

**ARTIFICIAL INTELLIGENCE (AI)**

**AI101B**

**Prime Numbers Generator & Checker**

**Report**

**BACHELOR OF TECHNOLOGY**

**in**

**Computer Science Engineering (AIML)**

**Section: B**



**SUBMITTED BY:**

**ISHA TYAGI**

**202401100400100**

**SUBMITTED TO:**

**MR. ABHISHEK SHUKLA**

# Index

S.N.	Outline	Page No.
1	Introduction	2
2	Methodology	3-6
3	Code	7-8
4	Output/Result	9

# INTRODUCTION

Prime numbers play a fundamental role in number theory and have significant applications in fields such as cryptography, computer science, and mathematics. A prime number is defined as a number greater than 1 that has no positive divisors other than 1 and itself. Identifying prime numbers and generating lists of primes are essential tasks in many algorithms and computational processes.

Ex: 2,3,5,7.....so on.

17 is only divided by 1 and itself.

This report includes a Prime Number Checker and Generator. The prime number checker checks if a number is prime, employing optimal algorithms to minimize computation time for large numbers. The prime number generator generates a list of primes up to a given limit, which can be utilized in any application that demands prime sequences. In this report, we will discuss the techniques, algorithms, and performance issues involved in creating both of these tools, emphasizing their effectiveness and usefulness in solving practical problems with prime numbers

# METHODOLOGY

## 1. Problem Definition

- **Objective:** The objective is to implement a solution that performs two tasks:
  1. Generate all prime numbers up to a given limit.
  2. Check if a user-provided number is prime.

## 2. Prime Number Definition

- A prime number is a number greater than 1 that has no divisors other than 1 and itself.
- The smallest prime number is 2, and it is also the only even prime number. All other even numbers are not prime.

## 3. Solution Overview

The code consists of two main functions:

1. **is\_prime(n):** This function checks whether a given number  $n$  is prime.
2. **generate\_primes(limit):** This function generates all prime numbers up to a given limit.

## 4. Steps Involved

### 1. Prime Checking (is\_prime function):

- **Input:** A single integer n.
- **Output:** A boolean value (True or False), indicating whether the number is prime.
- **Method:**
  - If n is less than or equal to 1, return False as it is not prime.
  - For numbers greater than 1, check divisibility starting from 2 up to the square root of n (i.e.,  $\sqrt{n}$ ).
  - If n is divisible by any number in this range, return False as it is not prime.
  - If no divisors are found, return True indicating that n is prime.

### 2. Prime Generation (generate\_primes function):

- **Input:** A limit limit (integer).
- **Output:** A list of prime numbers up to the given limit.
- **Method:**
  - Iterate through all numbers from 2 to limit.
  - For each number, call the is\_prime() function to check whether it is prime.

- If the number is prime, append it to the list of primes.
- Finally, return the list of prime numbers.

### 3. User Interaction:

- The user is prompted to enter a **limit** to generate primes up to that number.
- The user is then asked to enter a **number to check** if it is prime or not.
- Based on the input, the program generates a list of primes and checks the primality of the user-provided number.

### 5. Optimization

The **is\_prime(n)** function has been optimized to check divisibility only up to  $\sqrt{n}$ . This optimization significantly reduces the number of iterations compared to checking divisibility up to  $n-1$ .

- For example, to check if 29 is prime, we only need to check divisibility by numbers up to  $\sqrt{29} \approx 5$ , rather than checking all numbers up to 29.
- This approach helps improve performance, especially when dealing with larger numbers.

### 6. User Input and Output

- The program first prompts the user to enter a **limit**, which defines the upper bound for generating prime numbers.

- It outputs all prime numbers up to and including the given limit.
- Then, the user is asked to provide a **number to check** whether it is prime or not. The program outputs whether the number is prime.

## 7. Example Run

- **Input:**
  1. **Enter Limit:** 20
  2. **Enter number to check:** 17
- **Output:**
  - Prime numbers up to 20: [2, 3, 5, 7, 11, 13, 17, 19]

