ENR145 Computational Methods for Physicists and Engineers

Xiang Li

Spring 2026

COE COLLEGE

Instructions

Go to

www.menti.com

Enter the code

7679 6481

Or use QR code

Maybe ENR145 is different from other courses:

- No finals.
- Your grade will mostly be based on your accumulated learning achievements and final (dream) project.
- The project can be started since day 1, can be finished by mid-term.
- The course will try to cover lots of weird-ass skills. It’s OK if you hate (some of) them, but now it’s the time to try and to know that you hate them.
- I would like everyone to have chance practicing slide-making and presentation, so:
  - Show me you are good at that can ***boost*** your grade.
  - Show me you are getting better at that can ***boost*** your grade.

- That's how it works (the best case):

Beginning of a chapter

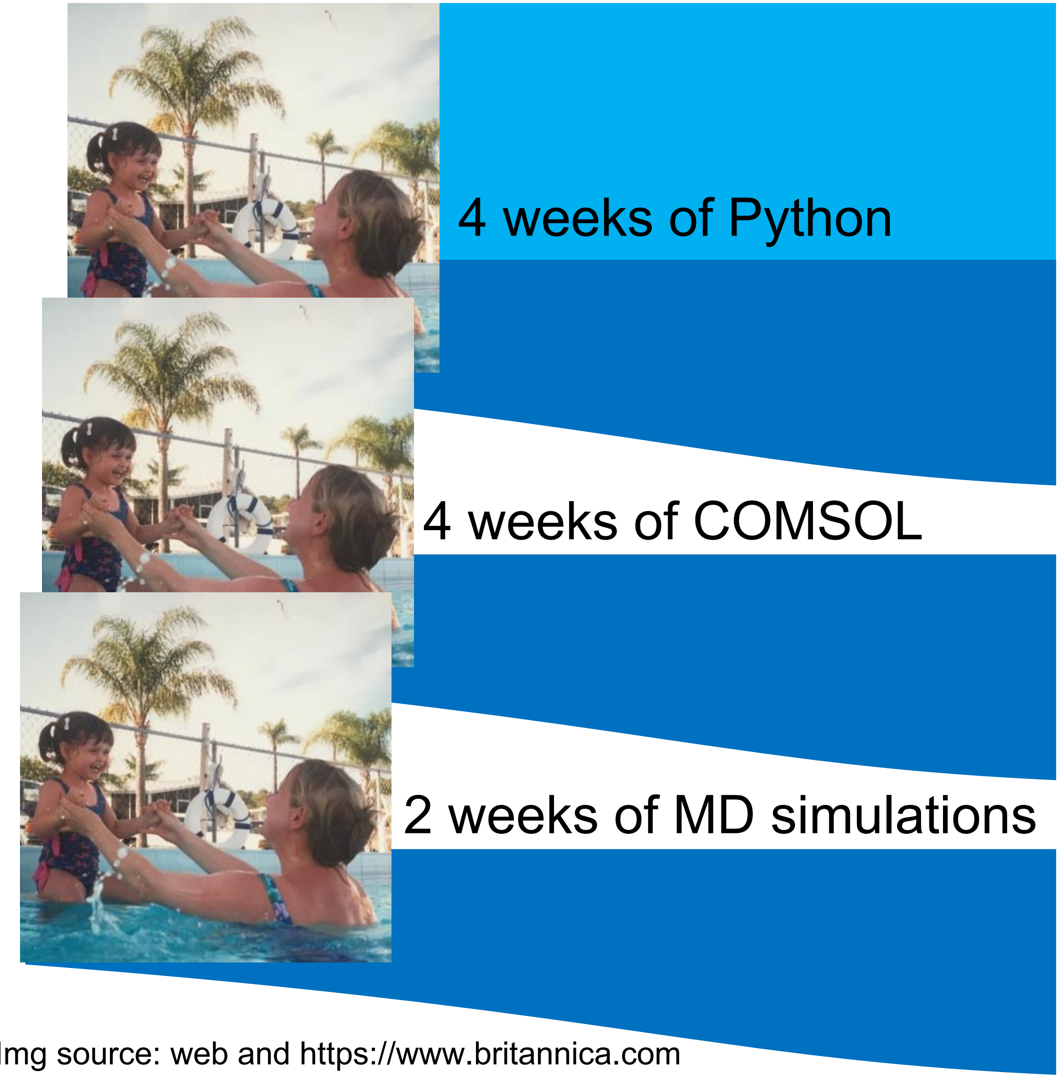
Years later:

COE COLLEGE

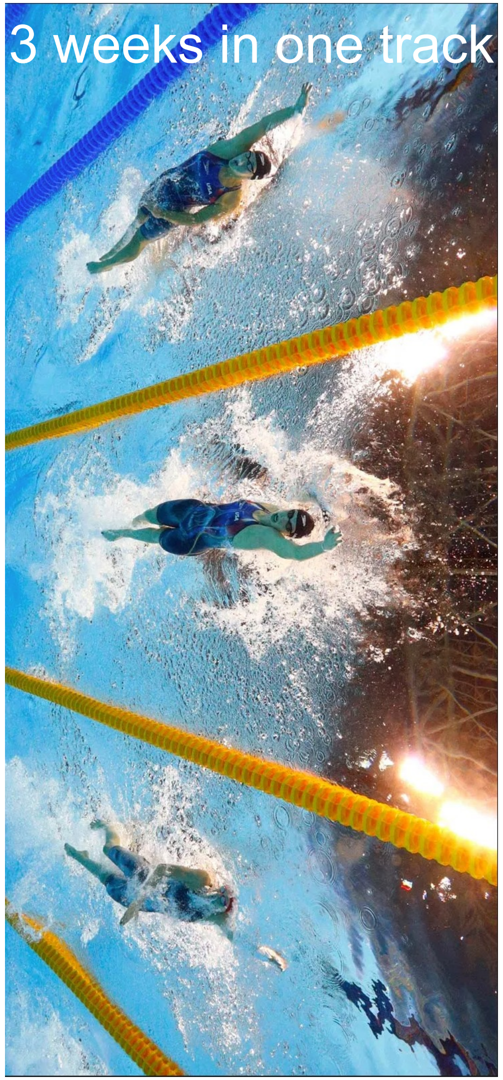
Img source: web and https://www.britannica.com

- How we are going to get there:

Beginning of a chapter



Your final project:

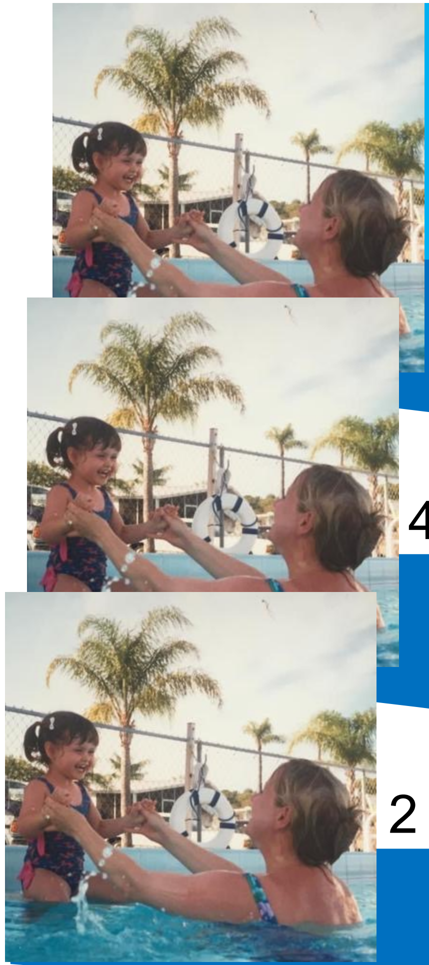


Years later:

COE COLLEGE

Img source: web and https://www.britannica.com

- Here’re the topics we are going to cover:



4 weeks of Python

- The (basic) art of coding
- Karel the Robot
- NumPy and linear algebra
- ODE and UMAP

4 weeks of COMSOL

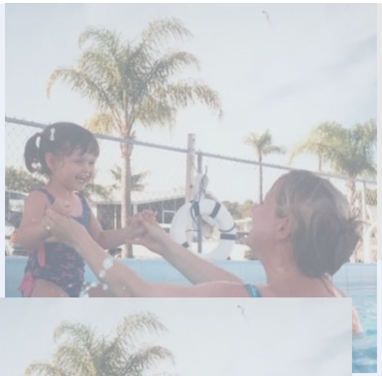
- Why/how/where we solve ODE and PDE
- FEA (structure) and CFD (heat transfer, fluid dynamics)
- CMSOL workshop
- Multiphysics

2 weeks of MD simulations

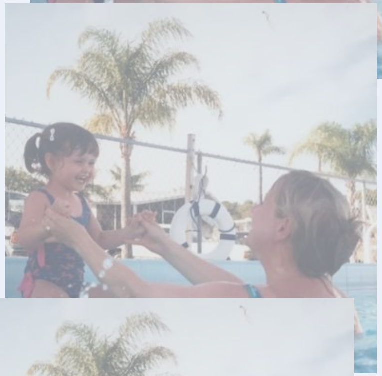
- Linux
- Fun time at clusters
- LAMMPS

Img source: web and https://www.britannica.com

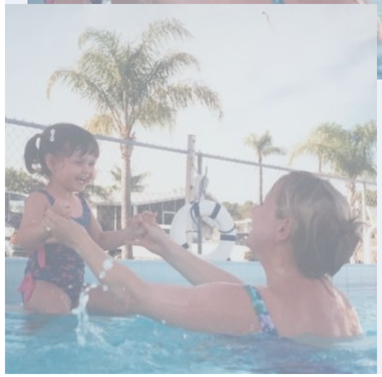
• Some topics I wish that we can cover:



4 weeks of Python



4 weeks of COMSOL



2 weeks of MD simulations

We will have “Digital Toolboxes”* sessions instead.

*cover anything from spread sheet to agentic AI.

4 weeks of physics engine

4 weeks of machine learning

4 weeks of data base and server

4 weeks of MATLAB and digital signals

Img source: web and https://www.britannica.com

- That’s how it NOT going to work (the worst case):

Your final project:

Disaster prevention:

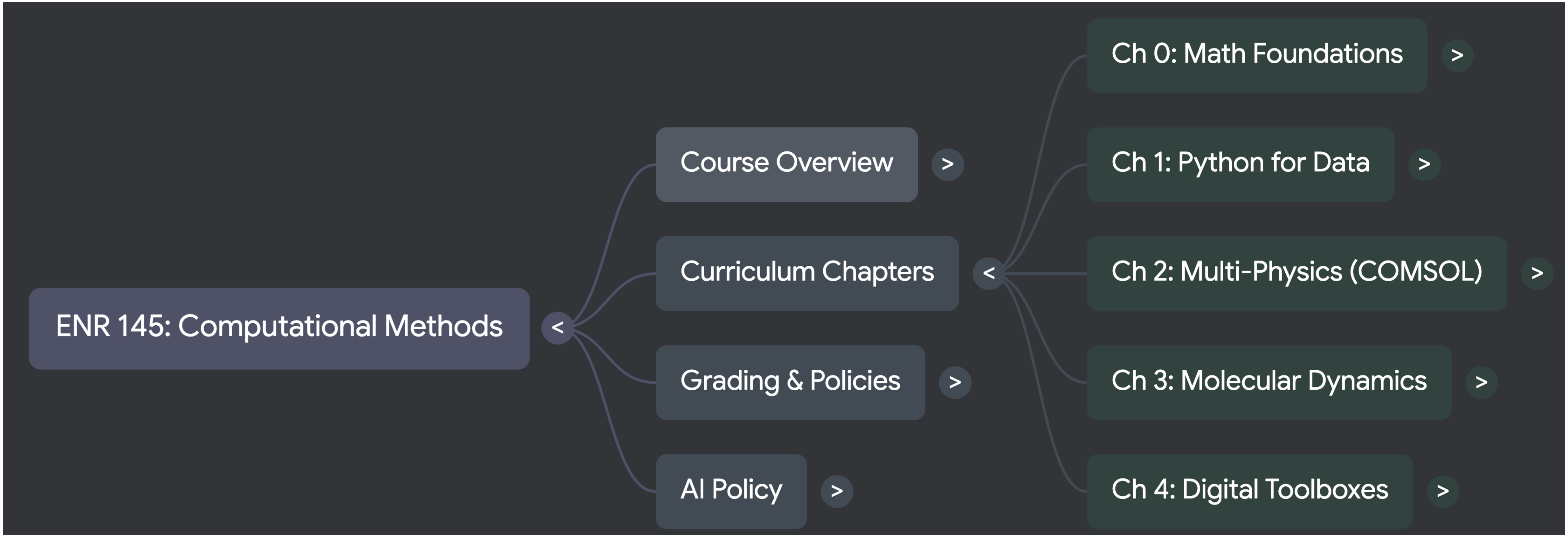
- Feedbacks. (not the last-minute ones)
- Study and practice.
- Work smarter, not harder.
- Find your own problem-solving style.
- And I am always here to help!
- We have quiz, tech-demo and check-points to keep track of the progress.

