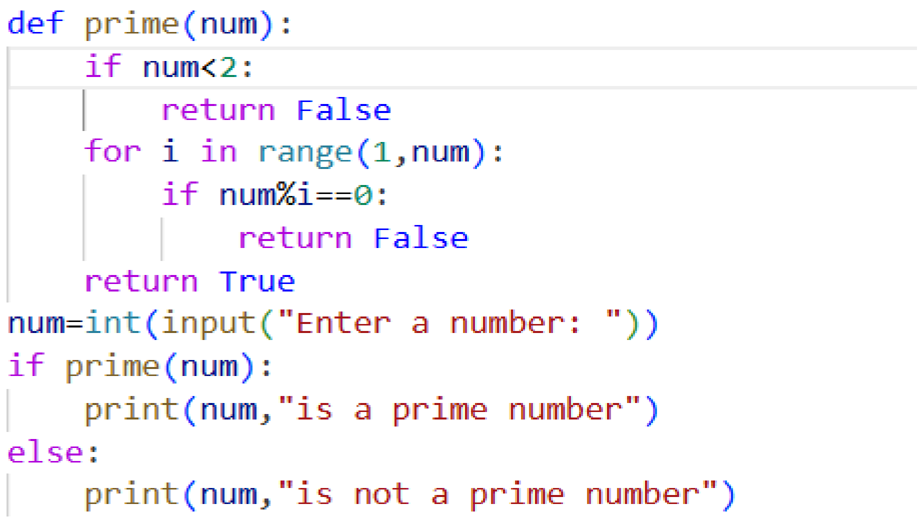
Lab Assignment 9.2

Program : B. Tech (CSE)
Course Title : AI Assisted Coding
Course Code : 23CS002PC304
Semester : III
Academic Session : 2025-2026
Name of Student : Sruthi
Enrollment No. : 2403A51L10
Batch No. : 51
Date : 06-02-2026

Task 1:

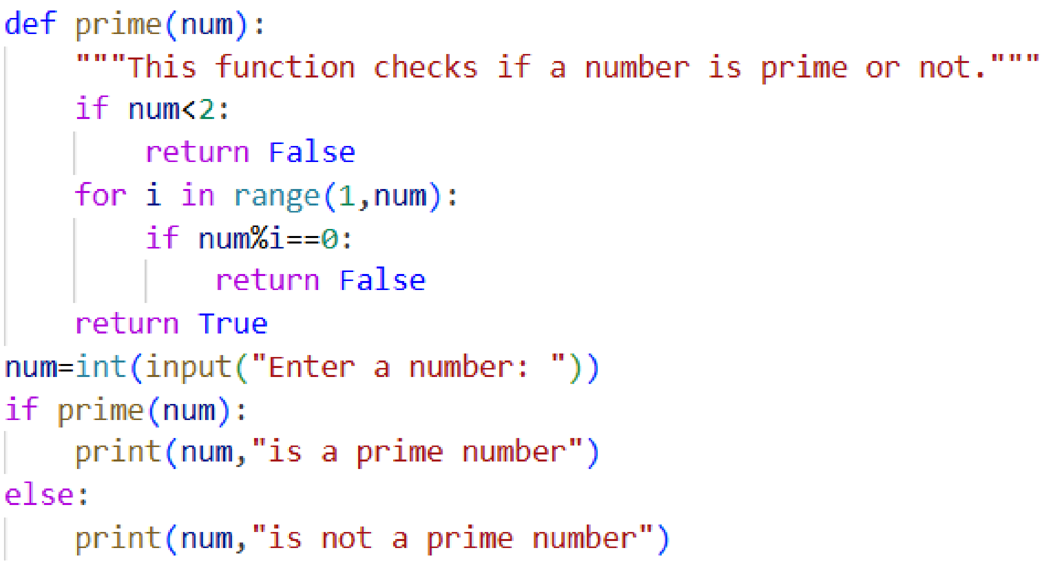
Code:



```
def prime(num):
    if num<2:
        return False
    for i in range(1,num):
        if num%i==0:
            return False
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

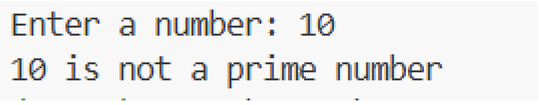
Prompt: Provide a short summary describing the purpose of the function.

Code:



```
def prime(num):
    """This function checks if a number is prime or not."""
    if num<2:
        return False
    for i in range(1,num):
        if num%i==0:
            return False
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

Output:



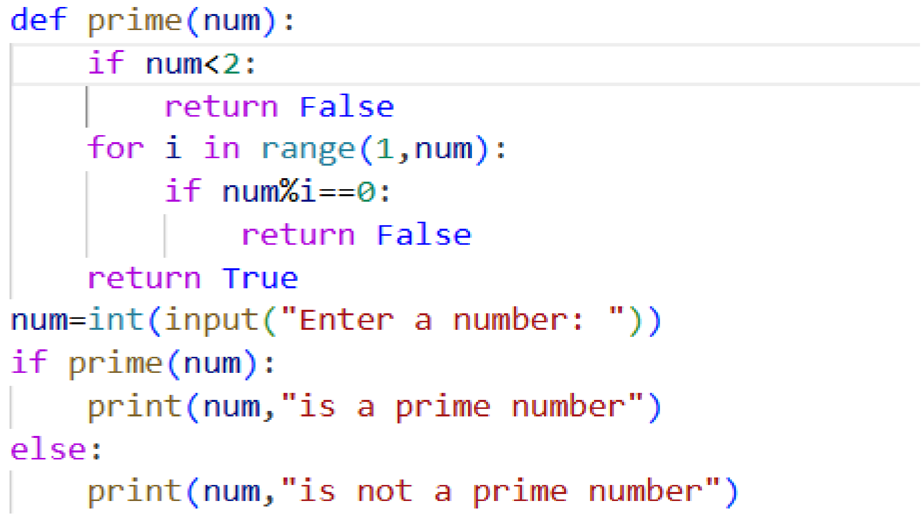
```
Enter a number: 10
10 is not a prime number
```

Explanation:

The first code is just a normal code. The code cannot understand to others.
The second tells the purpose of the code.

Task 2:

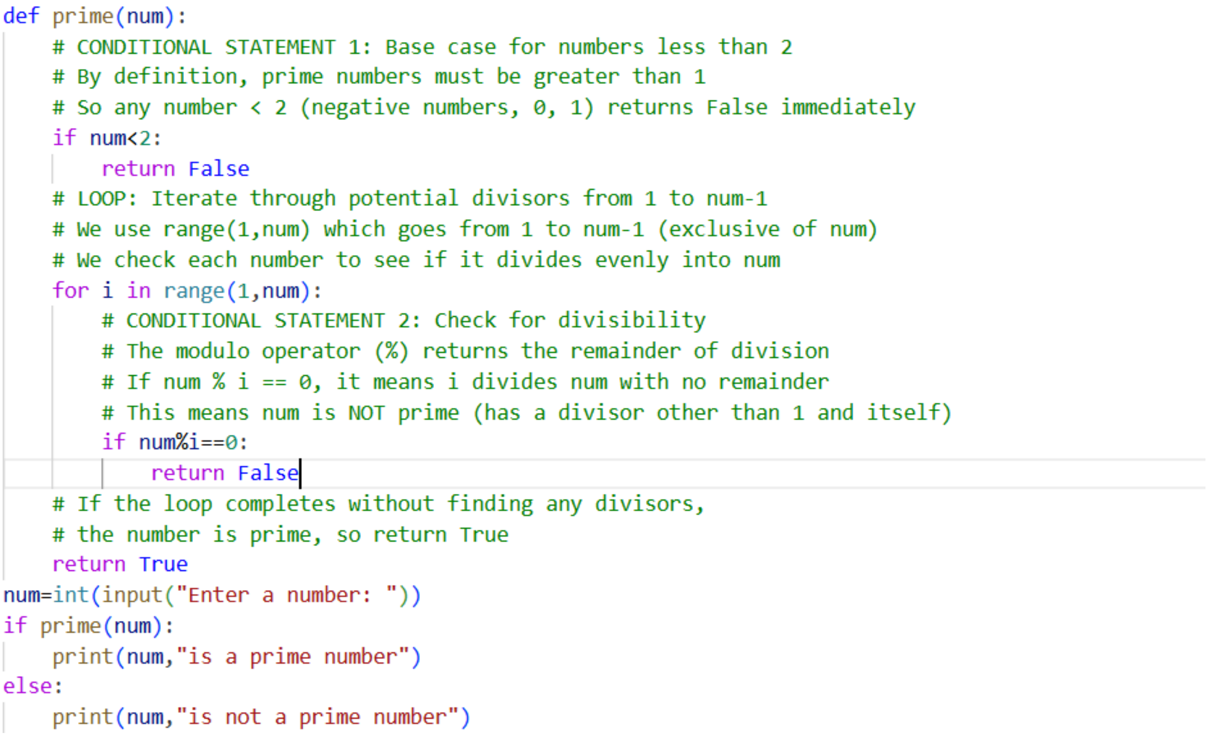
Code:



```python
def prime(num):
    if num<2:
        return False
    for i in range(1,num):
        if num%i==0:
            return False
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

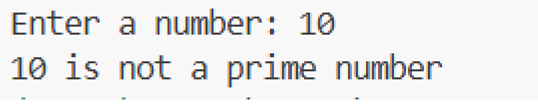
Prompt: Provide the document the logic behind the conditional statements and loops used in the function.

Code:



```python
def prime(num):
    # CONDITIONAL STATEMENT 1: Base case for numbers less than 2
    # By definition, prime numbers must be greater than 1
    # So any number < 2 (negative numbers, 0, 1) returns False immediately
    if num<2:
        return False
    # LOOP: Iterate through potential divisors from 1 to num-1
    # We use range(1,num) which goes from 1 to num-1 (exclusive of num)
    # We check each number to see if it divides evenly into num
    for i in range(1,num):
        # CONDITIONAL STATEMENT 2: Check for divisibility
        # The modulo operator (%) returns the remainder of division
        # If num % i == 0, it means i divides num with no remainder
        # This means num is NOT prime (has a divisor other than 1 and itself)
        if num%i==0:
            return False
    # If the loop completes without finding any divisors,
    # the number is prime, so return True
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

Output:



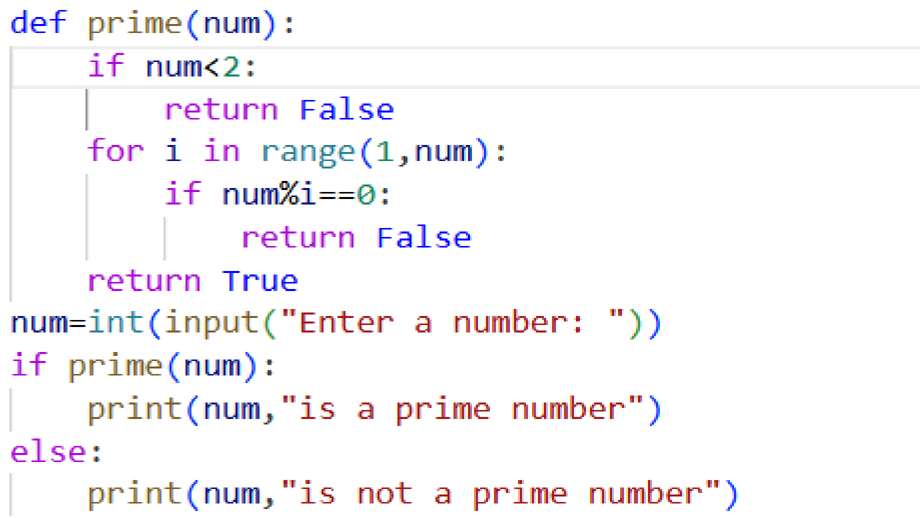
```
Enter a number: 10
10 is not a prime number
```

Explanation:

The first code is just a normal code. The code cannot understand to others.
And the second code gives the detailed explanation of loops & conditional statements used in the code.

