

Data, Math and Methods

Week 2, Gentle Math Recap



Intro

- Last class you played chess. You tried thinking about strategies, then like a computer to win a game, predict the opponent. Abstract, logical, mathematical, systematic thinking.
- Listen: [youtube.com/watch?v=cDIRT_NEMxo](https://www.youtube.com/watch?v=cDIRT_NEMxo)

AI playing games

- Deep Blue (1996)



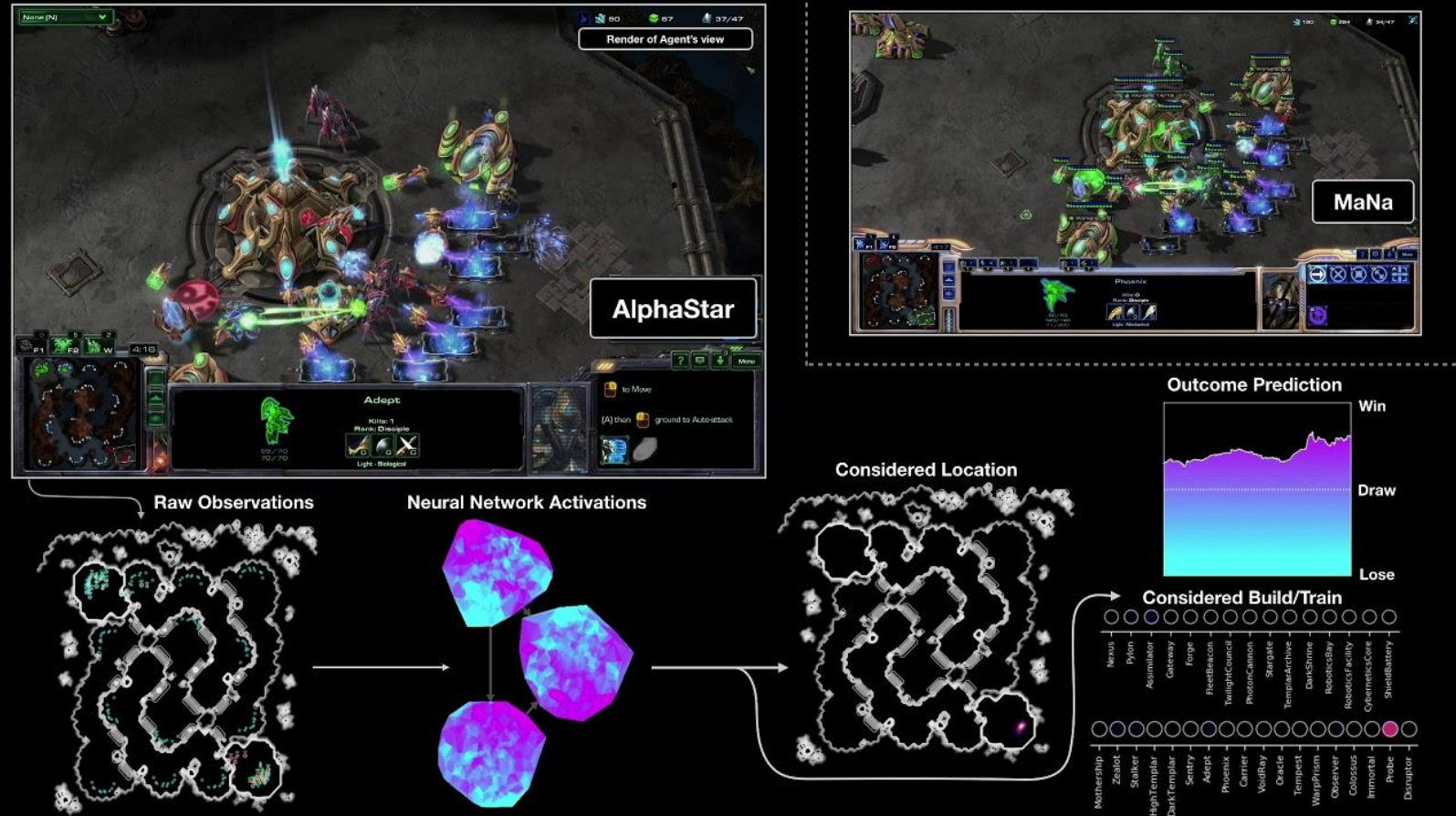
AI playing games

- Deep Blue (1996)
- Alpha GO (2015)
 - Self play



AI playing games

- Deep Blue (1996)
- Alpha GO (2015)
 - Self play
- Alpha Star (2019)
 - Neural Networks with Reinforcement Learning



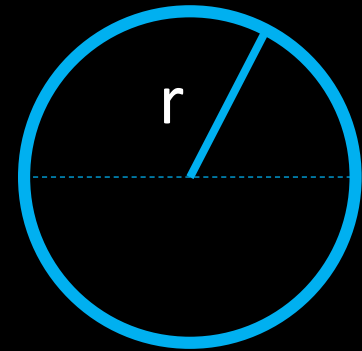
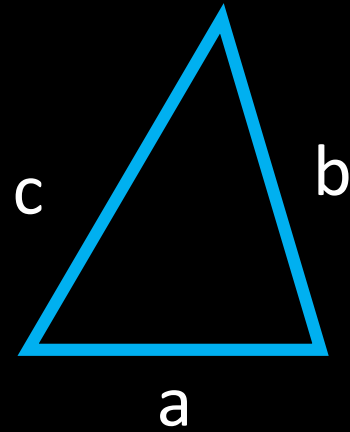
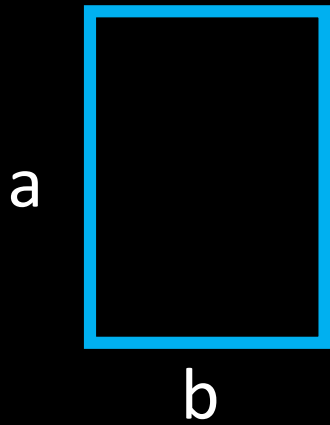
Motivation

- **Exciting times!**
- Examples where converting something into an abstract mathematical notation makes sense and helps the field.
- Real world problems are harder! Not possible to simulate reality (really exactly). We have only models which are being checked with real world experiments, compared.
 - Model of atmosphere => research of countering climate change effects. Then compare with real world experiments on smaller scale.

Algebra

- ... algebra is the study of mathematical symbols and the rules for manipulating these symbols ...
- (abstractions and rephrasing them into better solvable forms)

Volume of geometric shapes



- Area

$$a * b$$

$$p * (p - a) * (p - b) * (p - c)$$
$$p = (a + b + c) / 2$$

$$\pi r^2$$

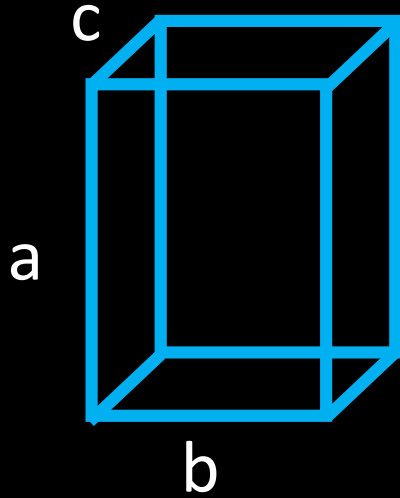
- Perimeter

$$2a + 2b$$

$$a + b + c$$

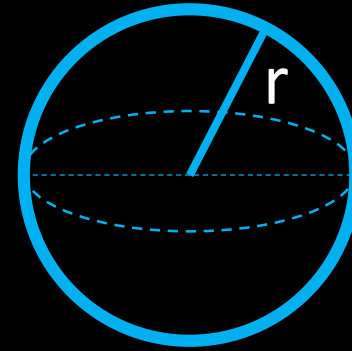
$$2\pi r \text{ (circumference)}$$

Volume of geometric shapes



- Volume

$$a*b*c$$



$$\frac{4}{3} \pi r^3$$

- Surface

$$2ab+2bc+2ac$$

$$4 \pi r^2$$

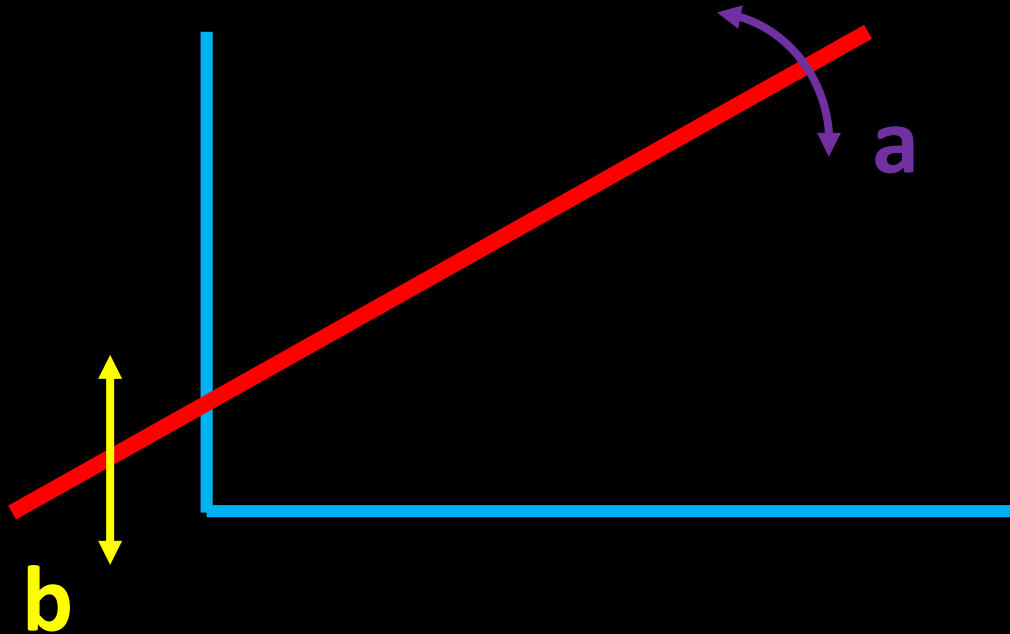
Equations

- Linear
- Quadratic
- Polynomial (wolfram)
- Multiple variables (wolfram)