COE COLLEGE

Img source: web and https://www.britannica.com
Doctor Strange image source: fandango.com movie promotion news 2016

Syllabus and course review:

- Moodle page: https://coewinmoodle.coe.edu/moodle/course/view.php?id=4725
- Course content page: https://xlicoe.github.io/ENR145/



This “mindmap” was generated with google notebooklm based on ENR 145 syllabus.

Rule-of-thumbs:

- Don’t spend more than 15-30 min getting stuck on a road block. (It’s time to get up.)
- Study–group, AI tutor, notes taking service, testing center… getting all the helps you need.
- Self-learn is encouraged and expected.

- AI policy:
  - you understood and could explain any AI generated contents (that you submitted).
  - Don’t submit an AI generated idea as your own idea.
  - I cannot grade your efforts based on AI generated contents, but I can grade your efforts based on documentation, presentation, and the clear understanding of the AI tooling.

Assignments

- Upload at Moodle page: https://coewinmoodle.coe.edu/moodle/course/section.php?id=73749
- Submission templates: https://coewinmoodle.coe.edu/moodle/mod/resource/view.php?id=369927
- Sometimes I will ask proof-of-efforts and proof-of-success by screenshots. It will be very helpful if the screenshots are nicely formatted and part of the slides.
- Pre-labs are tasks for you to try and complete before the course contents. If most of us having trouble getting it done, we will cover it in class. Otherwise, it can save us time talking about the fun stuff.

Example of pre-labs tasks:

- Get Google Colab and Python IDE of your choice. (If no prior experience, PyCharm is recommended).
  Colab: https://colab.research.google.com
  PyCharm: https://www.jetbrains.com/pycharm/
- Github: register an Github account https://github.com/ and try to join https://github.com/Coe-College
- CAD tool

Autodesk inventor, Autodesk Fusion or any tool of your choice.

Make a 3D Tesla valve based on following references:

- https://www.researchgate.net/publication/329154004_A_numerical_investigation_of_the_flow_of_nanofluids_through_a_micro_Tesla_valve
- https://www.ikts.fraunhofer.de/en/departments/structural_ceramics/processes_and_components/additive_hybrid_manufacturing/cr_ceramic_tesla_valve.html
- https://fluidpowerjournal.com/teslas-conduit/

COE COLLEGE

No matter what your dream project will be, ENR145 is here to help!

Before that, we need to acquire some basic **numerical** skills:

0.1) Counting in binary and hexadecimal numbers.

0.2) Boolean operation and truth table reading.

0.3) Intro to encoding, decoding, and error correction.

0.4) Deliverable: hand calculation of Hamming (16,11) extended encoding and error correction.

0.1) Let's go numerical

- Say bye-bye to base 10:

Base 10 (0,1,2,3,4,5,6,7,8,9):

$$(4321)_{10} = 4\times 10^3 + 3\times 10^2 + 2\times 10^1 + 1\times 10^0$$

Base 2 (0,1):

$$(1011)_2 = 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0$$

Base 16 (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F):

$$(\mathrm{FF}12)_{16} = 15\times 16^3 + 15\times 16^2 + 1\times 16^1 + 2\times 16^0$$

Looking up how we do base conversions manually and in python. I have a feeling it will be a pop-quiz/tech-demo.

The calculation of base 2 are pretty boring compared to base 10

Base 10	Base 2	Base 10	Base 2
$\begin{array}{r} 324 \\ +123 \\ \hline \end{array}$	$\begin{array}{r} 110 \\ +101 \\ \hline \end{array}$	$\begin{array}{r} 324 \\ \times 123 \\ \hline \end{array}$	$\begin{array}{r} 110 \\ \times 101 \\ \hline \end{array}$
$\begin{array}{r} 324 \\ -123 \\ \hline \end{array}$	$\begin{array}{r} 110 \\ -101 \\ \hline \end{array}$	$\begin{array}{r} 324 \\ \div 6 \\ \hline \end{array}$	$\begin{array}{r} 110 \\ \div 10 \\ \hline \end{array}$

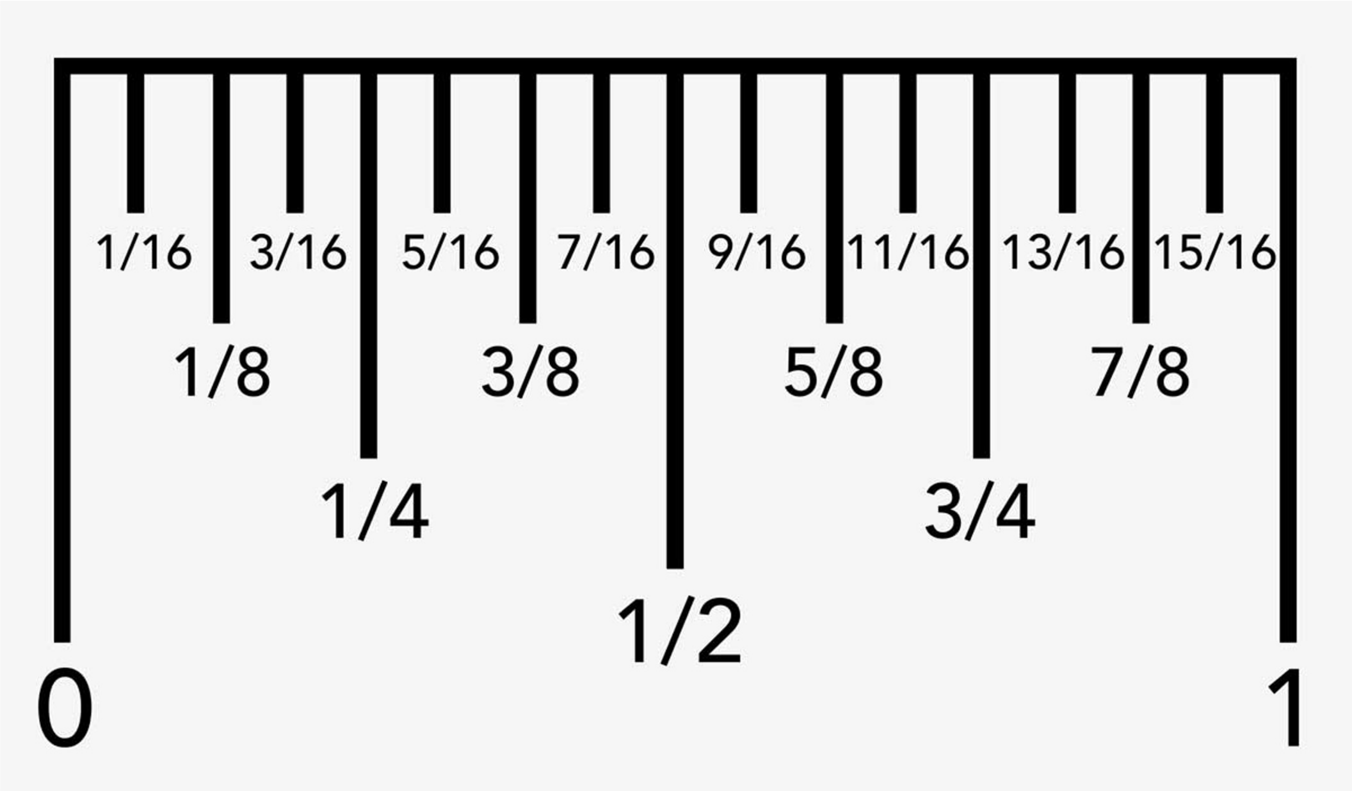
- Your base 10 arithmetic skills can be easily translated to base 2 ones.

