```
# Simulação do Circuito de Votação do Comitê

# Baseado na Unidade 1, Seção 2, "Sem medo de errar" do material "Lógica Computacional" [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.")

try:

    # 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 [5]
    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." [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 [7].
        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 [7].
        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) [8]
        # A Tabela Verdade é um "método exaustivo de geração de valorações para uma dada
fórmula" [8].
        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) [8]
        for a in [9]:
            for b in [9]:
                for c in [9]:
                    # 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).")
```

Explicação do Código e Conexão com os Conceitos:
Este código demonstra os princípios da **Lógica Simbólica/Formal** e da **Álgebra Booleana**, que surgiram no Período Booleano (1840-1910) e são a base da computação.

- **Representação de Entradas**: Os votos dos membros do comitê (Diretor Executivo, Vice-Diretor Financeiro, Vice-Diretor de Relações Institucionais) são representados por 1 (a favor) ou 0 (contra). Essa representação é análoga aos estados "fechado" (1) e "aberto" (0) de um interruptor em um circuito digital, um conceito fundamental na eletrônica digital e na Álgebra Booleana.
- **Lógica de Aprovação**: A condição para aprovação do projeto é que "o diretor executivo votar a favor e obtiver maioria". Para um comitê de três membros (A, B, C), se o Diretor Executivo (A) vota a favor, a maioria é alcançada se o Vice-Diretor Financeiro (B) ou o Vice-Diretor de Relações Institucionais (C) também votarem a favor. Isso se traduz na expressão lógica **A AND (B OR C)**.
- **Álgebra Booleana em Python**: O Python interpreta os valores inteiros 1 como `True` e 0 como `False` em contextos booleanos, permitindo a aplicação direta das operações lógicas `and` (conjunção) e `or` (disjunção).
- **Tabela-Verdade**: O código inclui a geração e exibição da tabela-verdade para a expressão lógica `A AND (B OR C)`. A tabela-verdade é um "método exaustivo de geração de valorações para uma dada fórmula", testando todas as combinações possíveis de entradas (2^n combinações, onde n é o número de proposições). Isso permite visualizar claramente todas as situações em que o projeto seria aprovado ou não.

Embora este exemplo seja da Seção 2, ele ilustra como os princípios matemáticos e lógicos discretos são aplicados na prática da computação, que é o foco geral da disciplina de Lógica Computacional

https://colab.research.google.com/notebooks/intro.ipynb#scrollTo=9qz04XZElAGV&line=60&uniqifier=1

Entendimento

A Matemática Discreta é a base da computação, lidando com informações finitas e separadas. Ela nos ajuda a resolver problemas de existência, contagem e otimização, usando a **Combinatória** para gerenciar recursos limitados. A **lista** é uma sequência ordenada, e o **Princípio da Multiplicação** e o **Fatorial** nos ajudam a contar as possibilidades. Conceitos como **Arranjo** (ordem importa) e **Combinação** (ordem não importa) são essenciais para organizar dados. Entender esses princípios é crucial para o raciocínio lógico e o desenvolvimento de software eficiente.