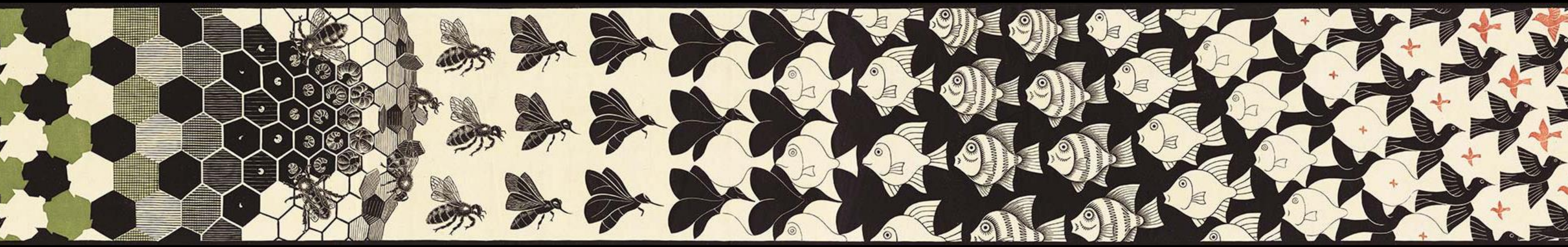
Data, Math and Methods

Week 3, Operators & Logic



~ Vítek Růžička

Today

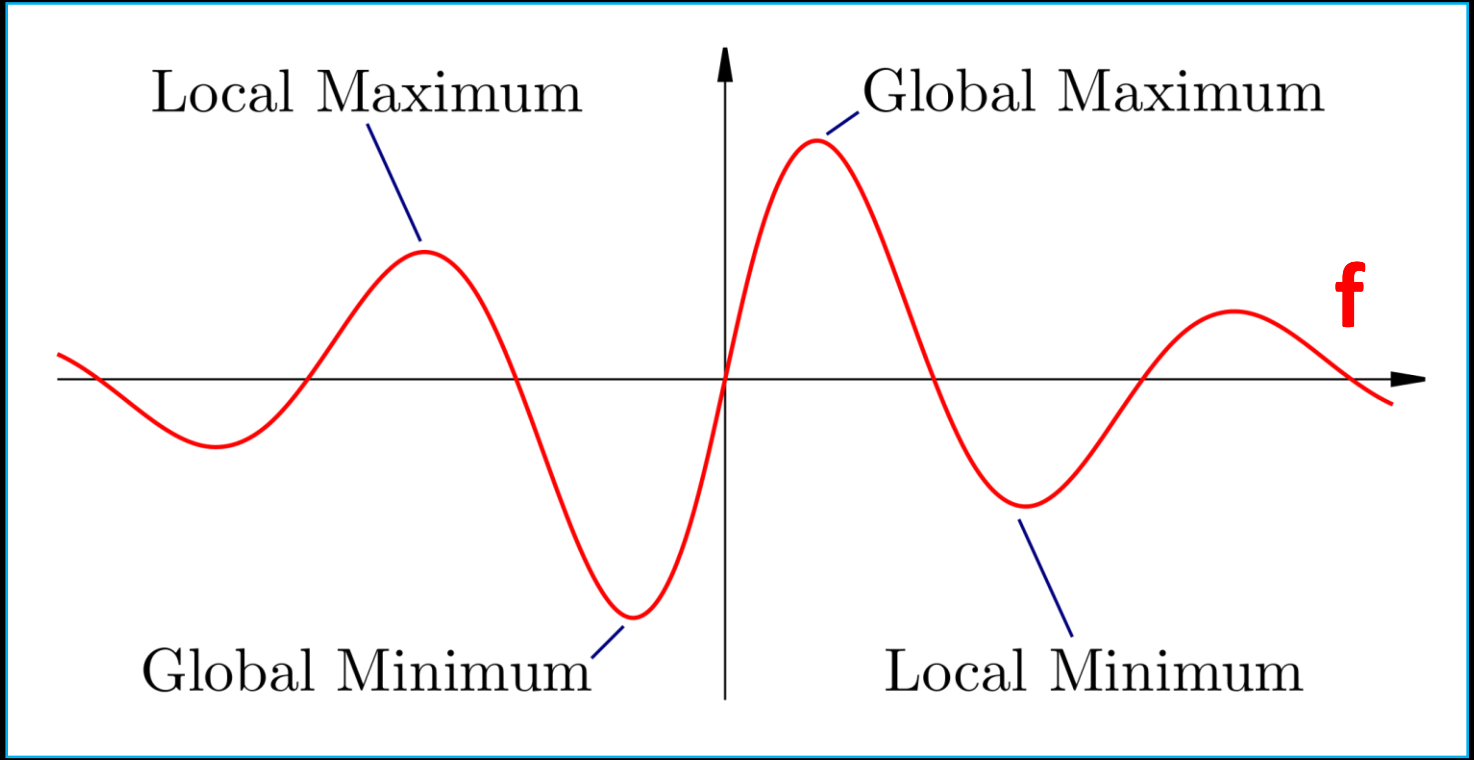
- Mathematic notation
- Operation properties
- Paper: Logic
- Code: Drawing a circle

$\forall$ $\exists$

For all Exists

Mathematical notation

- Let’s define something relatively simple using that notation:

