# Material Summary: High-school Maths

## Math in Real Life

* 1. **Mathematics in Nature**
* Honeycomb cells
  + Bees produce wax by consuming some of the honey they've made
  + Wax production takes time and energy (honey)
  + The hexagonal cells leave no unused space, and consume the least amount of wax and energy
* Snowflakes
  + All snowflakes are unique but they are perfectly symmetrical
  + Each arm (unless damaged) is identical
  + This makes them strong enough to stay together
* Romanesco broccoli
  + Each little floret looks exactly like the whole plant
  + This is called a fractal
  + Seen from above, the florets form a spiral
* Fibonacci spiral
  + Fibonacci spirals everywhere
  + Flowers, pinecones
  + Animal shells
  + Hurricanes
  + Galaxies
  1. **Mathematics in Music**
* Sound is a combination of waves travelling through the air
  + Each sound wave has a frequency (pitch)
  + Every note is associated with a certain frequency, for example [A4](https://www.youtube.com/watch?v=SsGhuUm_3-8) produces 440 oscillations every second ()
* Some combinations of tones sound pleasant, others sound harsh
  + Our ears like simple frequency ratios, e.g. is better than
  + All "good sounding" combinations of tones have simple ratios
* Example: "[A major](https://www.youtube.com/watch?v=VSZJ9GK6sXM)" chord
  + A4: , C#5: , E5:

## Methods

* 1. **Divide and Conquer**
  + Useful for any kind of problem:
    - Especially in algorithms and debugging
    - When invading countries
  + Assumption: complicated things are a combination of many, very simple things
    - Algorithms: [Merge sort](https://en.wikipedia.org/wiki/Merge_sort), [Discrete Fourier transform](https://en.wikipedia.org/wiki/Discrete_Fourier_transform)
    - Software architecture:
      * "I want to build an ecommerce system"  
         I want shop owners to add new products  
         I want to store products in the DB …   
         def save\_product(name, price)
    - Debugging
      * The bug is somewhere in my code …  
         the bug is ">=" instead of ">" on line 45 in user.py
  1. **The Scientific Method Steps**
* Ask a question
* Do some research
* Form a hypothesis
* Test the hypothesis with an experiment
  + Experiment works Analyze the data
  + Experiment doesn't work Fix experiment
* Results align with hypothesis OK
* Results don't align with hypothesis new question, new hypothesis
* Communicate the results
  1. **Why use the Scientific Method Steps**
* Useful when we're exploring something new
  + A new algorithm
  + A new codebase we've just been hired to work on
* Based on common logic
* Experiments
* Example: performance testing
  + Research: My logs show that this Web page on my server takes too much time to load
  + Hypothesis: This piece of code is too slow. I need to improve it
  + Control: Measure the runtime (in seconds)
  + Experiment: Try to fix the problem and repeat the runtime test
    - Did the fix bring a considerable performance gain?
  + Communication: Show the results and implement the fix

## Set Up Our Environment

**3.1 Anaconda**

* You can install the Python interpreter   
  and all libraries manually
  + Hard, boring and repetitive work
  + Error-prone
* Easy solution: platforms like Anaconda
  + Everything you need to get started with Python for science:   
    Python interpreter, packages (720+), package manager, IDE
* Download from [the Anaconda website](https://www.anaconda.com/products/individual)
* Current version (March 2023): Anaconda 2022.10 (Py3.9)
  + Choose your platform (Windows, Linux, or MacOS)
  + Follow the installer

**3.2 Setting Up an IDE**

* You can use the built-in IDE called Spyder
  + You can even use Notepad if that's your thing
* If you want to use another IDE,  
  you need to configure it to work with Python
  + Syntax highlighting, autocomplete, etc.
* Visual Studio Code
  + My preferred editor / IDE
  + [Python in VSCode – tutorial](https://code.visualstudio.com/docs/languages/python)
  + [Python extension](https://marketplace.visualstudio.com/items?itemName=ms-python.python)
* Visual Studio
  + [Python Tools](https://visualstudio.microsoft.com/vs/features/python/)

**3.3 Python Online**

* There are places where you can execute your code online
  + If you don't have access to Anaconda
  + Or you want to test something very quickly
* <https://www.python.org/shell/>
  + Provides a Python shell (CLI)
* <https://www.pythonanywhere.com/try-ipython/>
  + Provides an implementation of IPython (Interactive Python)
    - REPL (Read – Execute – Print Loop)
  + No major difference to the Python shell
* To share your code, you can use
  + [http://ideone.com](http://ideone.com/)
  + <https://gist.github.com/>
  + <http://pastebin.com/>

