2303A51563

batch=10

Task 1: Auto-Generating Function Documentation in a Shared Codebase

Scenario

You have joined a development team where several utility functions are already implemented, but the code lacks proper documentation. New team members are struggling to understand how these functions should be used.

Task Description

You are given a Python script containing multiple functions without any docstrings.

Using an AI-assisted coding tool:

• Ask the AI to automatically generate Google-style function docstrings for each function

• Each docstring should include:

o A brief description of the function

o Parameters with data types

o Return values

o At least one example usage (if applicable)

Experiment with different prompting styles (zero-shot or context-based) to observe quality differences.

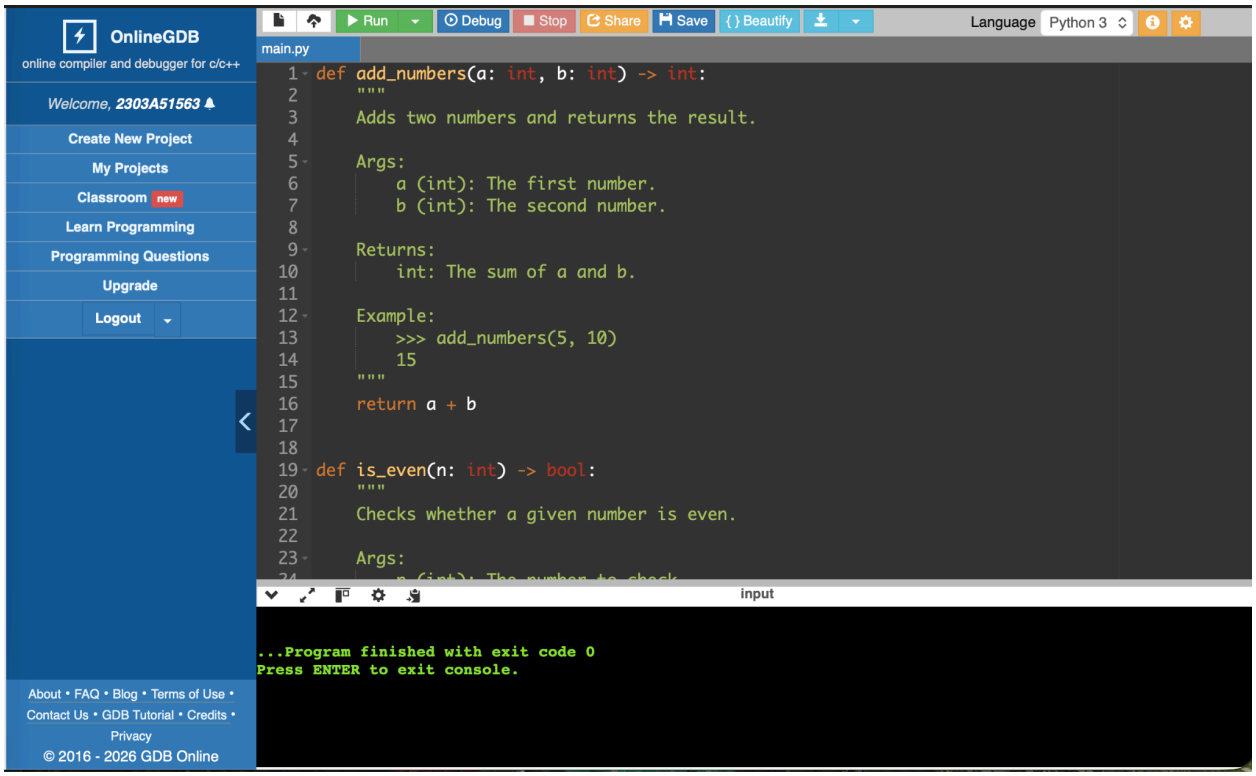
Expected Outcome

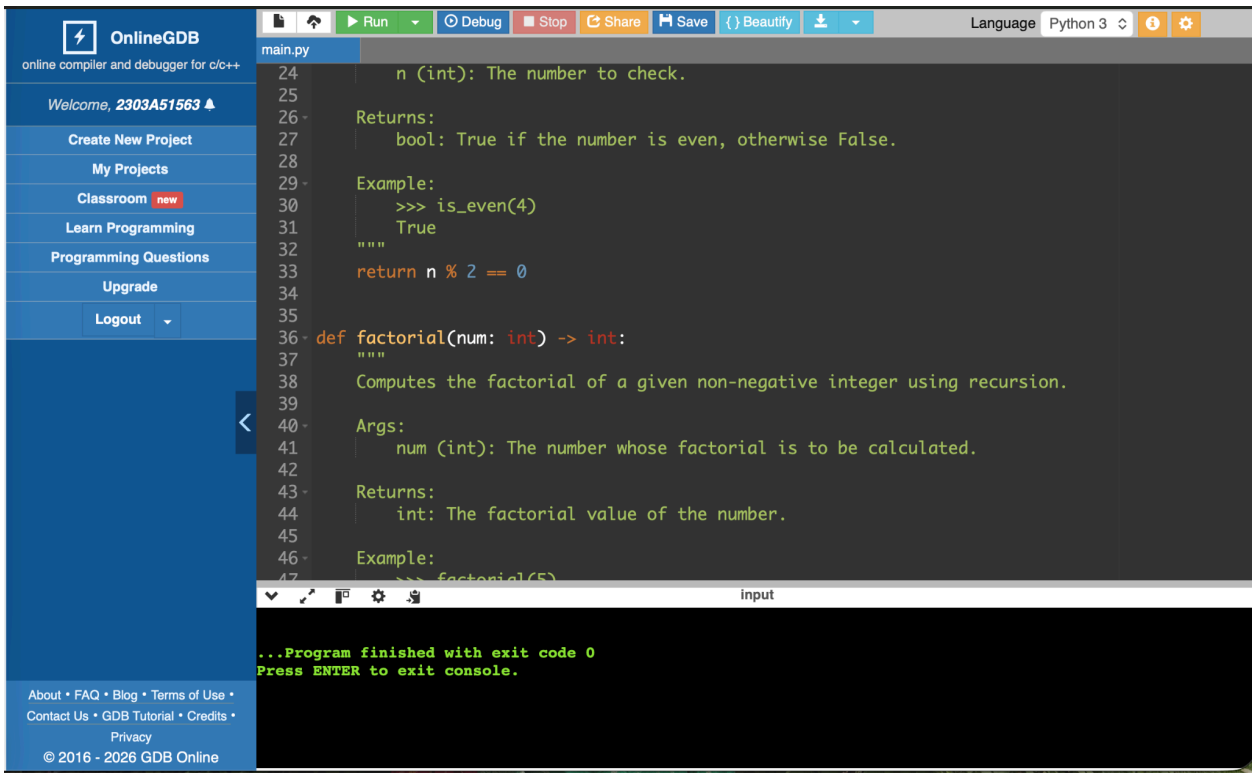
• A Python script with well-structured Google-style docstrings

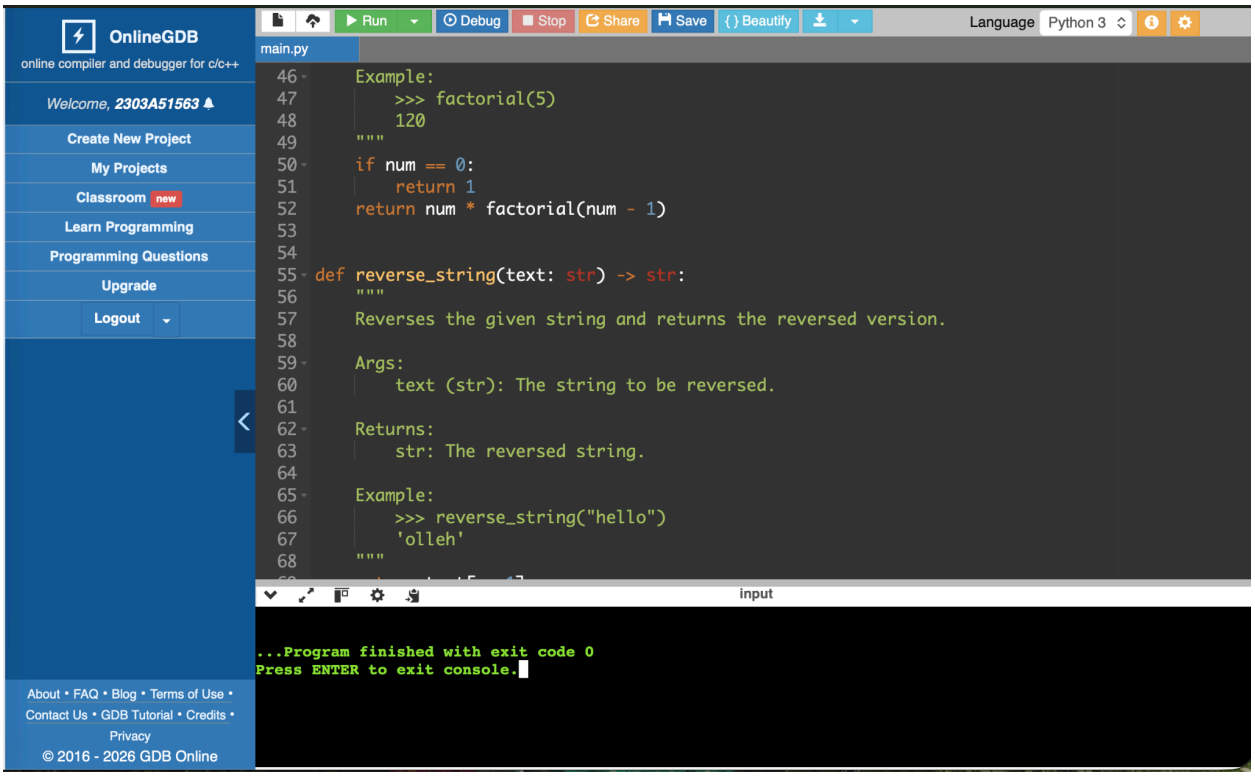
• Docstrings that clearly explain function behavior and usage

• Improved readability and usability of the codebase

Code


OnlineGDB
online compiler and debugger for c/c++
Welcome, 2303A51563
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Upgrade
Logout
Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
def add_numbers(a: int, b: int) -> int:
"""
Adds two numbers and returns the result.
Args:
a (int): The first number.
b (int): The second number.
Returns:
int: The sum of a and b.
Example:
>>> add_numbers(5, 10)
15
"""
return a + b
def is_even(n: int) -> bool:
"""
Checks whether a given number is even.
Args:
input
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2026 GDB Online


OnlineGDB
online compiler and debugger for c/c++
Welcome, 2303A51563
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Upgrade
Logout
Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
n (int): The number to check.
Returns:
bool: True if the number is even, otherwise False.
Example:
>>> is_even(4)
True
"""
return n % 2 == 0
def factorial(num: int) -> int:
"""
Computes the factorial of a given non-negative integer using recursion.
Args:
num (int): The number whose factorial is to be calculated.
Returns:
int: The factorial value of the number.
Example:
input
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2026 GDB Online



Task 2: Enhancing Readability Through AI-Generated Inline Comments

Scenario

A Python program contains complex logic that works correctly but is difficult to understand at first glance. Future maintainers may find it hard to debug or extend this code.

Task Description

You are provided with a Python script containing:

• Loops

• Conditional logic

• Algorithms (such as Fibonacci sequence, sorting, or searching)

Use AI assistance to:

• Automatically insert inline comments only for complex or non-obvious logic

• Avoid commenting on trivial or self-explanatory syntax

The goal is to improve clarity without cluttering the code.