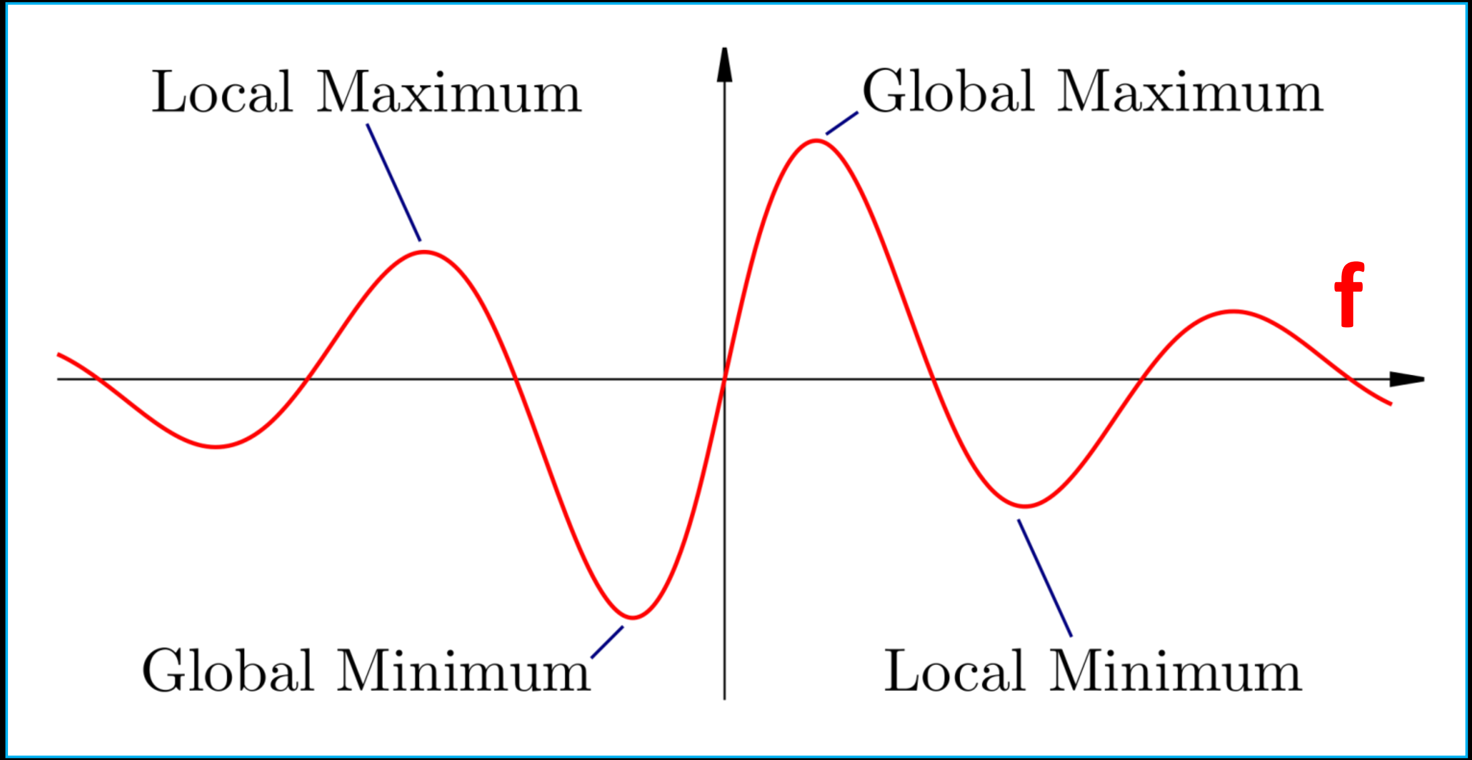


Local maximum

Global maximum

Mathematical notation

- Let’s define something relatively simple using that notation:



Local maximum

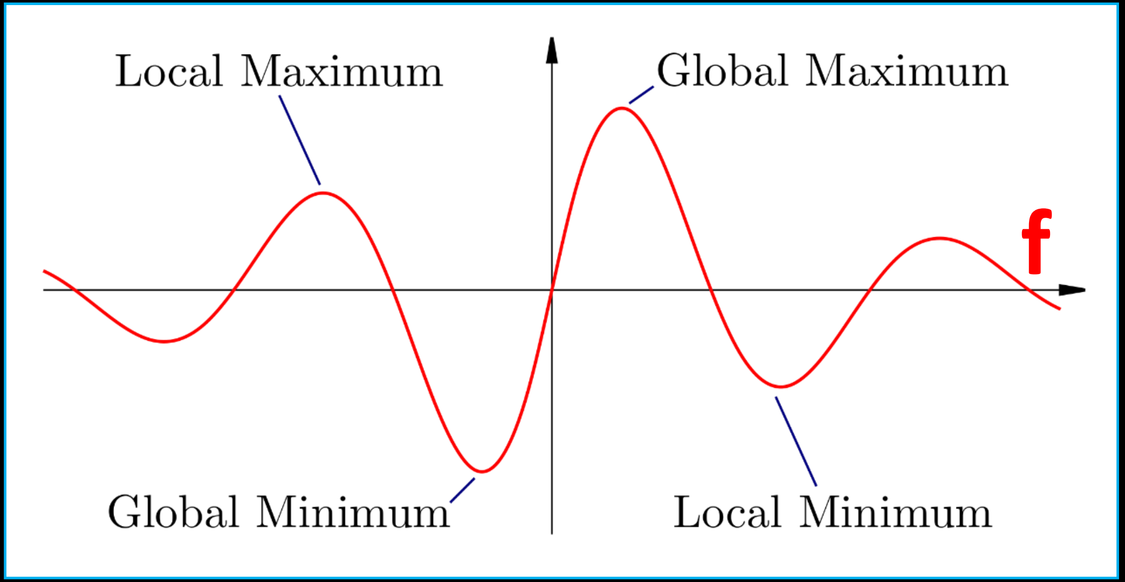
Global maximum

Let’s try explaining / defining one of them …

Function $f: X \rightarrow \mathbb{R}$

$\mathbb{R}$ … Domain of all Real numbers

X … Domain of all numbers we can put into the function (imagine it also as real numbers).

