

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 is a sidebar with navigation links: IDE, My Projects, Classroom (marked 'new'), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. Below these is an Auth0 login section and a footer with links like About, FAQ, Blog, Terms of Use, Contact Us, GDB Tutorial, Credits, Privacy, and a copyright notice for 2016-2026 GDB Online. The main editor area displays a Python file named 'main.py' with the following code:

```
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 console at the bottom shows the execution output: 'Enter a number: 5' followed by '5 is a prime number'. A status bar at the bottom indicates '...Program finished with exit code 0' and 'Press ENTER to exit console.'

Explanation

Explanation

- User enters a number n
- If $n \leq 1$, it is not prime
- Loop runs from 2 to $n-1$
- If divisible by any number \rightarrow 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 IDE interface. On the left is a sidebar with navigation links: IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. Below these is an Auth0 login section and a footer with links like About, FAQ, Blog, Terms of Use, etc. The main editor area displays a Python file 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, the input/output console shows the user entering '7' and the program outputting '7 is a prime number'. At the bottom, a status bar indicates '...Program finished with exit code 0' and 'Press ENTER to exit console.'

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
```

```
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
```

Enter a number: 8
8 is not a prime number

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

**OnlineGDB**

online compiler and debugger for c/c++

code.compile.run.debug.share.

IDE

My Projects

Classroom new


Learn Programming

Programming Questions

Opportunity for Intern

Sign Up

Login



You asked, we delivered!
Our Free Plan now includes a Custom Domain, 5 Actions, and 25,000 MAUs.

ADS VIA BUYSSELLADS

About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy

© 2016 - 2026 GDB Online

Run Debug Stop Share Save Beautify

Language Python 3


main.py

```
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:
```

input

Enter a number: 8
8 is not a prime number

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

**OnlineGDB**

online compiler and debugger for c/c++

code.compile.run.debug.share.

IDE

My Projects

Classroom new


Learn Programming

Programming Questions

Opportunity for Intern

Sign Up

Login



You asked, we delivered!
Our Free Plan now includes a Custom Domain, 5 Actions, and 25,000 MAUs.

ADS VIA BUYSSELLADS

About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy

© 2016 - 2026 GDB Online

Run Debug Stop Share Save Beautify

Language Python 3

main.py

```
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")
26
```

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")
```

```
1 # Basic Prime Check Approach
2
3 n = int(input("Enter number: "))
4
5 count = 0
6
7 for i in range(1, n + 1):
8     if n % i == 0:
9         count += 1
10
11 if count == 2:
12     print("Prime Number")
13 else:
14     print("Not Prime Number")
15
```

Enter number: 6
Not Prime Number

...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):
```

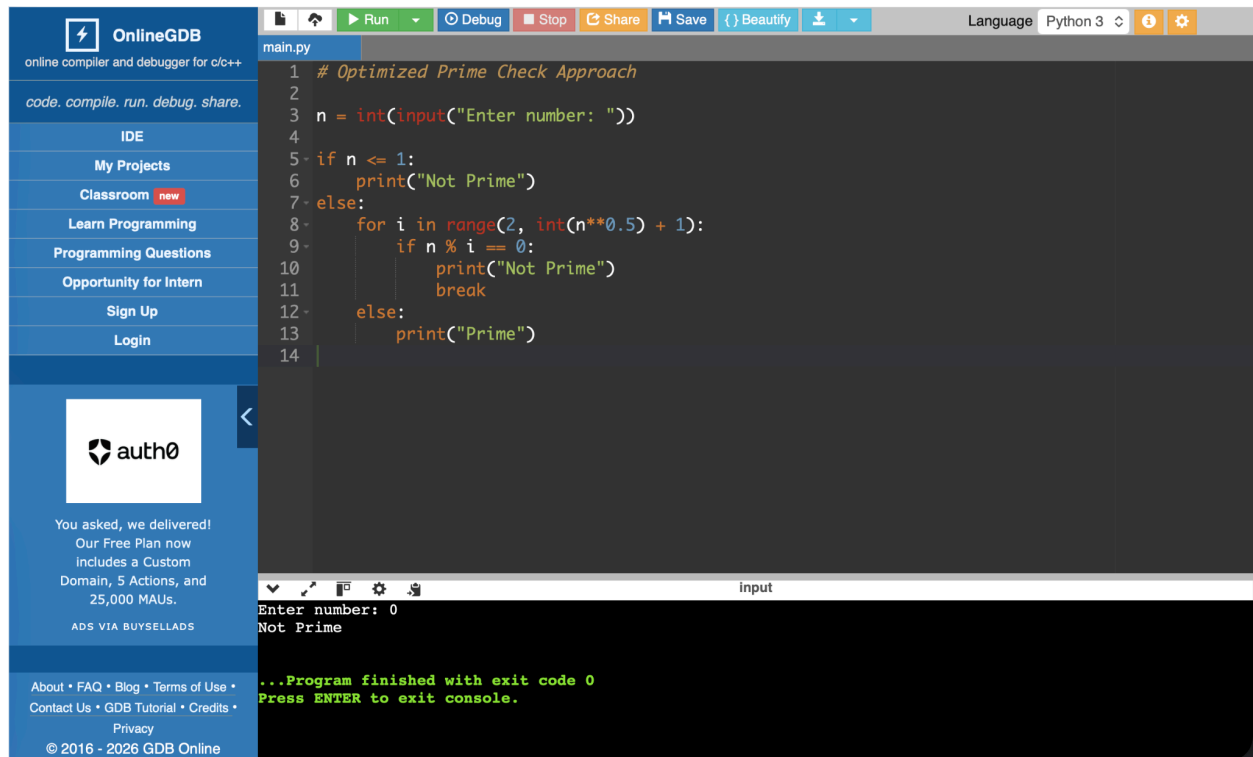
```
    if n % i == 0:
```

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

```
        break
```

else:

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



The screenshot shows the OnlineGDB IDE interface. On the left is a sidebar with navigation links: OnlineGDB, code.compile.run.debug.share, IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Opportunity for Intern, Sign Up, and Login. Below this is an auth0 login section and a footer with links like About, FAQ, Blog, Terms of Use, Contact Us, GDB Tutorial, Credits, Privacy, and a copyright notice for 2016-2026 GDB Online. The main editor area displays a Python file named main.py with the following code:

```
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
```

Below the code editor is an input field with the text "Enter number: 0" and a console output area showing "Not Prime". At the bottom, a status bar indicates "...Program finished with exit code 0" and "Press ENTER to exit console."

Explanation

- Counts total divisors
- Prime numbers have exactly 2 divisors

Complexity

- **$O(n)$** very slow