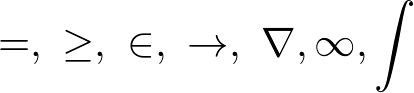
**3.4 Jupyter Notebook**

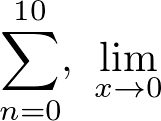
* A very nice and clean way to document your research
* Included in Anaconda
* Can create documents that contain live code, equations, visualizations and explanatory text
  + HTML / CSS / JavaScript
  + Markdown
  + Python
* Start:
  + use the Anaconda shortcut
  + type into the Command Prompt: **jupyter notebook**

**3.5 How to use Jupyter Notebook?**

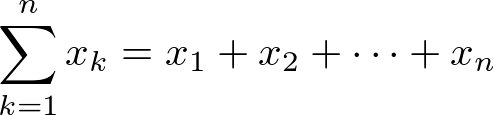
* Create a new notebook
  + New -> Python 3
* Every piece of text or code is in a cell
  + Text cells just contain text or Markdown
  + Code cells contain code (obviously)
  + Code can be executed
  + Jupyter "remembers" the code
* Execute cell: **Ctrl + Enter**
  + Or use the menus

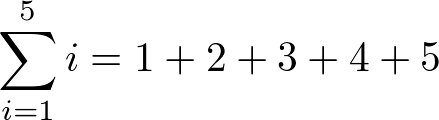
## Math Notation

* The basic symbols we use are numbers and letters
  + Usually English or Greek letters
* Special symbols:
* Indices:

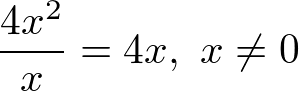


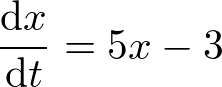
**4.1** **Other Useful Notations**

* [Scientific notation](https://en.wikipedia.org/wiki/Scientific_notation)
  + Used for very large or very small numbers
  + Numbers are expressed as decimals with **exactly one** digit   
    before the decimal point
  + All other digits are expressed as a power of 10
* Summation notation ("sigma" notation)
  + Used as a shorthand for writing long sums of numbers / symbols
    - Very similar to a for-loop
    - Greek capital "sigma" denotes the sum, the two numbers  
      below and above it denote the start and end points



**4.2 Equality Sign**

* Important as it has different meanings
  + Like programming: **"=", "==" and "==="**
* Identity
  + The two statements around "=" are always equal:
    - We can also use the "identity" symbol:
  + … for all "valid" symbols:
* Equation
  + The two statements are true only for specific values of the symbols



* Definition (we can also use or \documentclass{article}
  \usepackage{amsmath}
  \pagestyle{empty}
  \begin{document}

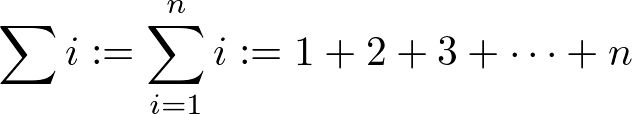
  $$ \overset{\text{def}}{=} $$

  \end{document} , or even ![\documentclass{article}
  \usepackage{amsmath}
  \pagestyle{empty}
  \begin{document}

  $$ \equiv $$

  \end{document}](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABwAAAASBAMAAAC6KaPXAAAAFVBMVEX///8AAAAAAAAAAAAAAAAAAAAAAAA9huxxAAAABnRSTlMAibtUq3aHAyuuAAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAGUlEQVQYGWMQS0MCSQwmLkjAlYF2YFDaCwA3LRq89EDaVgAAAABJRU5ErkJggg==))





## Linear Equations

* Equations of a variable
* is "on its own"
  + Not inside a function
  + No powers
* General form:
  + and : fixed numbers (parameters)
* Examples:
* Solutions of the parametric equation
  +  (every is a solution)
  +  (no solution)
  + (one solution, regardless of )
* Many simultaneous equations
  + To solve the system, we need to find values of the variable(s)  
    which satisfy **all equations** at once
  + Even if all individual equations have solutions, the system may  
    have no solution
* Solution:
  + Method 1: Solve one equation and substitute
  + Method 2: Use sum of equations
  + Later, we'll learn a faster way of solving these systems
* Example:

