040,000
Property & Equipment	0	160,000	160,000	150,000	130,000	220,000	230,000
Total	0	400,000	380,000	340,000	640,000	590,000	1,270,000
Capital Expenditures							
Technology	300,000	50,000	50,000	500,000	50,000	1,000,000	100,000
Property & Equipment	200,000	50,000	50,000	50,000	200,000	100,000	500,000
Total	500,000	100,000	100,000	550,000	250,000	1,100,000	600,000
Depreciation							
Technology							
2016	60,000	60,000	60,000	60,000	60,000	0	0
2017		10,000	10,000	10,000	10,000	10,000	0
2018			10,000	10,000	10,000	10,000	10,000
2019				100,000	100,000	100,000	100,000
2020					10,000	10,000	10,000
2021						200,000	200,000
2022							20,000
Subtotal	60,000	70,000	80,000	180,000	190,000	330,000	340,000
Property & Equipment							
2016	40,000	40,000	40,000	40,000	40,000	0	0
2017		10,000	10,000	10,000	10,000	10,000	0
2018			10,000	10,000	10,000	10,000	10,000
2019				10,000	10,000	10,000	10,000
2020					40,000	40,000	40,000
2021						20,000	20,000
2022							100,000
Subtotal	40,000	50,000	60,000	70,000	110,000	90,000	180,000
Total Depreciation	100,000	120,000	140,000	250,000	300,000	420,000	520,000
Closing Balance							
Technology	240,000	220,000	190,000	510,000	370,000	1,040,000	800,000
Property & Equipment	160,000	160,000	150,000	130,000	220,000	230,000	550,000
Total	400,000	380,000	340,000	640,000	590,000	1,270,000	1,350,000

5. Make a profit & loss statement and a balance sheet

The income statement and balance sheet are completed with the information from the accompanying schedules. Link depreciation to the PP&E schedule and interest to the debt schedule on the income statement. You may compute profits before taxes, taxes, and net income from there. Connect the closing PP&E balance and closing debt balance from the schedules on the balance sheet. Pulling forward last year's ending amount,

adding net income and capital raised, and deducting dividends or shares repurchased will give you shareholder's equity.

Profit and loss account for the year ended 30 April 2011

	Notes	2011 £	2010 £
Turnover	2	**384,821**	240,350
Cost of sales		**89,965**	44,942
Gross profit		**294,856**	195,408
Administrative expenses		**19,829**	36,628
		19,829	36,628
Operating profit	3	**275,027**	158,780
Other interest receivable and similar income		**-**	103
Profit on ordinary activities before taxation		**275,027**	158,883
Taxation on profit on ordinary activities	5	**57,059**	33,267
Profit for the financial year		**217,968**	125,616

6. Create a cash flow statement

After you've completed the income statement and balance sheet, you may use the reconciliation technique to create the cash flow statement. To calculate cash from operations, start with net income, subtract depreciation, and account for changes in non-cash working capital. Cash used for investing is a function of capital expenditures in the PP&E schedule, while cash used for financing is a result of the assumptions made regarding debt and equity financing.

Sample Corporation
Statement of Cash Flows
Year Ended December 31, 2020

Cash flows from operating activities	$ xxx
Cash flows from investing activities	xxx
Cash flows from financing activities	xxx
Net increase (decrease) in cash	xxx
Cash at the beginning of the year	xxx
Cash at the end of the year	$ xxx

7. Conduct a DCF analysis

After you've finished the three-statement model, you'll need to compute free cash flow and do a company appraisal. At the firm's cost of capital, the business's free cash flow is discounted back to today (its opportunity cost or required rate of return). We have a comprehensive set of courses that cover all the aforementioned processes in detail, with examples, templates, and step-by-step instructions.

	Actual				Forecast period					CAGR	CAGR
	2014A	2015A	2016A	2017A	2018E	2019E	2020E	2021E	2022E	2014-2017A	2018-2022E
Net sales	2 955	3 568	4 102	4 663	5 036	5 388	5 711	5 997	6 237	*16,4%*	*5,5%*
growth, %		*20,7%*	*15,0%*	*13,7%*	*8,0%*	*7,0%*	*6,0%*	*5,0%*	*4,0%*		
COGS	-2 098	-2 516	-2 831	-3 310	-3 544	-3 792	-4 019	-4 220	-4 389	16,4%	5,5%
Gross profit	**857**	**1 053**	**1 272**	**1 352**	**1 492**	**1 596**	**1 692**	**1 777**	**1 848**	**16,4%**	**5,5%**
margin, %	*29,0%*	*29,5%*	*31,0%*	*29,0%*	*29,6%*	*29,6%*	*29,6%*	*29,6%*	*29,6%*		
OPEX	-180	-200	-210	-215	-224	-232	-240	-249	-257	6,1%	3,5%
growth, %		*11,1%*	*5,0%*	*2,4%*	*4,0%*	*3,8%*	*3,6%*	*3,4%*	*3,3%*		
in % of net sales	*6,1%*	*5,6%*	*5,1%*	*4,6%*	*4,4%*	*4,3%*	*4,2%*	*4,1%*	*4,1%*		
EBITDA	**677**	**853**	**1 062**	**1 137**	**1 268**	**1 364**	**1 452**	**1 528**	**1 591**	***18,9%***	***5,8%***
margin, %	*22,9%*	*23,9%*	*25,9%*	*24,4%*	*25,2%*	*25,3%*	*25,4%*	*25,5%*	*25,5%*		
Depreciation	-49	-63	-75	-84	-88	-95	-100	-105	-110	19,8%	5,5%
in % of net sales	*1,6%*	*1,8%*	*1,8%*	*1,8%*	*1,8%*	*1,8%*	*1,8%*	*1,8%*	*1,8%*		
EBIT	**628**	**790**	**987**	**1 053**	**1 180**	**1 270**	**1 351**	**1 423**	**1 481**	***18,8%***	***5,9%***
Tax (30%)	-119	-138	-151	-164	-354	-381	-405	-427	-444	11,4%	5,9%
Capex		-79	-41	-169	-115	-123	-130	-136	-142	n.a.	0
in % of net sales		*2,2%*	*1,0%*	*3,6%*	*2,3%*	*2,3%*	*2,3%*	*2,3%*	*2,3%*		
Increase/Decrease in NWC		-25	-5	-2	-15	-14	-13	-11	-9		
Unlevered Free Cash Flow		**611**	**865**	**802**	**785**	**847**	**904**	**954**	**995**		

DCF-valuation	
Enterprise value ("EV")	**11 012**
Equity value ("market cap")	10 812
Price per share	1,08

Implied multiples	2017A	2018E	2019E	2020E	2021E
Sales	2,4x	2,2x	2,0x	1,9x	1,8x
EBITDA	9,7x	8,7x	8,1x	7,6x	7,2x
EBIT	10,5x	9,3x	8,7x	8,1x	7,7x

8. Include possibilities and sensitivity analysis

After you've finished with the DCF analysis and valuation portions, it's time to add sensitivity analysis and scenarios to the model. The goal of this study is to see how much changes in underlying assumptions will affect the company's worth (or some other statistic). This is particularly useful for determining the risk of an investment or for business planning (e.g., does the company need to seek money if sales volume declines by x%?).

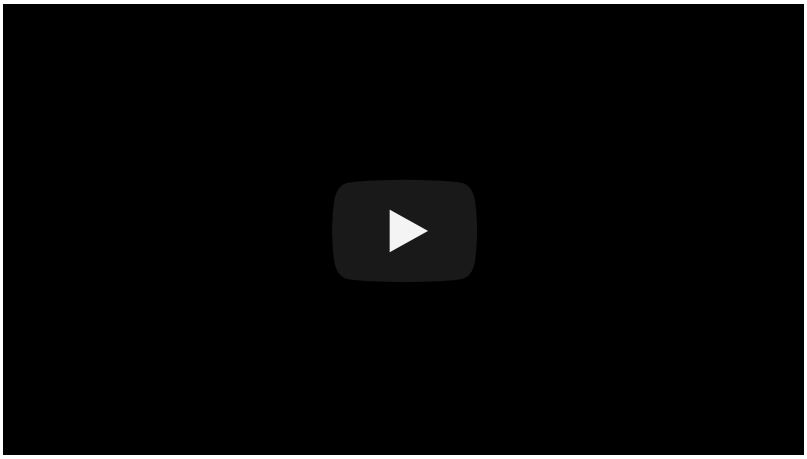
		Opportunistic		Disease Register		
	Scenarios	Mean cost/ patient compl. PACP (SD)	% change compared to base case	Mean cost/ patient compl. PACP (SD)	% change compared to base case	Differences in mean cost across delivery models (p)+
0	Base case (Figure 3a)	£53.22 (7.82)	–	£190.84 (38.98)	–	£137.62** (0.047)
1	National rollout of 'Let's get Moving' resource booklet	£20.58 (3.52)	-61.33%	£155.47 (35.18)	-18.53%	£134.89** (0.0401)
2	All consultations delivered by healthcare assistant	£44.94 (6.50)	-15.56%	£128.29 (23.74)	-32.78%	£83.35** (0.0451)
3	All support activities delivered by receptionist (NHS pay band 2)	£52.68 (7.81)	-1.01%	£178.29 (35.50)	-6.58%	£125.61** (0.0466)
4	Altering variability factors 1 to 3 simultaneously (Figure 3b)	£11.76 (1.20)	-77.9%	£80.36 (16.78)	-57.89%	£68.60** (0.0305)

9. Create graphs and charts

The ability to clearly communicate results is what distinguishes excellent financial analysts from outstanding ones. Charts and graphs are the most efficient method to display the findings of a financial model, which we go over in depth in our advanced Excel course as well as many of the specific financial modeling courses. Because most executives don't have the time or patience to investigate the model's inner workings, charts are far more useful.