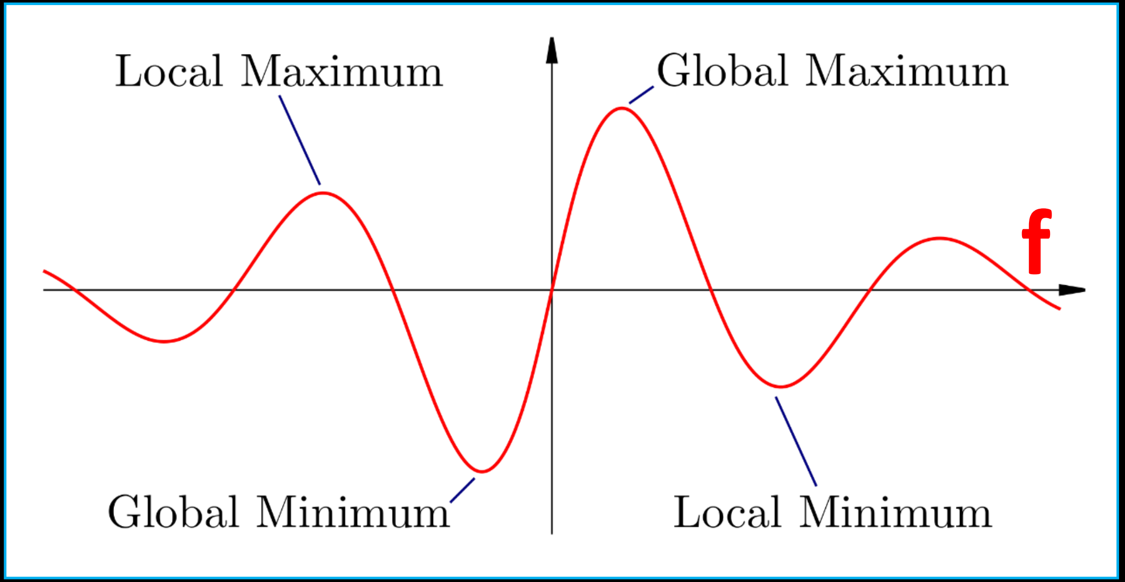


Number $c \in X$ … a number from that domain.

Function $f \colon X \to \mathbb{R}$

$\mathbb{R}$ ... Domain of all Real numbers

X ... Domain of all numbers we can put into the function (imagine it also as real numbers).



Number $c \in X$ ... a number from that domain.

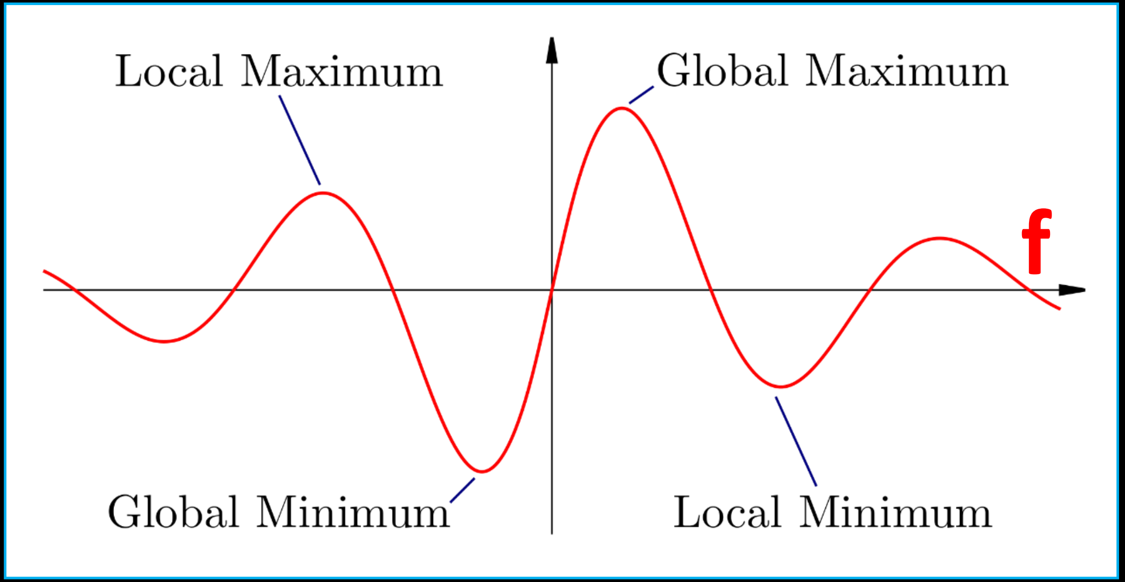
Definition:

Let $c \in X$ be the **global maximum point** of a function $f \colon X \to \mathbb{R}$ if:

Function $f: X \rightarrow \mathbb{R}$

$\mathbb{R}$ … Domain of all Real numbers

X … Domain of all numbers we can put into the function (imagine it also as real numbers).



Number $c \in X$ … a number from that domain.

Definition:

Let $c \in X$ be the **global maximum point** of a function $f: X \rightarrow \mathbb{R}$

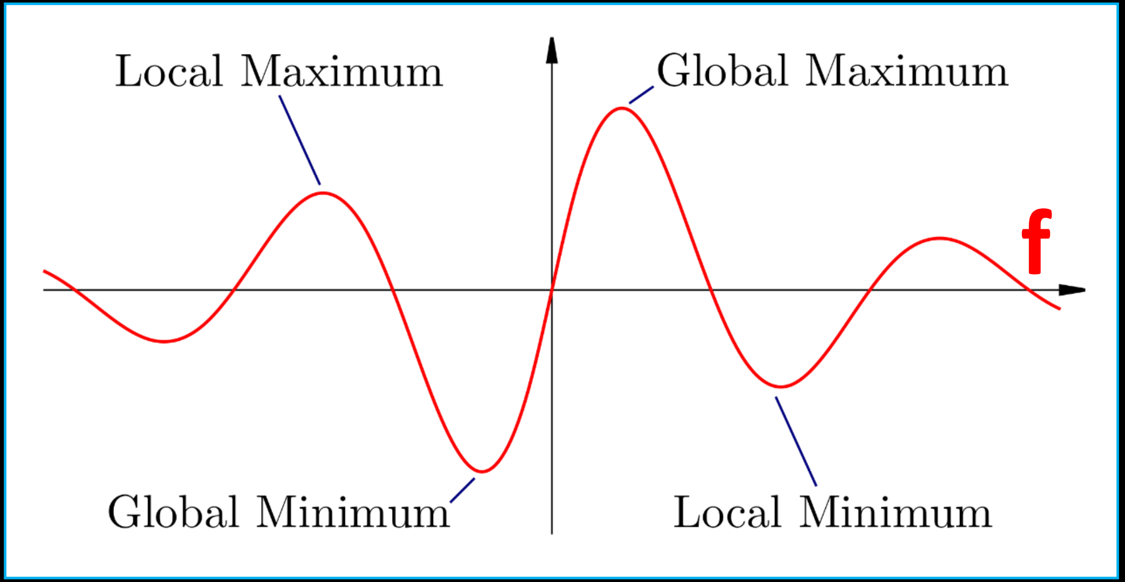
if: $\forall x \in X$

for all numbers

Function $f\colon X \to \mathbb{R}$

$\mathbb{R}$ ... Domain of all Real numbers

X ... Domain of all numbers we can put into the function (imagine it also as real numbers).



