AI MIDSEM – 1 EXAMINATION

NAME – PRASHASTHA SINGH

ROLL NO. – 202401100300176

BRANCH – CSE(AI)

SECTION – C

INTRODUCTION

Prime numbers are a fundamental concept in number theory, defined as natural numbers greater than 1 that have no positive divisors other than 1 and themselves. Prime numbers are essential in various areas, including cryptography, algorithms, and computational mathematics.

In this project, we aim to create two functionalities using Python:

1. **Prime Number Generator**: A tool that generates all prime numbers within a specified range.
2. **Prime Number Checker**: A function that checks if a given number is prime or not.

These functionalities will be implemented using efficient algorithms to ensure optimal performance and accuracy.

METHODOLOGY

The methodology for creating the **Prime Number Generator** and **Prime Number Checker** involves using well-established algorithms and Python functions to ensure the correctness and efficiency of the solution.

1. Prime Number Checker (iS_prime function)

To determine if a number n is prime:

- A prime number is only divisible by 1 and itself. So, if any number from 2 to n\sqrt{n}n divides n evenly, it’s not prime.
- We can optimize the checking by limiting the range to n\sqrt{n}n and skipping even numbers (after checking 2).

Steps:

1. If n is less than 2, it's not prime.
2. If n=2n = 2n=2, it’s prime (2 is the only even prime).
3. For numbers greater than 2, check divisibility from 3 up to n\sqrt{n}n, skipping even numbers.

4. If no divisor is found, n is prime.

2. Prime Number Generator (generate_primes function)

To generate prime numbers in a given range:

- We use the **Sieve of Eratosthenes** algorithm, which efficiently marks non-prime numbers and returns the primes up to a given number n.
- Alternatively, a simpler method can use the **is_prime function** to check all numbers in the range.

Steps:

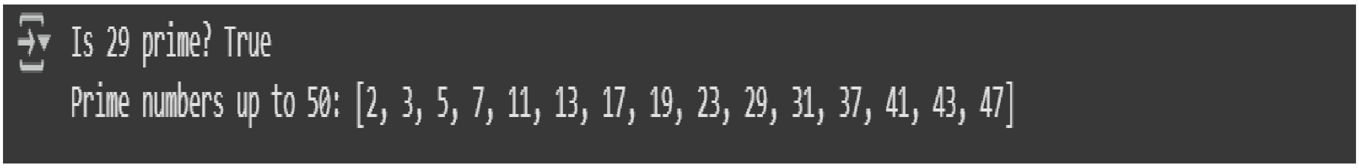
1. Start from 2 and check each number up to n using the **is_prime function**.
2. For each prime number found, add it to a list of primes, then return the list of primes.

CODE

```
import math


# Method 1: Prime Checker
def is_prime(n):
    """
    Function to check if a number 'n' is prime.
    A prime number is greater than 1 and divisible only by 1 and itself.
    """
    # Step 1: If n is less than or equal to 1, it's not a prime number.
    if n <= 1:
        return False


    # Step 2: 2 is the only even prime number, so return True if n is 2.
    if n == 2:
        return True


    # Step 3: Eliminate even numbers greater than 2 (they are not prime).
    if n % 2 == 0:
        return False

    # Step 4: Check divisibility from 3 to the square root of n.
    # Only check odd numbers (skip even numbers).
    for i in range(3, int(math.sqrt(n)) + 1, 2):
        if n % i == 0:
            return False   # If n is divisible by any of these, it's not prime.

    # Step 5: If no divisors were found, n is a prime number.
    return True


# Method 2: Prime Number Generator
def generate_primes(limit):
    """
    Function to generate all prime numbers up to a given limit.
    This function iterates through each number up to 'limit' and checks if it's prime.
    """
    primes = []  # List to store prime numbers
    # Step 1: Loop through all numbers from 2 to the specified 'limit'
    for num in range(2, limit + 1):
        # Step 2: Check if the number is prime using the is_prime function
        if is_prime(num):
            primes.append(num)  # Add the prime number to the primes list

    # Step 3: Return the list of prime numbers
    return primes


# Example usage
if __name__ == "__main__":
    # Example 1: Checking if a specific number is prime
    number = 29
    print(f"Is {number} prime? {is_prime(number)}")  # Expected output: True

    # Example 2: Generating prime numbers up to a specified limit
```

```
limit = 50
print(f"Prime numbers up to {limit}: {generate_primes(limit)}")
# Expected output: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

OUTPUT



```
Is 29 prime? True
Prime numbers up to 50: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

References and Credits:

1. **Prime Number Theory**:
    - A foundational concept in number theory that has been studied for centuries. More information can be found in standard mathematics textbooks such as:
        - *"Elementary Number Theory"* by David M. Burton.
        - *"An Introduction to the Theory of Numbers"* by G.H. Hardy and E.M. Wright.
2. **Sieve of Eratosthenes**:
    - The Sieve of Eratosthenes is an ancient algorithm used to find all primes up to any given limit. For an introduction, refer to:
        - *"The Art of Computer Programming, Volume 1: Fundamental Algorithms"* by Donald E. Knuth.
        - *"Introduction to Algorithms"* by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (often referred to as CLRS).
3. **Python Math Library**:
    - The Python math library was used to compute the square root and optimize prime checking. Official documentation can be found here:
        - Python Math Library Documentation: https://docs.python.org/3/library/math.html

4. **Python Documentation**:
    - Python’s official documentation provides a thorough guide on language features and standard libraries used for implementing algorithms.
        - Python 3 Documentation: https://docs.python.org/3/
5. **Online Resources and Tutorials**:
    - Many online resources, including Stack Overflow and Python documentation, were useful in shaping the logic and structure of the functions:
        - Stack Overflow: https://stackoverflow.com/
        - Real Python: https://realpython.com/
6. **Mathematical Algorithms**:
    - The algorithm for checking if a number is prime and for generating primes has been inspired by common algorithms used in computational mathematics and number theory.
        - *The Prime Number Theorem* and optimizations in prime checking can be further studied in online platforms like:
            - Khan Academy: https://www.khanacademy.org/math
            - GeeksforGeeks: https://www.geeksforgeeks.org/