High-School Maths

Establish a workflow, get to know our tools, review basic concepts

Yordan Darakchiev

Technical Trainer

iordan93@gmail.com

Table of Contents

- sli.do: #hs-maths
- Motivating examples
- Methods
  - Divide and conquer
  - Scientific method
- Setting up our environment
  - Python 3.6, Anaconda, Jupyter Notebook
- Math notation
  - Scientific notation
  - Summation
- Linear equations and systems of equations

Motivating Examples

Math in Real Life

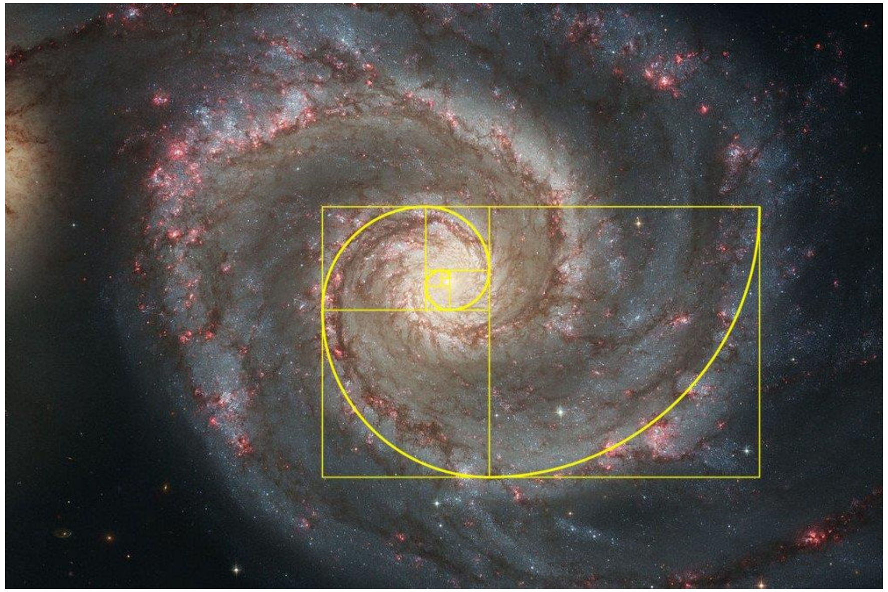
Mathematics in Nature

- 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

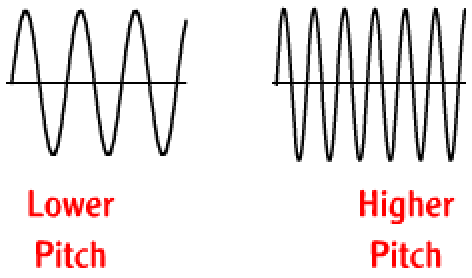
Mathematics in Nature (2)

- Romanesco broccoli
  - Each little floret looks exactly like the whole plant
    - This is called a fractal
  - Seen from above, the florets form a spiral
    - This is a Fibonacci spiral
- Fibonacci spirals everywhere
  - Flowers, pinecones
  - Animal shells
  - Hurricanes
  - Galaxies



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
    - E.g. A4 produces 440 oscillations every second ($440\ Hz$)
  - Some combinations of tones sound pleasant, others sound harsh
    - Our ears like simple frequency ratios, e.g. $2:3$ is better than $160:231$
    - All "good sounding" combinations of tones have simple ratios
  - Example: "A major" chord
    - A4: $440\ Hz$, C#5: $554{,}37\ Hz$, E5: $659{,}25\ Hz$
    - $A4: C\#5: E5 \approx 4:5:6$
    - $E5: A4 \approx 3:2$