Number $c \in X$ ... a number from that domain.

Definition:

Let $c \in X$ be the **global maximum point** of a function $f\colon X \to \mathbb{R}$

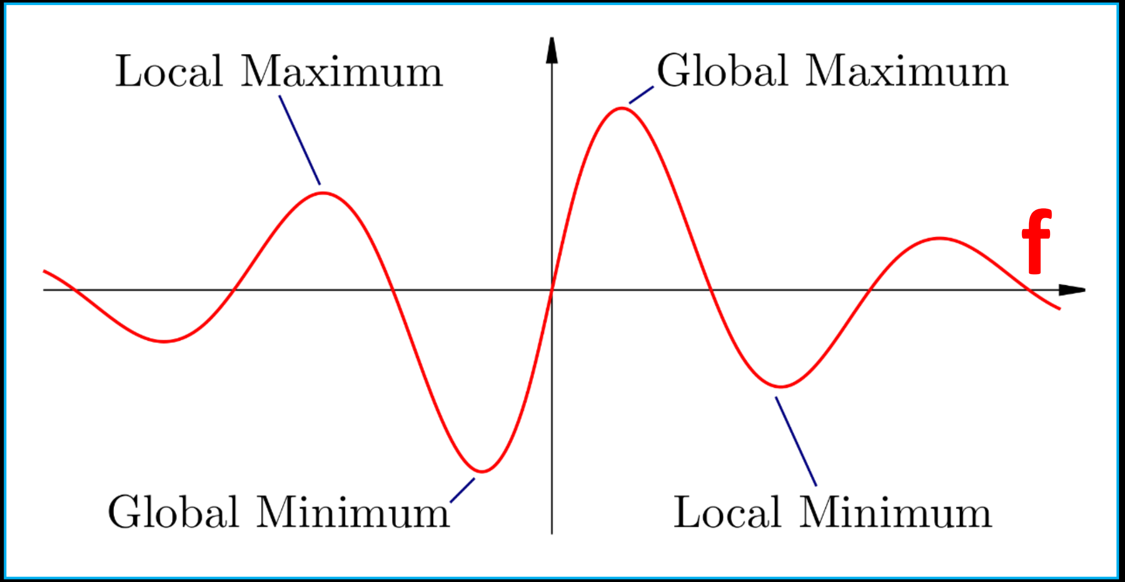
if: $\forall x \in X$ this is true: $f(c) \geq f(x)$.

for all numbers

Function $f\colon X \rightarrow \mathbb{R}$

$\mathbb{R}$ ... Domain of all Real numbers

X ... Domain of all numbers we can put into the function (imagine it also as real numbers).



Number $c \in X$ ... a number from that domain.

Definition:

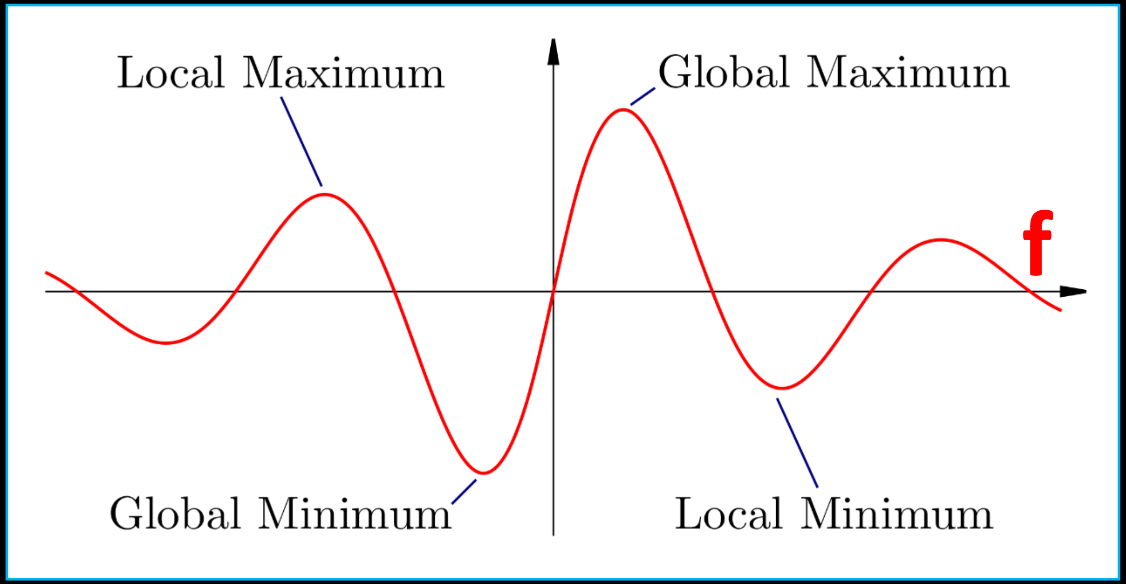
Let $c \in X$ be the **global maximum point** of a function $f\colon X \rightarrow \mathbb{R}$

if: $\forall x \in X$ this is true: $f(c) \geq f(x)$.

for all numbers

this point gives larger function value than any of the other numbers we try.

Global maximum



Definition:

Let $c \in X$ be the **global maximum point** of a function $f\colon X \to \mathbb{R}$

if: $\forall x \in X\colon\ f(c) \geq f(x)$.

In mathematics we lay brick by brick the basic building blocks (definitions) to then talk about more complicated concepts.

Mathematical notation helps us write in concise way in shared abstract language.

Operators

- What is an operator?

  +

  *

  -

  %

  not ()

  sin ()

  cos ()

  …

Operators

- What is an operator?