Discuss: the origin of base 16?

Discuss: the origin of base 16?

My theory:
An easy and fair way to compute with a weightless balance scale.

https://commons.wikimedia.org/w/index.php?curid=79229218



https://www.inchcalculator.com/how-to-read-a-ruler/

Why CS loves Hex(decimal) coding

Prefix for binary

$0b:\underbrace{0011}_{3}\underbrace{1001}_{9}\underbrace{0010}_{2}\underbrace{1111}_{F}\underbrace{1010}_{A}$ **This is too long.**

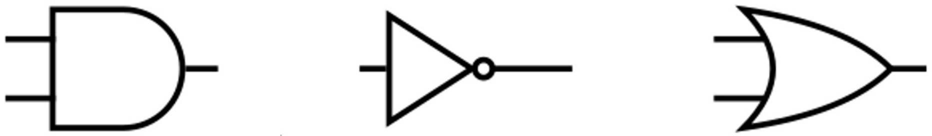
Prefix for hex

0x:392FA **This is not too long.**

0.2) Logic operation

- Boolean operations 1st gen: basic True (1) or False (0) algebra

AND NOT OR



AND gate truth table

Input		Output
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Inverter truth table

Input	Output
A	NOT A
0	1
1	0

OR gate truth table

Input		Output
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

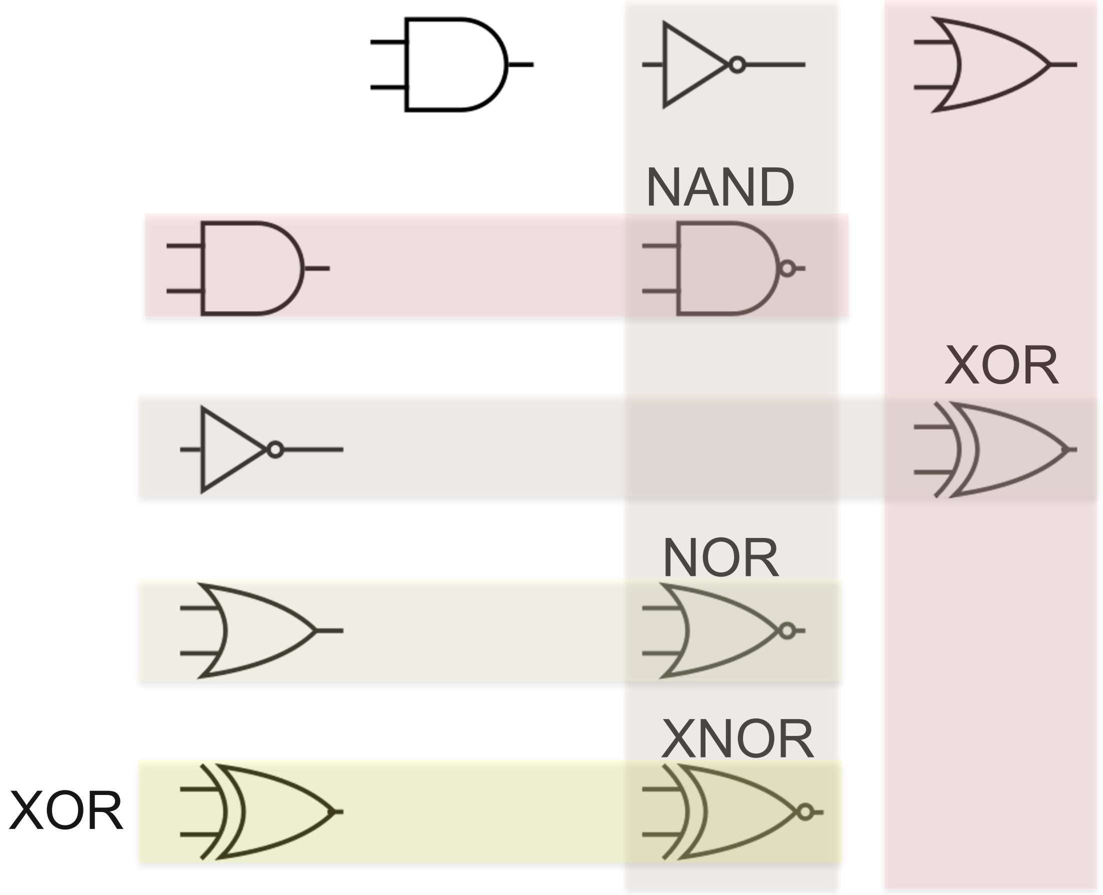
Boolean arithmetic symbols are messy

<table>
<tr><th></th><th>Viable symbols</th></tr>
<tr><td>AND</td><td>● * × & ∧</td></tr>
<tr><td>NOT</td><td>! - _ ‘ ¬</td></tr>
<tr><td>OR</td><td>+ | ∨</td></tr>
</table>

Truth table from: https://en.wikipedia.org/wiki/~_gate

Logic operation is like data manipulation

- Boolean operations 2nd gen:
  "logical logic operation"



"reverse AND"

NAND gate truth table

Input		Output
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

"reverse OR"

NOR gate truth table

Input		Output
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

"exclusive OR"

