



*Mid semester examination – Introduction To AI*

Name – Naitik Kukreja

Branch – CSE(AIML)

Section – B

Roll no. – 47

University roll no. – 202401100400120

Problem Statement:- Prime number generator and checker.

*\*\*Prime Number Generator and Checker\*\**

### **### Introduction**

Prime numbers play a crucial role in mathematics and computer science, particularly in cryptography and number theory. A prime number is a natural number greater than 1 that has exactly two distinct positive divisors: 1 and itself. This report explains a Python program that performs two main functions:

1. Checking whether a given number is prime.
2. Generating all prime numbers up to a specified limit.

The program is designed for efficiency using mathematical principles, such as checking divisibility only up to the square root of a number.

### **### Code Explanation**

The Python program consists of two main functions: `is_prime(n)` and `prime_generator(limit)`. It also includes a user interaction section to accept input for both generating primes and checking if a number is prime.

**#### 1.** Checking if a Number is Prime (`is_prime(n)`)

The function `is_prime(n)` determines whether a number is prime using an efficient approach:

```
``python code
def is_prime(n):
    """Check if a number is prime."""
    if n < 2:
        return False # Numbers less than 2 are not prime
    for i in range(2, int(n ** 0.5) + 1): # Check divisibility up to the
square root of n
        if n % i == 0:
            return False # If divisible, not a prime number
    return True # If no divisors found, it's prime
``
```

#### **\*\*Explanation:\*\***

- If `n` is less than 2, it is not a prime number.
- The loop iterates from 2 to `sqrt(n)`, since a larger factor of `n` would have a corresponding smaller factor.
- If `n` is divisible by any of these numbers, it returns `False`.
- Otherwise, it returns `True`, confirming `n` is prime.

## **#### 2. Generating Prime Numbers**

`(prime_generator(limit))`

This function generates prime numbers up to a given limit:

```
``python code
def prime_generator(limit):
    """Generate prime numbers up to a given limit."""
    for num in range(2, limit + 1): # Iterate from 2 to the given limit
        if is_prime(num): # Check if the number is prime
            yield num # Yield the prime number
``
```

#### **\*\*Explanation:\*\***

- The function iterates over numbers from 2 to `limit`.
- For each number, it calls `is\_prime(num)`, checking whether it is prime.
- If `is\_prime(num)` returns `True`, the number is yielded (produced as output) dynamically using the `yield` statement, which makes it a generator function.

### **#### 3. User Interaction**

The program includes a section for user input and execution:

```
``python code
if __name__ == "__main__":
    limit = int(input("Enter the limit for prime generation: ")) # Get
user input for prime generation limit
    print("Prime numbers up to", limit, ":", list(prime_generator(limit)))
# Print generated prime numbers

    num = int(input("Enter a number to check if it's prime: ")) # Get
user input for prime check
    print(f"{num} is prime:" if is_prime(num) else f"{num} is not prime")
# Print prime check result
``
```

### **\*\*Explanation:\*\***

- The user first inputs a limit, and the program generates all prime numbers up to that limit using `prime\_generator(limit)`.
- Then, the user enters a number to check if it is prime using `is\_prime(num)`, and the result is displayed.

### **### Example Execution**

When the program runs, it produces output like the following:

```
``
```

Enter the limit for prime generation: 20

Prime numbers up to 20: [2, 3, 5, 7, 11, 13, 17, 19]

Enter a number to check if it's prime: 29

29 is prime:

...

### **Efficiency Considerations**

- The `is_prime(n)` function uses a mathematical optimization by iterating only up to `sqrt(n)`, which significantly reduces the number of checks compared to a naive approach.
- The `prime_generator(limit)` function avoids unnecessary computations by leveraging `is_prime(n)`.
- The use of a generator (`yield`) improves efficiency when working with large values, as it does not store all prime numbers in memory at once.

### **Conclusion**

This Python program efficiently generates prime numbers and verifies if a number is prime using optimized mathematical operations. Prime number identification is essential in areas like cryptography, data security, and computational mathematics. This implementation balances efficiency and clarity, making it suitable for educational and practical applications.