  +

  *

  -

  %

  not ()

  sin ()

  cos ()

  …

But rather than calculating some basic examples using these (which would be easy), let's talk about the rules and properties they have…

Operators

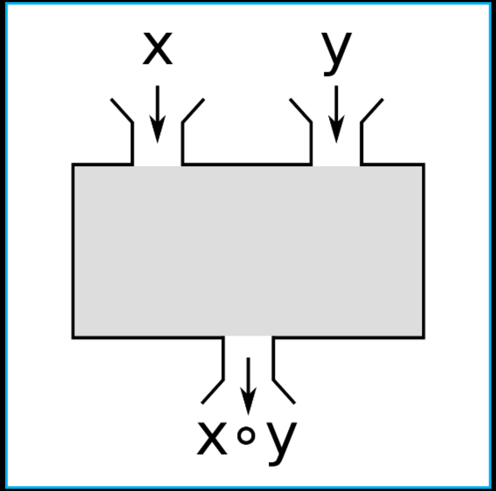
- There are rules about priority of evaluation:
  For example:

    - *, % would be first
    - +, - would go after

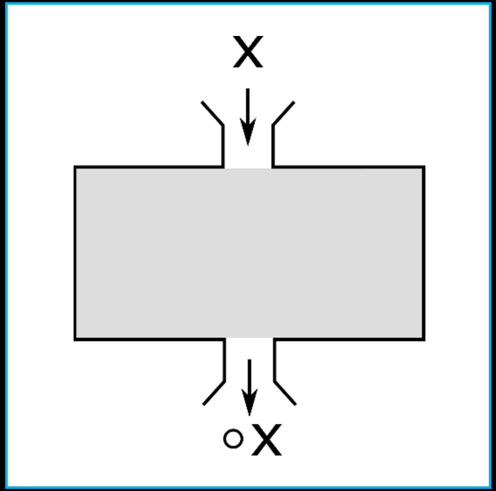
- Sometimes order matters, sometimes not; this also influences how we can reformate the whole formula (simplify).

Operators

- Binary operation
  - Takes two inputs
  - For example: a+b, a*b



- Unary operation
  - Takes a single input
  - For example: not(a), sin(a)



Operators properties

- **Associative property:**

- *(human readable definition :))*
  Rearranging the parentheses in such an expression will not change its value.

  $(2+3)+5 \quad = \quad 2+(3+5)$
  $(4*2)*8 \quad = \quad 4*(2*8)$
  $(3-1)-3 \quad \neq \quad 3-(1-3)$ because $(2)-3 \neq 3-(-2)$

Operators properties

- **Associative property:**

A binary operation ■ is called associative if:

$$\forall a, b, c \in \mathbb{R}: (a \blacksquare b) \blacksquare c = a \blacksquare (b \blacksquare c)$$

Operators properties

- **Associative property:**

A binary operation $*$ is called associative if:

$$\forall a, b, c \in \mathbb{R}: (a * b) * c = a * (b * c)$$

Operators properties

- **<u>Associative property:</u>**

A binary operation $*$ is called associative if:

$$\forall a, b, c \in \mathbb{R}: (a * b) * c = a * (b * c)$$

If: for any three numbers we can just *willy nilly* swap the brackets like this

For example:

- This is true for:
- This is not true for:

Operators properties

- **Associative property:**

A binary operation $*$ is called associative if:

$$\forall a, b, c \in \mathbb{R}: (a * b) * c = a * (b * c)$$

If: for any three numbers we can just *willy nilly* swap the brackets like this

For example:

- This is true for: $*$ multiplication, + addition
- This is not true for:

Operators properties

- <u>**Associative property:**</u>

A binary operation $*$ is called associative if:

$$\forall a, b, c \in \mathbb{R}: (a * b) * c = a * (b * c)$$

If: for any three numbers we can just *willy nilly* swap the brackets like this

For example:

- This is true for: $*$ multiplication, + addition
- This is not true for: - subtraction

Operators properties

- **Associative property:**

A binary operation $*$ is called associative if:

$$\forall a, b, c \in \mathbb{R}: (a * b) * c = a * (b * c)$$

Real world example:

"concatenating two words together" – is associative

Operators properties

- **Associative property:**

A binary operation $*$ is called associative if:

$$\forall a, b, c \in \mathbb{R}: (a * b) * c = a * (b * c)$$

As a function:

Function f is associative if: $f(f(x, y), z) = f(x, f(y, z))$

Operators properties

- **Commutative property:**

- *(human readable definition :))*
  A binary operation is commutative if **changing the order of the operands does not change the result**.

  $2+5 \quad = \quad 5+2$
  $4*2 \quad = \quad 2*4$
  $10-2 \quad \neq \quad 2-10 \qquad$ because $8 \neq -8$

Operators properties

- **Commutative property:**

A binary operation ■ is called commutative if:

$$\forall a, b \in \mathbb{R}: a \blacksquare b = b \blacksquare a$$

Operators properties

- **Commutative property:**

A binary operation $*$ is called commutative if:

$$\forall a, b \in \mathbb{R}: a * b = b * a$$

Operators properties

- **Commutative property:**

A binary operation $*$ is called commutative if:

$$\forall a, b \in \mathbb{R}: a * b = b * a$$

Real world example:

- *"Putting on socks resembles a* ***commutative*** *operation since which sock is put on first is unimportant. Either way, the result (having both socks on), is the same. In contrast, putting on underwear and trousers is* ***not commutative****."*
- *"Concatenation of two stings" – is* ***not commutative***

Operators properties

- **Commutative property:**

A binary operation $*$ is called commutative if:

$$\forall a, b \in \mathbb{R}: a * b = b * a$$

As a function:

Function f is commutative if: $f(a, b) = f(b, a)$

Operators properties

- Operation **Addition** (+) is *associative* and *commutative*.
  - Associative $\forall a, b, c \in \mathbb{R}: (a + b) + c = a + (b + c)$
  - Commutative $\forall a, b \in \mathbb{R}: a + b = b + a$

Operators properties

- Operation **Concatenation**, adding two word together …
  - Associative $\forall a, b, c \in \mathbb{R}: (a + b) + c = a + (b + c)$
    - (T+E)+A = T+(E+A)

Operators properties

- Operation **Concatenation**, adding two word together …
  - Associative $\forall a, b, c \in \mathbb{R}: (a + b) + c = a + (b + c)$
    - (T+E)+A = T+(E+A)
    - **Yes**, it is … … so we swap parentheses around

Operators properties

- Operation **Concatenation**, adding two word together …
  - Associative $\forall a, b, c \in \mathbb{R}: (a + b) + c = a + (b + c)$
    - (T+E)+A = T+(E+A)
    - **Yes**, it is … … so we swap parentheses around
  - Commutative $\forall a, b \in \mathbb{R}: a + b = b + a$
    - TE + A = A + TE ?