XOR gate truth table

Input		Output
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

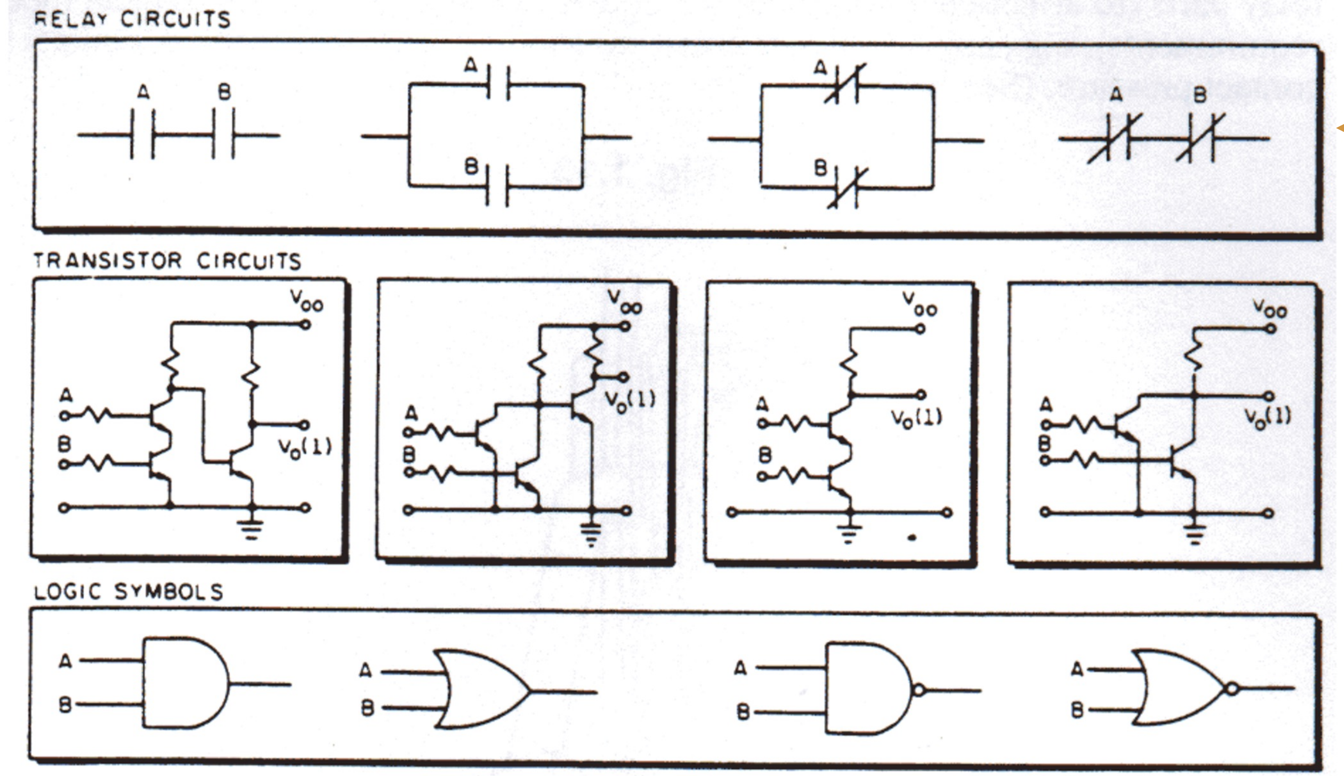
"reverse XOR"

XNOR gate truth table

Input		Output
A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

Truth table from: https://en.wikipedia.org/wiki/~_gate

Logic operation in the early days: a literal digital logic gate



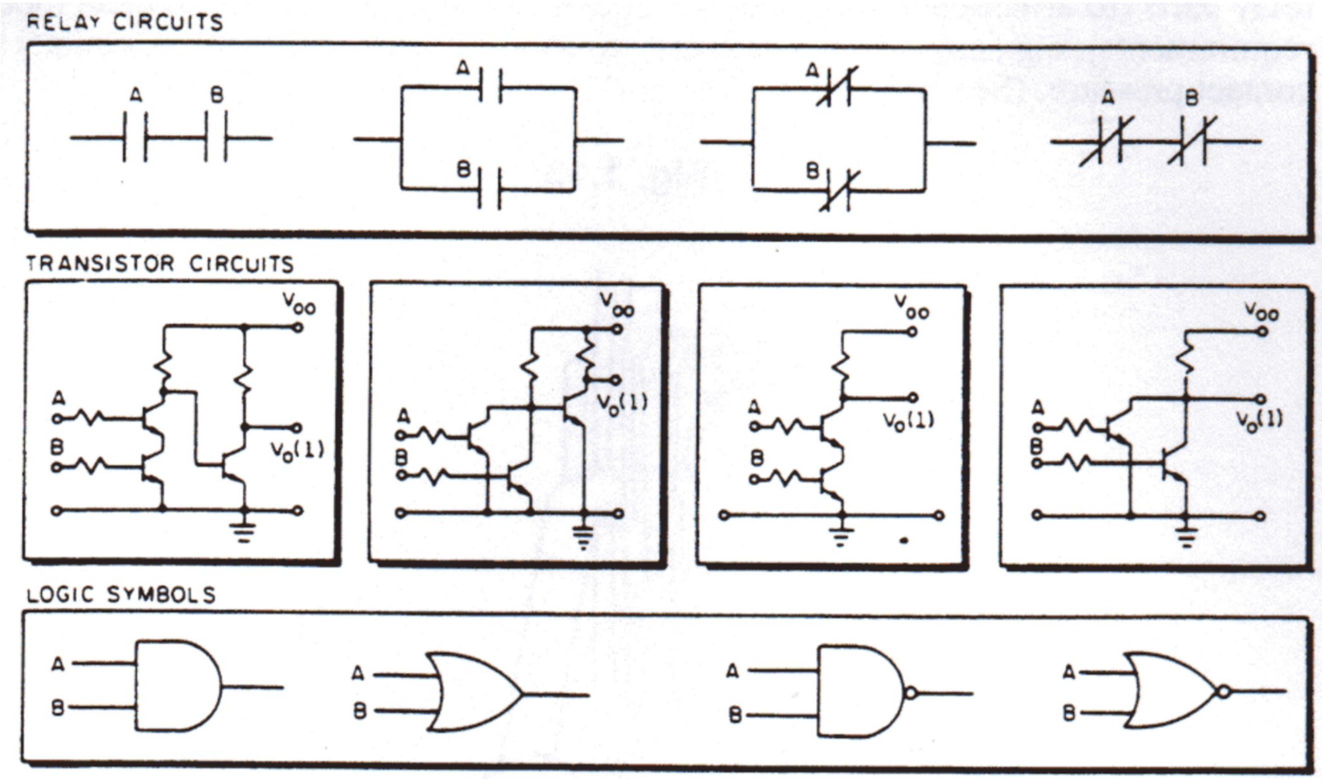
Engineer's Relay Handbook, 5th edition, Relay and Switch Industry Association (RSIA), 1966

Fig. 1.12 Symbols used in cam-operated timer control.