Methods

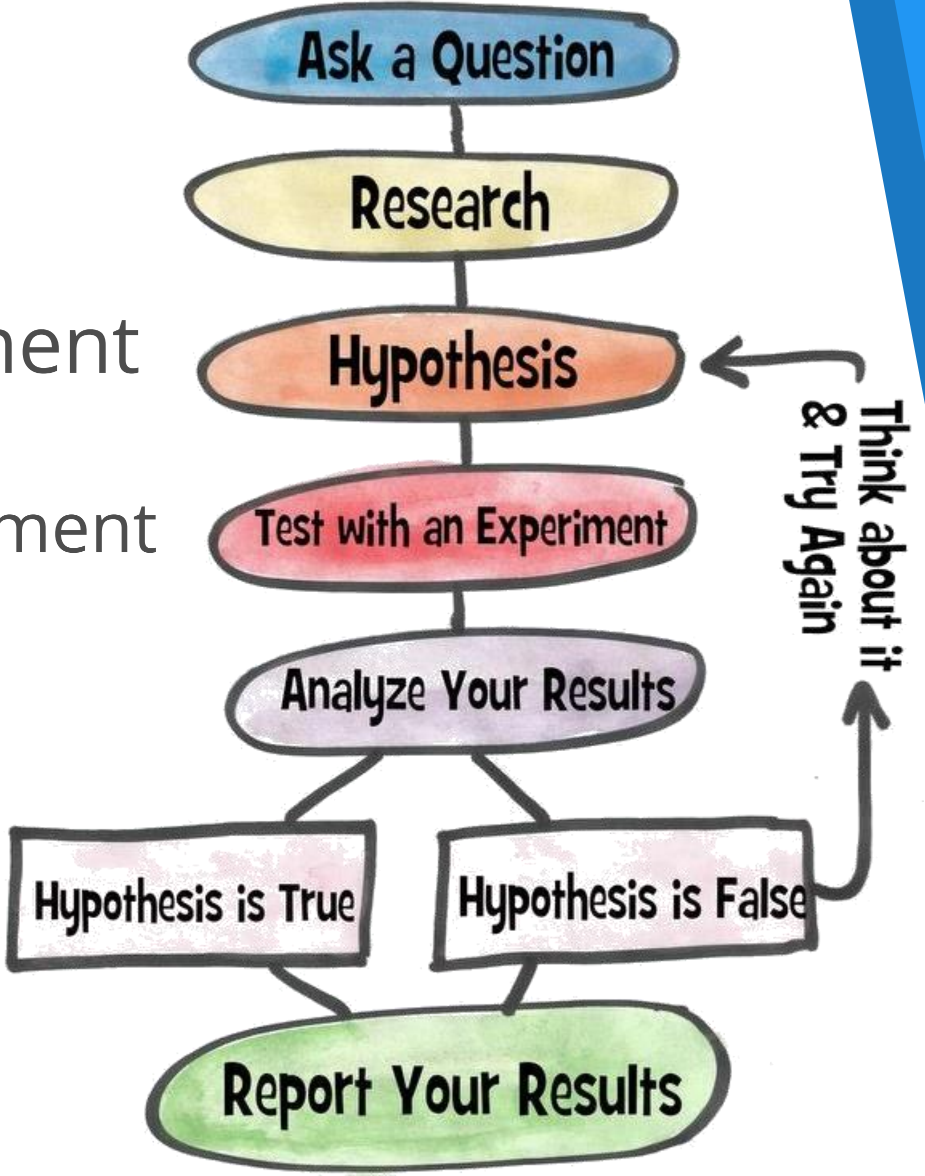
How not to get lost

Divide and conquer

- Useful for any kind of problem
  - Especially in algorithms and debugging
  - … also when invading countries
- Assumption: Complicated things are a combination of many, very simple things
  - Algorithms: Merge sort, 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`

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



Why use the Scientific Method?

- 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

Setting Up Our Environment

Getting ready to conquer math, science and programming

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 https://www.anaconda.com/download/
- Current version (April 2018): Anaconda 5.1.0
  - Choose your platform (Windows, Linux, or MacOS)
  - Download the **Python 3.6** version
  - Follow the installer

Setting Up an IDE (Optional)

- 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 have to configure it to work with Python
  - Syntax highlighting, autocomplete, etc.
- If you're using Visual Studio
  - Python Tools
  - https://www.visualstudio.com/vs/python/
- Visual Studio Code
  - If you prefer something lightweight, Visual Studio Code is a good alternative
  - https://code.visualstudio.com/docs/languages/python

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
- 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://pythonfiddle.com/ or http://pastebin.com/

Jupyter Notebook

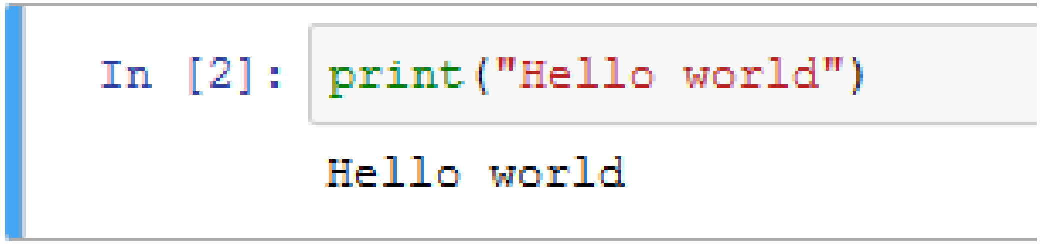
- A very nice and clean way to document your research
- Included in Anaconda
- Can create documents that contain live code, equations, visualizations and explanatory text
  - HTML / CSS / JavaScript
  - Markdown
  - $\LaTeX$
  - Python
- Start – use the Anaconda shortcut
  - …or type into the Command Prompt

```
jupyter notebook
```