Operators properties

- Operation **Concatenation**, adding two word together …
  - Associative $\forall a, b, c \in \mathbb{R}: (a + b) + c = a + (b + c)$
    - (T+E)+A = T+(E+A)
    - **Yes**, it is … … so we swap parentheses around
  - Commutative $\forall a, b \in \mathbb{R}: a + b = b + a$
    - TE + A = A + TE ?
    - **No**, it’s not … … but we can’t change the two arguments

Operators properties

- Why do we care about these properties?
  - Given some of them, we can say something about the operator in general. For example if we can swap it around in the formula / if we can collapse it into brackets and overall simplify the problem.

Lesson?

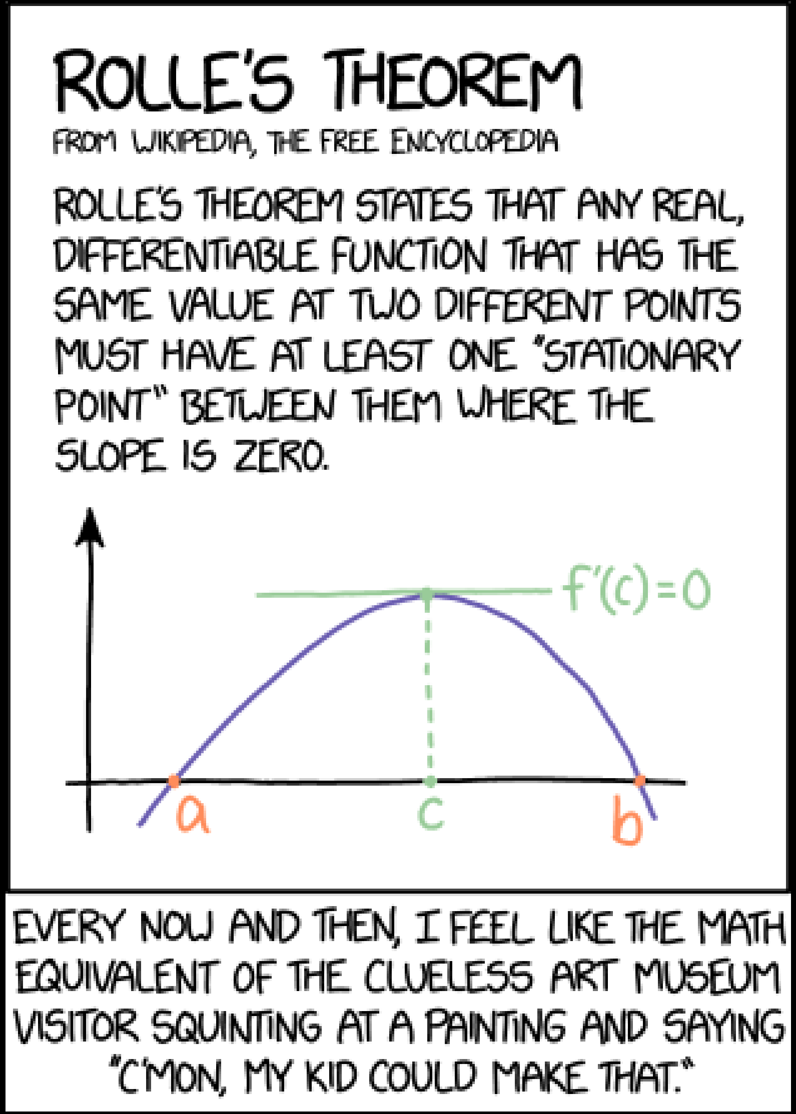
- Lesson of this part is …

  … that Math includes definitions of even the smallest building blocks (*who needs to describe behavior of + ?)*

  … *kinda* allows modularity/re-definition of these basic steps

  … builds from bottom up (definitions of the smallest particle –> slowly describing more complex composites)

Pause 1


ROLLE'S THEOREM
FROM WIKIPEDIA, THE FREE ENCYCLOPEDIA
ROLLE'S THEOREM STATES THAT ANY REAL, DIFFERENTIABLE FUNCTION THAT HAS THE SAME VALUE AT TWO DIFFERENT POINTS MUST HAVE AT LEAST ONE "STATIONARY POINT" BETWEEN THEM WHERE THE SLOPE IS ZERO.
f'(c)=0
a
c
b
EVERY NOW AND THEN, I FEEL LIKE THE MATH EQUIVALENT OF THE CLUELESS ART MUSEUM VISITOR SQUINTING AT A PAINTING AND SAYING "C'MON, MY KID COULD MAKE THAT."

Logic

- Single unit of some more complicated formula is called *variable*. We can think of it as one bit (it can be 0/1 or if you'd like False/True).

a
0
1

This can also be understood as a programming variable – so that variable **a** is either 0 or 1. *(Kinda like enumerating all the possibilities)*

Logic - NOT

- Combinations of several *variables* make up a more complicated logic formula. What makes this formula is the operators between variables *(similarly to having individual numbers and making up formulas from them using +,-,*, …)*

$\boldsymbol{a}$
0
1

A simple unary operator is **the negation: NOT**

$\boldsymbol{\neg a}$
1
0

It flips the values.

NOT $\boldsymbol{a}$ … $\boldsymbol{\neg a}$

(in python: **not** a)

Logic - AND

- There are also some basic binary operators (now we need all possible values for both $\boldsymbol{a}$ and $\boldsymbol{b}$):
- **AND**

$\boldsymbol{a}$	$\boldsymbol{b}$
0	0
0	1
1	0
1	1

$\boldsymbol{a \wedge b}$
?
?
?
?

AND will be True only when both of the variables were True.

Logic - AND

- There are also some basic binary operators (now we need all possible values for both $\boldsymbol{a}$ and $\boldsymbol{b}$):
- **AND**

$\boldsymbol{a}$	$\boldsymbol{b}$
0	0
0	1
1	0
1	**1**

$\boldsymbol{a \wedge b}$
0
0
0
1

AND will be True only when both of the variables were True.

Logic - OR

- **OR**

a	b
0	0
0	1
1	0
1	**1**

$a \vee b$
0
1
1
1

OR will be True if at least one of the variables was True.

Logic

- **Implication**: relationship between statements that holds true when one logically "follows from" other.

a	b
0	0
0	1
1	0
1	**1**

$a \Rightarrow b$
1
1
0
1

If the first one isn’t True, then the implied one may or may not be true and its alright ($0 \Rightarrow 1$, $0 \Rightarrow 0$ both evaluate to True).

