```
# Importa a biblioteca 'itertools', que possui ferramentas eficientes
# para criar combinações e outras estruturas iteráveis.
# A informação sobre a biblioteca 'itertools' não está nas fontes;
# é um conhecimento padrão da linguagem de programação Python.
import itertools

# --- Passo 1: Definição do Conjunto Original ---
# De acordo com as fontes, um conjunto é uma coleção não ordenada de objetos.
# Em Python, podemos representar um conjunto usando chaves {}.
# Vamos usar o conjunto do seu desafio na Unidade 2, Seção 1.
conjunto_original = {1, 2, 3, 4}

print(f"Conjunto Original (A): {conjunto_original}")
print("-" * 30)

# --- Passo 2: Cálculo da Cardinalidade e do Número de Subconjuntos ---
# Cardinalidade é o número de elementos de um conjunto.
cardinalidade = len(conjunto_original)
print(f"A cardinalidade do conjunto |A| é: {cardinalidade}")

# A teoria afirma que o número total de subconjuntos é 2 elevado à cardinalidade.
num_subconjuntos_teorico = 2**cardinalidade
print(f"Número teórico de subconjuntos (2^|A|): {num_subconjuntos_teorico}")
print("-" * 30)


# --- Passo 3: Geração e Listagem de Todos os Subconjuntos ---
# Vamos gerar e listar todos os subconjuntos, desde os de cardinalidade 0 (conjunto vazio)
# até os de cardinalidade 4 (o próprio conjunto).

print("Listando todos os subconjuntos possíveis:")

todos_subconjuntos = []

# O laço 'for' a seguir irá iterar para criar subconjuntos de todos os tamanhos possíveis,
# de 0 até a cardinalidade do conjunto original.
for tamanho in range(cardinalidade + 1):
    # A função 'combinations' da biblioteca 'itertools' gera todas as combinações
    # possíveis de um determinado tamanho. Isso corresponde a encontrar
    # todos os subconjuntos de uma cardinalidade específica.
    subconjuntos_de_tamanho_n = itertools.combinations(conjunto_original, tamanho)

    # Adiciona os subconjuntos encontrados à nossa lista principal
    todos_subconjuntos.extend(subconjuntos_de_tamanho_n)
```

```
# Imprime cada subconjunto encontrado de forma organizada.
# Note que a saída da função 'combinations' é uma tupla, mas representa
# o conceito de subconjunto.
for subconjunto in todos_subconjuntos:
    # O conjunto vazio é representado por {} ou ∅ na teoria.
    # Aqui, ele será uma tupla vazia ().
    print(f"Subconjunto: {subconjunto if subconjunto else '{}'}")

print("-" * 30)
print(f"Total de subconjuntos gerados: {len(todos_subconjuntos)}")

# Verificação final para confirmar que o número gerado bate com o teórico.
if len(todos_subconjuntos) == num_subconjuntos_teorico:
    print("\nSucesso! O número de subconjuntos gerados corresponde ao valor teórico de
2^|A|.")
else:
    print("\nAlgo deu errado. A contagem não bateu com o valor teórico.")
```