10. Run the model through a stress test and an audit

Your job does not end after the model is completed. Then, to determine if the model responds as predicted, start stress-testing severe situations. It's also critical to utilize the auditing tools presented in our financial modeling fundamentals course to ensure that the data is correct and that all of the formulae are operating properly.



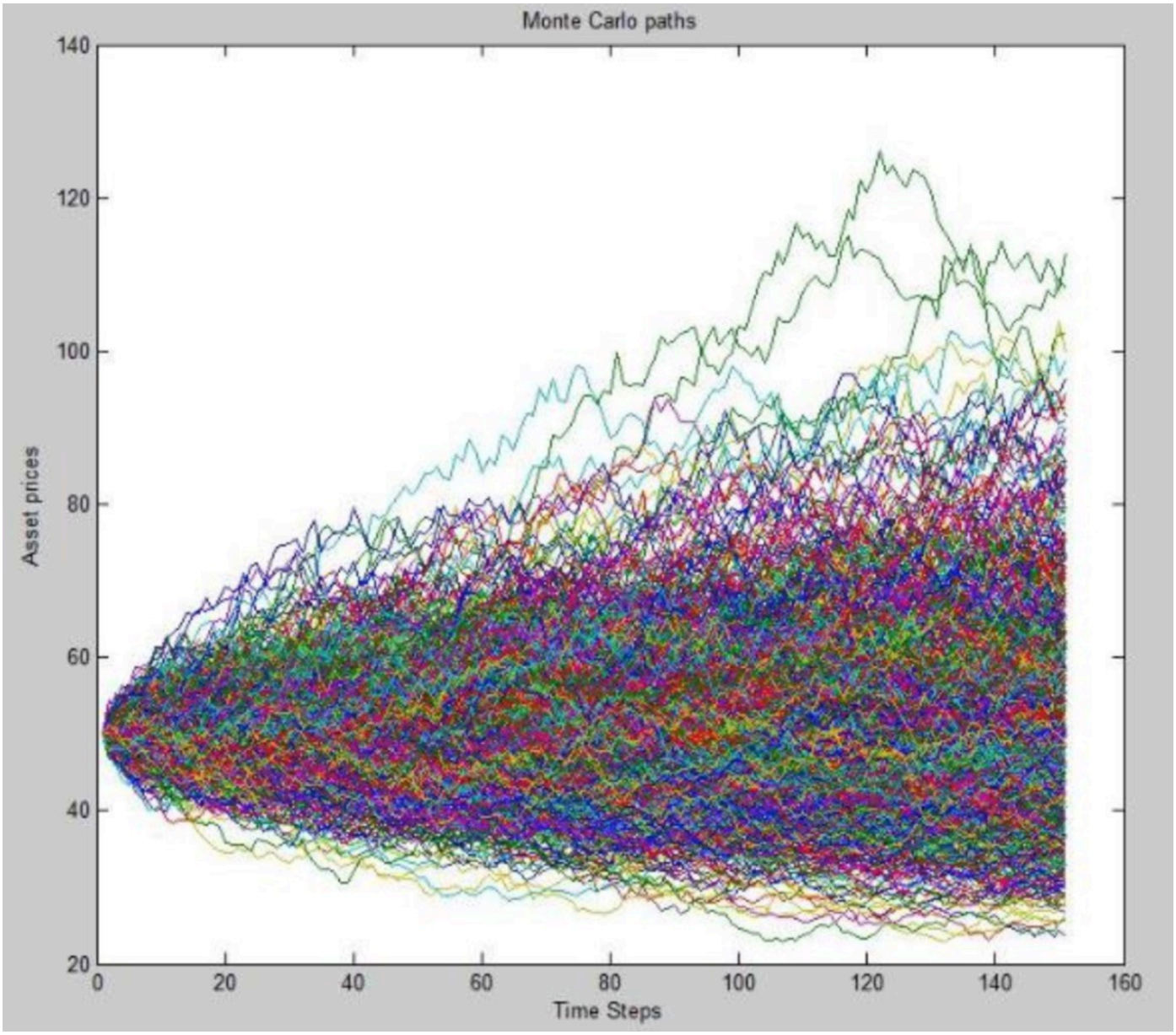
Introduction to Monte Carlo Simulations

Objectives:

- Students will be able to explain what the Monte Carlo simulation is.
- Students will be able to describe the importance of the Monte Carlo simulation.

Monte Carlo simulation most often refers to a computerized mathematical technique that allows people to account for risk in quantitative analysis and decision-making

The first occurrence of the Monte Carlo simulation method may be traced back to 1944. This approach has been subjected to several interpretations and definitions. Thus, we can conclude that it has undergone a lengthy evolution and development process. The method's ability to produce long sequences of random numbers was an early concern. In the beginning, pseudo-random numbers were employed, but as computer technology advanced, this barrier was removed.

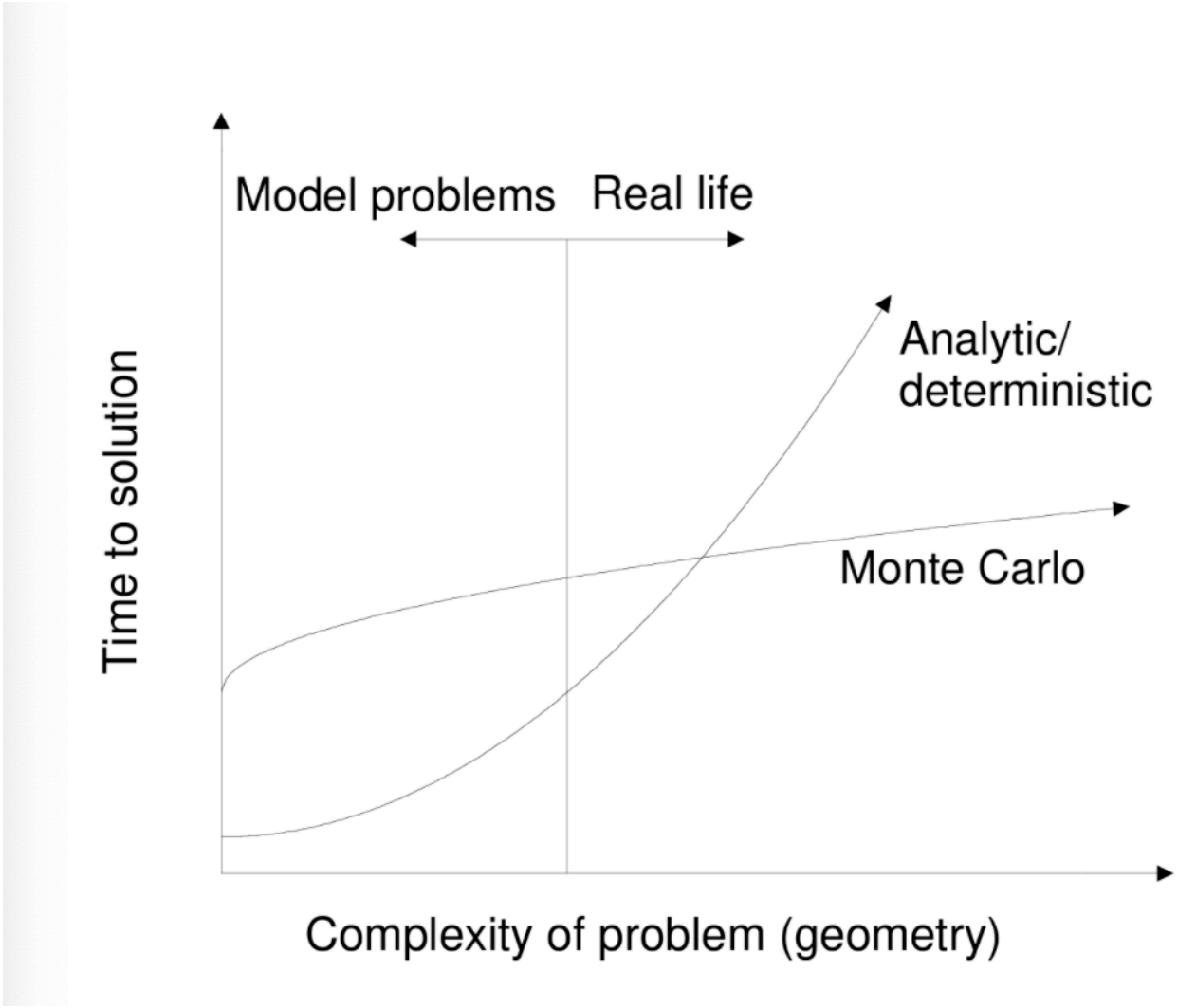


Cost Estimating Uncertainty Using Monte Carlo Techniques, published by Paul F. Dienemann in 1966 as part of a research project performed by the RAND Corporation for the US military, is one of the most important publications on the Monte Carlo approach in investment project selection. The purpose of this study was to explore how to choose a project depending on its cost. The author stressed that a single deterministic number is not a suitable selection indication, and that we need stochastic factors such as average, standard deviation, asymmetry, and others to make the best project selection decision.

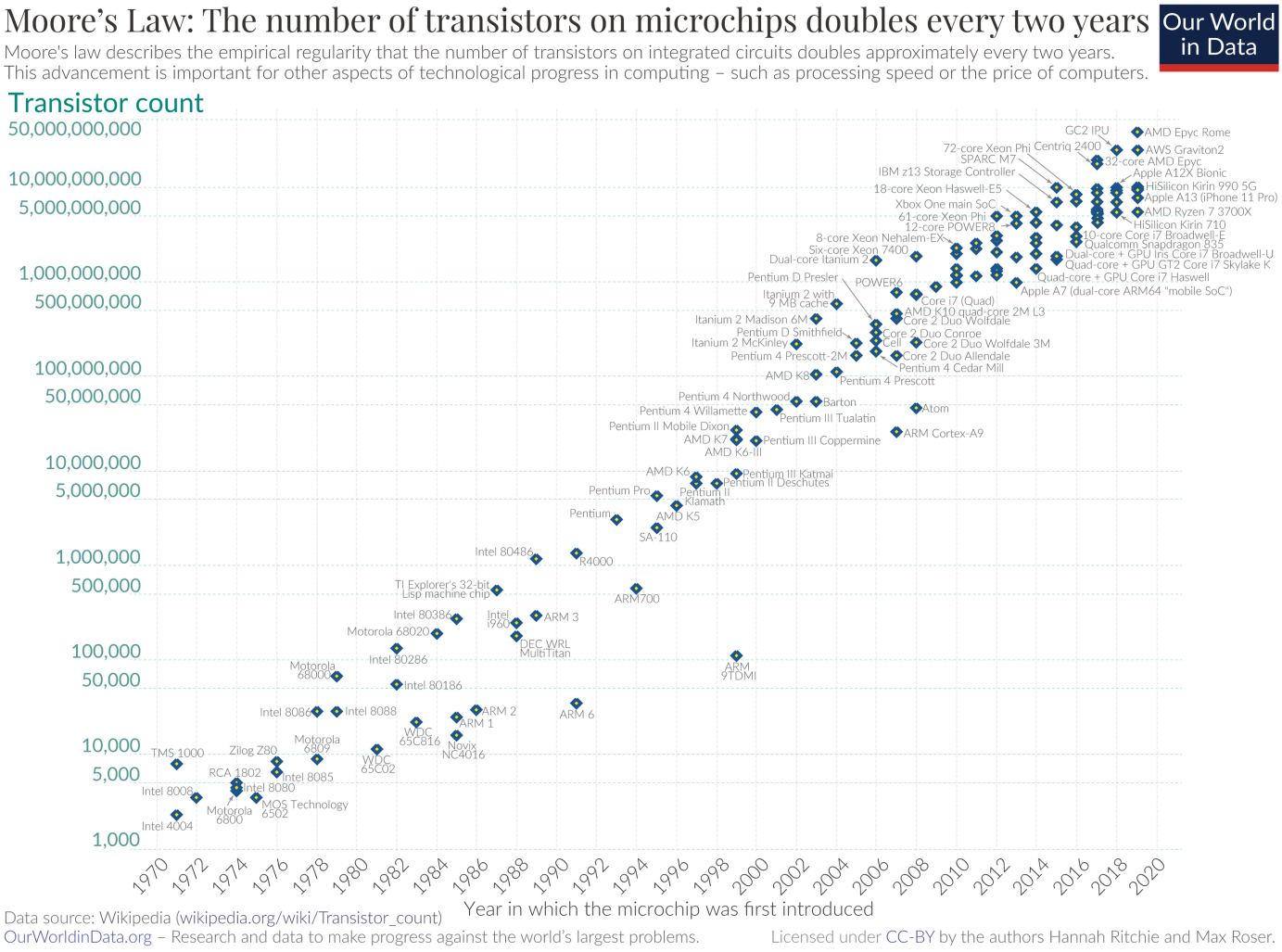
There would be considerable motive to invent Monte Carlo if it did not exist.

As previously stated, the trinity of measurement, theory, and Monte Carlo is essential to the output of both basic and applied research. Monte Carlo is sometimes viewed as a “rival” to other macroscopic computation methods, which we shall refer to as deterministic and/or analytic approaches. Although supporters of either technique might get rabid in their discussions, a scientist should first consider, "What do I want to accomplish?" and then, "What is the most effective way to achieve it?" The right answer will be “Deterministic” at times and “Monte Carlo” at other times.

However, there are two unavoidable facts. The first is that macroscopic theory, particularly transport theory, gives a wealth of information and enables for the development of sophisticated intuition about how macroscopic particle fields should behave. Monte Carlo will have a hard time competing with this. Monte Carloists are quite similar to experimentalists when it comes to finding the characteristics of macroscopic field behavior. Without theory to guide them, the process of discovery is trial and error, with occasional great intuition thrown in for good measure. When it comes to the difficulty of an issue, however, Monte Carlo approaches grow more beneficial as the complexity of the problem increases, regardless of how that is assessed.



Another unavoidable truth is that computers are becoming faster and less expensive at an exponential rate. Moore's Law is the name given to this phenomenon. The subject of when this exponential increase in computer performance will come to an end has sparked some heated discussion.



There are two methods for increasing speed. Computer circuits may be made smaller and smaller as technology develops, and signal movement within a chip takes less time. Smaller chips can also be put closer together, resulting in shorter interchip communication times. One of the issues with smaller sizes is that heat becomes an issue. This is addressed by running these circuits at lower voltages and putting these chips in sophisticated heat-dissipating assemblies, some of which have cooling fans installed directly on them. The “quantum limit” is another issue with tiny sizes. The state of a switch can grow so tiny that it is no longer well-defined (as in the classical example), and there is a risk that a switch, once thrown, would spontaneously reverse, generating undesired effects. Another advantage of compact size is that more circuitries may be packed into the same silicon real estate, allowing for the implementation of more sophisticated signal processing algorithms. Furthermore, ongoing study into the design of these

algorithms is leading in more efficient processing per transistor. All of these factors combine to produce the exponential rise in computer speed per unit cost that we witness today.

Other doubters claim that market pressures aren't in favor of faster, less expensive computers. Fast computing has historically been driven by the demand for it in science, major business, and the military. The attractiveness of personal computers to the home market and their accessibility to small businesses are driving their rise. The personal computer has been so successful that the mainframe computer business has been pushed into niche industries. Some believe that consumers, particularly in home markets, will eventually stop driving demand so hard. What is the minimum speed need for a home computer? is a common comment made from this viewpoint. However, because applications often evolve to keep up with new technology, this claim may be unfounded. In the housing market, “keeping up with the Joneses” is still an issue.

Another advantage of Monte Carlo is that it uses a minimal quantity of data and a maximum number of floating-point operations. Maximum data and minimum floating-point operation techniques are frequently used in deterministic computations. Because communication bottlenecks, whether from the CPU to cache memory, cache memory to main memory, or main memory to large storage devices (typically disk), are frequently the cause of data processing bottlenecks, modern computer architecture favors the Monte Carlo model, which emphasizes iteration and minimizes data storage. Monte Carlo is just another weapon in the theorist's or experimentalist's arsenal. The significance of analytical progress cannot be overstated.

Because of the minimal computing work required in comparison to the difficulty of the issues that may be addressed, this approach is now useful for solving a wide range of problems with little effort. The Monte Carlo approach uses a random uniformly distributed number generator in the [0, 1] interval to produce fake values for a probabilistic variable, as well as the

cumulative distribution function associated with these stochastic variables.

Economic decision simulation may be used to solve a variety of problems involving operational rules, policies, and procedures, such as those involving choice adaptation, decision control, and pricing policy. The application of simulation techniques is not, in reality, a decision-making process. In order to reach the goal, solving issues using simulation approaches necessitates the employment of interactive algorithms and the existence of well-defined phases. Random variables produced by a random number generator are commonly used as input data.

The approach algorithm is demonstrated in five phases, each of which is interactive:

- **Phase 1:** Create a parametric model, y = f (x1, x2, ..., xq)
- **Phase 2:** Create a random input set of data, xi1, xi2, ..., xiq
- **Phase 3:** Calculate effectively and memorize results as yi
- **Phase 4:** Repeat steps 2 and 3 for i = 1 to n (n >= 5000)
- **Phase 5:** Analyze the findings using histograms, confidence intervals, and other statistical indicators generated by the simulation, among other things.

A parametric deterministic model creates a collection of input variables that are then reported to a set of output variables. The input variables are randomized (being represented by a random distribution) in the stochastic model of uncertainty propagation, and the outcome will be random as well, generally following the Normal Distribution. Monte Carlo simulation is based on this idea.

On a final note, under these assumptions, we know that the least squares estimators are the best linear unbiased estimators of the regression parameters. And if the random errors are normal, we may be certain that the estimators have normal distributions in repeated experiments. It's tough to comprehend the concept of "repeated trials." Random number generators

are used in Monte Carlo simulation experiments to imitate the random nature of data collection. We define a data creation technique and produce samples of fake data in Monte Carlo simulations. Then we use the data we've collected to "test out" estimate approaches. We generate a large number of N-samples and investigate the estimators' repeated sampling characteristics. This allows us to investigate how statistical techniques operate in perfect and less-than-ideal situations. This is critical since economic, commercial, and social science facts aren't always (or even generally) as lovely as our assumptions.

For the simple linear regression model, the data creation procedure is as follows:

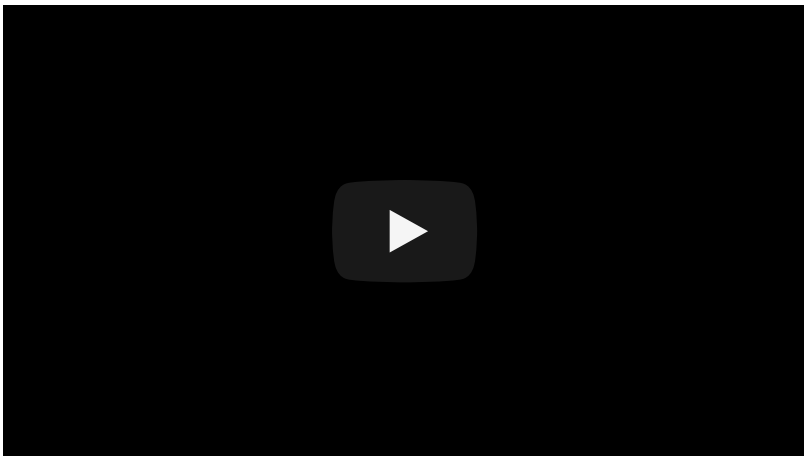
$y_i = E(y_i|x_i) + e_i = \beta_1 + \beta_2 x_i + e_i, i = 1, \ldots, N$

By adding a random error ei to the regression function E, each value of the dependent variable yi is acquired, or produced (yi|xi). We construct values for the systematic component of the regression relationship E(yi|xi) and add the random error ei to mimic yi values. This is similar to a physical experiment in which the variables are fixed, and the experiment is conducted. Because of random uncontrolled mistakes, the outcome varies from trial to trial.