Otherwise we can’t imply False from True.

Logic

- **Equals**: It gives the functional value true if both functional arguments have the same logical value, and false if they are different.

a	b
0	**0**
0	1
1	0
1	**1**

$a \Leftrightarrow b$
1
0
0
1

True when the inputs are the same.

Logic – basic:

- **Overview of the basic operators:**

not, and, or, implies, equals

a	b	$\neg a$	$a \wedge b$	$a \vee b$	$a \Rightarrow b$	$a \Leftrightarrow b$
0	0	1	0	0	1	**1**
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	**1**	**1**	**1**	**1**

Logic – extended:

- **XOR:** exclusive OR.
- **NAND:** not AND.
- **NOR**: not OR.

a	b
0	**0**
0	1
1	0
1	**1**

a XOR b
0
1
1
0

$a \wedge b$	a NAND b
0	**1**
0	**1**
0	**1**
1	0

$a \vee b$	a NOR b
0	**1**
1	0
1	0
1	0

Logic – tasks:

- **Task 1:** enumerate the True/False values for the following formula:

$$\neg(\boldsymbol{a} \vee \boldsymbol{b})$$

$\boldsymbol{a}$	$\boldsymbol{b}$
0	**0**
0	1
1	0
1	**1**

$\neg(\boldsymbol{a} \vee \boldsymbol{b})$
?
?
?
?

Logic – tasks:

- **Task 2:** enumerate the True/False values for the following formula:

$$\neg(\boldsymbol{a} \Leftrightarrow \boldsymbol{b})$$

$\boldsymbol{a}$	$\boldsymbol{b}$
0	**0**
0	1
1	0
1	**1**

$\neg(\boldsymbol{a} \Leftrightarrow \boldsymbol{b})$
?
?
?
?

Q: is it similar to anything else?

Logic – tasks:

- **Task 3:** enumerate the True/False values for the following formula:

$$\neg(\boldsymbol{a} \Rightarrow \neg\boldsymbol{b})$$

$\boldsymbol{a}$	$\boldsymbol{b}$
0	**0**
0	1
1	0
1	**1**

$\neg(\boldsymbol{a} \Rightarrow \neg\boldsymbol{b})$
?
?
?
?

Q: is it similar to anything else?

Logic – tasks:

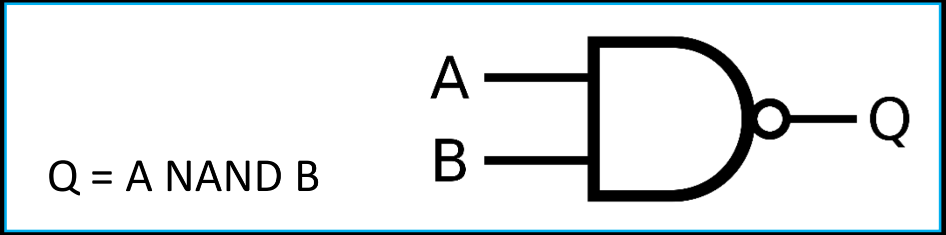
- **Task 4:** enumerate the True/False values for the following formula:

$$(\boldsymbol{a} \vee \boldsymbol{b}) \wedge (\neg \boldsymbol{a} \wedge \boldsymbol{b})$$

$\boldsymbol{a}$	$\boldsymbol{b}$
0	**0**
0	1
1	0
1	**1**

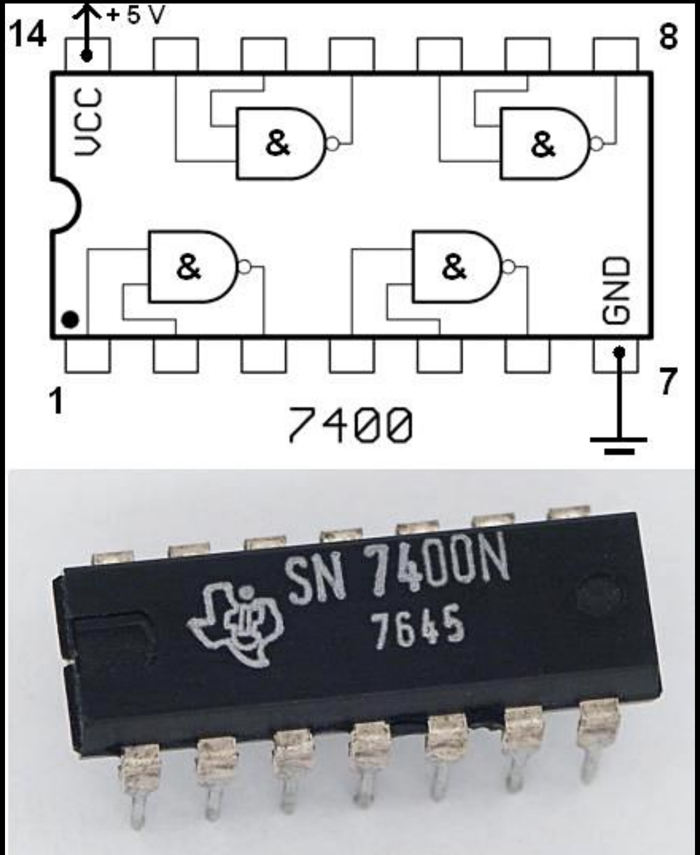
…
?
?
?
?

NAND is all you need



a	*b*
0	0
0	1
1	0
1	1

a NAND *b*
1
1
1
0



NAND is all you need

a	b
0	0
0	1
1	0
1	1

a NAND b
1
1
1
0

a	a	a NAND a
0	0	**1**
0	0	**1**
1	1	**0**
1	1	**0**

NOT a

NAND is all you need

a	b
0	0
0	1
1	0
1	1

a NAND b
1
1
1
0

(**a** NAND **b**) NAND (**a** NAND **b**)
?
?
?
?

What is this?

NAND is all you need

a	b
0	0
0	1
1	0
1	1

a NAND b
1
1
1
0

(**a** NAND **a**) NAND (**b** NAND **b**)
?
?
?
?

What is this?

NAND is all you need

- NAND Logic
- All the rest of conversions at All the rest at: https://en.wikipedia.org/wiki/NAND_logic

- There is also NOR Logic
  - “For example, the first embedded system, the Apollo Guidance Computer, was built exclusively from NOR gates, about 5,600 in total for the later versions.”