Task 3:

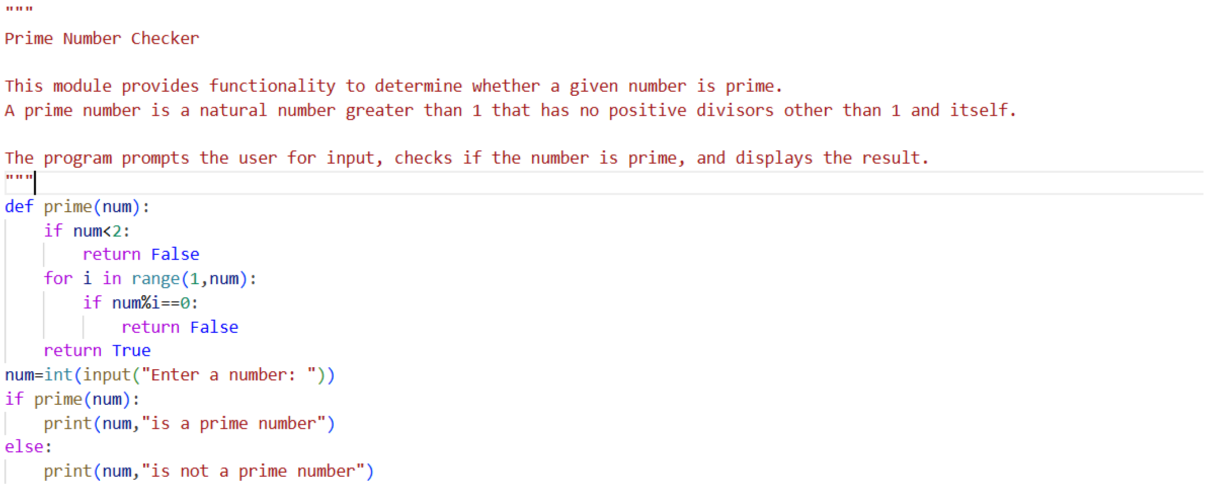
Code:



```python
def prime(num):
    if num<2:
        return False
    for i in range(1,num):
        if num%i==0:
            return False
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

Prompt: Generate a high-level overview describing the functionality of an entire Python file.

Code:

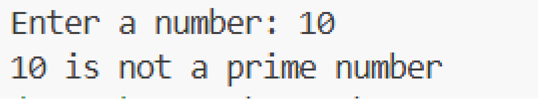


```python
"""
Prime Number Checker

This module provides functionality to determine whether a given number is prime.
A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.

The program prompts the user for input, checks if the number is prime, and displays the result.
"""
def prime(num):
    if num<2:
        return False
    for i in range(1,num):
        if num%i==0:
            return False
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

Output:



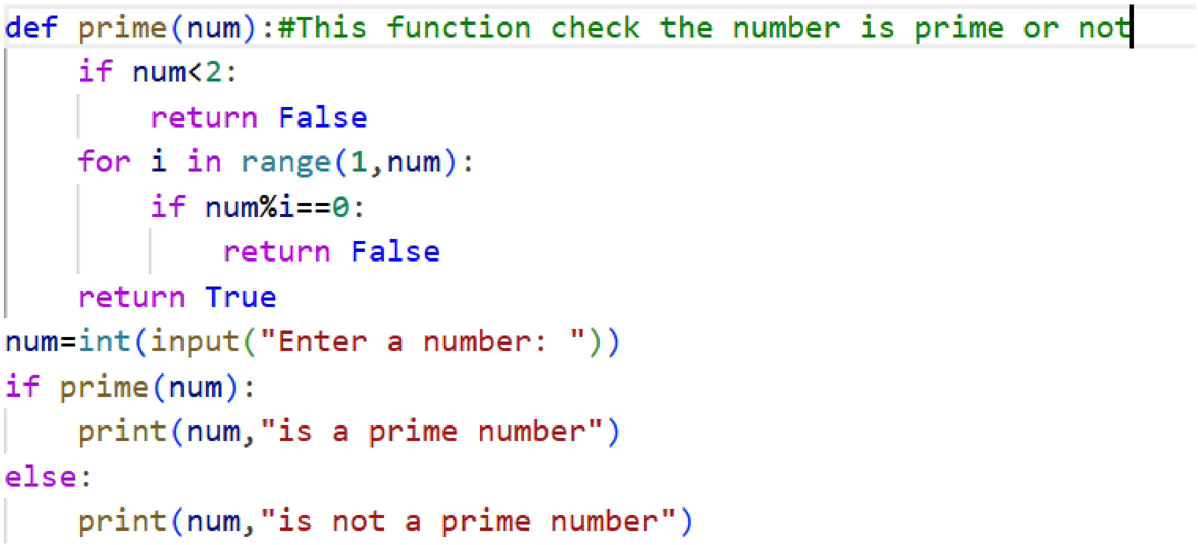
```
Enter a number: 10
10 is not a prime number
```

Explanation:

The first code is just a normal code. The code cannot understand to others.
And the second code explains the functionality of code.