How to Use Jupyter

- 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



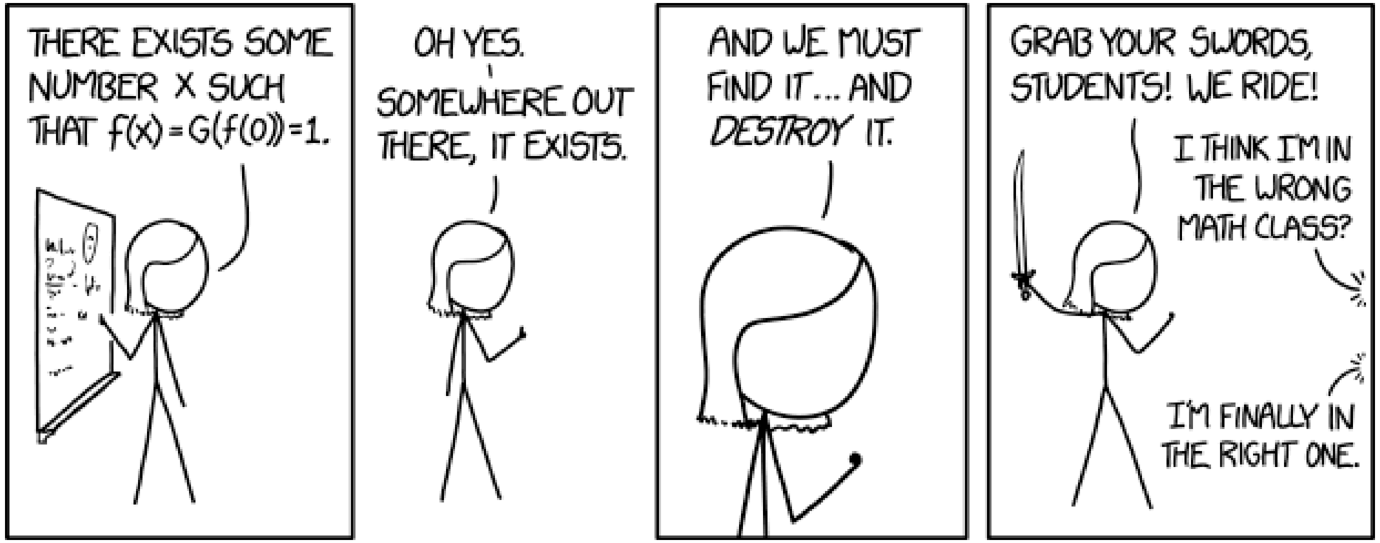
- Execute cell: **`Ctrl + Enter`**
  - Or use the menus

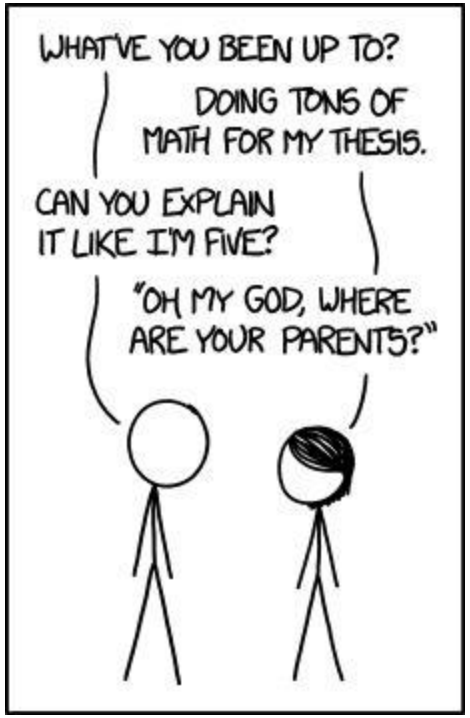
Math Notation

How to write more quickly and concisely

Math Notation

- The basic symbols we use are numbers and letters
  - Usually English or Greek letters
- Special symbols: $=,\ \geq,\ \in,\ \rightarrow,\ \nabla, \infty, \int$
- Indices: $\sum_{n=0}^{10},\ \lim_{x \to 0}$





