Testar Se É Num Primo

Para construir uma rotina para testar números primos em um programa, podemos seguir uma abordagem simples de verificação. Aqui está um exemplo básico em pseudocódigo:

python

Copiar código

função é\_primo(n):

se n <= 1:

retornar Falso # números menores ou iguais a 1 não são primos

para cada i de 2 até a raiz quadrada de n:

se n % i == 0:

retornar Falso # encontrou um divisor, portanto não é primo

retornar Verdadeiro # se nenhum divisor foi encontrado, é primo

# Exemplo de uso:

para cada número n de 1 a 20:

se é\_primo(n):

imprimir(n, "é primo")

senão:

imprimir(n, "não é primo")

**Explicação:**

1. **Função é\_primo(n)**: Esta função recebe um número n e verifica se é primo ou não.
   * Primeiro, verifica se n é menor ou igual a 1. Números menores ou iguais a 1 não são primos.
   * Em seguida, itera de 2 até a raiz quadrada de n. Para cada número i durante essa iteração:
     + Verifica se n é divisível por i (ou seja, n % i == 0). Se for, n não é primo porque tem um divisor diferente de 1 e ele mesmo.
   * Se nenhum divisor é encontrado durante o loop, n é primo.
2. **Exemplo de uso**: Testa todos os números de 1 a 20 e imprime se são primos ou não.

Esta é uma abordagem básica e eficiente para verificar se um número é primo. Em linguagens de programação como Python, essa lógica pode ser implementada diretamente com ajustes para lidar com números maiores e otimizar ainda mais o desempenho se necessário.