Task 4:

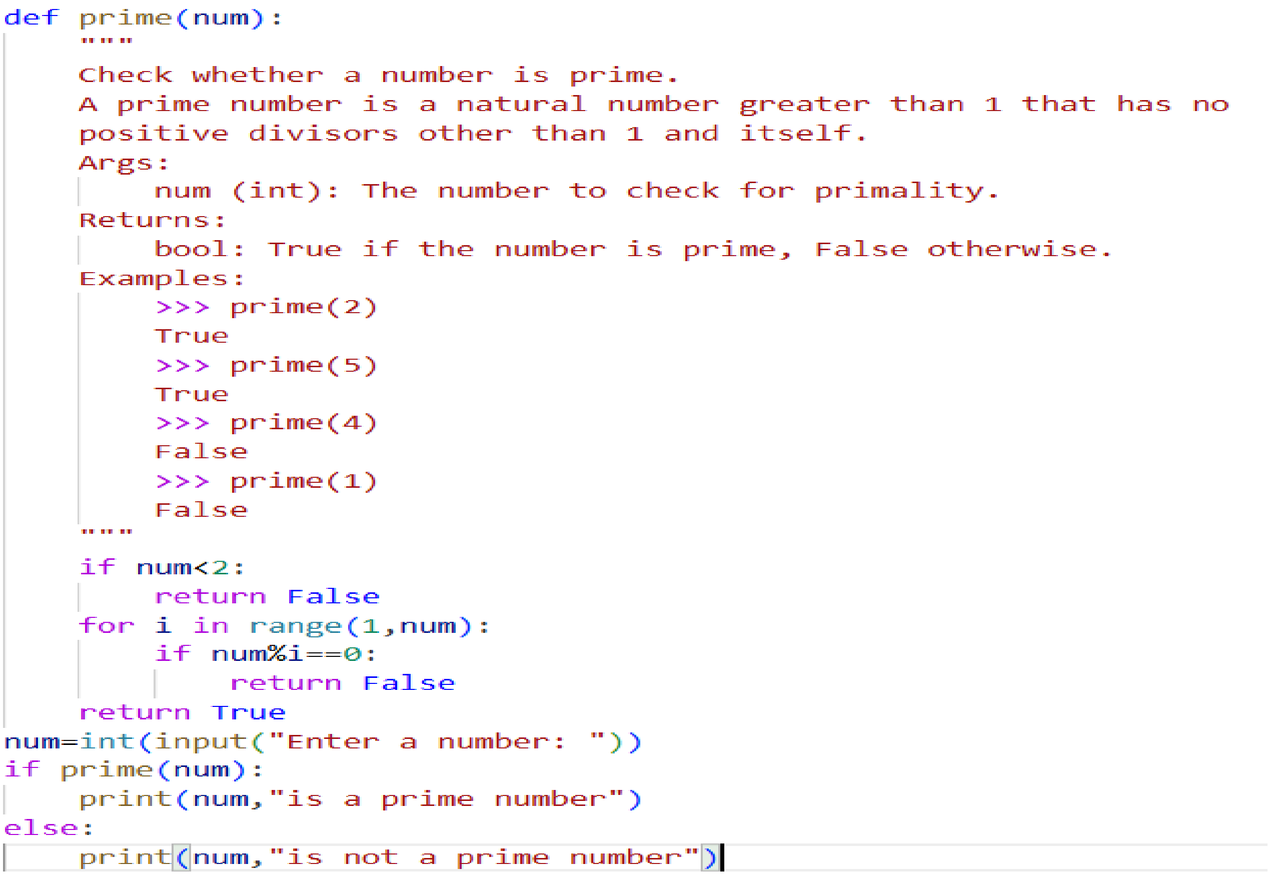
Code:



```python
def prime(num):#This function check the number is prime or not
    if num<2:
        return False
    for i in range(1,num):
        if num%i==0:
            return False
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

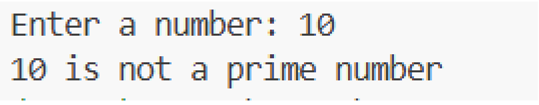
Prompt: A Python file with a clear and concise file-level overview at the beginning.

Code:



```python
def prime(num):
    """
    Check whether a number is prime.
    A prime number is a natural number greater than 1 that has no
    positive divisors other than 1 and itself.
    Args:
        num (int): The number to check for primality.
    Returns:
        bool: True if the number is prime, False otherwise.
    Examples:
        >>> prime(2)
        True
        >>> prime(5)
        True
        >>> prime(4)
        False
        >>> prime(1)
        False
    """
    if num<2:
        return False
    for i in range(1,num):
        if num%i==0:
            return False
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

Output:



```
Enter a number: 10
10 is not a prime number
```

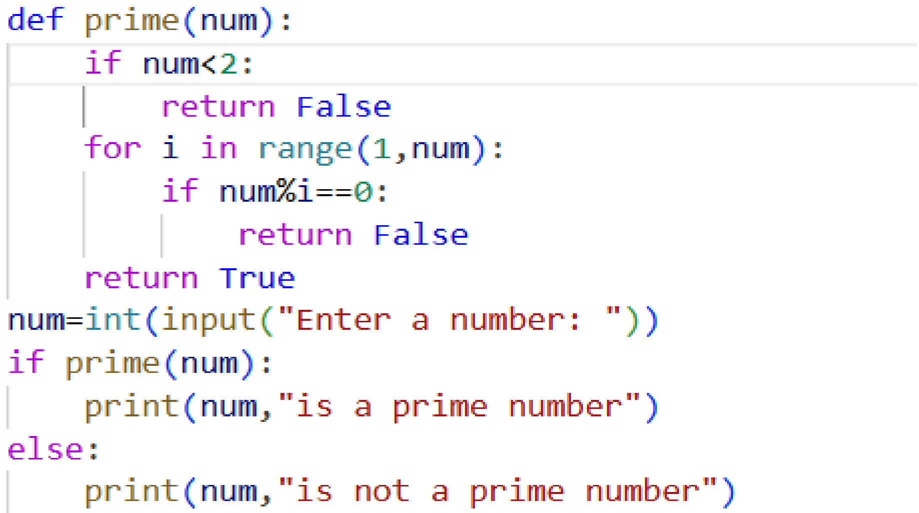
Explanation:

The first code just explains what the functions does.

But the second code explains with different examples.

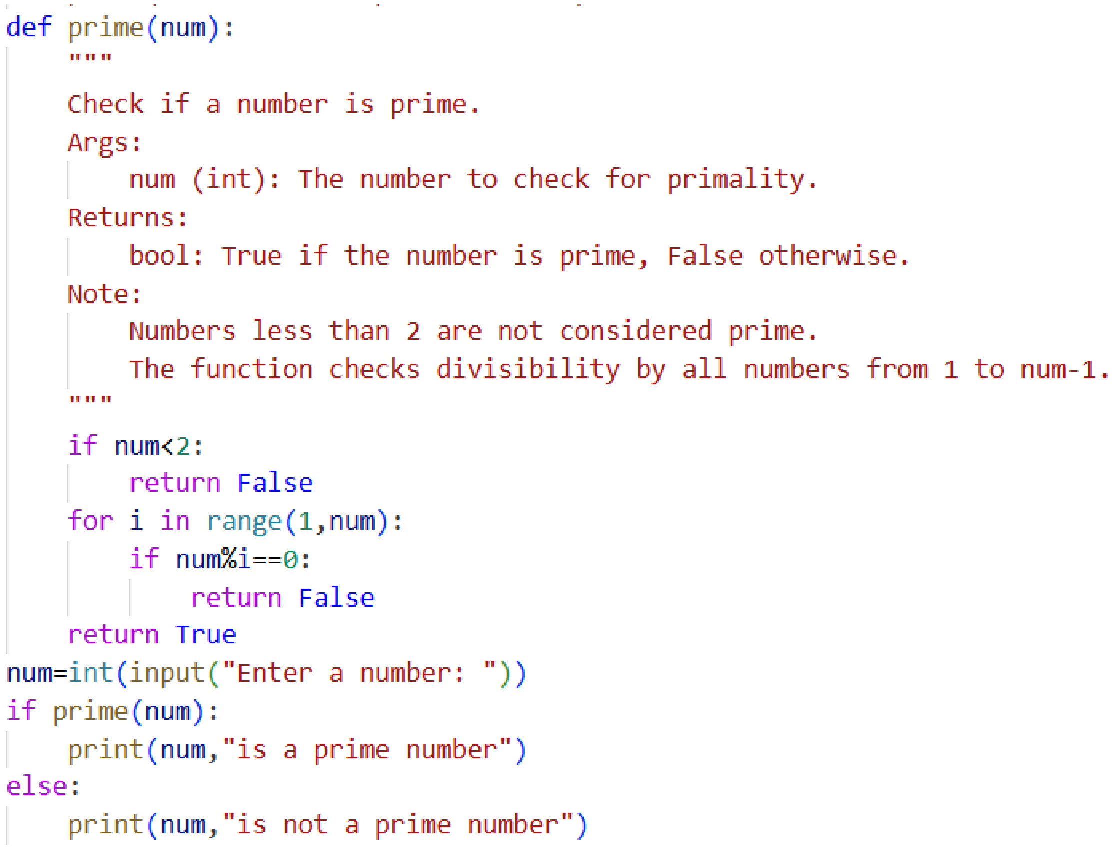
Task 5:

Code:



```python
def prime(num):
    if num<2:
        return False
    for i in range(1,num):
        if num%i==0:
            return False
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

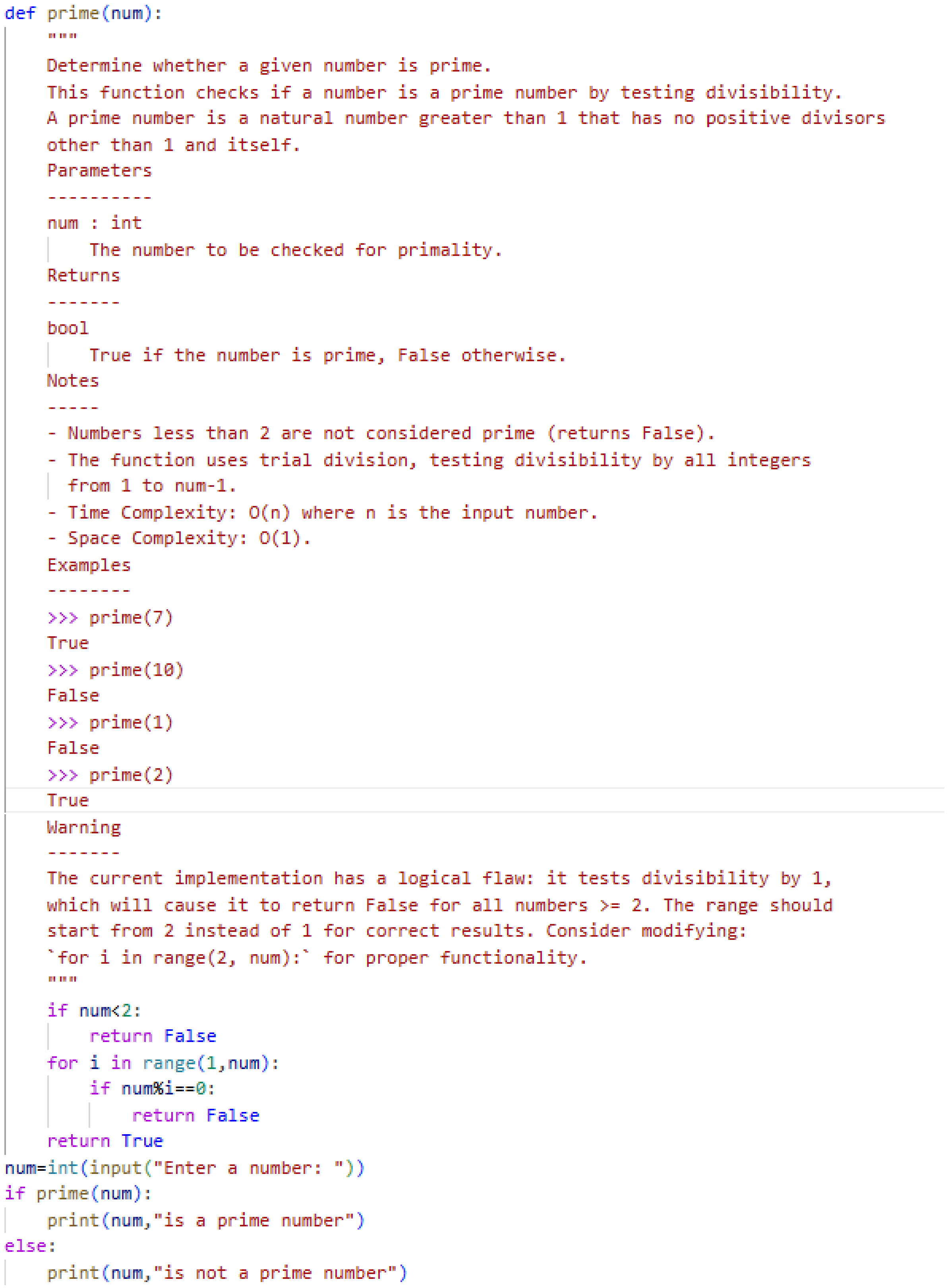
Prompt: Provide a brief explanation of the code



