Na **Seção 2 da Unidade 1 – Evolução da Lógica**, a gente acompanha como a lógica foi se transformando ao longo do tempo e como isso se conecta direto com programação e algoritmos.

A lógica é basicamente o estudo do raciocínio e dos princípios que definem se um argumento é válido ou não. Ela evoluiu em **três grandes períodos**:

Período Aristotélico (390 a.C. – 1840 d.C.)
Aristóteles criou a ideia do **silogismo** (se as premissas são verdadeiras, a conclusão também será). Essa fase ficou conhecida como **Lógica Clássica**, com três princípios fundamentais: Identidade (A = A), Não Contradição (algo não pode ser verdadeiro e falso ao mesmo tempo) e Terceiro Excluído (ou é verdadeiro ou é falso, sem meio-termo).

Período Booleano (1840 – 1910)
Aqui surge a **Lógica Simbólica/Formal**. O destaque é **George Boole**, criador da **Álgebra Booleana**, onde tudo se resolve em 0 (falso) e 1 (verdadeiro). Esse modelo é a base da computação, porque conecta operações lógicas com circuitos e programação. Também entram em cena **Cantor**, com a Teoria dos Conjuntos, e **Frege**, que trouxe o cálculo proposicional e de predicados, além de símbolos e quantificadores.

Período Atual (a partir de 1910)
Ganha força com Russell e Whitehead e abre espaço para as **lógicas não clássicas**: paracompletas, paraconsistentes, modais e principalmente a **lógica fuzzy**, que trabalha com valores entre 0 e 1 (ótima para IA e sistemas que lidam com incerteza).

No fim, a seção mostra como a **álgebra booleana** é aplicada na prática, tipo em circuitos digitais ou num sistema de votação, usando **tabelas-verdade** para mapear condições lógicas.

O código gerado foi:

```
# Simulação do Circuito de Votação do Comitê
# Baseado na Unidade 1, Seção 2, "Sem medo de errar" do material
"Lógica Computacional" [1-3]

print("--- Simulação de Votação do Comitê Diretor ---")
print("O projeto será aprovado se o Diretor Executivo votar 'Sim' E
houver maioria.")
print("Para 'Sim' digite 1, para 'Não' digite 0.")

# Entradas dos votos dos membros do comitê
# Representamos 1 como 'Voto a Favor' e 0 como 'Voto Contra',
```

```
# análogo ao estado 'fechado' (1) e 'aberto' (0) do interruptor em um
circuito [4]
try:
    voto_diretor_executivo = int(input("Voto do Diretor Executivo (A):
"))
    voto_vice_diretor_financeiro = int(input("Voto do Vice-Diretor
Financeiro (B): "))
    voto_vice_diretor_relacoes = int(input("Voto do Vice-Diretor de
Relações Institucionais (C): "))

    # Validação das entradas para garantir que são 0 ou 1
    # Esta validação, embora não explicitamente detalhada nas fontes, é
uma boa prática de programação.
    if not all(v in {0, 1} for v in [voto_diretor_executivo,
voto_vice_diretor_financeiro, voto_vice_diretor_relacoes]):
        print("\nErro: Por favor, digite apenas 0 ou 1 para os votos.")
    else:
        # Lógica de aprovação do projeto conforme as regras:
        # "o diretor executivo votar a favor e obtiver maioria." [5, 6]
        # Para 3 membros (A, B, C), se A já votou a favor, a maioria
significa (B OU C) [7].
        # A expressão lógica é, portanto, A AND (B OR C) [7].
        # O Python interpreta 1 como True e 0 como False em contextos
booleanos [8].
        projeto_aprovado = bool(voto_diretor_executivo) and
(bool(voto_vice_diretor_financeiro) or
bool(voto_vice_diretor_relacoes))

        # Exibição do resultado
        print("\n--- Resultado da Votação ---")
        # Uma luz se acenderá caso o projeto seja aprovado, e
permanecerá apagada caso contrário [4, 5].
        if projeto_aprovado:
            print("Luz de aprovação: ACESA (Projeto APROVADO!)")
        else:
            print("Luz de aprovação: APAGADA (Projeto NÃO APROVADO.)")

        # Demonstração da Tabela Verdade para a expressão A AND (B OR
C) [9]
        # A Tabela Verdade é um "método exaustivo de geração de
valorações para uma dada fórmula" [10].
        print("\n--- Tabela Verdade para Aprovação (A AND (B OR C))
---")
```

```
        print("A | B | C | Aprovação")
        print("--------------------")
        # Itera sobre todas as 8 combinações possíveis de 0s e 1s para
A, B, C (2^n combinações, onde n é o número de proposições) [11]
        for a in [12]:
            for b in [12]:
                for c in [12]:
                    # Aplica a mesma lógica booleana para cada
combinação da tabela verdade
                    resultado_combinacao = bool(a) and (bool(b) or
bool(c))
                    # Exibe os valores 0 ou 1 e o resultado da
aprovação (1 para aprovado, 0 para reprovado)
                    print(f"{a} | {b} | {c} | {'1 (APROVADO)' if
resultado_combinacao else '0 (REPROVADO)'}")
        print("--------------------")

except ValueError:
    # Este bloco `except ValueError` trata o erro caso o usuário não
digite um número inteiro.
    # Esta é uma estrutura de tratamento de exceções em Python, que não
é diretamente abordada nas fontes,
    # mas é fundamental para robustez de software.
    print("\nErro: Entrada inválida. Por favor, digite um número (0 ou
1).")
```

Apresentando o funcionamento:

```
--- Simulação de Votação do Comitê Diretor ---
O projeto será aprovado se o Diretor Executivo votar 'Sim' E houver
maioria.
Para 'Sim' digite 1, para 'Não' digite 0.
Voto do Diretor Executivo (A): 1
Voto do Vice-Diretor Financeiro (B): 0
Voto do Vice-Diretor de Relações Institucionais (C): 0

--- Resultado da Votação ---
Luz de aprovação: APAGADA (Projeto NÃO APROVADO.)

--- Tabela Verdade para Aprovação (A AND (B OR C)) ---
A | B | C | Aprovação
--------------------
12 | 12 | 12 | 1 (APROVADO)
--------------------
```