Other Useful Notations

- 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
  - $15\ 000 = 1{,}5.10^4$
  - $0{,}000015 = 1{,}5.10^{-5}$
- 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

$$\sum_{i=1}^{5} i = 1 + 2 + 3 + 4 + 5 \qquad \sum_{k=1}^{n} x_k = x_1 + x_2 + \cdots + x_n$$

Equality Sign

- Important as it has different meanings
  - Similar to programming: **"="**, **"=="** and **"==="**
- **Identity**
  - The two statements around "=" are always equal: $x(x+3) = x^2 + 3x$
    - We can also use the "identity" symbol: $(a+b)^2 \equiv a^2 + 2ab + b^2$
  - ... for all "valid" symbols: $\frac{4x^2}{x} = 4x, \ x \neq 0$
- **Equation**
  - The two statements are true only for specific values of the symbols

$$2x + 5 = 4, \ x = -0.5 \qquad x^2 - 1 = 0, \ x = \pm 1 \qquad \frac{\mathrm{d}x}{\mathrm{d}t} = 5x - 3$$

- **Definition** (we can also use $:=$ or $\stackrel{\text{def}}{=}$, or even $\equiv$)

$$\sum i := \sum_{i=1}^{n} i := 1 + 2 + 3 + \cdots + n$$

Linear Equations

Simple, yet very useful

Linear Equations – Review

- Equations of a **variable** **x**
- **x** is "on its own"
  - Not inside a function (e.g. $\sin(x)$, $\frac{1}{x}$, e^x)
  - No power (e.g. x^3)
- General form: $ax + b = 0$
  - **a** and **b**: fixed numbers (**parameters**)
- Examples
  - $2x + 3 = 0$
  - $2(2x + 3) - 3x - 3(-4 + 3x) = 12$
- Solutions of the parametric equation
  - $a = 0, b = 0 \Rightarrow 0.x = 0, \ \forall x$ (every **x** is a solution)
  - $a = 0, b \neq 0 \Rightarrow 0.x = -b$ (no solution)
  - $a \neq 0, \Rightarrow x = -b/a$ (one solution, regardless of **b**)

Exercise: Linear Equations

- Write a Python function which solves a linear equation given the definition from the previous slide
  - The function should accept the **a** and **b** as arguments
  - The function should return
    - The solution, if there is only one
    - **nan** if there is no solution
    - Empty list **[]** if all x satisfy the equation

```python
import math
def solve_linear_equation(a, b):
  if a == 0:
    if b == 0:
      return []
    else:
      return math.nan
  else:
    return -b / a
```

```python
solve_linear_equation(0, 0) # []
solve_linear_equation(0, 5) # nan
solve_linear_equation(5, 0) # 0.0
solve_linear_equation(5, 5) # -1.0
solve_linear_equation(2.5, -5.3) # 2.12
```

Linear Systems of Equations – Review

- 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

$$\left|\begin{array}{rcl} 4x + 3y & = & 7 \\ 3x + 5y & = & 8 \\ x - 2y & = & -1 \end{array}\right.$$

Solving a Linear System

$$\left|\begin{array}{rcl} 4x+3y & = & 7 \\ 3x+5y & = & 8 \\ x-2y & = & -1 \end{array}\right.$$

$$(\mathbf{3}) : x = -1 + 2y$$

$$\begin{aligned}(\mathbf{3}) \to (\mathbf{2}) : & 3(-1+2y) + 5y = 8 \\ & -3 + 6y + 5y = 8 \\ & 11y = 11 \\ & \boxed{y = 1}\end{aligned}$$

$$\begin{aligned}(\mathbf{2}) \to (\mathbf{3}) : & x = -1 + 2.1 \\ & \boxed{x = 1}\end{aligned}$$

$$(\mathbf{1}) : 4.1 + 3.1 = 7$$

$\Rightarrow (x, y) = (1, 1)$ is the only solution of the system

- Note: The numbers of equations and variables matter
  - E.g. this system is "overdetermined"
  - We'll learn more about this later

Summary

- Motivating examples
- Methods
  - Divide and conquer
  - Scientific method
- Setting up our environment
  - Python 3.6, Anaconda, Jupyter notebook
- Math notation
  - Scientific notation
  - Summation
- Linear equations and systems of equations

Questions?