Monte Carlo Simulations

Objectives:

- Students will be able to summarize the video on the Monte Carlo Simulation (MIT OpenCourseWare).



The following content is from https://www.toptal.com/finance/financial-modeling/python-and-finance

It is a step-by-step tutorial with using Python and finance together, giving you insight on how Stefan Thelin, who chronicled this tutorial, creates a simplified version of the Monte Carlo simulation with Python.

Step-by-step Tutorial of Using Python and Finance Together

What follows is a step-by-step tutorial showing how to create a simplified version of the Monte Carlo simulation but using Python instead of the @RISK plugin for Excel.

Monte Carlo methods rely on random sampling to obtain numerical results. One such application is to draw random samples from a probability distribution representing uncertain potential future states of the world where

variables or assumptions can take on a range of values.

It is helpful to do the Monte Carlo simulation on a simplified DCF valuation model instead of the more common examples you see showing valuation of options or other derivatives, since for this we don't need any math beyond the basics of calculating the financial statements and discounting cash flows, allowing us to focus on the Python concepts and tools. Please note though that this basic tutorial model is meant to illustrate the key concepts, and is not useful as-is for any practical purposes. I also won't touch on any of the more academic aspects of Monte Carlo simulations.

The tutorial assumes that you are familiar with the basic building blocks of programming, such as variables and functions. If not, it might be helpful to take 10 minutes to check the key concepts in for example this introduction.

The Starting Point and Desired Outcome

I start with the same very simplified DCF valuation model used in the Monte Carlo simulation tutorial. It has some key line items from the three financial statements, and three highlighted input cells, which in the Excel version have point estimates that we now want to replace with probability distributions to start exploring potential ranges of outcomes.

The intention of this tutorial is to give finance professionals new to Python an introduction not only to what a useful program might look like, but an introduction also to the iterative process you can use to develop it. It, therefore, has two parts:

1. First, I develop a working prototype using a straightforward approach which I think is easy to follow and not completely unlike the process one could use to start this project if you were to start from scratch.
2. Then, after having developed the working prototype, I walk through the process of refactoring - changing the structure of code without changing its functionality. You may want to stick around for that part - it

is a more elegant solution than the first one, and, as a bonus, it is about 75x faster in terms of execution time.

1. Developing a Working Prototype

Setting up the Jupyter Notebook

The Jupyter notebook is a great tool for working with Python interactively. It is an interactive Python interpreter with cells that can contain code, Markdown text, images, or other data. For this tutorial I used the Python Quant Platform, but I can also recommend Colaboratory by Google, which is free and runs in the cloud. Once there, simply select "New Python 3 Notebook" in the "File" menu, and you are ready to go.

Having done that, the next step is to import the third-party packages we need for data manipulation and visualizations, and tell the program that we want to see charts inline in our notebook, instead of in separate windows:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

A note before we start naming our first variables. As I already highlighted, readability is one of Python's strengths. Language design goes a long way to support that, but everyone writing code is responsible for making it readable and understandable, not only for others but also for themselves.
As Eagleson's Law states, "Any code of your own that you haven't looked at for six or more months might as well have been written by someone else."

A good rule of thumb is to name the components of your program in such a way that you minimize the need for separate comments that explain what your program does.

With that in mind, let's move on.

Creating the Financial Statements

There are many ways that we can work with existing spreadsheet data in Python. We could, for example, read a sheet into a Pandas DataFrame with one line of code using the `read_excel` command. If you want a tighter integration and real-time link between your spreadsheet and Python code, there are both free and commercial options available to provide that functionality.

Since the model here is very simple, and to focus us on the Python concepts, we will be recreating it from scratch in our script. At the end of the first part, I will show how you can export what we have created to a spreadsheet.

As a first step towards creating a Python representation of the financial statements, we will need a suitable data structure. There are many to choose from, some built into Python, others from various libraries, or we can create our own. For now, let's use a Series from the Pandas library to have a look at its functionality:

```
years = ['2018A', '2019B', '2020P', '2021P', '2022P', '2023P']
sales = pd.Series(index=years)
sales['2018A'] = 31.0  sales
```

This input and its corresponding output is shown below:

```
In [2]: years = ['2018A', '2019B', '2020P', '2021P', '2022P', '2023P']
        sales = pd.Series(index=years)
        sales['2018A'] = 31.0
        sales

Out[2]: 2018A    31.0
        2019B     NaN
        2020P     NaN
        2021P     NaN
        2022P     NaN
        2023P     NaN
        dtype: float64
```

With the first three lines we have created a data structure with an index consisting of years (each marked to show if it is Actual, Budget or Projected), a starting value (in millions of euros, as in the original DCF model), and empty (NaN, “Not a Number”) cells for the projections. The fourth line prints a representation of the data - in general, typing the name of a variable or other objects in the interactive interpreter will usually give you a sensible representation of it.

Next, we declare a variable to represent the projected annual sales growth. At this stage, it is a point estimate, the same figure as in our original DCF model. We want to first use those same inputs and confirm that our Python version performs the same and gives the same result as the Excel version, before looking at replacing point estimates with probability distributions. Using this variable, we create a loop that calculates the sales in each year of the projections based on the previous year and the growth rate:

```
growth_rate = 0.1
for year in range(1, 6):    sales[year] = sales[year - 1] * (1
+ growth_rate)    sales
```

We now have projected sales, instead of NaN:

```
In [3]: growth_rate = 0.1

        for year in range(1, 6):
            sales[year] = sales[year - 1] * (1 + growth_rate)

        sales

Out[3]: 2018A    31.00000
        2019B    34.10000
        2020P    37.51000
        2021P    41.26100
        2022P    45.38710
        2023P    49.92581
        dtype: float64
```

Using the same approach, we continue through the financial statements, declaring variables as we need them and performing the necessary calculations to eventually arrive at free cash flow. Once we get there we can check that what we have corresponds to what the Excel version of the DCF model says.

```
ebitda_margin = 0.14
depr_percent = 0.032
ebitda = sales * ebitda_margin
depreciation = sales * depr_percent
ebit = ebitda - depreciation
nwc_percent = 0.24
nwc = sales * nwc_percent
change_in_nwc = nwc.shift(1) - nwc
capex_percent = depr_percent
capex = -(sales * capex_percent)
tax_rate = 0.25
tax_payment = -ebit * tax_rate
tax_payment = tax_payment.apply(lambda x: min(x, 0))
free_cash_flow = ebit + depreciation + tax_payment + capex +
change_in_nwc
free_cash_flow
```

This gives us the free cash flows:

```
In [4]: ebitda_margin = 0.14
        depr_percent = 0.032

        ebitda = sales * ebitda_margin
        depreciation = sales * depr_percent
        ebit = ebitda - depreciation
```

```
In [5]: nwc_percent = 0.24

        nwc = sales * nwc_percent
        change_in_nwc = nwc.shift(1) - nwc
        capex_percent = depr_percent
        capex = -(sales * capex_percent)

        tax_rate = 0.25

        tax_payment = -ebit * tax_rate
        tax_payment = tax_payment.apply(lambda x: min(x, 0))
        free_cash_flow = ebit + depreciation + tax_payment + capex + change_in_nwc

        free_cash_flow
```

```
Out[5]: 2018A         NaN
        2019B    2.018100
        2020P    2.219910
        2021P    2.441901
        2022P    2.686091
        2023P    2.954700
        dtype: float64
```

The one line above that perhaps needs a comment at this stage is the second `tax_payment` reference. Here, we apply a small function to ensure that in scenarios where profit before tax becomes negative, we won't then have a positive tax payment. This shows how effectively you can apply custom functions to all cells in a Pandas Series or DataFrame. The actual function applied is, of course, a simplification. A more realistic model for a larger valuation exercise would have a separate tax model that calculates actual cash taxes paid based on a number of company-specific factors.

Performing the DCF Valuation

Having arrived at projected cash flows, we can now calculate a simple terminal value and discount all cash flows back to the present to get the DCF result. The following code introduces indexing and slicing, which allows us to access one or more elements in a data structure, such as the Pandas Series object.

We access elements by writing square brackets directly after the name of the structure. Simple indexing accesses elements by their position, starting with zero, meaning that `free_cash_flow[1]` would give us the second element. `[-1]` is shorthand for accessing the last element (the last year's cash flow is used to calculate the terminal value), and using a colon gives us a slice, meaning that `[1:]` gives us all elements except the first one, since we don't want to include the historical year `2018A` in our DCF valuation.

```
cost_of_capital = 0.12
terminal_growth = 0.02
terminal_value = ((free_cash_flow[-1] * (1 + terminal_growth))
/
                  (cost_of_capital - terminal_growth))
discount_factors = [(1 / (1 + cost_of_capital)) ** i for i in
range (1,6)]
dcf_value = (sum(free_cash_flow[1:] * discount_factors) +
             terminal_value * discount_factors[-1])
dcf_value
```

```
In [6]: cost_of_capital = 0.12
        terminal_growth = 0.02

        terminal_value = ((free_cash_flow[-1] * (1 + terminal_growth)) /
                          (cost_of_capital - terminal_growth))

        discount_factors = [(1 / (1 + cost_of_capital)) ** i for i in range (1,6)]

        dcf_value = (sum(free_cash_flow[1:] * discount_factors) +
                     terminal_value * discount_factors[-1])

        dcf_value

Out[6]: 25.794384011137922
```

That concludes the first part of our prototype - we now have a working DCF model, albeit a very rudimentary one, in Python.

Exporting the Data

Before moving on to the actual Monte Carlo simulation, this might be a good time to mention the exporting capabilities available in the Pandas package. If

you have a Pandas DataFrame object, you can write that to an Excel file with one line using the `to_excel` method. There is similar functionality to export to more than a dozen other formats and destinations as well.

```
output = pd.DataFrame([sales, ebit, free_cash_flow],
                      index=['Sales', 'EBIT', 'Free Cash
Flow']).round(1)
output.to_excel('Python DCF Model Output.xlsx')
output
```

```
In [7]: output = pd.DataFrame([sales, ebit, free_cash_flow],
                              index=['Sales', 'EBIT', 'Free Cash Flow']).round(1)

        output.to_excel('Python DCF Model Output.xlsx')

        output
```

Out[7]:

	2018A	2019B	2020P	2021P	2022P	2023P
Sales	31.0	34.1	37.5	41.3	45.4	49.9
EBIT	3.3	3.7	4.1	4.5	4.9	5.4
Free Cash Flow	NaN	2.0	2.2	2.4	2.7	3.0

Creating Probability Distributions for Our Monte Carlo Simulation

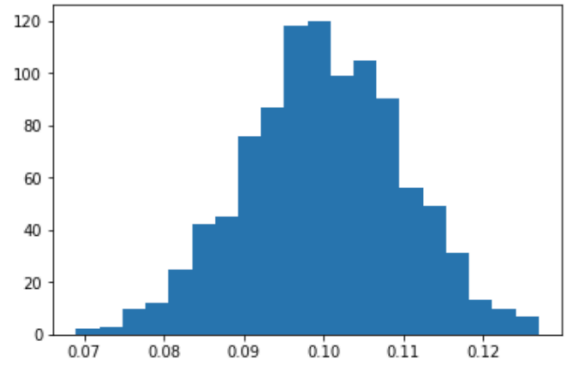
Now we are ready to tackle the next challenge: to replace some of the point estimate inputs with probability distributions. While the steps up to this point may have seemed somewhat cumbersome compared to building the same model in Excel, these next few lines will give you a glimpse of how powerful Python can be.

Our first step is to decide how many iterations we want to run in the simulation. Using 1,000 as a starting point strikes a balance between getting enough data points to get sensible output plots, versus having the simulation finish within a sensible time frame. Next, we generate the actual distributions. For the sake of simplicity, I generated three normal distributions here, but the NumPy library has a large number of

distributions to choose from, and there are other places to look as well, including the Python standard library. After deciding which distribution to use, we need to specify the parameters required to describe their shape, such as mean and standard deviation, and the number of desired outcomes.

```
iterations = 1000
sales_growth_dist = np.random.normal(loc=0.1, scale=0.01,
size=iterations)
ebitda_margin_dist = np.random.normal(loc=0.14, scale=0.02,
size=iterations)
nwc_percent_dist = np.random.normal(loc=0.24, scale=0.01,
size=iterations)
plt.hist(sales_growth_dist, bins=20)
plt.show()
```

```
In [8]: iterations = 1000
        sales_growth_dist = np.random.normal(loc=0.1, scale=0.01, size=iterations)
        ebitda_margin_dist = np.random.normal(loc=0.14, scale=0.02, size=iterations)
        nwc_percent_dist = np.random.normal(loc=0.24, scale=0.01, size=iterations)

        plt.hist(sales_growth_dist, bins=20)
        plt.show()
```



Here you could argue that EBITDA should not be a separate random variable independent from sales but instead correlated with sales to some degree. I would agree with this, and add that it should be driven by a solid understanding of the dynamics of the cost structure (variable, semi-variable and fixed costs) and the key cost drivers (some of which may have their own probability distributions, such as for example input commodities prices), but I leave those complexities aside here for the sake of space and clarity.

The less data you have to inform your choice of distribution and parameters, the more you will have to rely on the outcome of your various due diligence workstreams, combined with experience, to form a consensus view on ranges of likely scenarios. In this example, with cash flow projections, there will be a large subjective component, which means that visualizing the probability distributions becomes important. Here, we can get a basic visualization, showing the sales growth distribution, with only two short lines of code. This way we can quickly view any distribution to eyeball one that best reflects the team's collective view.

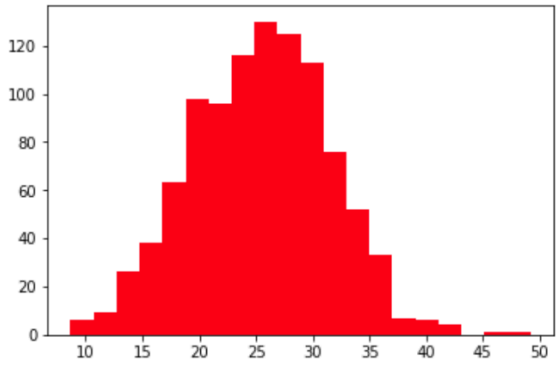
Now we have all the building blocks we need to run the simulation, but they are not in a convenient format for running the simulation. Here is the same code we have worked with thus far but all gathered in one cell and rearranged into a function for convenience:

```
def run_mcs():

    # Create probability distributions
    sales_growth_dist = np.random.normal(loc=0.1, scale=0.01,
size=iterations)
    ebitda_margin_dist = np.random.normal(loc=0.14, scale=0.02,
size=iterations)
    nwc_percent_dist = np.random.normal(loc=0.24, scale=0.01,
size=iterations)

    # Calculate DCF value for each set of random inputs
    output_distribution = []
    for i in range(iterations):
        for year in range(1, 6):
            sales[year] = sales[year - 1] * (1 +
sales_growth_dist[0])
        ebitda = sales * ebitda_margin_dist[i]
        depreciation = (sales * depr_percent)
        ebit = ebitda - depreciation
        nwc = sales * nwc_percent_dist[i]
```

```
        change_in_nwc = nwc.shift(1) - nwc
        capex = -(sales * capex_percent)
        tax_payment = -ebit * tax_rate
        tax_payment = tax_payment.apply(lambda x: min(x, 0))
        free_cash_flow = ebit + depreciation + tax_payment +
capex + change_in_nwc

        # DCF valuation
        terminal_value = (free_cash_flow[-1] * 1.02) /
(cost_of_capital - 0.02)
        free_cash_flow[-1] += terminal_value
        discount_factors = [(1 / (1 + cost_of_capital)) ** i
for i in range (1,6)]
        dcf_value = sum(free_cash_flow[1:] * discount_factors )
        output_distribution.append(dcf_value)

    return output_distribution
```

We can now run the whole simulation and plot the output distribution, which will be the discounted cash flow value of this company in each of the 1,000 iterations, with the following code. The command is not Python code but a notebook shorthand that measures the time to run something (you could instead use the Python function from the standard library). It depends on the computer you run it on, but this version needs 1-2 seconds to run the 1,000 iterations and visualize the outcome.

```
In [10]: %time plt.hist(run_mcs(), bins=20, color='r')
         plt.show()
CPU times: user 1.38 s, sys: 4 ms, total: 1.38 s
Wall time: 1.35 s
```



For more information on EBITDA

2. Refining the Prototype

Refactoring refers to the process of rewriting existing code to improve its structure without changing its functionality, and it can be one of the most fun and rewarding elements of coding. There can be several reasons to do this. It might be to:

1. Organize the different parts in a more sensible way.
2. Rename variables and functions to make their purpose and workings clearer.
3. Allow and prepare for future features.
4. Improve the execution speed, memory footprint or other resource utilization.

To show what one step in that process might look like, I cleaned up the prototype that we just walked through by collecting all initial variables in one place, rather than scattered throughout as in the prototype script, and optimized its execution speed through a process called *vectorization*.

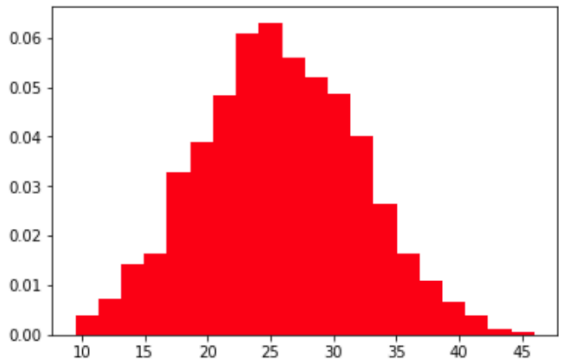
It now looks cleaner and easier to understand:

```
# Key inputs from DCF model
years = 5
starting_sales = 31.0
capex_percent = depr_percent = 0.032
sales_growth = 0.1
ebitda_margin = 0.14
nwc_percent = 0.24
tax_rate = 0.25
# DCF assumptions
r = 0.12
g = 0.02
# For MCS model
iterations = 1000
sales_std_dev = 0.01
ebitda_std_dev = 0.02
nwc_std_dev = 0.01


def run_mcs():

    # Generate probability distributions
    sales_growth_dist = np.random.normal(loc=sales_growth,
                                         scale=sales_std_dev,
                                         size=(years,
iterations))
    ebitda_margin_dist = np.random.normal(loc=ebitda_margin,
                                          scale=ebitda_std_dev,
                                          size=(years,
iterations))
    nwc_percent_dist = np.random.normal(loc=nwc_percent,
                                        scale=nwc_std_dev,
                                        size=(years,
iterations))

    # Calculate free cash flow
    sales_growth_dist += 1
    for i in range(1, len(sales_growth_dist)):
        sales_growth_dist[i] *= sales_growth_dist[i-1]
```

```
    sales = sales_growth_dist * starting_sales
    ebitda = sales * ebitda_margin_dist
    ebit = ebitda - (sales * depr_percent)
    tax = -(ebit * tax_rate)
    np.clip(tax, a_min=None, a_max=0)
    nwc = nwc_percent_dist * sales
    starting_nwc = starting_sales * nwc_percent
    prev_year_nwc = np.roll(nwc, 1, axis=0)
    prev_year_nwc[0] = starting_nwc
    delta_nwc = prev_year_nwc - nwc
    capex = -(sales * capex_percent)
    free_cash_flow = ebitda + tax + delta_nwc + capex
    # Discount cash flows to get DCF value
    terminal_value = free_cash_flow[-1] * (1 + g) / (r - g)
    discount_rates = [(1 / (1 + r)) ** i for i in range (1,6)]
    dcf_value = sum((free_cash_flow.T * discount_rates).T)
    dcf_value += terminal_value * discount_rates[-1]

    return dcf_value
```

The main difference you will notice between this version and the previous one is the absence of the `for i in range(iterations)` loop. Using NumPy's array operation, this version runs in 18 milliseconds compared to the 1.35 seconds for the prototype version - roughly 75x faster.

```
In [4]: %time plt.hist(run_mcs(), bins=20, density=True, color="r")
        plt.show()
CPU times: user 20 ms, sys: 0 ns, total: 20 ms
Wall time: 18 ms
```



I'm sure that further optimization is possible, since I put together both the prototype and refined version in a short time solely for the purpose of this tutorial.

Taking it Further

This tutorial showed some of the powerful features of Python, and if you were to develop this further the opportunities are almost endless. You could for example:

- Scrape or download relevant company or sector statistics from web pages or other data sources, to help inform your choice of assumptions and probability distributions.
- Use Python in quantitative finance applications, such as in an automated trading algorithm based on fundamental and/or macroeconomic factors.
- Build exporting capabilities that generate output in a spreadsheet and/or presentation format, to be used as part of your internal transaction review and approval process, or for external presentations.

I haven't even touched upon what you could also do with the various web, data science, and machine learning applications that have contributed to Python's success.

In Summary: A Useful Language for Your Financial Toolbox

This article gave an introduction to the Python programming language, listed some of the reasons why it has become so popular in finance and showed how to build a small Python script. In a step-by-step tutorial, I walked through how Python can be used for iterative prototyping, interactive financial analysis, and for application code for valuation models, algorithmic trading programs and more.

For me, at the end of the day, the killer feature of Python technology is that it is simply fun to work with! If you enjoy problem-solving, building things and making workflows more efficient, then I encourage you to try it out. I would love to hear what you have done with it or would like to do with it.

Introduction to Statistical Use Cases in FinTech

Objectives:

- Students will complete an extensive review of current and potential statistical use cases in FinTech.

Cutting-edge innovations like blockchain and artificial intelligence are ushering in new ways of doing business. Fields including electronic payments, banking, insurance, personal loans, and wealth management are all getting a digital facelift. Statistics show that the tide is turning as established companies realize the potential and necessity of these new technologies.In recent years, the financial industry has experienced disruptions by Fintech which are related to providing consumers with alternatives to traditional options.

Consumers expect to have a seamless digital experience when handling their money and investments. To stay afloat, financial companies must provide such services to their consumers. More and more now, there are partnerships being established between traditional companies and Fintech startups to provide a business-to-business approach. This is to offer its technology to larger companies and more consumers.

FinTech companies depend, heavily, on machine learning, artificial intelligence (AI), predictive analysis and data science to make financial decisions simplified while providing optimal solutions. Data science is used in FinTech through the following examples:

- **Robo-Advisors** - process starts with collecting information about the client through an online survey where the client's profile is captured, such as their financial status, risk capacity, future financial goals, etc and then the data is used to processed to provide financial advice or automatically invest client assets in instruments and asset classes best suited for their needs and goals.
- **Risk Analysis** - use logistic regression to predict the risk of customers and separate good borrowers from bad ones.
- **Fraud Detection** - leverage big data and data analytics techniques where vast amounts of online fraudulent transactions can be used and modeled in a way that can help us flag or predict fraud in future transactions.
- **Customer Acquisition and Retention** - an algorithm could be built to predict what additional products or services the customer would like to purchase based on their historical purchase behavior.
- **Insurance Products** - claims department in an insurance company uses data science algorithms to separate fraudulent from non-fraudulent transactions.

Scientific Libraries & NumPy

Objectives:

- Students will review the benefits of libraries for FinTech programming
- Students will review the role of NumPy FinTech programming

A large amount of data science and machine learning is built on a few specific branches of mathematics. Specifically, Linear Algebra and Statistics. Linear Algebra deals with computation involving matrices. The Python libraries used to conduct efficient matrix calculations are called Numpy and Scipy.

NumPy

NumPy is a library for Python that supports large multidimensional arrays and matrices. Later on, we will manipulate datasets using the Pandas library which is built around NumPy. NumPy has a lot of utility and is able to perform complex mathematical operations quickly and efficiently. In Machine Learning using Python, these arrays will be the data structure used for handing off our transformed data to our Machine Learning algorithm with the help of Scikit-Learn.

Learning the basics of NumPy will help you become a data manipulation master and is a fast way to perform large amounts of matrix operations locally.

Additional Resources:

- Documentation
- NumPy Reference PDF
- Wikipedia
- Chapter 2: Introduction to NumPy

Quiz Review

Correct answers are in bold.

NumPy is a Python Library that supports large multidimensional arrays and matrices.

- **A. True**
- B. False

NumPy is not efficient for manipulating data.

- A. True
- **B. False**

Intro: Financial Modeling using Python

Objectives:

- Students will be able to provide explanations on how Python is used for Financial Modeling.
- Students will be able to apply Python in a Financial Modeling scenario.

Financial modeling using Python is a method of building a model using the Python programming language. The language allows coders to modify and analyze Excel spreadsheets and automate certain tasks. As an example, the task of copying data from one spreadsheet to another can be automated with code, and searching for errors can be dramatically sped up.

Python is one of the popular programming languages used in finance. Guido van Rossum created Python, which was released for the first time in 1991. Currently, it is one of the programming languages used in financial modeling .

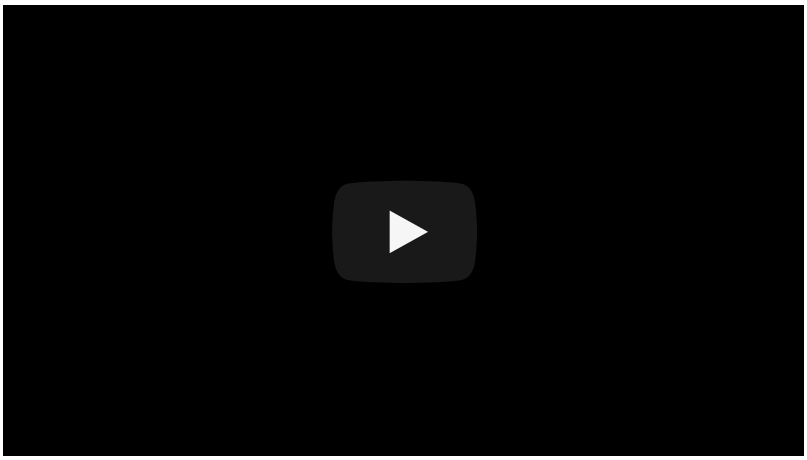
Companies used to stay within their industry, but they eventually turn their attention to tech firms and take advantage of innovations and tools that make handling financial transactions much easier, especially in managing large volumes of data.

Morgan Stanley, the Investment Banking Company, quoted on their website

> *Over the long term, tech advances such as artificial intelligence (AI) and blockchain will clearly play a role in the evolution of banking..... In order to remain competitive, banks will need to update technology on the back end in order to deliver a seamless experience on the front end since customers will have little tolerance for glitchy apps no matter how sleek*

> *the user interface.*

and to stress the “evolution of banking”, investment banks and financial institutions will need very talented researchers and engineers in the fields of computer science and data science with the support of the finest of technologies. They will not only collect, clean and organize data but will also create applications based on mathematical models, maintain servers that process collected data, and tweak those models continuously.



Financial Modeling using Python

Objectives:

- Students will be able to explain how Python is used for financial modeling.
- Students will be able to describe the process of using Python for Excel integration.

