

2303A521563

batch=10

Task 1: AI-Generated Logic Without Modularization (Prime Number Check Without Functions)

❖ Scenario

➢ You are developing a basic validation script for a numerical learning application.

❖ Task Description

Use GitHub Copilot to generate a Python program that:

➢ Checks whether a given number is prime

➢ Accepts user input

➢ Implements logic directly in the main code

➢ Does not use any user-defined functions

❖ Expected Output

➢ Correct prime / non-prime result

➢ Screenshots showing Copilot-generated code suggestions

➢ Sample inputs and outputs

# Task 1: Prime Number Check Without Functions

```
n = int(input("Enter a number: "))

if n <= 1:
    print(n, "is not a prime number")
else:
    prime = True

    for i in range(2, n):
        if n % i == 0:
            prime = False
            break

    if prime:
        print(n, "is a prime number")
    else:
        print(n, "is not a prime number")
```

The screenshot shows the OnlineGDB IDE interface. On the left, there's a sidebar with links like 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Opportunity for Intern', 'Sign Up', and 'Login'. Below this is an 'auth0' advertisement. The main workspace has tabs for 'main.py' and 'input'. The code in 'main.py' is:

```

1 # Task 1: Prime Number Check Without Functions
2
3 n = int(input("Enter a number: "))
4
5 if n <= 1:
6     print(n, "is not a prime number")
7 else:
8     prime = True
9
10    for i in range(2, n):
11        if n % i == 0:
12            prime = False
13            break
14
15    if prime:
16        print(n, "is a prime number")
17    else:
18        print(n, "is not a prime number")
19

```

The 'input' tab shows the program's interaction with the user:

```

input
Enter a number: 5
5 is a prime number
...Program finished with exit code 0
Press ENTER to exit console.

```

## Explanation

### Explanation

- User enters a number `n`
- If `n <= 1`, it is not prime
- Loop runs from `2` to `n-1`
- If divisible by any number → not prime

### Time Complexity

- Worst case:  $O(n)$
- Not efficient for large numbers

### Task 2: Efficiency & Logic Optimization (Cleanup)

#### ❖ Scenario

The script must handle larger input values efficiently.

#### ❖ Task Description

Review the Copilot-generated code from Task 1 and improve it by:

- Reducing unnecessary iterations
- Optimizing the loop range (e.g., early termination)
- Improving readability
- Use Copilot prompts like:
  - “Optimize prime number checking logic”

- “Improve efficiency of this code”

Hint:

Prompt Copilot with phrases like

“optimize this code”, “simplify logic”, or “make it more readable”

❖ Expected Output

➢ Original and optimized code versions

➢ Explanation of how the improvements reduce time complexity

# Task 2: Optimized Prime Number Check

```
n = int(input("Enter a number: "))
```

```
if n <= 1:  
    print(n, "is not a prime number")
```

```
else:
```

```
    prime = True
```

```
    for i in range(2, int(n**0.5) + 1):
```

```
        if n % i == 0:
```

```
            prime = False
```

```
            break
```

```
    if prime:
```

```
        print(n, "is a prime number")
```

```
    else:
```

```
        print(n, "is not a prime number")
```

The screenshot shows the OnlineGDB interface. On the left, there's a sidebar with links like 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Opportunity for Intern', 'Sign Up', and 'Login'. A central window displays a Python script named 'main.py' with the following code:

```

1 # Task 2: Optimized Prime Number Check
2
3 n = int(input("Enter a number: "))
4
5 if n <= 1:
6     print(n, "is not a prime number")
7 else:
8     prime = True
9
10    for i in range(2, int(n**0.5) + 1):
11        if n % i == 0:
12            prime = False
13            break
14
15    if prime:
16        print(n, "is a prime number")
17    else:
18        print(n, "is not a prime number")
19

```

Below the code editor, a terminal window shows the output of running the program with the input '7':

```

input
Enter a number: 7
7 is a prime number

...Program finished with exit code 0
Press ENTER to exit console.

```

A sidebar on the left also displays an 'auth0' advertisement.

### Task 3: Modular Design Using AI Assistance (Prime Number Check Using Functions)

#### ❖ Scenario

The prime-checking logic will be reused across multiple modules.

#### ❖ Task Description

Use GitHub Copilot to generate a function-based Python program that:

- Uses a user-defined function to check primality
- Returns a Boolean value
- Includes meaningful comments (AI-assisted)
- ❖ Expected Output
- Correctly working prime-checking function
- Screenshots documenting Copilot's function generation
- Sample test cases and outputs

# Task 2: Optimized Prime Number Check

# Task 3: Prime Number Check Using Function

```
def is_prime(n):
```

```
    """
```

This function checks whether a number is prime or not.