Linear Functions

$$f(x): ax + b$$

$$y = ax + b$$



- Linear function is a line, parameters **a**, **b** influence it's location

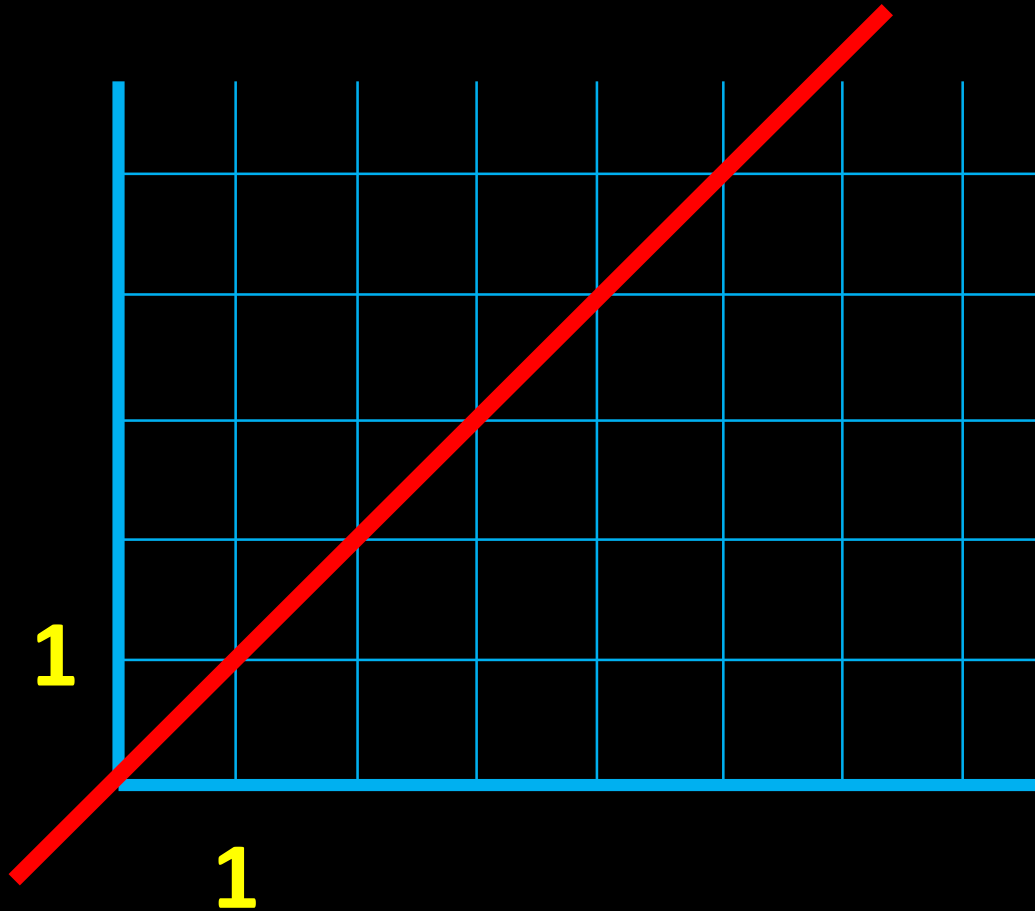
Linear Functions - example

$$f(x): ax + b$$

$$a = 1$$

$$b = 0$$

$$f(x): x$$



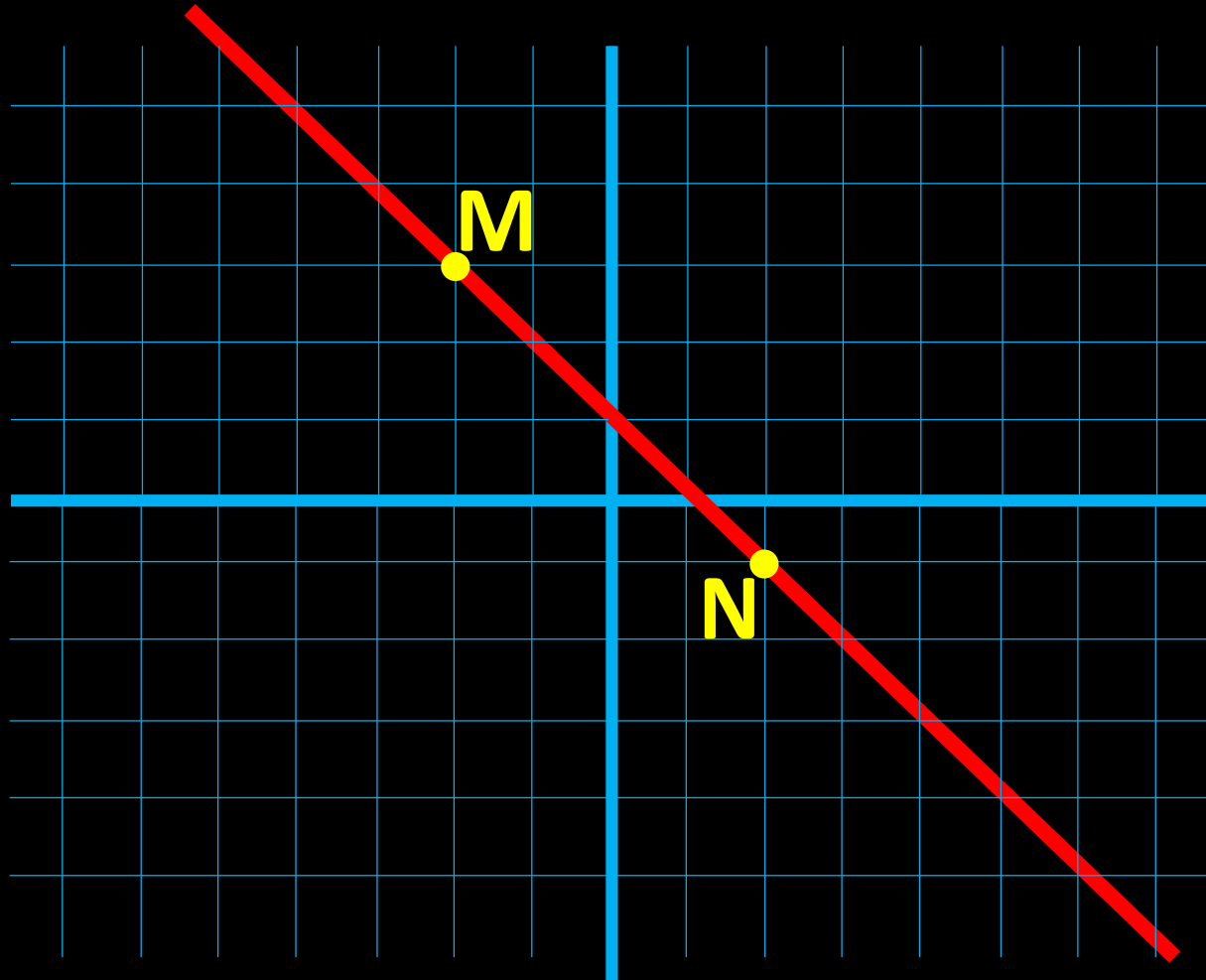
Linear Functions – given by points

$$f(x): ax + b$$

Find values for a, b so that the points M and N are on this line:

$$M = [-2; 3]$$

$$N = [2; -1]$$



Linear Functions – given by points

$$f(x): ax + b$$

Find values for **a**, **b** so
that the points **M** and **N**
are on this line:

$$M = [-2; 3]$$

$$N = [2; -1]$$

Points are on the line, which means:

$$M = [m_x, m_y] = [-2, 3]$$

$$N = [n_x, n_y] = [2, -1]$$

Are on:

$$ax + b = y$$

$$a*m_x + b = m_y$$

$$a*n_x + b = n_y$$

Linear Functions – given by points

$$f(x): \mathbf{ax + b}$$

Find values for **a,b** so
that the points **M** and **N**
are on this line:

$$M = [-2; 3]$$

$$N = [2; -1]$$

Points are on the line, which means:

$$M = [m_x, m_y] = [-2, 3]$$

$$N = [n_x, n_y] = [2, -1]$$

Are on:

$$\mathbf{ax + b = y}$$

$$a*m_x + b = m_y$$

$$a*-2 + b = 3$$

$$a*n_x + b = n_y$$

$$a*2 + b = -1$$

... (solve from these two equations) ...