Python

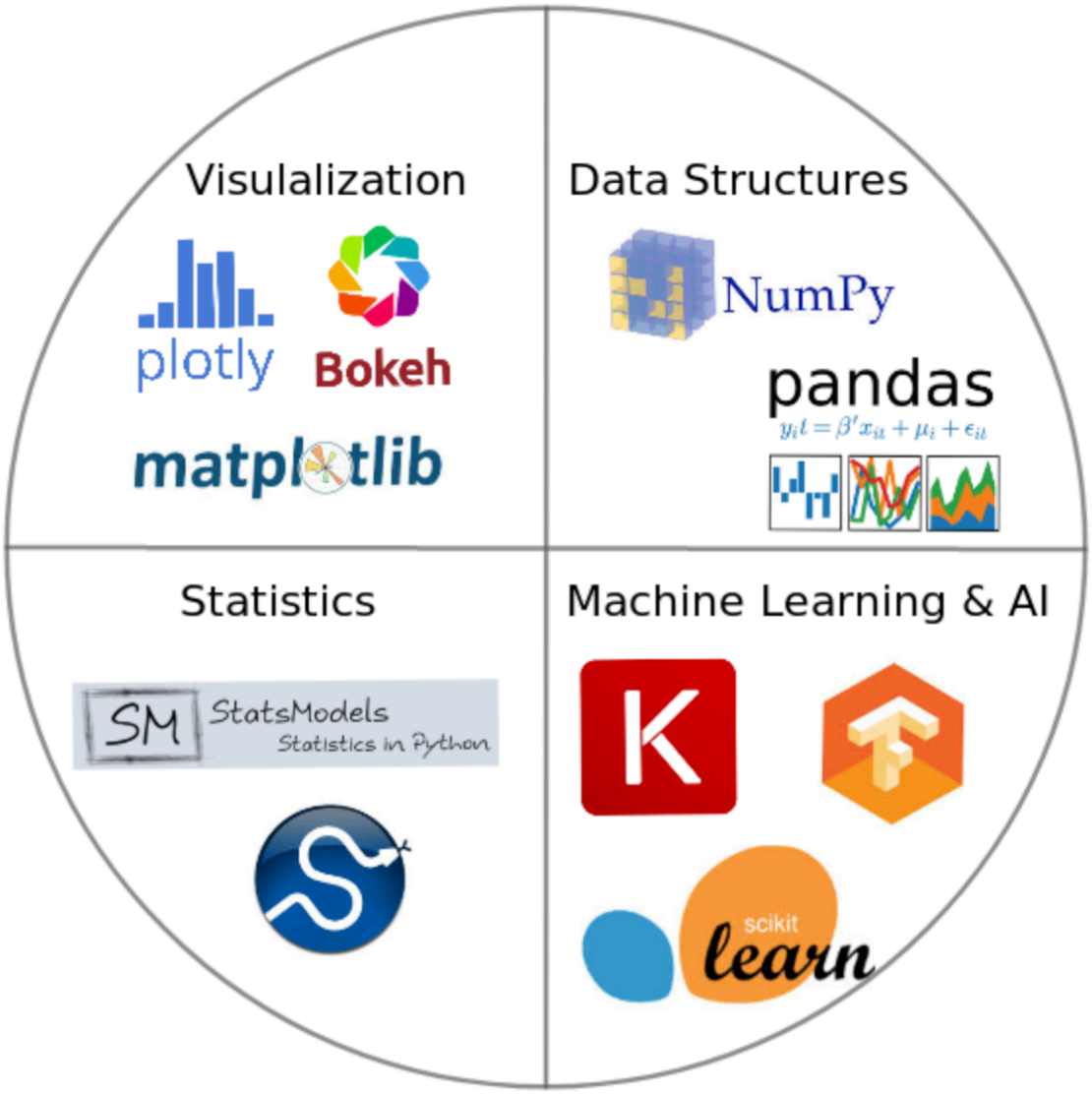
According to Investopedia,

> *Fintech, a portmanteau of 'financial technology,' is used describe new tech that seeks to improve and automate the delivery and use of financial services. At its core, fintech is utilized to help companies, business owners and consumers better manage their financial operations, processes and lives by utilizing specialized software and algorithms that are used on computers and, increasingly, smartphones.*

From the data published in "The Pulse of Fintech in 2018" by KPMG

Python as an ecosystem

Easier implementation coupled with vast ocean of libraries for mathematical computations, data collection, machine learning, visualization of data and even development of applications based on those models have transformed Python into a complete ecosystem for data science projects.



Python's ecosystem for data science

What is Financial Modeling in Python?

> *Financial Modeling in Python refers to the method that is used to build a financial model using high-level python programming language that has a rich collection of built-in data types. This language can be used for modification and analysis of excel spreadsheets as well as automation of certain tasks that exhibit repetition. Given that financial models use spreadsheets extensively, Python has become one of the most popular programming languages in the field of finance.*

Extending Python

There are certain limitations in Python that can be overcome with the extension modules using C. These extension modules can be used to add new built-in object types to Python and can call functions from the C library. A certain set of functions, macros, and variables available in Python API to support such extensions. The header '*Python.h*' is included in a C source file for Python API.

Python Excel Integration

Some of the Python Excel integration tools that can be used to supercharge the existing excel functionality are as follows:

- **xlwings:** This package can be used to move the backend processing from VBA to Python. After that, the users can continue using Excel seamlessly while using each control button to call Python scripts.
- **Jupyter Notebook:** It allows users to leverage Python for creating interactive, shareable, and web-based documents that can contain visualizations, code, and text.
- **Pandas Library:** It can be used to quickly load data from excel spreadsheets into SQL database or pandas DataFrames. In either case, data can be analyzed and explored swiftly.

Python Data Model

Objects are the underlying essence of a Python data model. All the data in a Python program is either represented by objects straightaway or by the relationship between objects. An object can be recognized by its identity, type, and value.

1. **Identity:** It refers to the address of an object in the memory, and it never changes once created.
2. **Type:** It defines the operations that an object supports along with the possible value for that object type.

3. **Value:** The value of an object may change. The ones that change are known as mutable, while the unchangeable ones are known as immutable.

Misconceptions about Python

- It is a pure scripting language as it uses simple syntax and cross-platform support.
- It doesn't have a compiler like other languages.
- It lacks scalability, and as such, it can't support any significantly large user base.
- It is perceived to be very slow.
- It doesn't support concurrency.

Significance of Financial Modeling in Python

Python has grown to become one of the most popular programming languages used for financial modeling. Companies nowadays seek innovative tools for handling large volumes of financial data in a much easier way and Python fits into that criteria perfectly.

Practical Business Python

Taking care of business, one python script at a time

Financial Modeling Techniques

Objectives:

- Students will be able to explain the four financial modeling techniques discussed in this lesson.
- Students will be able to go in detail regarding assumptions in financial modeling.

Techniques

A financial model represents the financial performance of a company for both the past and future. Models being very cohesive it's also advisable to build a financial model in excel. Knowledge of Excel, knowledge of accounting and knowledge of financial modeling techniques, corporate finance, understanding the company's operations are some of the financial modeling skill sets required in an individual in order to build a model.

Financial Modeling Techniques

Mr. Raj, a research analyst prepared a financial model on company ABC and unfortunately got sick and went on leave. During his absence, the market moved exactly opposite to his expectations and the financial model of company ABC required the changes as per the current situation. Due to Mr. Raj Absence, his assistant Mr. Saurabh is asked to incorporate the necessary changes in the financial model of company ABC.Mr. Raj knew how to prepare a financial model but he lacked knowledge of important financial modeling techniques.

Mr. Saurabh opens the model and gets confused looking at the model as he is not able to find out which one is the right cell in which changes need to be incorporated. In some cells, due to interlinkages, there is no value that can

be seen.

What do you think why did Mr. Saurabh faced a lot of problems with the financial model. Do you think a model which another person is unable to understand is a good model?? According to me the answer to this question is No.

A good financial model should always be:-

- Realistically based on reasonable and defensible assumptions and projections
- Flexible and adaptable to dynamic working schedules (or modules)
- Easy to follow, should not intimidate the reader

Wondering how can a model have these features. So let's learn some important financial modeling techniques and make a model flexible and easy to understand.

Financial Modeling Techniques are as follows:

1. Financial modeling techniques – Historical data

Your assumption for the future years is based on your historical. So it is very important to gather the right data from the right source. While gathering data remember one thing you are an analyst, not an auditor. So if the annual reports published by the company do not tally don't panic and sit to tally them.

2. Financial modeling techniques – Assumption

Financial models need to have clear and well-defined assumptions which are Referred to as 'drivers' or 'inputs' these are based on a thorough understanding of the business

Assumptions should reflect business realities and expectations

In order to come up with an assumption analyzing the historical plays a vital role. To analyze the historicals one should do a ratio analysis of the company financials and come up with answers to a certain question like

- Whether a certain ratio has declined or is growing
- What are the reasons behind this declining or growing percentage
- How would it affect future

The other criteria which one should consider while making an assumption are

- No bias should get into the assumptions on the business
- Clearly, understanding the expected changes in future performance
- Understand Management expectations
- Check out what other analysts think about the company

3. Financial modeling techniques – Color coding /Linkages

Formatting is very important in anything you prepare. In financial modeling, color coding is one of the formattings which one needs to take care of.

Let's consider an example and try to understand why color coding is so important.

You have prepared a financial model but the color of all the numbers are the same and you are on leave. There is some very important news that has been published which would change the assumptions that you had made for that particular company and your colleague wants to come up with the target price. In order to come up with a target price, your colleague has to change certain things in the model. Since it has the same color throughout your colleague is finding very difficult to find the right cell in which changes need to do.

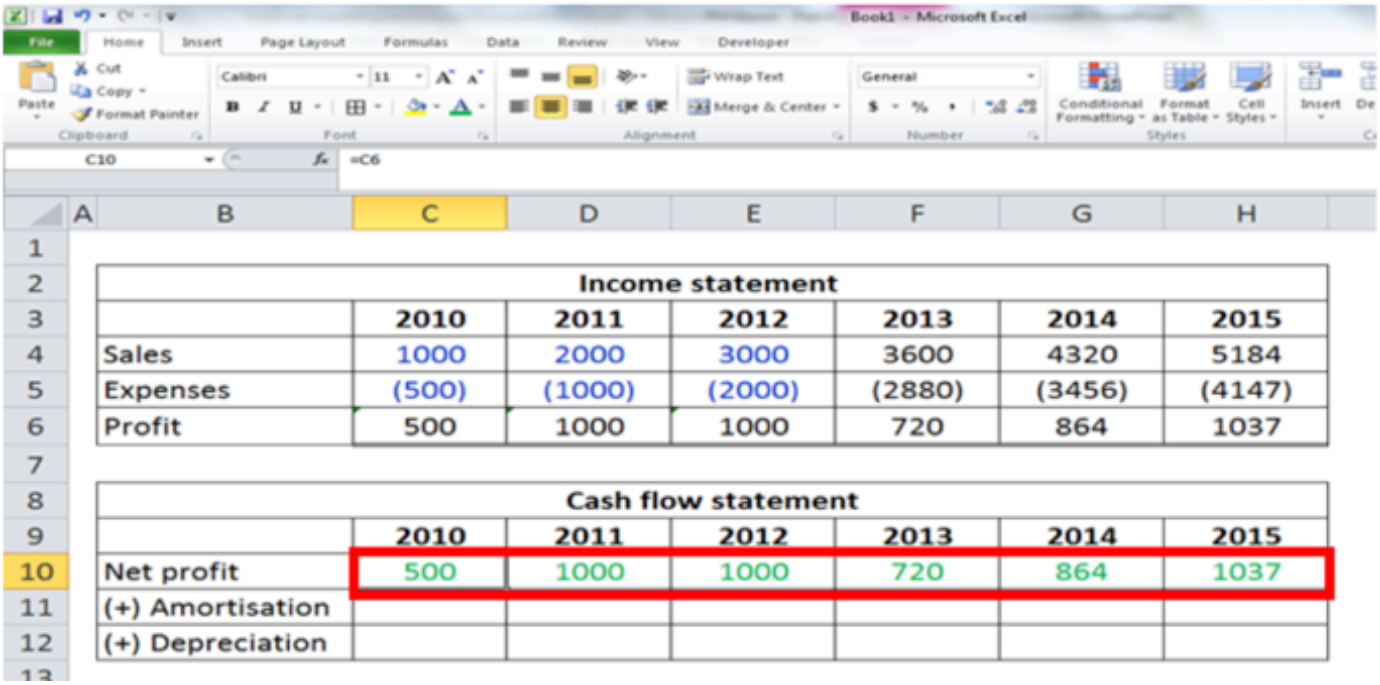
What can be done to overcome this situation?

A right color coding would solve this problem. So there should always be different color coding for Historical inputs, formulas, and linkages. This would help your colleague to understand the financial model and make the necessary changes in the right cell.

We have used certain color coding

Historical inputs in Blue

Formulas in Black Linkages in green



	A	B	C	D	E	F	G	H
1								
2		Income statement						
3			2010	2011	2012	2013	2014	2015
4		Sales	1000	2000	3000	3600	4320	5184
5		Expenses	(500)	(1000)	(2000)	(2880)	(3456)	(4147)
6		Profit	500	1000	1000	720	864	1037
7								
8		Cash flow statement						
9			2010	2011	2012	2013	2014	2015
10		Net profit	500	1000	1000	720	864	1037
11		(+) Amortisation						
12		(+) Depreciation						

4. Financial Modeling Techniques – a Circular reference

A circular reference is a series of references where the last object references the first, resulting in a closed-loop.

Below is an example:

You need to calculate the net income from the income statement. While calculating the net income, interest income is one of the items that need to be calculated. You are calculating net income as a percentage on the ending cash and cash balances which get calculated in the cash flow statement.

Over here you are assuming that the entire cash balance we have deposited in a bank.

Income Sheet (Rs m)	Year 1
Net Sales	
(-)Direct Costs	
Gross Profit	
(-)Selling, General & Admin Costs	
EBITDA	
(-)Depreciation/Amortisation	
EBIT	
(-)Interest Expense	
(+)Interest Income	
Pretax Income	
(-)Income Taxes	
Net Income	

Here to calculate the right net income you need to calculate interest income. Interest income will not be calculated unless you prepare a cash flow statement. So, see what is required to prepare the cash flow statement.

Cash Flow Statement	Year 1
Operating Activities	
Net Income	
Depreciation/Amortization	
Change in Working Capital	
Cash Flow from Operating Activities	
Investment Activities	
Capital Expenditures	
Additions to Intangibles	
Cash Flow from Investing Activities	
Financing Activities	
Issuance/ (Repayment) of Long-Term Debt	
Issuance/ (Repurchase of) Equity	
Cash Flow from Financing Activities	
Net Change in Cash	
Beginning Cash Balance	
Ending Cash Balance	

You can see here we need net income to calculate the ending cash balance which will be used in calculating interest income.

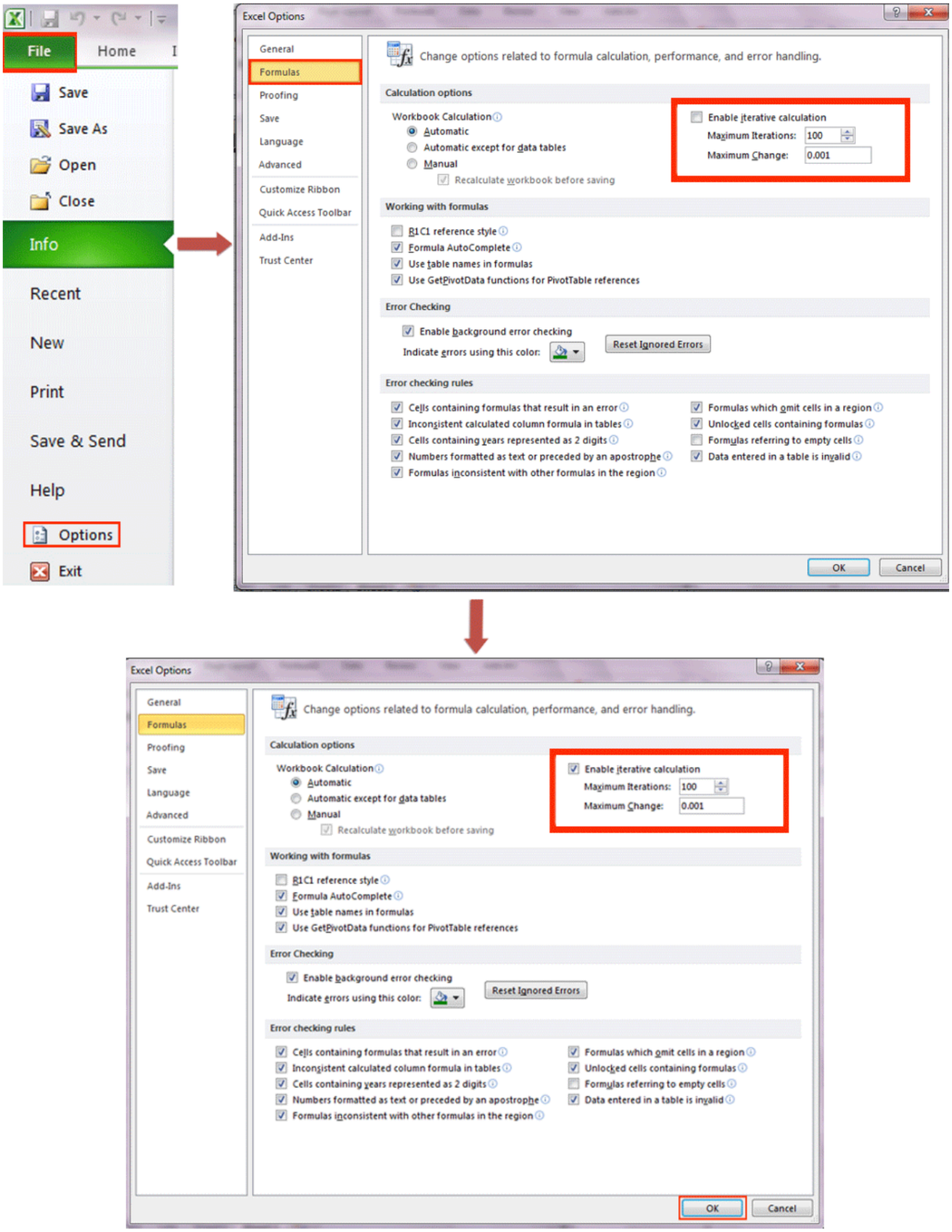
Cash balances	Year 1
Average Cash Balance	
Interest Rate	
Interest Income	

To do all this:

First, you will calculate net income by considering interest income to be 0. This net income will get linked to the cash flow statement through which you will be able to find the ending cash balance. Then this ending cash balance will get linked to average cash balances which will help calculate the interest income. Later, you will link this interest income to the income statement and find out the right net income balance. So, you must be wondering whether the new net income figure will get reflected in the cash flow statement.. Yes, through circular reference, this entire process will be done automatically and accordingly the other figures in the income statement, balance sheet and cash flow statement would also be changed. But remember one thing excel cannot calculate automatically when the model contains a circular reference. You need to Turn ON “Iterations” in order to resolve the situation.

Go to

File >>>Options >>>> Formulas >>>>> Enable iterative calculation >>>> OK


File
Home
Save
Save As
Open
Close
Info
Recent
New
Print
Save & Send
Help
Options
Exit
Excel Options
General
Formulas
Proofing
Save
Language
Advanced
Customize Ribbon
Quick Access Toolbar
Add-Ins
Trust Center
Change options related to formula calculation, performance, and error handling.
Calculation options
Workbook Calculation
Automatic
Automatic except for data tables
Manual
Recalculate workbook before saving
Enable iterative calculation
Maximum Iterations: 100
Maximum Change: 0.001
Working with formulas
R1C1 reference style
Formula AutoComplete
Use table names in formulas
Use GetPivotData functions for PivotTable references
Error Checking
Enable background error checking
Indicate errors using this color:
Reset Ignored Errors
Error checking rules
Cells containing formulas that result in an error
Inconsistent calculated column formula in tables
Cells containing years represented as 2 digits
Numbers formatted as text or preceded by an apostrophe
Formulas inconsistent with other formulas in the region
Formulas which omit cells in a region
Unlocked cells containing formulas
Formulas referring to empty cells
Data entered in a table is invalid
OK
Cancel