AI ASSISTED CODING

LAB ASSIGNMENT-1

J.Sai Kumar || Batch:-09 || 2303A51562

Task-1: Write a python program to check a given number Prime number or not without using functions.

Code:-

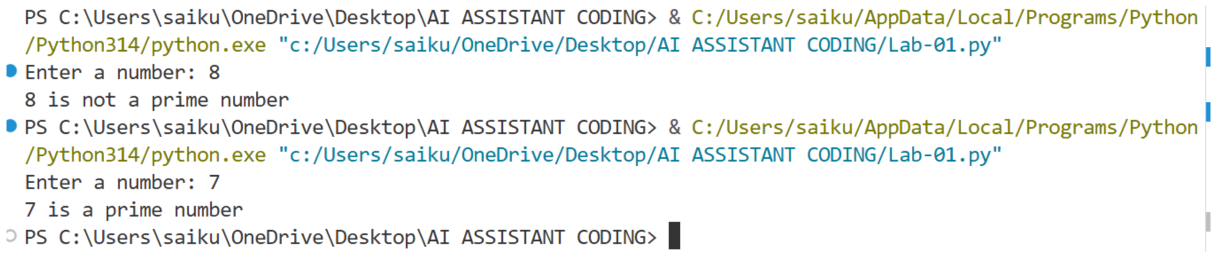
```python
num = int(input("Enter a number: "))

if num < 2:
    print(f"{num} is not a prime number")
else:
    is_prime = True

    for i in range(2, num):
        if num % i == 0:
            is_prime = False
            break

    if is_prime:
        print(f"{num} is a prime number")
    else:
        print(f"{num} is not a prime number")
```

Output:



```
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/saiku/AppData/Local/Programs/Python
/Python314/python.exe "c:/Users/saiku/OneDrive/Desktop/AI ASSISTANT CODING/Lab-01.py"
Enter a number: 8
8 is not a prime number
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/saiku/AppData/Local/Programs/Python
/Python314/python.exe "c:/Users/saiku/OneDrive/Desktop/AI ASSISTANT CODING/Lab-01.py"
Enter a number: 7
7 is a prime number
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING>
```

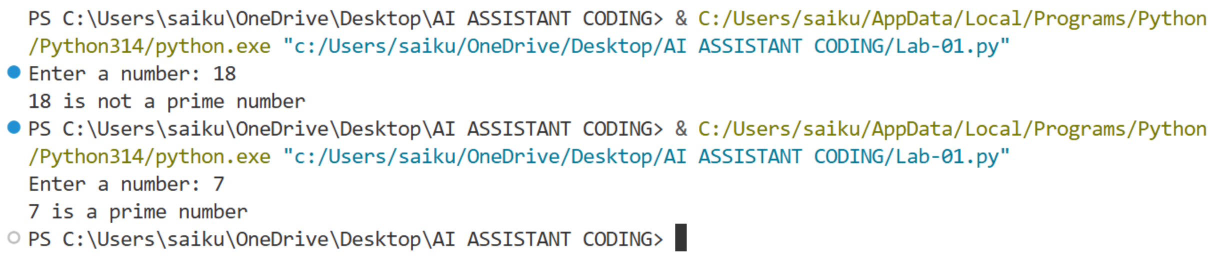
Task-2: Optimize the above code using functions.

Code:-

```
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True

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

Output:



```
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/saiku/AppData/Local/Programs/Python
/Python314/python.exe "c:/Users/saiku/OneDrive/Desktop/AI ASSISTANT CODING/Lab-01.py"
Enter a number: 18
18 is not a prime number
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/saiku/AppData/Local/Programs/Python
/Python314/python.exe "c:/Users/saiku/OneDrive/Desktop/AI ASSISTANT CODING/Lab-01.py"
Enter a number: 7
7 is a prime number
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING>
```

Observation:-

In **Task-01**, the program checks whether a number is prime by trying to divide it with every number from **2 up to N−1**. If any of these numbers divides the given number exactly, it is not prime. Although this method works correctly, it does a lot of unnecessary checking, especially when the number is large, which makes the program slower.

In **Task-02**, the program uses a smarter approach. Instead of checking all the way up to N−1, it only checks numbers from **2 up to √N**. This is because if a number has a factor greater than √N, there must also be a smaller factor below √N. So checking beyond √N is not needed.

Because of this optimization, the program runs faster while still giving correct results. For example, when **N = 25**, it is enough to check divisibility only up to **5**, instead of checking all the way up to **24**.