Linear Functions – given by points

$$f(x): ax + b$$

Find values for **a**, **b** so
that the points **M** and **N**
are on this line:

$$M = [-2; 3]$$

$$N = [2; -1]$$

Points are on the line, which means:

$$M = [m_x, m_y] = [-2, 3]$$

$$N = [n_x, n_y] = [2, -1]$$

Are on:

$$ax + b = y$$

$$a \cdot m_x + b = m_y$$

$$a \cdot -2 + b = 3$$

$$a \cdot n_x + b = n_y$$

$$a \cdot 2 + b = -1$$

... (solve from these two equations) ...

$$a = -1$$

$$b = 1$$

$$\text{Line: } -x + 1 = y$$

Linear Functions

Super simple example, right?

$$f(x): ax + b$$

Wolfram Alpha command:

line points (-2,3) (2,-1)

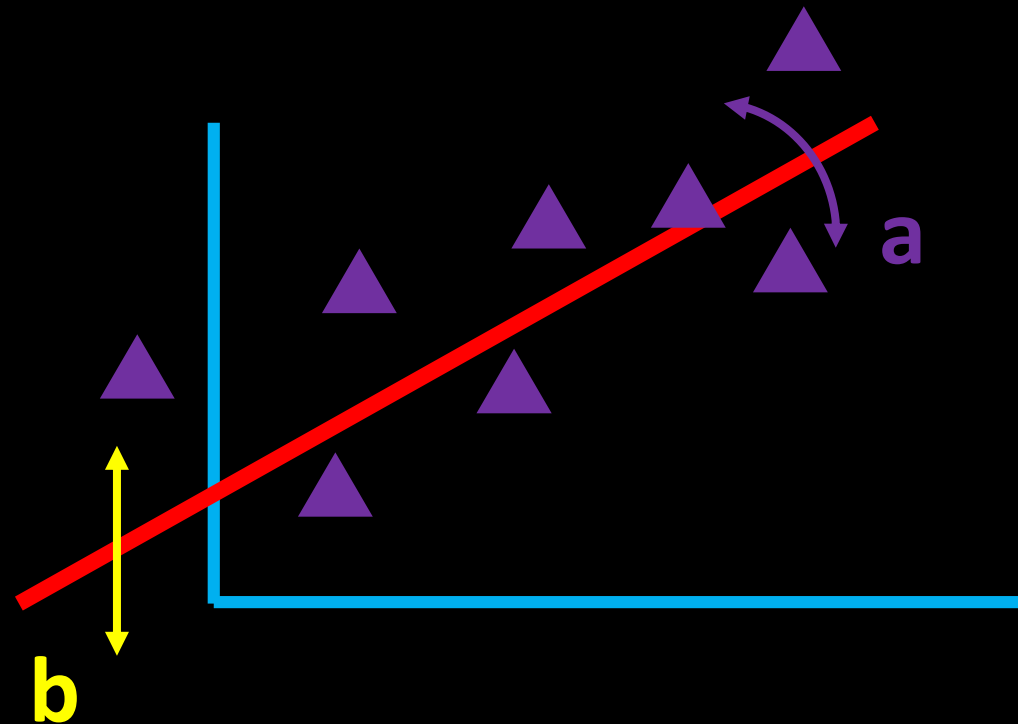
[wolframalpha.com/input/?i=line points \(-2,3\) \(2,-1\)](https://www.wolframalpha.com/input/?i=line+points+(-2,3)+(2,-1))

Simplest Machine Learning algo.:

Linear regression:

$$f(x): ax + b$$

Find a , b , so that
the line follows
the data

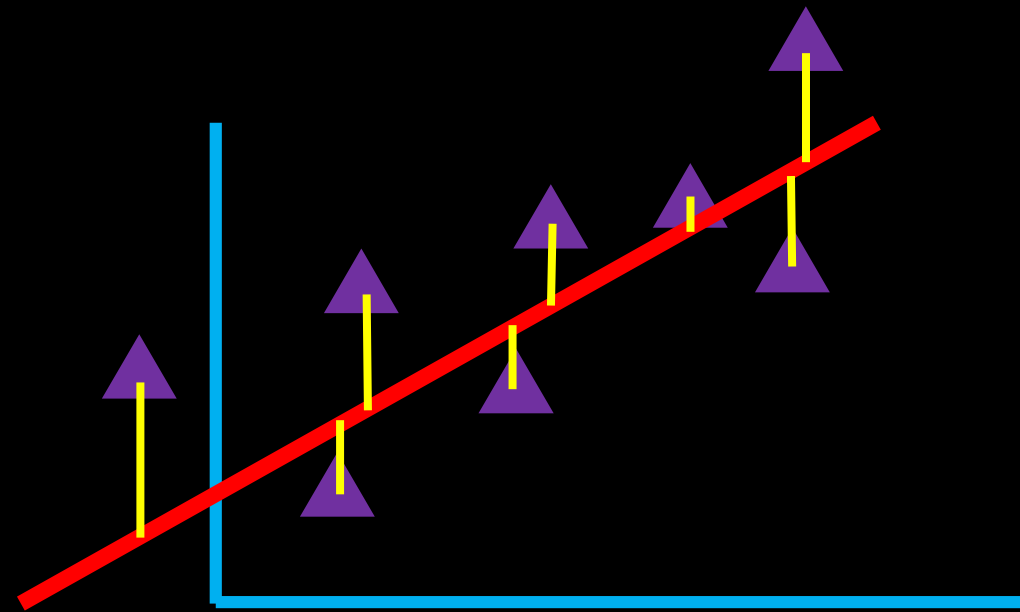


Simplest Machine Learning algo.:

Linear regression:

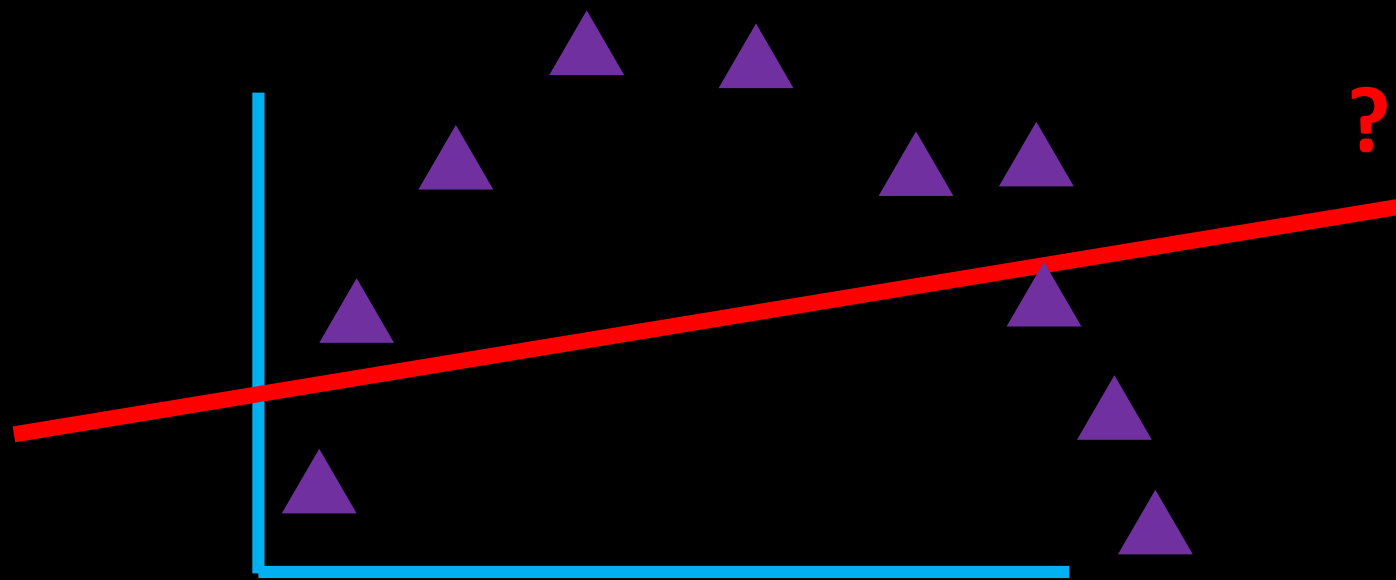
$$f(x): ax + b$$

Find a , b , so that
the line follows
the data



minimize the sum of the
squared distances from the line

Simplest Machine Learning algo.:



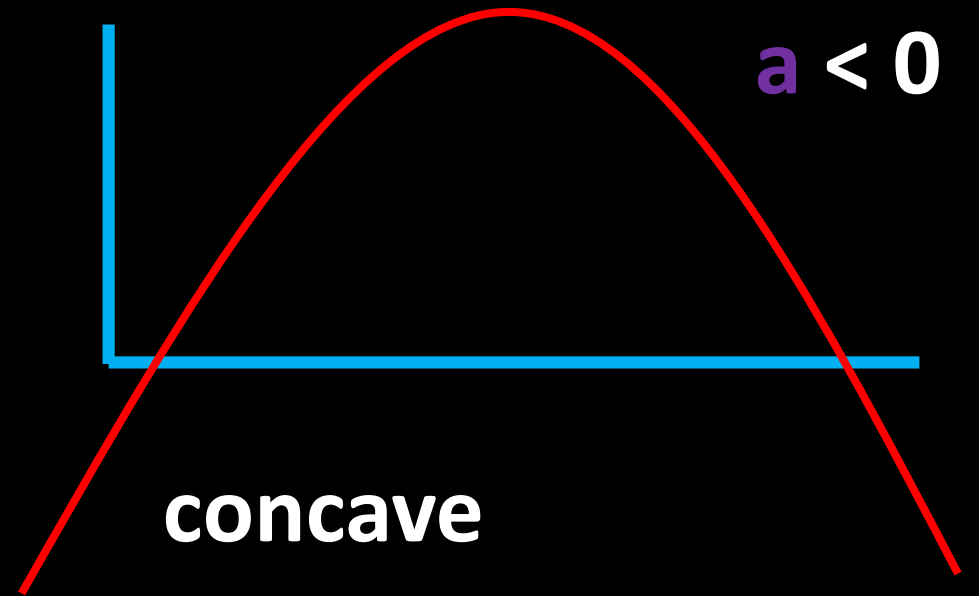
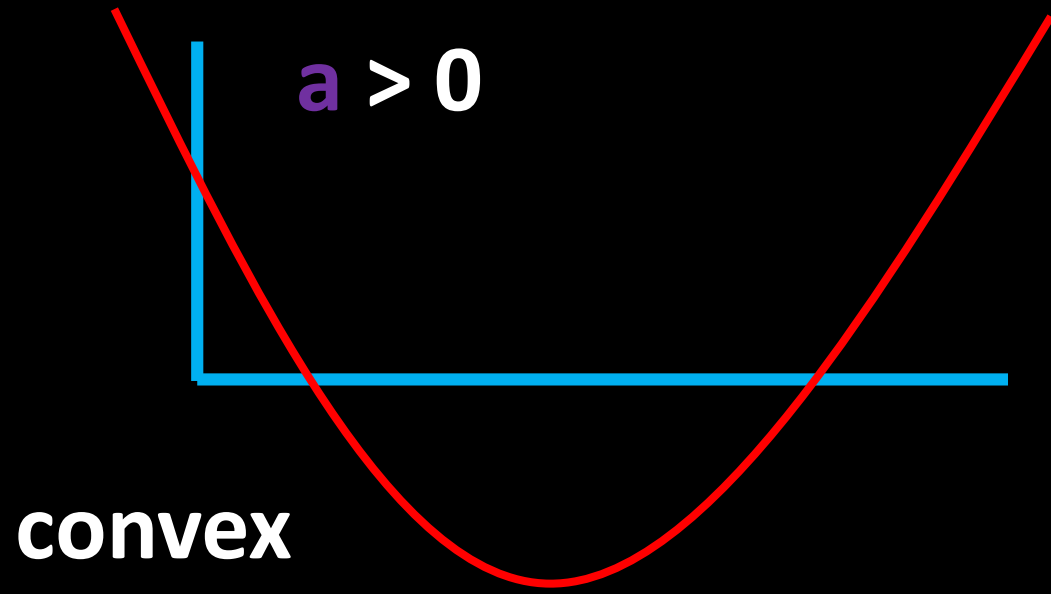
Depending on the points, might not be the most suitable ...

Simplest Machine Learning algo.:

See [demo](#) for
Least-Squares Regression

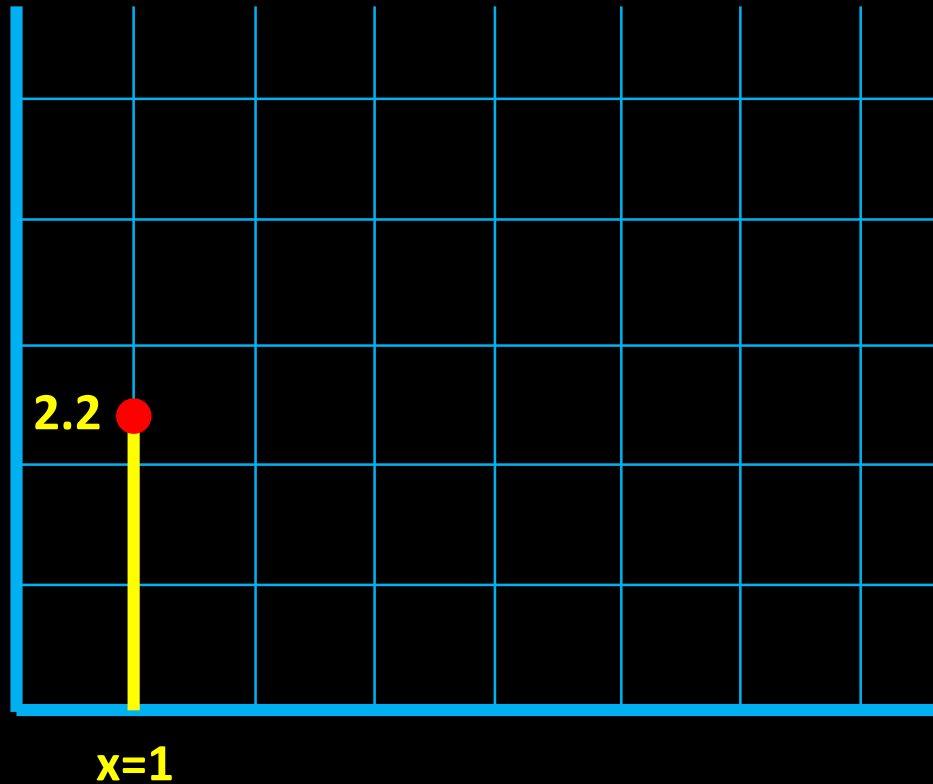
Quadratic Functions

$$f(x): ax^2 + bx + c$$



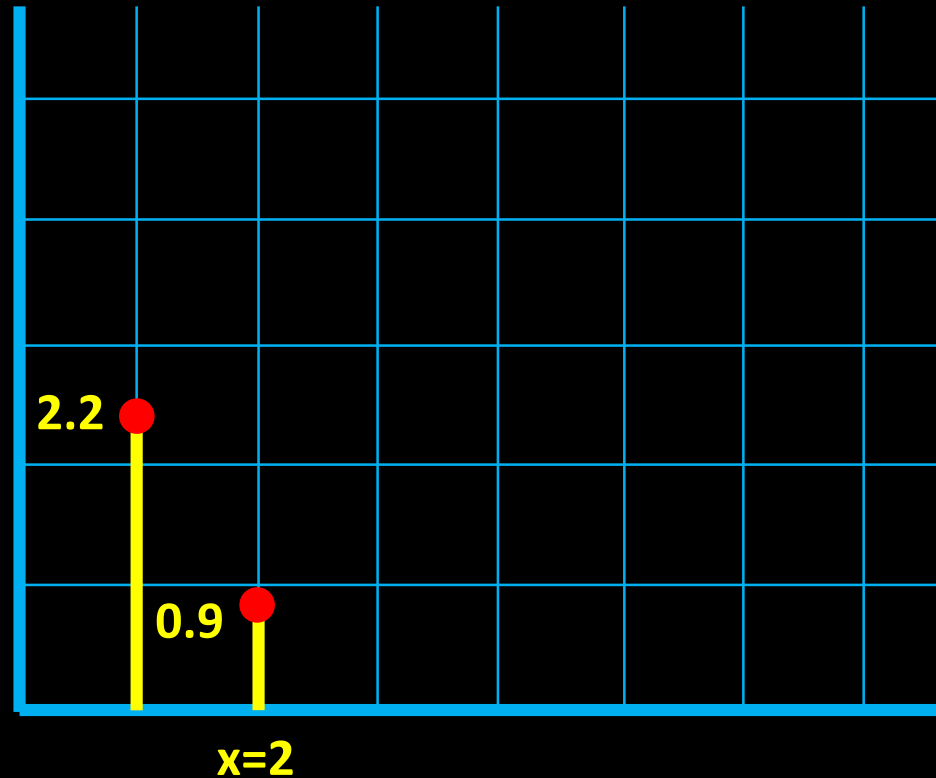
Quadratic Functions – finding minimum

$$f(x): ax^2 + bx + c$$



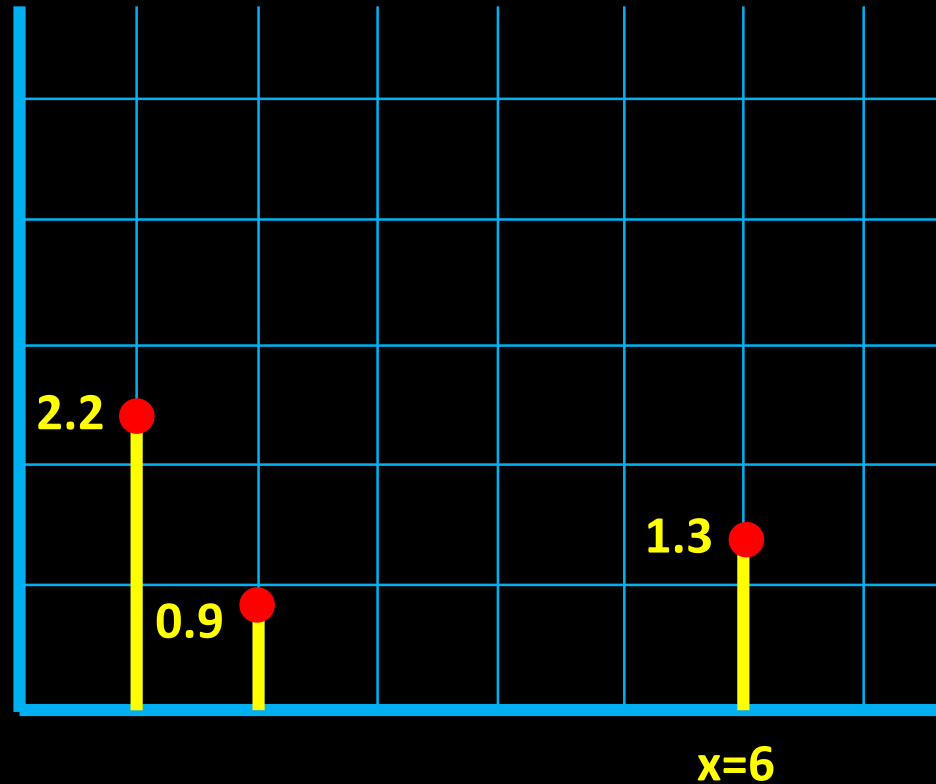
Quadratic Functions – finding minimum

$$f(x): ax^2 + bx + c$$



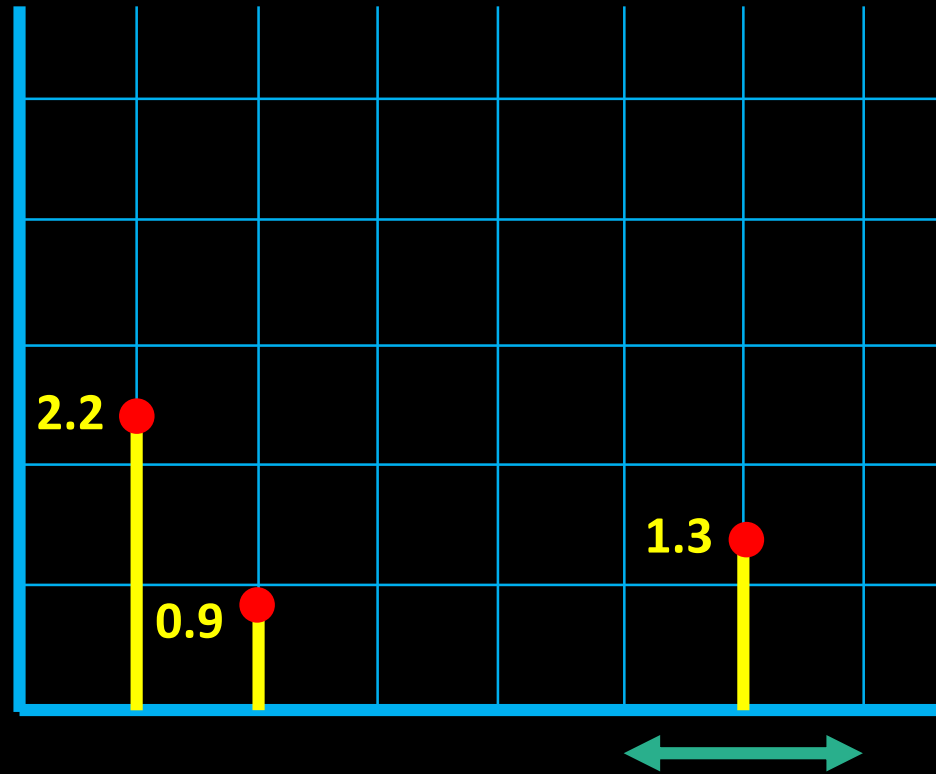
Quadratic Functions – finding minimum

$$f(x): ax^2 + bx + c$$



Quadratic Functions – finding minimum

$$f(x): ax^2 + bx + c$$



Where would you sample next?

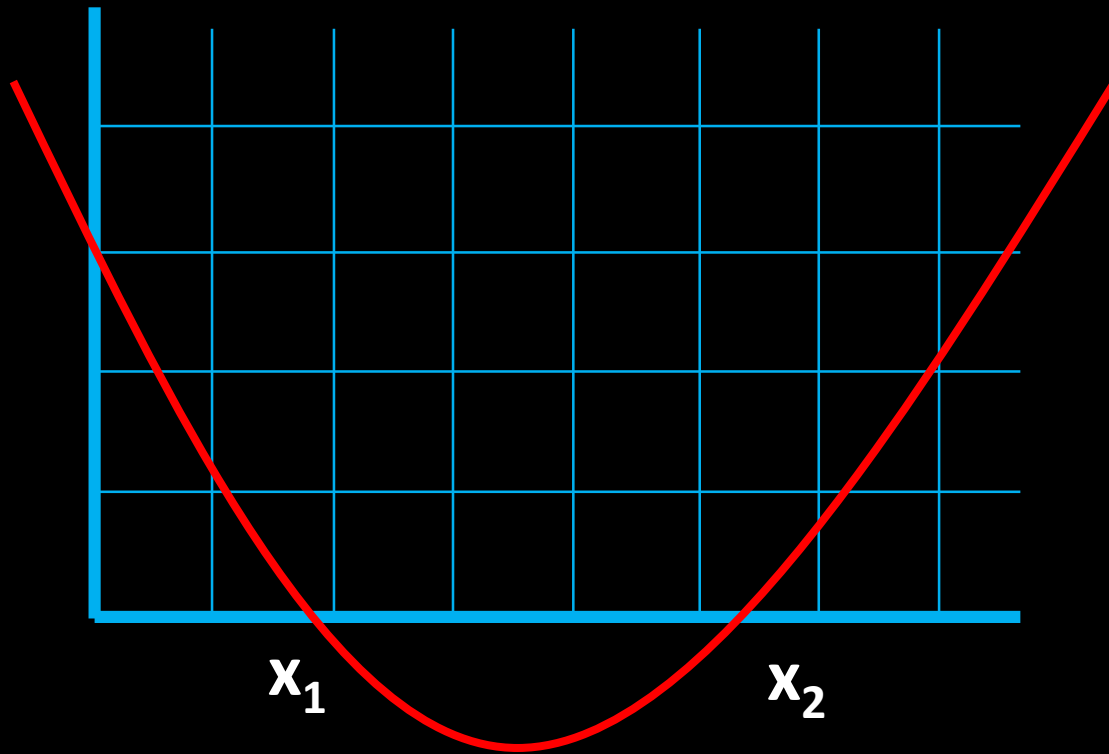
Quadratic Functions – finding minimum

$$f(x): ax^2 + bx + c$$



Quadratic Functions – intersections with x

$$f(x): ax^2 + bx + c$$



$$ax^2 + bx + c = 0$$

x	indeterminate variable
a	quadratic coefficient
b	linear coefficient
c	constant coefficient

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$x_{12} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Quadratic Functions – intersections with x

$$f(x): ax^2 + bx + c$$

$$x_{12} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Wolfram Alpha command:

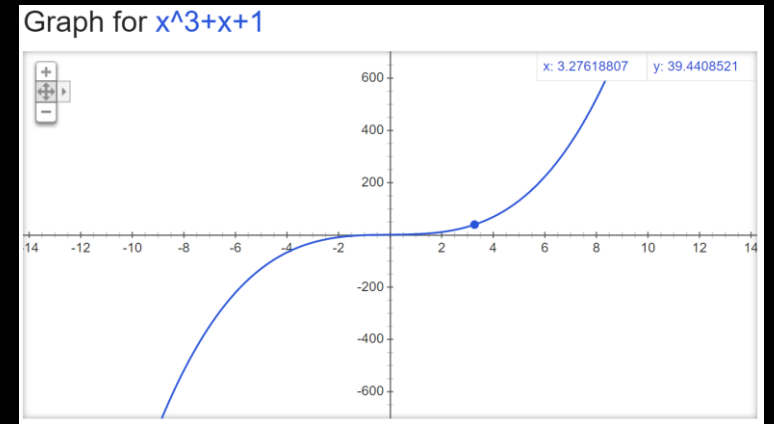
quadratic formula calculator

[wolframalpha.com/input/?i=quadratic+formula+calculator](https://www.wolframalpha.com/input/?i=quadratic+formula+calculator)

Polynomial Functions

For example:

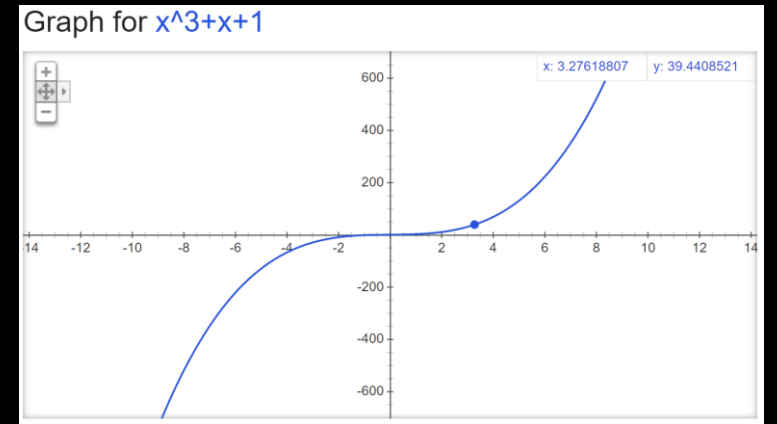
$$f(x): x^3 + x + 1$$



Polynomial Functions

For example:

$$f(x): x^3 + x + 1$$



More generally:

$$f(x): a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

ps: quadratic functions are also polynomials of 2 degree

Polynomial Functions

$$f(x): a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Wolfram alpha:

[wolframalpha.com/examples/mathematics/algebra/polynomials/](https://www.wolframalpha.com/examples/mathematics/algebra/polynomials/)

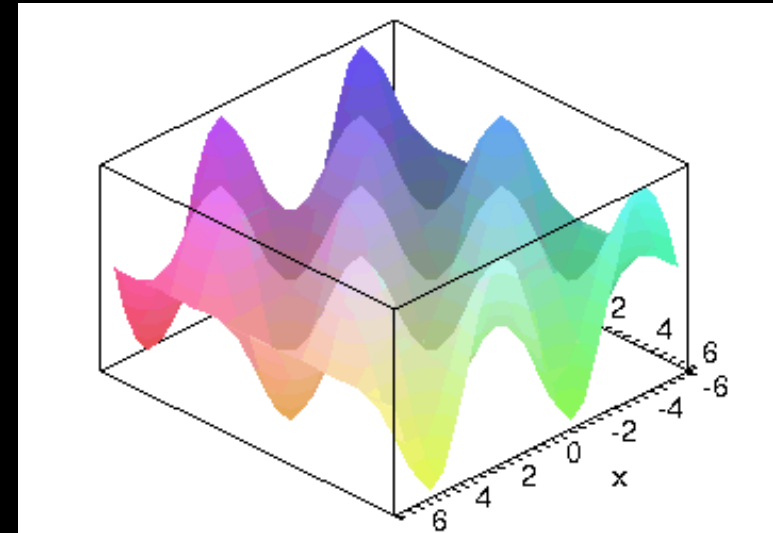
Multivariate Functions

So far...

$$f(x): x^3 + x + 1$$

Multiple variables – x, y, z, \dots

$$f(x, y): x^3 + y + 12$$



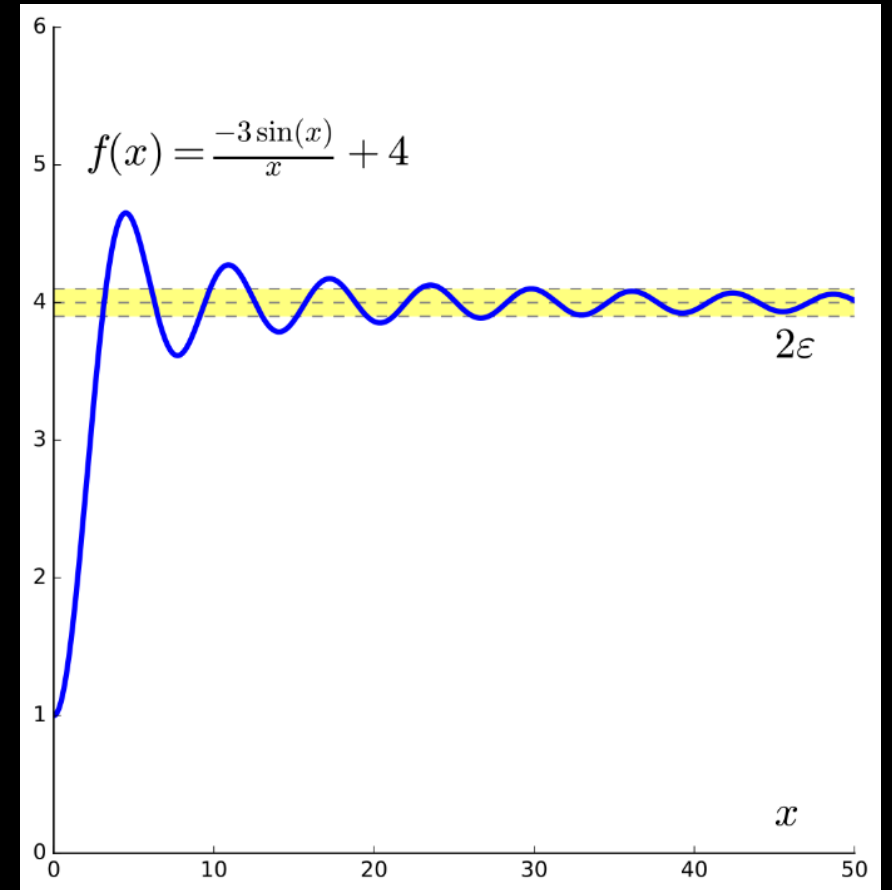
1 variable (x) – visualize in 2D, 2 variables (x, y) – vis. in 3D, ...

Functions

- Slope
- Min, Max, intersection with 0
- Visualization, inspection (wolfram)

Limits

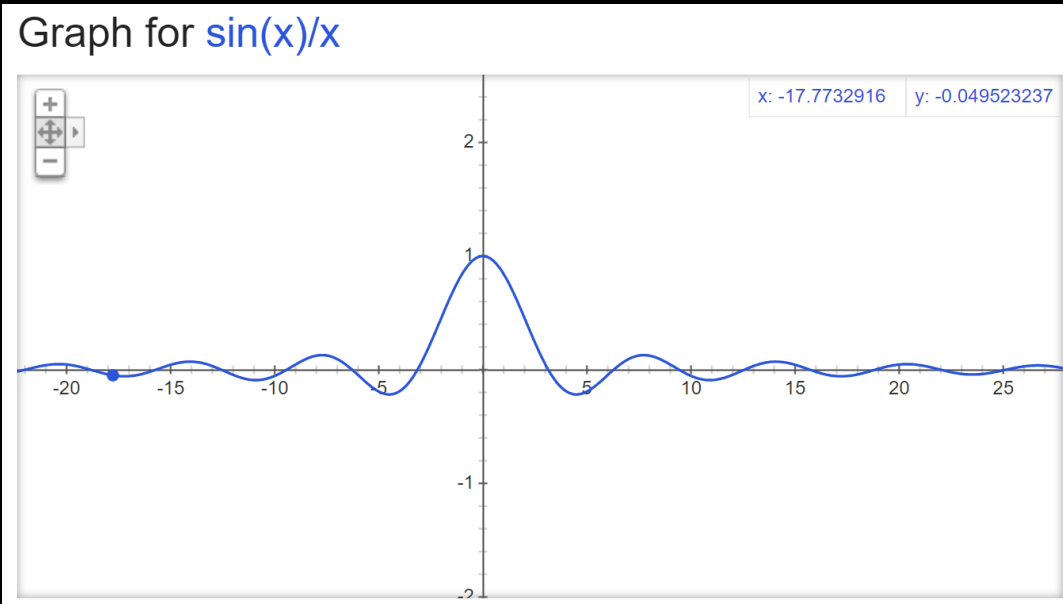
- In mathematics, a limit is the value that a function "approaches" as the input "approaches" some value.



Limits

$$f(x): (\sin x)/x$$

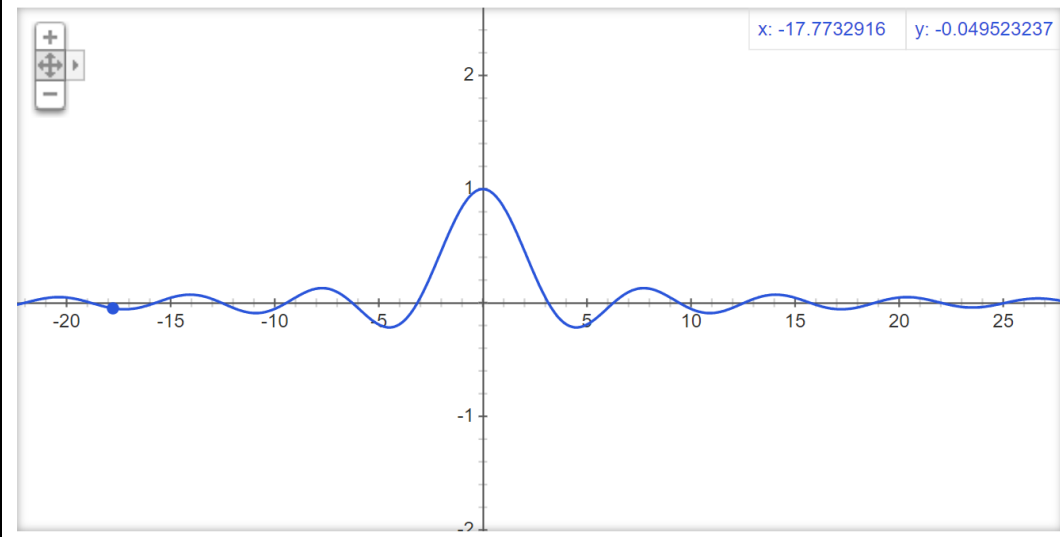
- $f(0) = \text{err, dividing by } 0 = \text{not defined}$



Limits

$$f(x): (\sin x)/x$$

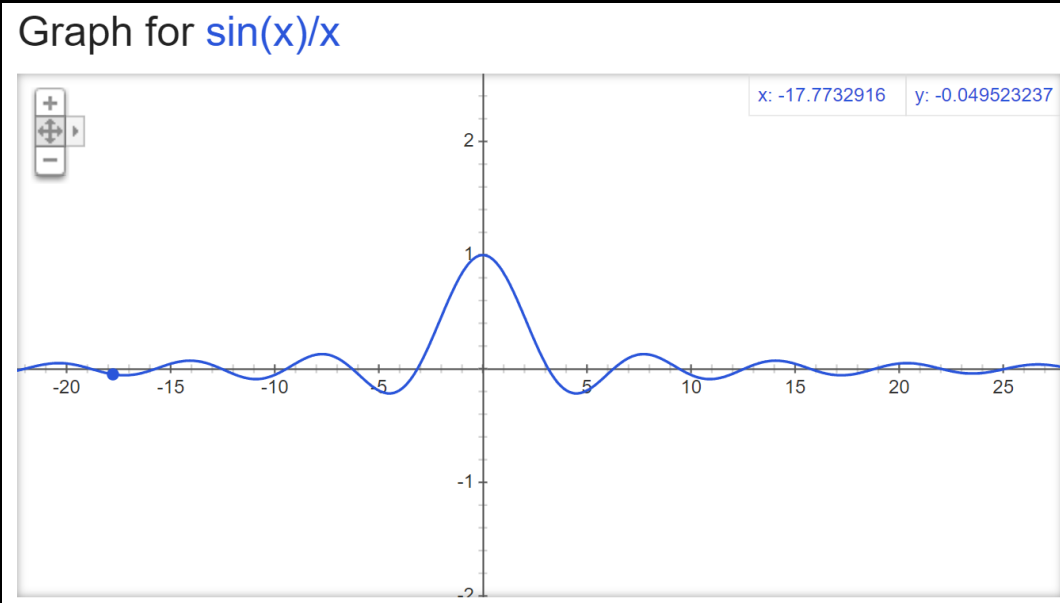
Graph for $\sin(x)/x$



- $f(0) = \text{err, dividing by } 0 = \text{not defined}$
- $f(1) = 0.841471\dots$
- $f(0.1) = 0.998334\dots$
- $f(0.01) = 0.999983\dots$

Limits

$$f(x): (\sin x)/x$$



- $f(0) = \text{err, dividing by } 0 = \text{not defined}$
 - $f(1) = 0.841471\dots$
 - $f(0.1) = 0.998334\dots$
 - $f(0.01) = 0.999983\dots$
-
- Although the function $(\sin x)/x$ is **not defined at zero**, as x becomes closer and closer to zero, $(\sin x)/x$ becomes arbitrarily close to 1. In other words, the limit of $(\sin x)/x$ as x approaches zero equals 1.

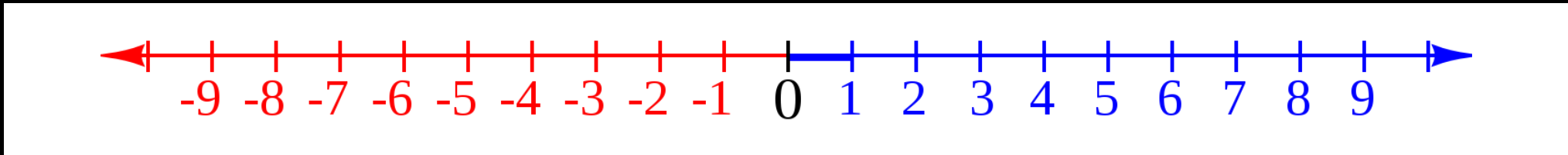
Pause 1

Number representations

- Integers
 - Natural numbers
 - Real numbers
 - Complex numbers
- Int
 - Float
 - Double, ...

Integers

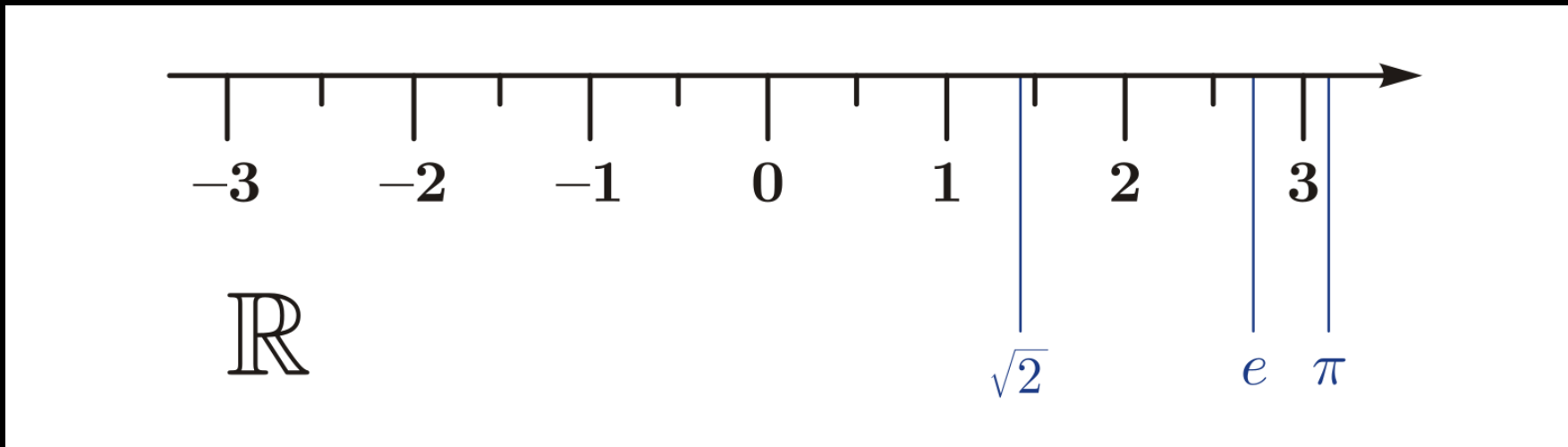
- \mathbb{Z}
- Integers can be thought of as discrete, equally spaced points on an infinitely long number line.



- \mathbb{N}
- Natural numbers are non-negative integers (\mathbb{N}_0 includes 0).

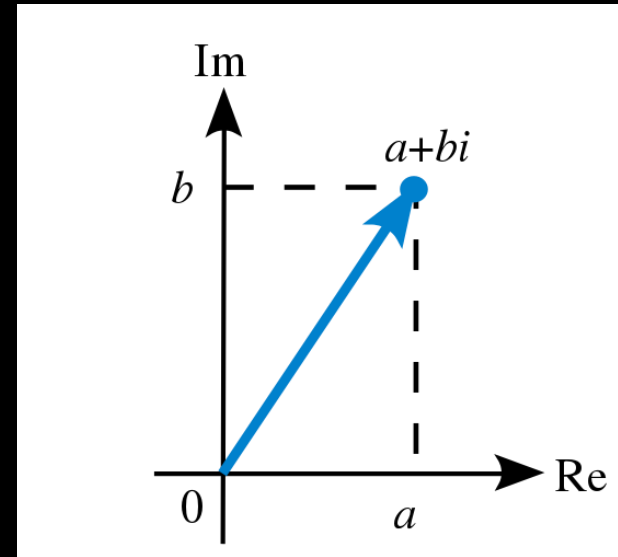
Real numbers

- \mathbb{R}
- In mathematics, a real number is a value of a continuous quantity that can represent a distance along a line.

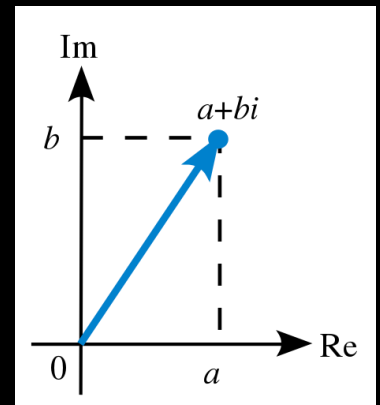
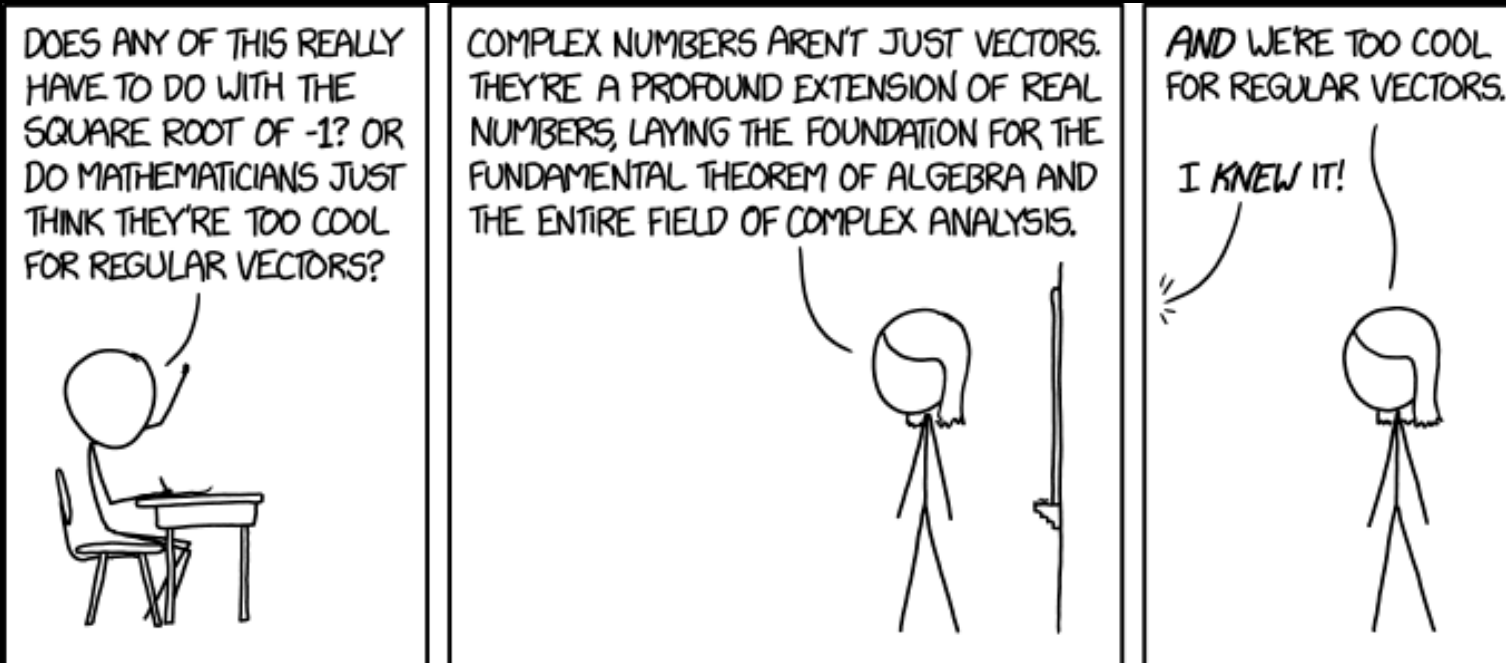


Complex numbers

- \mathbb{C}
- A complex number is a number that can be expressed in the form $a+bi$, where a and b are real numbers, and i is a solution of the equation $x^2=-1$. Because no real number satisfies this equation, i is called an imaginary number.



Complex numbers



Integers in programming

- Variable which holds integer number. It can be signed (allows negative values) or unsigned (doesn't).
- Number of bytes corresponds to what is the minimal and maximal number we can store in it:

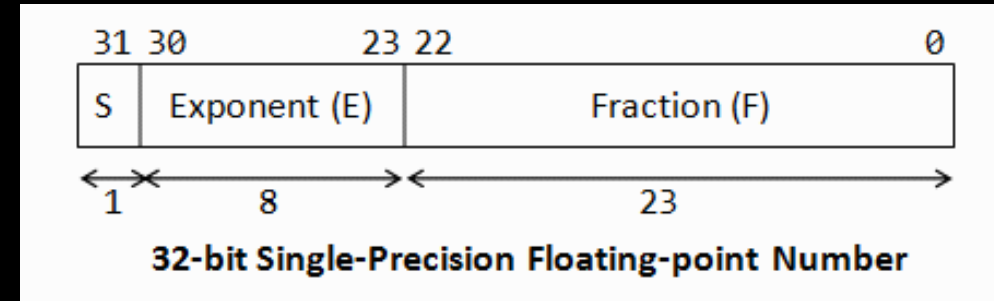
[https://en.wikipedia.org/wiki/Integer_\(computer_science\)](https://en.wikipedia.org/wiki/Integer_(computer_science))

- In Python:

```
import sys
print(sys.maxsize)
# 2147483647
```

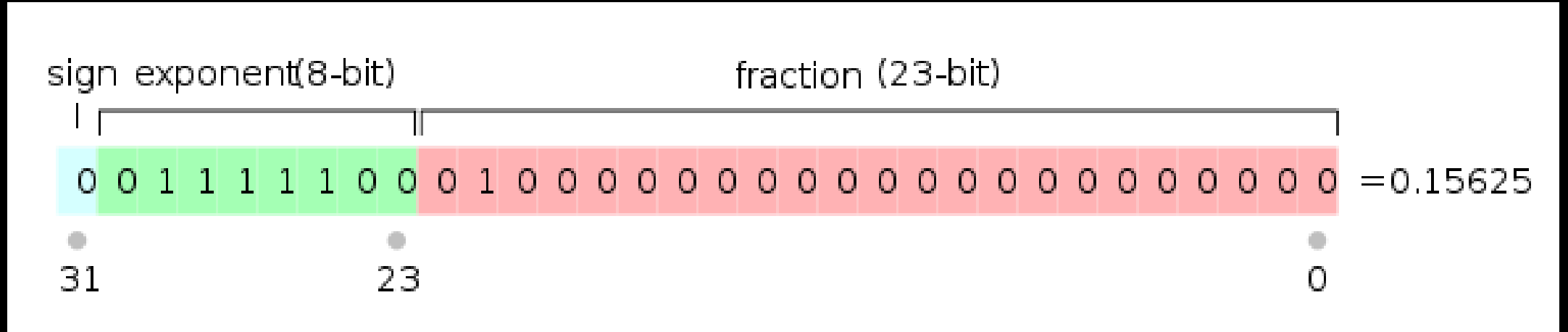
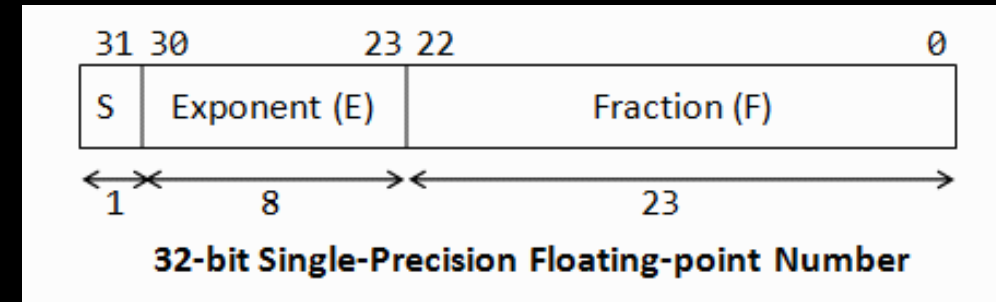
Floats in programming

- Variables which hold real numbers.
- 32 bits used as Sign, Exponent, and Fraction



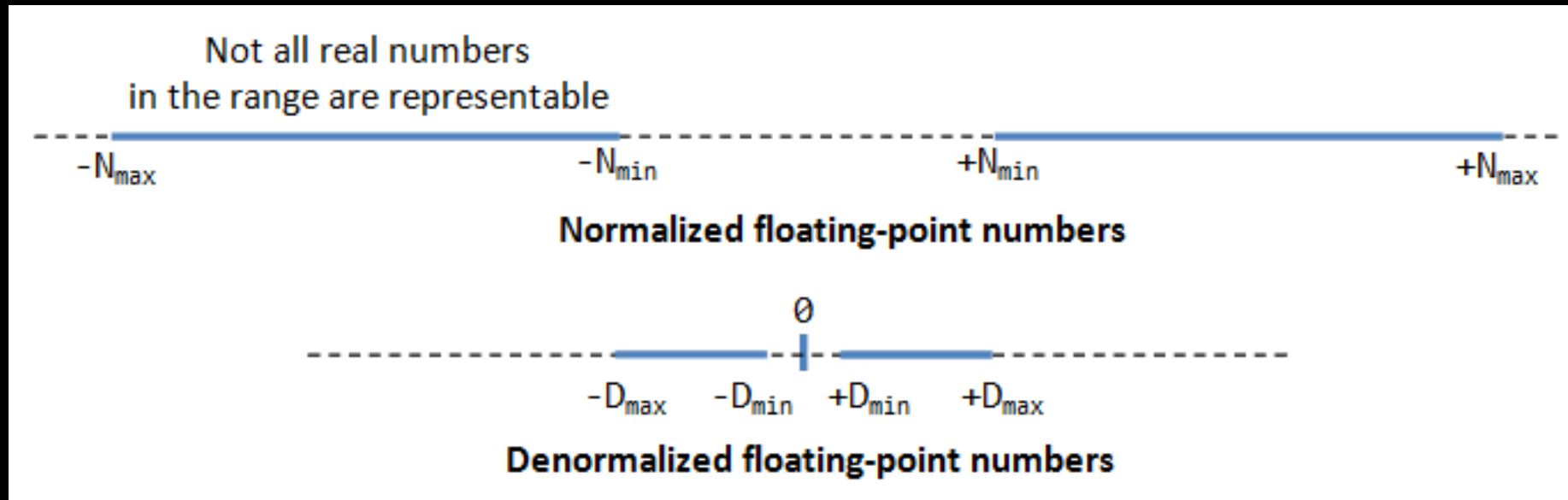
Floats in programming

- Variables which hold real numbers.
- 32 bits used as Sign, Exponent, and Fraction



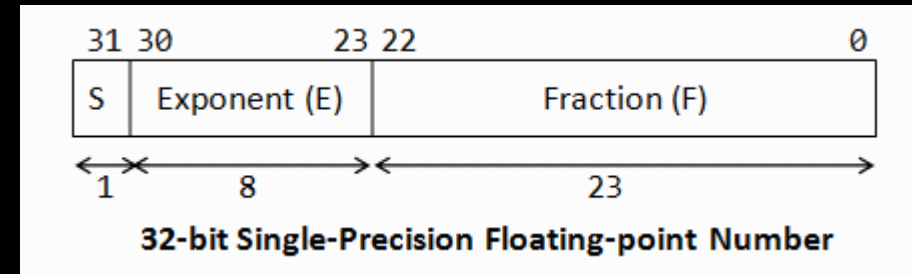
Floats in programming

- S = Sign, E = Exponent, F = Fraction
- For large numbers “normalized” $(-1)^S * 1.F * 2^{(E-127)}$
- For small numbers “denormalized” (E=0) $(-1)^S * 0.F * 2^{(-126)}$



Floats in programming

- In Python:



```
import sys
print(sys.float_info)
# sys.float_info(max=1.7976931348623157e+308, max_exp=1024,
max_10_exp=308, min=2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307,
dig=15, mant_dig=53, epsilon=2.220446049250313e-16, radix=2, rounds=1)
```


Pause 2

Task 1 - Python

- Quadratic function solver

$$x_{12} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Ask the user for values of a,b,c in $f(x)$: $ax^2 + bx + c$
- Then solve for the intersections
- *Hint*: depending on the value of the discriminant $D = b^2 - 4ac$ we have
 - $D > 0$... two solutions
 - $D == 0$... one solution
 - $D < 0$... no real solution

Task 2 - Python

- **Plot quadratic function**

```
import matplotlib.pyplot as plt

x_values=[]
y_values=[]

# TODO: sample the x and y values!

fig= plt.figure()
axes=fig.add_subplot(111)
axes.plot(x_values,y_values)
plt.show()
```

- Starter code:

git [week02_math-recap/w02_plot_quadratic_function.py](#) / link [colab](#)