## CODE

```
def is_prime(n):
    #Prime Number Checker
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1): # check upto sqrt(n)
        if n % i == 0:
            return False
    return True

def generate_primes(limit):
    #Generating Prime Numbers
    primes = [] # list for all prime numbers upto given limit
    for num in range(2, limit + 1):
        if is_prime(num):
            primes.append(num)
    return primes

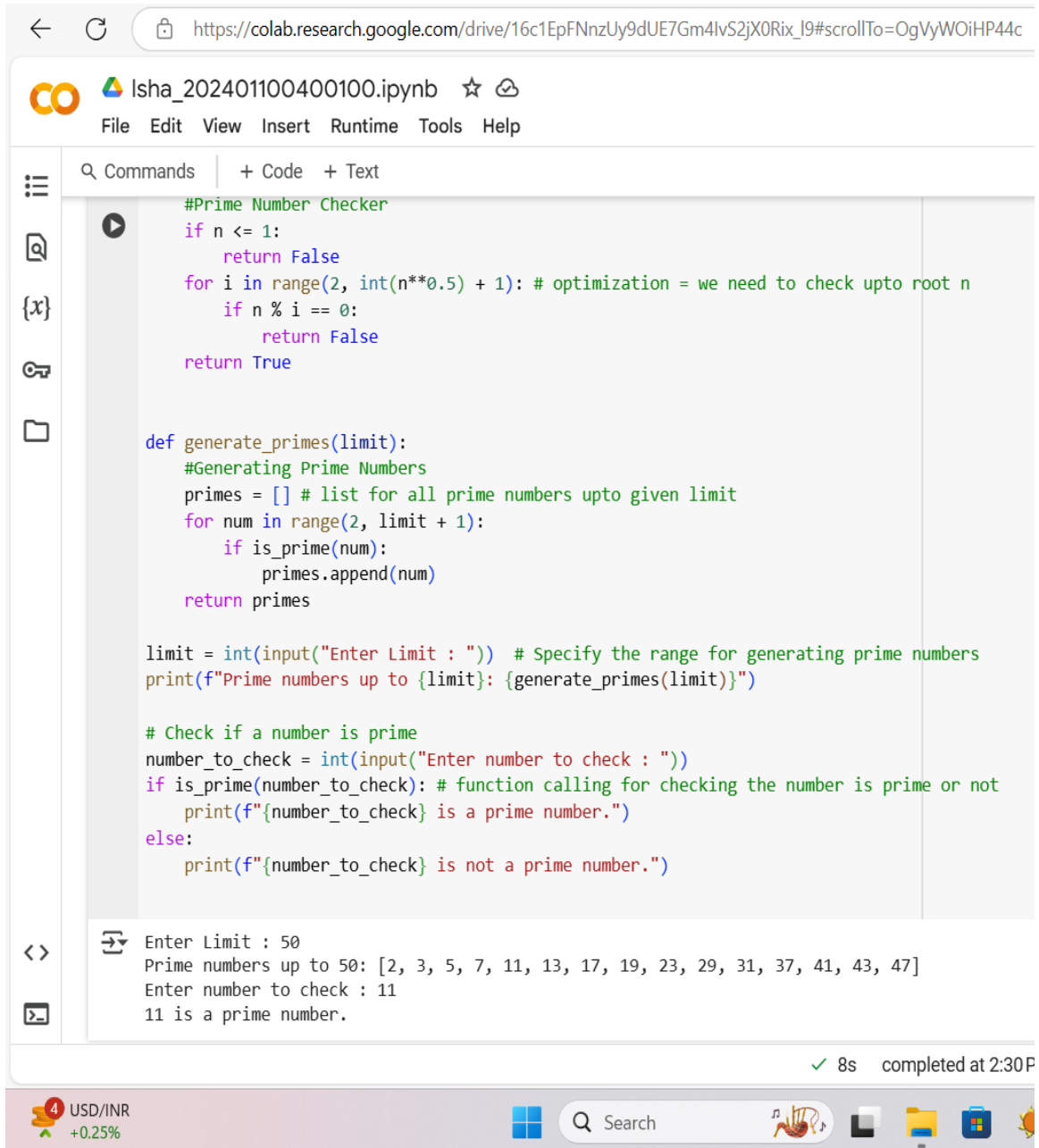
limit = int(input("Enter Limit : ")) # Specify the range for generating
prime numbers
print(f"Prime numbers up to {limit}: {generate_primes(limit)}")

# Check if a number is prime
```



```
number_to_check = int(input("Enter number to check : "))  
if is_prime(number_to_check):  
    print(f"{number_to_check} is a prime number.")  
else:  
    print(f"{number_to_check} is not a prime number.")
```

# OUTPUT



https://colab.research.google.com/drive/16c1EpFnzUy9dUE7Gm4lvS2jX0Rix\_I9#scrollTo=OgVyWOiHP44c

Isha\_202401100400100.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
#Prime Number Checker
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1): # optimization = we need to check upto root n
        if n % i == 0:
            return False
    return True

def generate_primes(limit):
    #Generating Prime Numbers
    primes = [] # list for all prime numbers upto given limit
    for num in range(2, limit + 1):
        if is_prime(num):
            primes.append(num)
    return primes

limit = int(input("Enter Limit : ")) # Specify the range for generating prime numbers
print(f"Prime numbers up to {limit}: {generate_primes(limit)}")

# Check if a number is prime
number_to_check = int(input("Enter number to check : "))
if is_prime(number_to_check): # function calling for checking the number is prime or not
    print(f"{number_to_check} is a prime number.")
else:
    print(f"{number_to_check} is not a prime number.")
```

Enter Limit : 50  
Prime numbers up to 50: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]  
Enter number to check : 11  
11 is a prime number.

8s completed at 2:30P

USD/INR +0.25%

Search