Guardian Electric Series 200 Relay

Available on ebay by seller sharkbate111

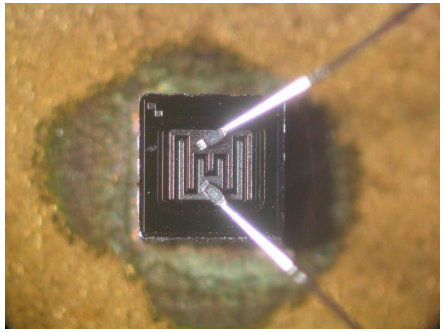
Logic operation in the early days: a literal digital logic gate



Engineer's Relay Handbook, 5th edition, Relay and Switch Industry Association (RSIA), 1966

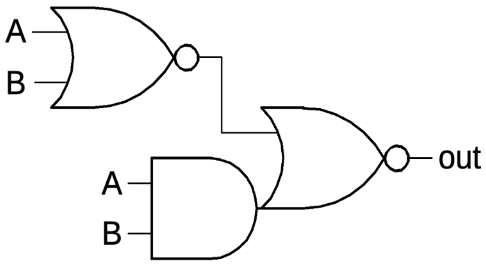
Fig. 1.12 Symbols used in cam-operated timer control.

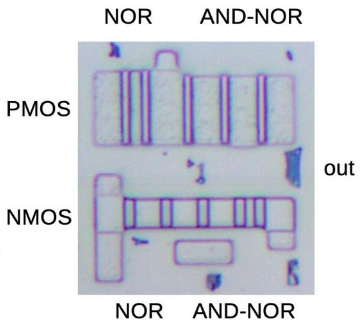
Transistor–transistor logic (TTL) built with BJT



https://electronics.stackexchange.com/

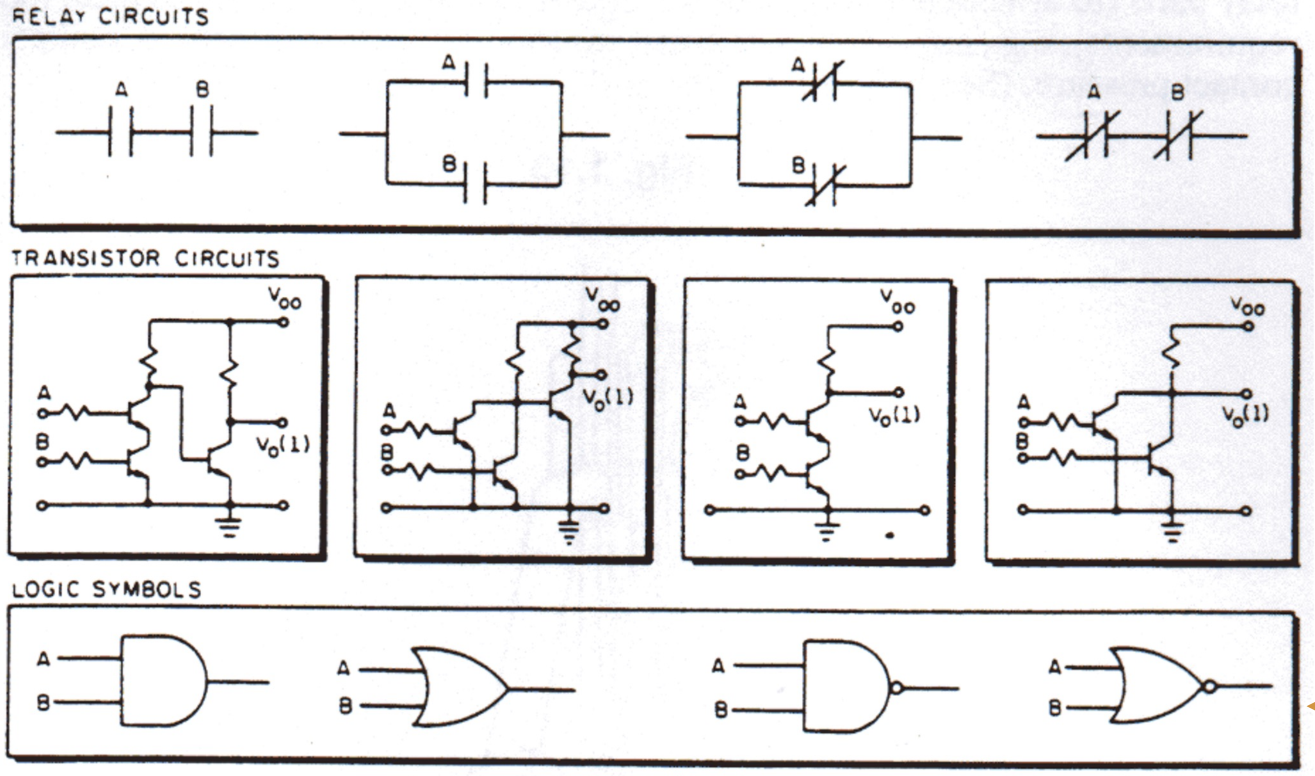
Other than power electronics, BJT was mostly replaced by CMOS:





https://www.righto.com/2023/12/386-xor-circuits.html

Logic operation in the early days: a literal digital logic gate



Engineer's Relay Handbook, 5th edition, Relay and Switch Industry Association (RSIA), 1966

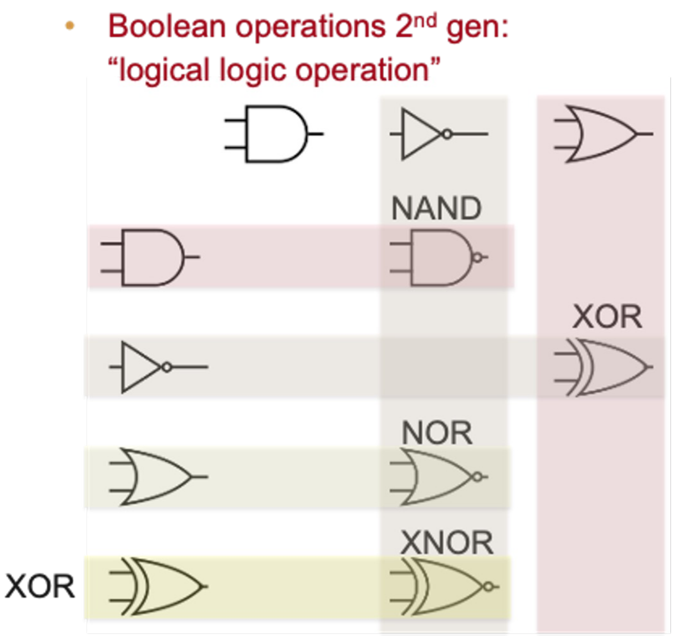
Fig. 1.12 Symbols used in cam-operated timer control.

