Logic and Computation

Exploring Boolean operations, Python examples, and mathematical notation.

Table of Contents

Introduction

This presentation demonstrates slides with:

- Dynamic **backgrounds**
- Highlighted **code blocks**
- **Mathematical notation**
- Logical **operations and truth tables**

Everything is written in Markdown and styled with a custom Marp theme.

Long Text Section 📝

In the field of computer science, **logic and computation** form the foundation upon which modern systems are built.
Boolean logic, introduced by George Boole in the 19th century, established a mathematical framework for reasoning about truth and falsehood.
This binary perspective — where every statement is either *true* or *false* — has since become the basis of digital electronics, programming languages, and algorithmic design.

Computers operate on a series of logical decisions that are fundamentally Boolean in nature.
Every conditional statement, from a simple `if` clause in Python to the control logic of a microprocessor, relies on evaluating truth values.
Over time, this simple idea has evolved into more complex logical systems such as **fuzzy logic**, **modal logic**, and **temporal logic**, each extending the expressive power of reasoning in different contexts.

In practical programming, logic manifests not only through conditionals but also through **control structures**, **error handling**, and **decision trees**.
A developer often writes logic that determines how data flows through a program, how it reacts to unexpected input, or how it makes optimized decisions based on real-time conditions.
Even in artificial intelligence, Boolean foundations persist beneath the layers of neural networks — determining activation, propagation, and optimization through mathematical logic.

Boolean Operations ⚙

Operation	Symbol	Example	Result
AND	∧ / `and`	`1 and 0`	`0`
OR	∨ / `or`	`1 or 0`	`1`
XOR	⊕ / `^`	`1 ^ 1`	`0`
NOT	¬ / `not`	`not 1`	`0`

```python
a, b = True, False

print(a and b)  # AND
print(a or b)   # OR
print(a ^ b)    # XOR
print(not a)    # NOT
```

Python Logic Example 💻

```python
def logic_ops(a: bool, b: bool) → dict:
    return {
        "AND": a and b,
        "OR": a or b,
        "XOR": a ^ b,
        "NAND": not (a and b),
    }

result = logic_ops(True, False)

for op, value in result.items():
    print(f"{op}: {value}")
```

This example shows a simple function returning multiple logical evaluations.

Mathematical Expressions 🧮

Basic quadratic equation:

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

Logical equivalences:

$$A \wedge B = B \wedge A \; A \vee (B \vee C) \quad = (A \vee B) \vee C \; \neg(A \wedge B) = \neg A \vee \neg B$$

Summary

- Boolean logic is fundamental to computing.
- Python provides intuitive operators for it.
- Math and logic can be elegantly combined in Markdown.
- Custom themes enhance clarity and style.

Thank You!