```python
def prime(num):
    """
    Check if a number is prime.
    Args:
        num (int): The number to check for primality.
    Returns:
        bool: True if the number is prime, False otherwise.
    Note:
        Numbers less than 2 are not considered prime.
        The function checks divisibility by all numbers from 1 to num-1.
    """
    if num<2:
        return False
    for i in range(1,num):
        if num%i==0:
            return False
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

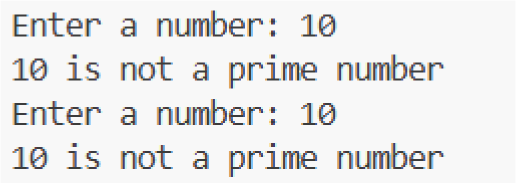
Prompt: Provide a detailed explanation of the code

Code:



```
def prime(num):
    """
    Determine whether a given number is prime.
    This function checks if a number is a prime number by testing divisibility.
    A prime number is a natural number greater than 1 that has no positive divisors
    other than 1 and itself.
    Parameters
    ----------
    num : int
        The number to be checked for primality.
    Returns
    -------
    bool
        True if the number is prime, False otherwise.
    Notes
    -----
    - Numbers less than 2 are not considered prime (returns False).
    - The function uses trial division, testing divisibility by all integers
      from 1 to num-1.
    - Time Complexity: O(n) where n is the input number.
    - Space Complexity: O(1).
    Examples
    --------
    >>> prime(7)
    True
    >>> prime(10)
    False
    >>> prime(1)
    False
    >>> prime(2)
    True
    Warning
    -------
    The current implementation has a logical flaw: it tests divisibility by 1,
    which will cause it to return False for all numbers >= 2. The range should
    start from 2 instead of 1 for correct results. Consider modifying:
    `for i in range(2, num):` for proper functionality.
    """
    if num<2:
        return False
    for i in range(1,num):
        if num%i==0:
            return False
    return True
num=int(input("Enter a number: "))
if prime(num):
    print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

Output:



```
Enter a number: 10
10 is not a prime number
Enter a number: 10
10 is not a prime number
```

Comparison table:

Aspect	**Brief Explanation Code**	**Detailed Explanation Code**
Completeness	Provides a basic overview of the function's purpose.	Offers a comprehensive description including parameters, return values, notes, examples, and warnings.
Clarity	Simple and straightforward, but may lack depth.	Clear and detailed, making it easier to understand the function's behaviour and edge cases.
Accuracy	Contains a logical flaw in the implementation (checks in divisibility by 1).	Accurately describes the function's intent but also highlights the flaw in the implementation, suggesting a correction.
Usefulness	Useful for quick reference but may lead to misunderstandings due to lack of detail.	Highly useful for developers needing in-depth understanding and correct usage of the function.