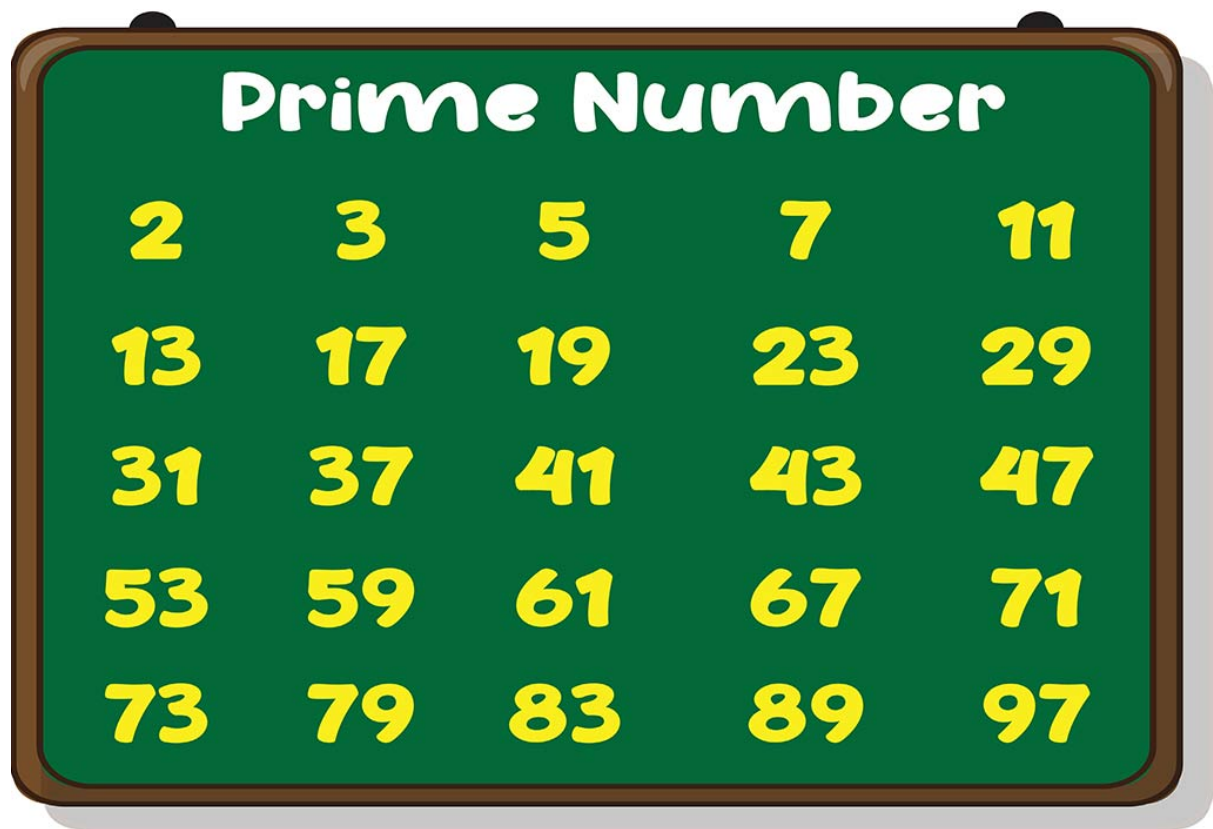
```
def is_prime(n):
    """Check if a number is prime."""
    if n < 2:
        return False # Numbers less than 2 are not prime
    for i in range(2, int(n ** 0.5) + 1): # Check divisibility up to the square root of n
        if n % i == 0:
            return False # If divisible, not a prime number
    return True # If no divisors found, it's prime

def prime_generator(limit):
    """Generate prime numbers up to a given limit."""
    for num in range(2, limit + 1): # Iterate from 2 to the given limit
        if is_prime(num): # Check if the number is prime
            yield num # Yield the prime number

# Example usage
if __name__ == "__main__":
    limit = int(input("Enter the limit for prime generation: ")) # Get user input for prime generation limit
    print("Prime numbers up to", limit, ":", list(prime_generator(limit))) # Print generated prime numbers

    num = int(input("Enter a number to check if it's prime: ")) # Get user input for prime check
    print(f"{num} is prime:" if is_prime(num) else f"{num} is not prime") # Print prime check result
```

Enter the limit for prime generation: 20  
Prime numbers up to 20 : [2, 3, 5, 7, 11, 13, 17, 19]  
Enter a number to check if it's prime: 29  
29 is prime:



Prime Number				
2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71
73	79	83	89	97

**References:**

1. Chat GPT
2. Class Notes
3. WIKIPEDIA