The foundation of the digital and information age: https://www.youtube.com/watch?v=cNn34cIKFpo&t=2s

COE COLLEGE

“Universal Gate”

- We can use AND, OR, and NOT to build any gates:

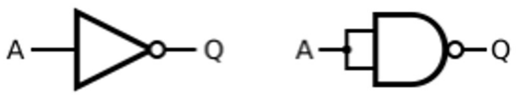


- We can build any gates with NAND gate:

“Universal Gate”

- We can build any gates with NAND gate:

Desired NOT Gate | **NAND Construction**

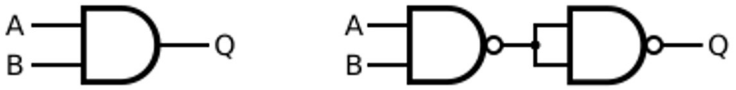


Q = NOT(**A**) | = **A** NAND **A**

Truth Table

Input A	Output Q
0	1
1	0

Desired AND Gate | **NAND Construction**

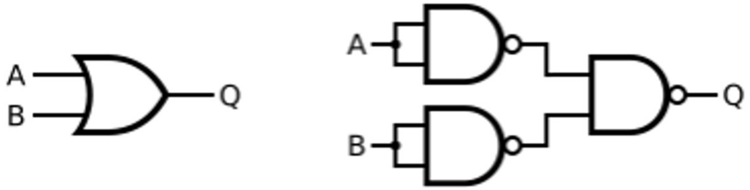


Q = **A** AND **B** | = (**A** NAND **B**) NAND (**A** NAND **B**)

Truth Table

Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1

Desired OR Gate | **NAND Construction**



Q = **A** OR **B** | = (**A** NAND **A**) NAND (**B** NAND **B**)

Truth Table

Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	1

Why NAND?

https://en.wikipedia.org/wiki/NAND_logic

COE COLLEGE

One can dig deeper into the Boolean operations:

Boolean expressions of the 16 functions between two variables.

Function Name	Function description	Boolean Expression
Null	FALSE (0)	0
AND	AND	$A \cdot B$
Inhibition	A NOT B	A/B
Transfer	A	A
Inhibition	B NOT A	B/A
Transfer	B	B
Exclusive-OR	XOR	$A \oplus B$
OR	OR	$A+B$
NOR	NOR	$A \downarrow B$
XNOR	XNOR	$A \odot B$
Complement	NOT B	B'
Implication	A OR NOT B	$A+B'$
Complement	NOT A	A'
Implication	NOT A OR B	$A'+B$
NAND	NAND	$A \uparrow B$
Identity	TRUE (1)	1

Fundamental Theorems & Postulates of Boolean Algebra

Identities:	(1)	$X + 0 = X$	(1D)	$X \cdot 1 = X$
Null Elements:	(2)	$X + 1 = 1$	(2D)	$X \cdot 0 = 0$
Indempotency:	(3)	$X + X = X$	(3D)	$X \cdot X = X$
Involution (Double Negation):	(4)			$(X')' = X$
Complements:	(5)	$X + X' = 1$	(5D)	$X \cdot X' = 0$
Commutativity:	(6)	$X + Y = Y + X$	(6D)	$X \cdot Y = Y \cdot X$
Associativity:	(7)	$(X+Y)+Z = X+(Y+Z)$	(7D)	$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$
Distributivity:	(8)	$X \cdot Y+X \cdot Z = X \cdot (Y+Z)$	(8D)	$(X+Y) \cdot (X+Z) = X+Y \cdot Z$
Combining:	(9)	$X \cdot Y + X \cdot Y' = X$	(9D)	$(X+Y) \cdot (X+Y') = X$
Covering:	(10)	$X + X \cdot Y = X$	(10D)	$X \cdot (X+Y) = X$
DeMorgan's Laws:	(12)	$(X \cdot Y \cdot Z)' = X'+Y'+Z'$	(12D)	$(X+Y+Z)' = X' \cdot Y' \cdot Z'$
Consensus:	(17)	$X \cdot Y+X' \cdot Z+Y \cdot Z = X \cdot Y+X' \cdot Z$	(17D)	$(X+Y) \cdot (X'+Z) \cdot (Y+Z) = (X+Y) \cdot (X'+Z)$

Shannon Expansion:

(18) $F(X,Y,Z) = X \cdot F(1,Y,Z) + X' \cdot F(0,Y,Z)$

(18D) $F(X,Y,Z) = (X+F(0,Y,Z)) \cdot (X'+F(1,Y,Z))$

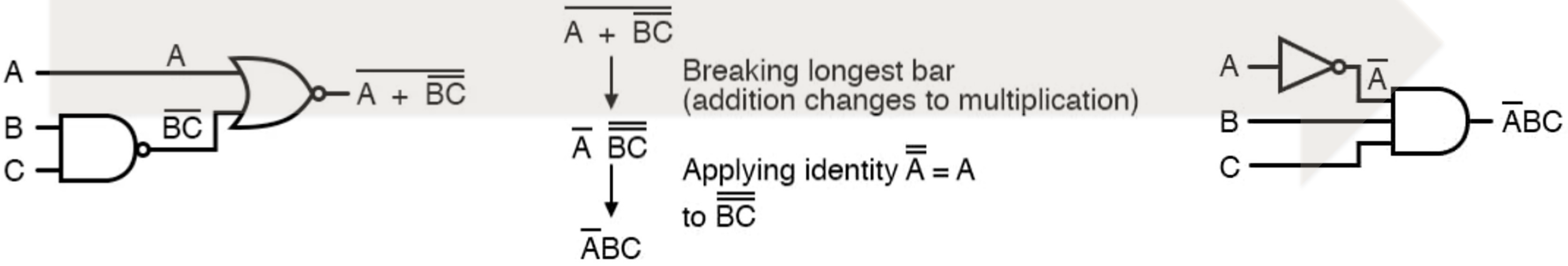
CS211, Rutgers 2013 notes

But we will only touch base on two: De Morgan

- De Morgan's laws (1/2)

not (A OR B) = (not A) AND (not B)
not (A AND B) = (not A) OR (not B)

$$\overline{(A \cdot B)} \equiv (\overline{A} + \overline{B})$$
$$\overline{(A + B)} \equiv (\overline{A} \cdot \overline{B})$$



Example from: https://www.allaboutcircuits.com/textbook/digital/chpt-7/demorgans-theorems/

De Morgan's laws is a way to **mathematically** simplified a circuit, but not always realistic for integrated circuits (IC).

COE COLLEGE

But we will only touch base on two: K-map

But we will only pouch base on two:

- Karnaugh map (K-map) (2/2)

NAND gate truth table		
Input		Output
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

K-map of NAND gate

B \ A	0	1
0	**1**	**1**
1	**1**	**0**

K-map grouping rules

1. No zeros allowed.
2. No diagonals.
3. Only power of 2 number of cells in each group.
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. Overlapping allowed.
7. Wrap around allowed.
8. Fewest number of groups possible.

Grouping rules: https://physics.umd.edu/~drew/spr07/Karnaugh%20Maps%20-%20Rules%20of%20Simplification.htm

K-map

- Karnaugh map (K-map) (2/2), continued

K-map grouping rules

1. No zeros allowed.
2. No diagonals.
3. Only power of 2 number of cells in each group.
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. Overlapping allowed.
7. Wrap around allowed.
8. Fewest number of groups possible.

(A **AND** B) **OR** (**NOT**C **AND** D)

(A×B)+(!C ×D)

	CD \ AB	00	01	11	10
!C×!D	00	**0**	**0**	**1**	**0**
!C×D	01	**1**	**1**	**1**	**1**
C×D	11	**0**	**0**	**1**	**0**
C×!D	10	**0**	**0**	**1**	**0**

K-map

- Karnaugh map (K-map) (2/2), in class practice

K-map grouping rules

1. No zeros allowed.
2. No diagonals.
3. **Only power of 2 number of cells in each group.**
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. Overlapping allowed.
7. **Wrap around allowed.**
8. Fewest number of groups possible.

CD \ AB	00	01	11	10
00	**0**	**0**	**1**	**1**
01	**1**	**1**	**0**	**0**
11	**0**	**0**	**0**	**0**
10	**0**	**0**	**1**	**1**

K-map

- Karnaugh map (K-map) (2/2), in class practice

K-map grouping rules

1. No zeros allowed.
2. No diagonals.
3. **Only power of 2 number of cells in each group.**
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. Overlapping allowed.
7. **Wrap around allowed.**
8. Fewest number of groups possible.

(!A × !C × D)+(A × !D)

AB / CD	00	01	11	10
00	**0**	**0**	**1**	**1**
01	**1**	**1**	**0**	**0**
11	**0**	**0**	**0**	**0**
10	**0**	**0**	**1**	**1**

Coe College

K-map

- Karnaugh map (K-map) (2/2), 1 more in class practice

K-map grouping rules

1. No zeros allowed.
2. No diagonals.
3. **Only power of 2 number of cells in each group.**
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. **Overlapping allowed.**
7. Wrap around allowed.
8. Fewest number of groups possible.

A \ BC	00	01	11	10
0	**0**	**1**	**1**	**1**
1	**1**	**1**	**1**	**0**

Coe College

K-map

- Karnaugh map (K-map) (2/2), 1 more in class practice

K-map grouping rules

1. No zeros allowed.
2. No diagonals.
3. **Only power of 2 number of cells in each group.**
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. **Overlapping allowed.**
7. Wrap around allowed.
8. Fewest number of groups possible.

$(A \times !B) + (!A \times B) + C$

A \ BC	00	01	11	10
0	**0**	**1**	**1**	**1**
1	**1**	**1**	**1**	**0**

Better/worse than K-map to get Boolean equation out of truth table

- Other than K-map, the more mechanical way to turn truth map to Boolean equations:

Step 1

A	B	C	
0	0	**1**	$C_0 = !A \times !B$
0	1	**1**	$C_1 = !A \times B$
1	0	**1**	$C_2 = A \times !B$
1	1	**0**	

Step 2

$$C = C_0 + C_1 + C_2$$

Step 3

$C = !A \times !B + !A \times B + A \times !B$

$= !A(!B + B) + A \times !B$ Complements: (5) $X + X' = 1$

$= !A + A \times !B$

$= !A \times (1 + !B) + A \times !B$ Null Elements: (2) $X + 1 = 1$

$= !A + !A\,!B + A\ !B$

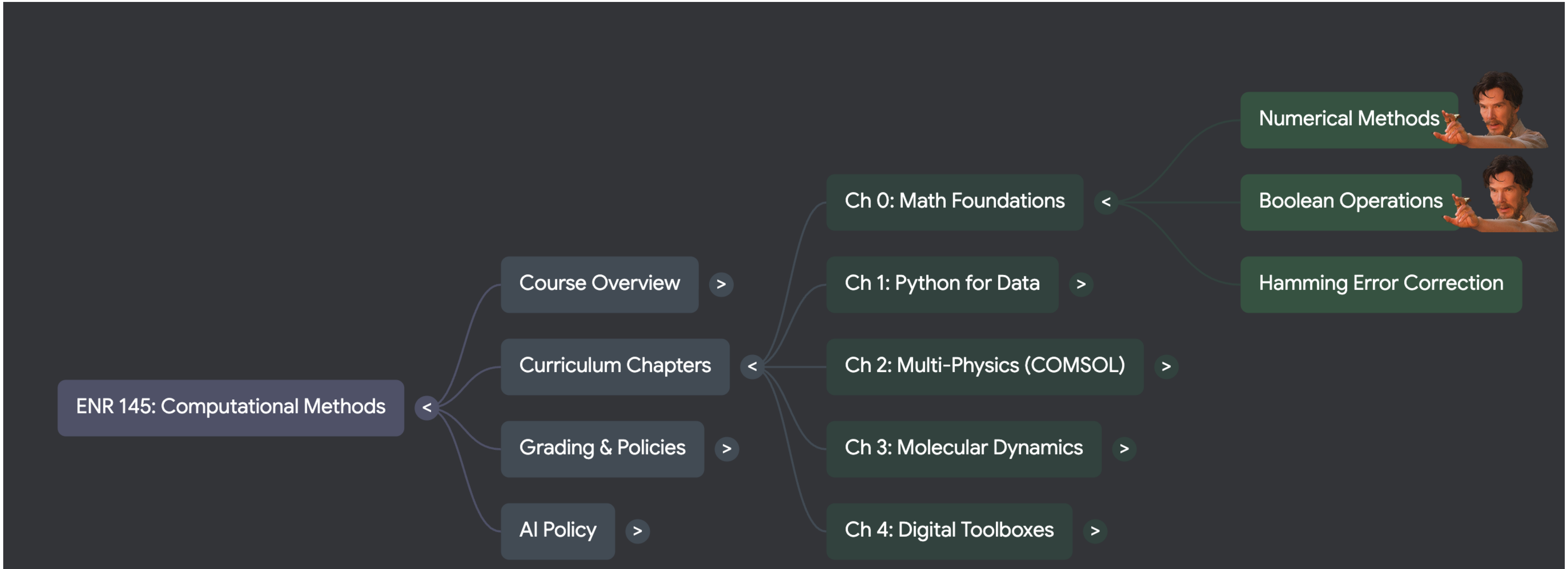
$= !A + (!A + A) \times !B$

$= !A + !B$

- This expression is also called **Sum of Products (SOP)**.
- There are also **Product of Sums**, for sure.

COE COLLEGE

Course recap:



This “mindmap” was generated with google notebooklm based on ENR 145 syllabus.

Doctor Strange image source: fandango.com movie promotion news 2016