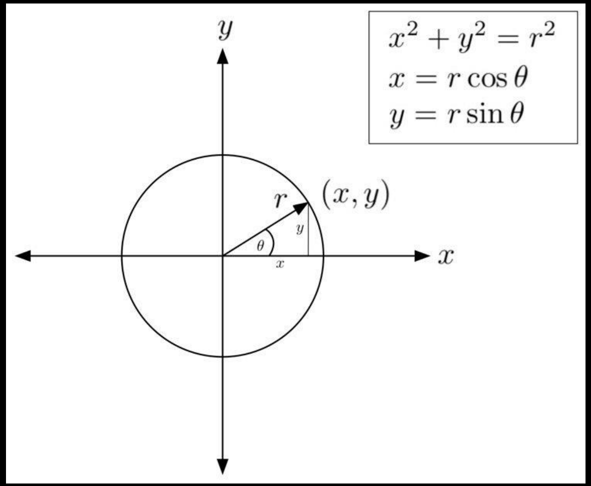
Pause 2

Code: circle

- Inspiration from the demo at: http://www.generative-gestaltung.de/2/sketches/?01_P/P_2_2_3_01

Task: points on circle

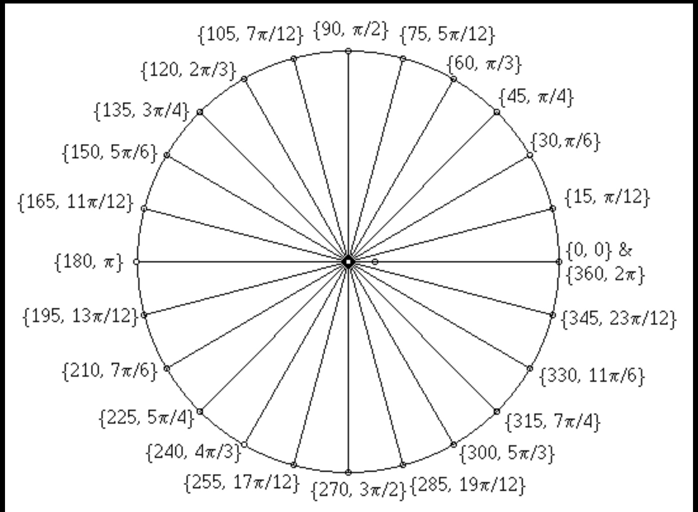
- Points on circle given by a formula:
  - Get (x,y) from the parameter of angle and radius



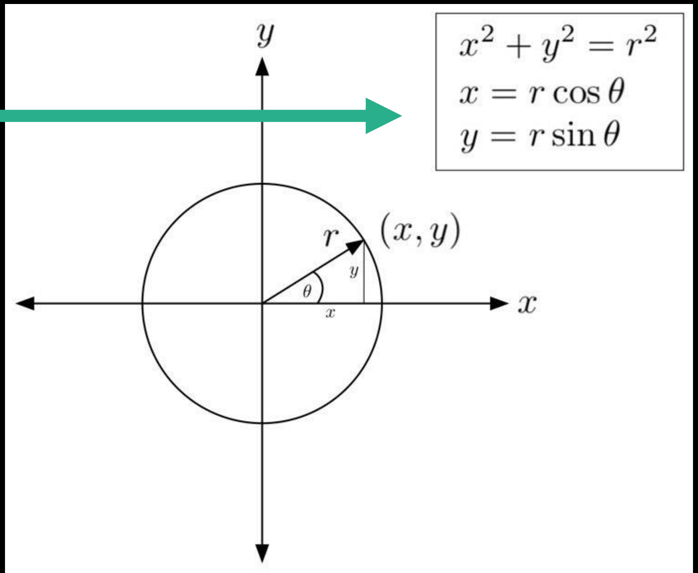
- Task – make a python code which would draw a circle (drawing it point by point, don't use functions to draw it directly)
- **Starter code:** our github [w3_circle_matplotlib_starter.py] or [w3_circle_tkinter_starter.py] (depends on which libraries you have installed … *matplotlib* is probably easier to setup, *tkinter* worked faster as a demo)
- *Note: use your local PCs for this and don't use Colab – we want to have it interactive!*

Task: points on circle

- Formula for (x_point,y_point) from (x,y,r):
  - Angle calculation:
    - We are converting degrees (0 to 360) into radians (0 to 2π)
    - For **t** going from 0 to 360:



- angle = float(**t**) * pi / 180.0
- x_point = r * cos(angle) + x_center
- y_point = r * sin(angle) + y_center

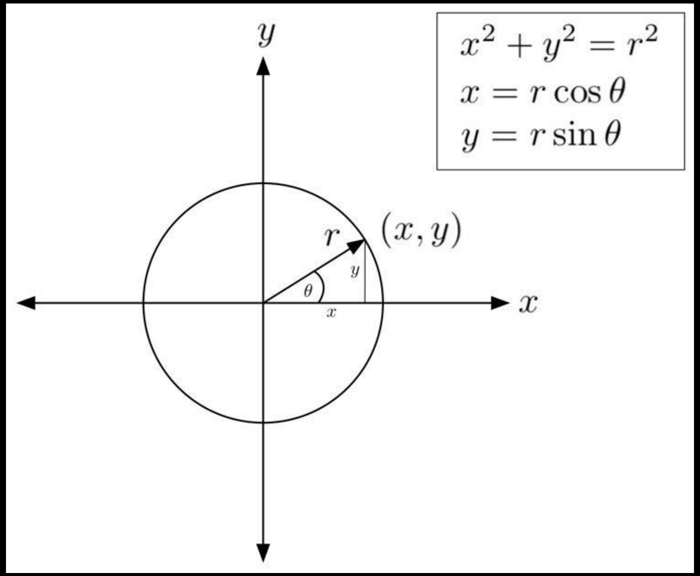


- *Hint: use math.pi, math.cos() and math.sin()*

Task: points on circle

- angle = float(**t**) * pi / 180.0
- x_point = r * cos(angle) + x_center
- y_point = r * sin(angle) + y_center

Hint: use math.pi, math.cos() and math.sin()



- Advanced:
  - <u>**Animate the circles**</u> – for example by changing the parameter **r**
  - *Hint:* **r** could also be given by sin(t),
    this would give it a wave like behavior: