Title Page

Problem Statement :

Prime number generator and checker

Name: Yogesh

Roll No.: 202401100300288

Course: INTRODUCTION TO AI

Institution: KIET

Introduction

A **prime number** is a natural number greater than 1 that has only two divisors: **1 and itself**. Prime numbers play a crucial role in mathematics, computer science, cryptography, and artificial intelligence. Identifying prime numbers efficiently is essential for various computational applications, including encryption algorithms and data security.

This project, **Prime Number Generator and Checker**, is designed to:

1. **Check whether a given number is prime** by verifying its divisibility properties.
2. **Generate a list of prime numbers up to a user-specified limit** using efficient algorithms.

To achieve these functionalities, the project implements optimized techniques such as **Trial Division** and **Sieve of Eratosthenes** to ensure accuracy and performance. The program is designed to handle different edge cases, such as negative numbers, zero, and non-prime numbers, while maintaining a user-friendly approach.

By developing this tool, we explore fundamental mathematical concepts and their applications in computational problem-solving. The project serves as a **foundation for advanced topics like AI-based number theory analysis and cryptographic security systems**.

Methodology

The Prime Number Generator and Checker is developed using a structured approach to ensure accuracy and efficiency. The methodology involves problem analysis, algorithm selection, implementation, and testing. Below are the key steps followed in this project:

1. Understanding the Problem

- **A prime number is a natural number greater than 1 that is divisible only by 1 and itself.**
- **The problem requires two main functionalities:**
    1. **Checking whether a given number is prime.**
    2. **Generating all prime numbers up to a given limit.**

2. Algorithm Selection

To implement the prime number detection and generation efficiently, we selected the following algorithms:

A. Prime Number Checking (Trial Division Method)

- **A number N is checked for divisibility from 2 to √N.**
- **If N is divisible by any number in this range, it is not prime.**
- **This method significantly reduces the number of checks compared to a naive approach.**

B. Prime Number Generation (Sieve of Eratosthenes)

- **This algorithm generates all prime numbers up to a given limit N efficiently.**
- **Steps:**
    1. **Create a boolean list of size N+1, initialized as True.**
    2. **Set 0 and 1 as False (not prime).**
    3. **Starting from 2, mark all multiples as False (composite numbers).**
    4. **Continue this process up to √N, leaving only prime numbers as True.**
- **The Sieve of Eratosthenes has a time complexity of O(N log log N), making it much faster than checking each number individually.**

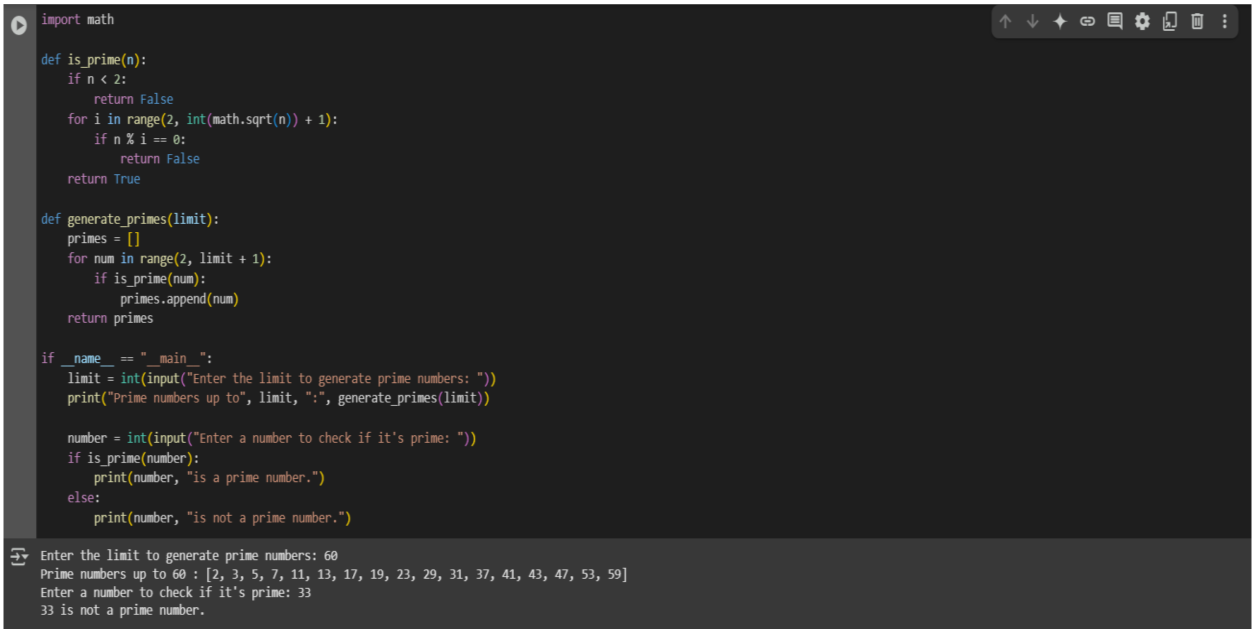
3. Implementation Approach

- **The program is written in Python for easy readability and implementation.**
- **It follows a modular approach, where prime checking and generation are handled by separate functions.**
- **The user is prompted to enter a number to check or input a limit for generating primes.**
- **The program processes the input using the selected algorithms and displays the results**

Code

```
import math


def is_prime(n):
    if n < 2:
        return False
    for i in range(2,
int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True


def generate_primes(limit):
    primes = []
    for num in range(2, limit
+ 1):
        if is_prime(num):

primes.append(num)
    return primes


if __name__ ==
"__main__":
    limit = int(input("Enter
the limit to generate prime
numbers: "))
    print("Prime numbers up
to", limit, ":",
generate_primes(limit))


    number = int(input("Enter
a number to check if it's
prime: "))
    if is_prime(number):
        print(number, "is a
prime number.")
    else:
        print(number, "is not a
prime number.")
```

Output



```python
import math

def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True

def generate_primes(limit):
    primes = []
    for num in range(2, limit + 1):
        if is_prime(num):
            primes.append(num)
    return primes

if __name__ == "__main__":
    limit = int(input("Enter the limit to generate prime numbers: "))
    print("Prime numbers up to", limit, ":", generate_primes(limit))

    number = int(input("Enter a number to check if it's prime: "))
    if is_prime(number):
        print(number, "is a prime number.")
    else:
        print(number, "is not a prime number.")
```

```
Enter the limit to generate prime numbers: 60
Prime numbers up to 60 : [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59]
Enter a number to check if it's prime: 33
33 is not a prime number.
```

Credits

- Mathematical background: Wikipedia - Prime Numbers
- Algorithm: GeeksforGeeks - Sieve of Eratosthenes
- Python Documentation: Python.org
- Code : Chatgpt