Questao 2

Codigo Fonte:

```
import itertools
import io
import os
from contextlib import redirect_stdout
from fpdf import FPDF
from fpdf.enums import XPos, YPos # Importar para a nova sintaxe

# --- Definição das Expressões Lógicas ---

def evaluate_E1(p, q, r, s):
    """
    Avalia a expressão E1 = (s -> (p -> r)) ^ (~s -> (p -> r v q)) ^ s
    Lembrando que (A -> B) é equivalente a (~A v B)
    """
    # Termo 1: s -> (p -> r)  =>  (~s) v (~p v r)
    term1 = (not s) or ((not p) or r)

    # Termo 2: ~s -> (p -> (r v q))  =>  (~(~s)) v (~p v (r v q))  =>  s v (~p v r v q)
    term2 = s or ((not p) or r or q)

    # Termo 3: s
    term3 = s

    return term1 and term2 and term3

def evaluate_E2(p, q, r, s):
    """
    Avalia a expressão E2 = (p ^ q ^ ~r ^ s) v ~(p v s)
    """
    # Termo 1: (p ^ q ^ ~r ^ s)
    term1 = p and q and (not r) and s

    # Termo 2: ~(p v s)
    term2 = not (p or s)

    return term1 or term2

def solve_and_capture_output():
    """
    Constrói a tabela verdade, compara os resultados e imprime a análise.
    O output é capturado para ser usado no PDF.
    """
    # CORREÇÃO: Usando caracteres ASCII para as fórmulas
    print("Questao 2: Verificacao de Equivalencia Logica")
    print("E1 = (s -> (p -> r)) ^ (~s -> (p -> r v q)) ^ s")
    print("E2 = (p ^ q ^ ~r ^ s) v ~(p v s)")
    print("-" * 60)
```

```
    print("\nConstruindo a Tabela Verdade:\n")

    # Cabeçalho da tabela
    header = f"{'p':<5} | {'q':<5} | {'r':<5} | {'s':<5} || {'E1':<7} | {'E2':<7} | {'Equivalentes?':<15}"
    print(header)
    print("=" * len(header))

    are_equivalent = True

    # Define os valores possíveis para cada variável
    possible_values = [True, False]
    # Gera todas as combinações (produtos cartesianos) desses valores para 4 variáveis
    truth_combinations = list(itertools.product(possible_values, repeat=4))

    for combo in truth_combinations:
        p, q, r, s = combo

        res_e1 = evaluate_E1(p, q, r, s)
        res_e2 = evaluate_E2(p, q, r, s)

        match = (res_e1 == res_e2)
        if not match:
            are_equivalent = False

        # Imprime a linha da tabela
        row_str = (f"{str(p):<5} | {str(q):<5} | {str(r):<5} | {str(s):<5} || "
                   f"{str(res_e1):<7} | {str(res_e2):<7} | {('Sim' if match else 'NAO <-'):<15}")
        print(row_str)

    # Conclusão final
    print("\n" + "=" * len(header))
    print("\nConclusao:")
    if are_equivalent:
        print("As expressoes E1 e E2 SAO logicamente equivalentes.")
    else:
        print("As expressoes E1 e E2 NAO SAO logicamente equivalentes.")

def generate_pdf_report(code_content, output_content):
    """Gera um PDF com o código e o output."""
    pdf = FPDF()
    pdf.add_page()

    # Título Principal
    pdf.set_font("Courier", 'B', 16)
    # CORREÇÃO: Usando a nova sintaxe recomendada em vez de ln=True
    pdf.cell(0, 10, 'Questao 2', new_x=XPos.LMARGIN, new_y=YPos.NEXT, align='C')
    pdf.ln(10)

    # Seção do Código Fonte
    pdf.set_font("Courier", 'B', 12)
    pdf.cell(0, 10, 'Codigo Fonte:', new_x=XPos.LMARGIN, new_y=YPos.NEXT)
    pdf.set_font("Courier", '', 8)
    pdf.multi_cell(0, 5, code_content)
```

```
    pdf.add_page()

    # Seção do Output (Tabela Verdade e Conclusão)
    pdf.set_font("Courier", 'B', 12)
    pdf.cell(0, 10, 'Output da Execucao:', new_x=XPos.LMARGIN, new_y=YPos.NEXT)
    pdf.set_font("Courier", '', 10)
    pdf.multi_cell(0, 5, output_content)

    pdf_file_name = "resultado_questao_2.pdf"
    pdf.output(pdf_file_name)
    return pdf_file_name

# --- Bloco Principal de Execução ---
if __name__ == "__main__":
    output_buffer = io.StringIO()
    with redirect_stdout(output_buffer):
        solve_and_capture_output()
    output_content = output_buffer.getvalue()

    try:
        # CORREÇÃO: Lê o próprio arquivo e substitui os caracteres especiais para evitar erro no PDF
        with open(__file__, 'r', encoding='utf-8') as f:
            code_content = f.read()
    except Exception as e:
        code_content = f"Nao foi possivel ler o arquivo do codigo: {e}"

    try:
        pdf_file = generate_pdf_report(code_content, output_content)
        print(f"\nPDF '{pdf_file}' gerado com sucesso no diretorio atual!")
    except Exception as e:
        print(f"\nOcorreu um erro ao gerar o PDF: {e}")
```

Output da Execucao:

```
Questao 2: Verificacao de Equivalencia Logica
E1 = (s -> (p -> r)) ^ (~s -> (p -> r v q)) ^ s
E2 = (p ^ q ^ ~r ^ s) v ~(p v s)
------------------------------------------------------------

Construindo a Tabela Verdade:

p     | q     | r     | s     || E1      | E2      | Equivalentes?
=====================================================================
True  | True  | True  | True  || True    | False   | NAO <-
True  | True  | True  | False || False   | False   | Sim
True  | True  | False | True  || False   | True    | NAO <-
True  | True  | False | False || False   | False   | Sim
True  | False | True  | True  || True    | False   | NAO <-
True  | False | True  | False || False   | False   | Sim
True  | False | False | True  || False   | False   | Sim
True  | False | False | False || False   | False   | Sim
False | True  | True  | True  || True    | False   | NAO <-
False | True  | True  | False || False   | True    | NAO <-
False | True  | False | True  || True    | False   | NAO <-
False | True  | False | False || False   | True    | NAO <-
False | False | True  | True  || True    | False   | NAO <-
False | False | True  | False || False   | True    | NAO <-
False | False | False | True  || True    | False   | NAO <-
False | False | False | False || False   | True    | NAO <-

=====================================================================

Conclusao:
As expressoes E1 e E2 NAO SAO logicamente equivalentes.
```