Returns True if prime, otherwise False.

```
    """
```

```
if n <= 1:
```

```
return False

for i in range(2, int(n**0.5) + 1):
    if n % i == 0:
        return False
```

```
return True
```

The screenshot shows the OnlineGDB IDE interface. On the left, there's a sidebar with links for 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Opportunity for Intern', 'Sign Up', and 'Login'. A banner from auth0 is displayed, stating: 'You asked, we delivered! Our Free Plan now includes a Custom Domain, 5 Actions, and 25,000 MAUs.' Below the sidebar are links for 'About', 'FAQ', 'Blog', 'Terms of Use', 'Contact Us', 'GDB Tutorial', 'Credits', and 'Privacy', along with the copyright notice '© 2016 - 2026 GDB Online'.

The main workspace shows a Python script named 'main.py' with the following code:

```
1 # Task 2: Optimized Prime Number Check
2
3 n = int(input("Enter a number: "))
4
5 if n <= 1:
6     print(n, "is not a prime number")
7 else:
8     prime = True
9
10    for i in range(2, int(n**0.5) + 1):
11        if n % i == 0:
12            prime = False
13            break
14
15    if prime:
16        print(n, "is a prime number")
17    else:
18        print(n, "is not a prime number")
19
```

Below the code editor is a terminal window with the following output:

```
input
Enter a number: 8
8 is not a prime number

...Program finished with exit code 0
Press ENTER to exit console.
```

The screenshot shows the OnlineGDB IDE interface. On the left, there's a sidebar with links like 'My Projects', 'Classroom', 'Learn Programming', etc. The main area has tabs for 'main.py' and 'output'. The code in 'main.py' is:

```
1 # Task 3: Prime Number Check Using Function
2
3 def is_prime(n):
4     """
5         This function checks whether a number is prime or not.
6         Returns True if prime, otherwise False.
7     """
8
9     if n <= 1:
10        return False
11
12    for i in range(2, int(n**0.5) + 1):
13        if n % i == 0:
14            return False
15
16    return True
17
18
19 # Main Program
20 num = int(input("Enter a number: "))
21
22 if is_prime(num):
23     print(num, "is a prime number")
24 else:
25     print(num, "is not a prime number")
```

The 'output' tab shows the console output:

```
input
Enter a number: 8
8 is not a prime number

...Program finished with exit code 0
Press ENTER to exit console.
```

This screenshot shows the same OnlineGDB IDE interface, but the code in 'main.py' has been optimized:

```
3 def is_prime(n):
4     """
5         This function checks whether a number is prime or not.
6         Returns True if prime, otherwise False.
7     """
8
9     if n <= 1:
10        return False
11
12    for i in range(2, int(n**0.5) + 1):
13        if n % i == 0:
14            return False
15
16    return True
17
18
19 # Main Program
20 num = int(input("Enter a number: "))
21
22 if is_prime(num):
23     print(num, "is a prime number")
24 else:
25     print(num, "is not a prime number")
```

The 'output' tab shows the console output:

```
input
Enter a number: 8
8 is not a prime number

...Program finished with exit code 0
Press ENTER to exit console.
```

explanation

**Optimization Done:**

- Instead of checking till  $n-1$
- We check only till  $\sqrt{n}$

Because:

If  $n$  has a factor greater than  $\sqrt{n}$ , it must also have a factor smaller than  $\sqrt{n}$ .

---

## Time Complexity

- Before:  $O(n)$
- After:  $O(\sqrt{n})$   Faster for large inputs

Task 4: Comparative Analysis –With vs Without Functions

❖ Scenario

You are participating in a technical review discussion.

❖ Task Description

Compare the Copilot-generated programs:

- Without functions (Task 1)
- With functions (Task 3)
- Analyze them based on:
  - Code clarity
  - Reusability
  - Debugging ease
  - Suitability for large-scale applications
- ❖ Expected Output

Comparison table or short analytical report

# Basic Prime Check Approach

```
n = int(input("Enter number: "))
```

```
count = 0
```

```
for i in range(1, n + 1):
```

```
    if n % i == 0:
```

```
        count += 1
```

```
if count == 2:
```

```
    print("Prime Number")
```

```
else:
```

```
    print("Not Prime Number")
```

The screenshot shows the OnlineGDB IDE interface. On the left, there's a sidebar with navigation links like 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Opportunity for Intern', 'Sign Up', and 'Login'. A 'auth0' logo is also present. The main area has a dark theme with a light-colored code editor. The code editor window is titled 'main.py' and contains the following Python script:

```
# Basic Prime Check Approach
n = int(input("Enter number: "))
count = 0
for i in range(1, n + 1):
    if n % i == 0:
        count += 1
if count == 2:
    print("Prime Number")
else:
    print("Not Prime Number")
```

Below the code editor, there's an 'input' field containing '6' and the output console showing 'Not Prime Number'. At the bottom, it says '...Program finished with exit code 0 Press ENTER to exit console.'

## Explanation

- Counts total divisors
- Prime numbers have exactly 2 divisors

## Complexity

- $O(n)$  very slow

Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to Prime Checking)

❖ Scenario

Your mentor wants to evaluate how AI handles alternative logical strategies.

❖ Task Description

Prompt GitHub Copilot to generate:

- A basic divisibility check approach
- An optimized approach (e.g., checking up to  $\sqrt{n}$ )

❖ Expected Output

- Two correct implementations

➢ Comparison discussing:

- Execution flow
- Time complexity
- Performance for large inputs
- When each approach is appropriate

# Optimized Prime Check Approach

```
n = int(input("Enter number: "))
```

```
if n <= 1:
```

```

print("Not Prime")

else:

for i in range(2, int(n**0.5) + 1):

    if n % i == 0:

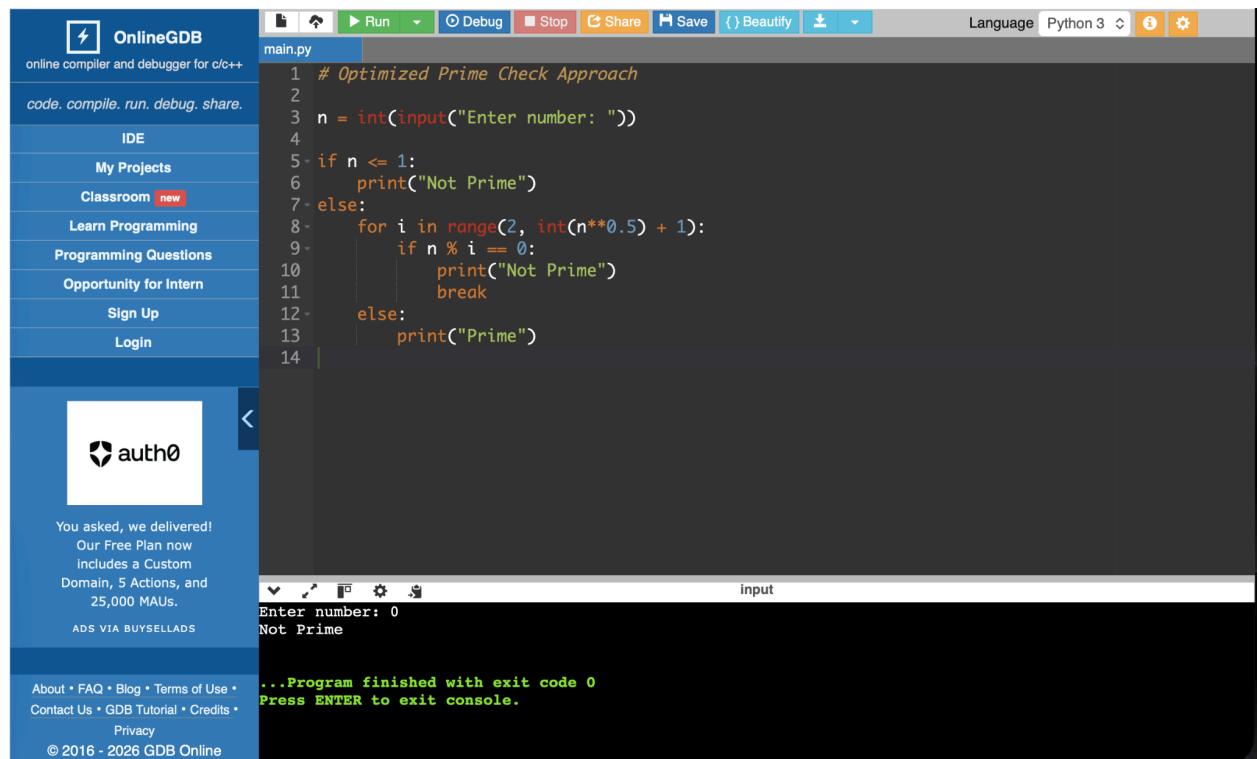
        print("Not Prime")

        break

else:

    print("Prime")

```



The screenshot shows the OnlineGDB IDE interface. On the left, there's a sidebar with navigation links like 'IDE', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Opportunity for Intern', 'Sign Up', and 'Login'. A banner from auth0 is displayed, stating: 'You asked, we delivered! Our Free Plan now includes a Custom Domain, 5 Actions, and 25,000 MAUs.' At the bottom of the sidebar, there are links for 'About', 'FAQ', 'Blog', 'Terms of Use', 'Contact Us', 'GDB Tutorial', 'Credits', 'Privacy', and '© 2016 - 2026 GDB Online'.

The main window has a toolbar at the top with 'Run', 'Debug', 'Stop', 'Save', 'Beautify', and other icons. The language is set to 'Python 3'. The code editor contains the following Python script:

```

main.py
1 # Optimized Prime Check Approach
2
3 n = int(input("Enter number: "))
4
5 if n <= 1:
6     print("Not Prime")
7 else:
8     for i in range(2, int(n**0.5) + 1):
9         if n % i == 0:
10             print("Not Prime")
11             break
12 else:
13     print("Prime")
14

```

The terminal below the code editor shows the output of running the script with the input '0', which prints 'Not Prime'. The status bar at the bottom indicates '...Program finished with exit code 0'.

## Explanation

- Counts total divisors
- Prime numbers have exactly 2 divisors

## Complexity

- $O(n)$  very slow