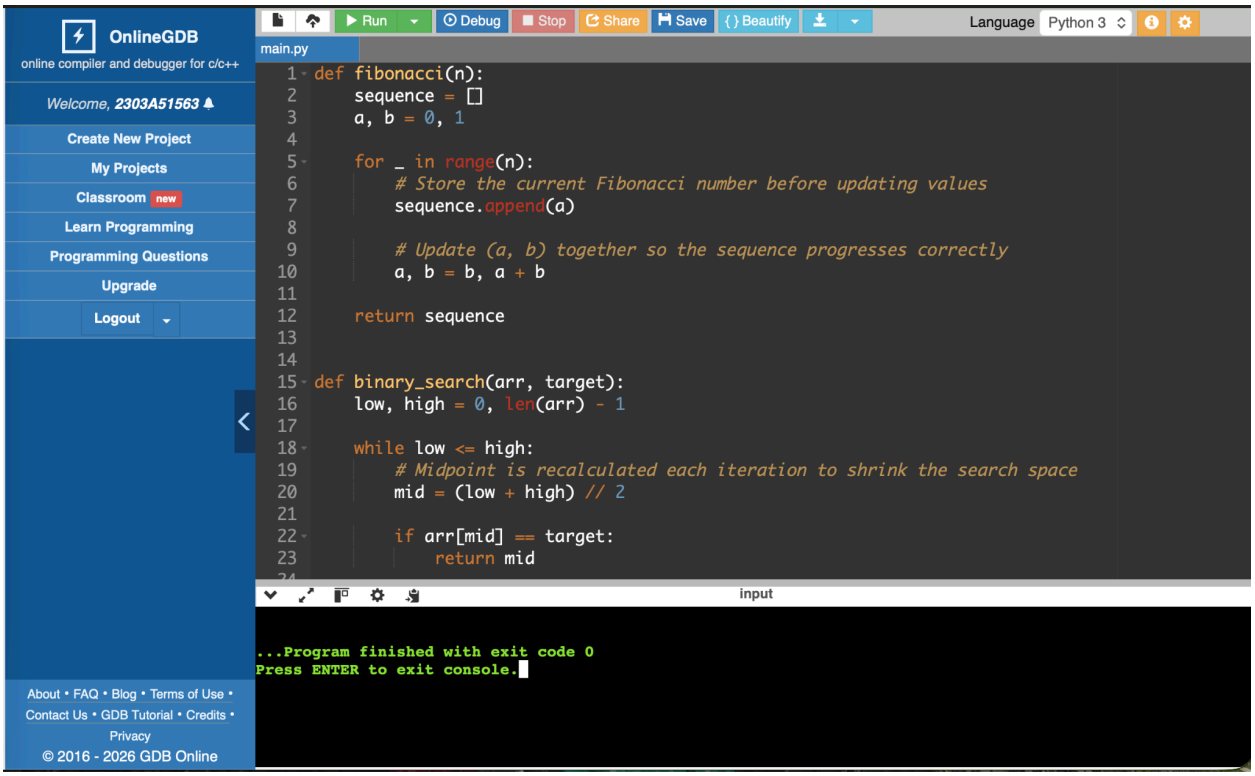
Expected Outcome

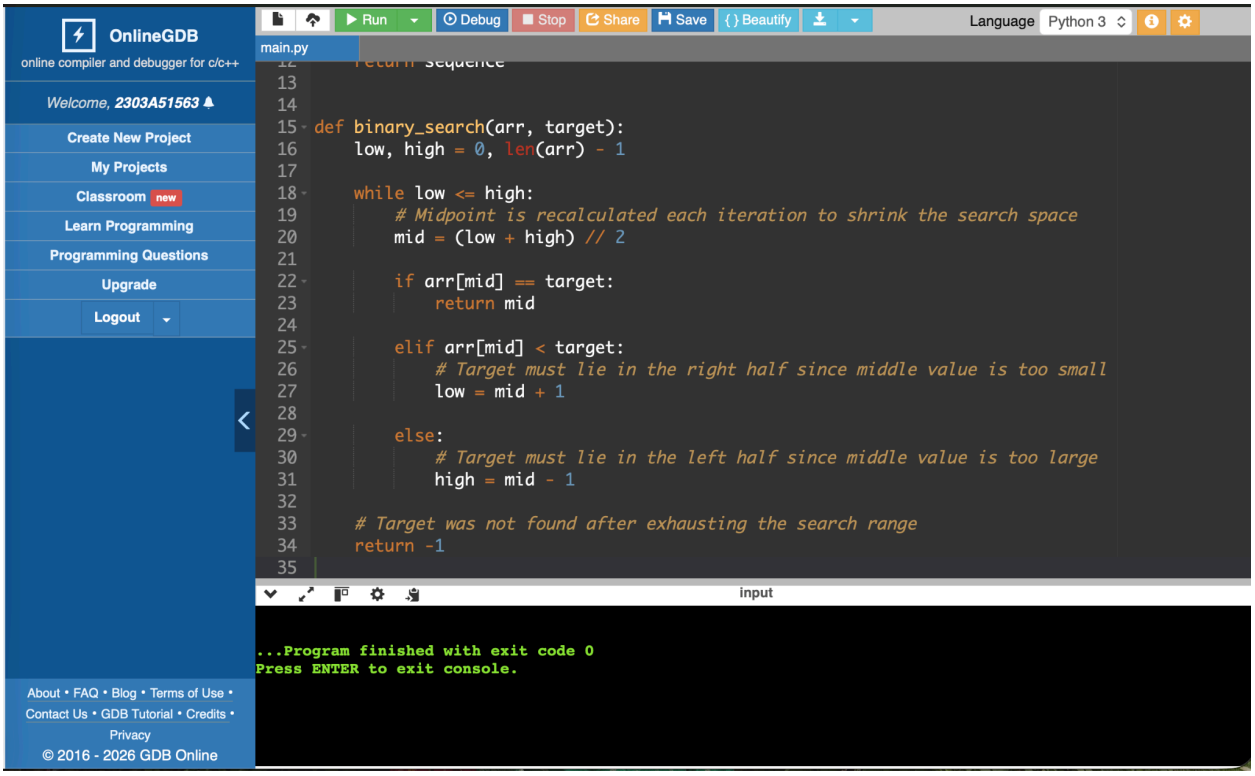
• A Python script with concise, meaningful inline comments

• Comments that explain why the logic exists, not what Python syntax does

• Noticeable improvement in code readability

Code





ask 3: Generating Module-Level Documentation for a Python Package

Scenario

Your team is preparing a Python module to be shared internally (or

uploaded to a repository). Anyone opening the file should immediately understand its purpose and structure.

Task Description

Provide a complete Python module to an AI tool and instruct it to automatically generate a module-level docstring at the top of the file that includes:

• The purpose of the module
• Required libraries or dependencies
• A brief description of key functions and classes
• A short example of how the module can be used

Focus on clarity and professional tone.

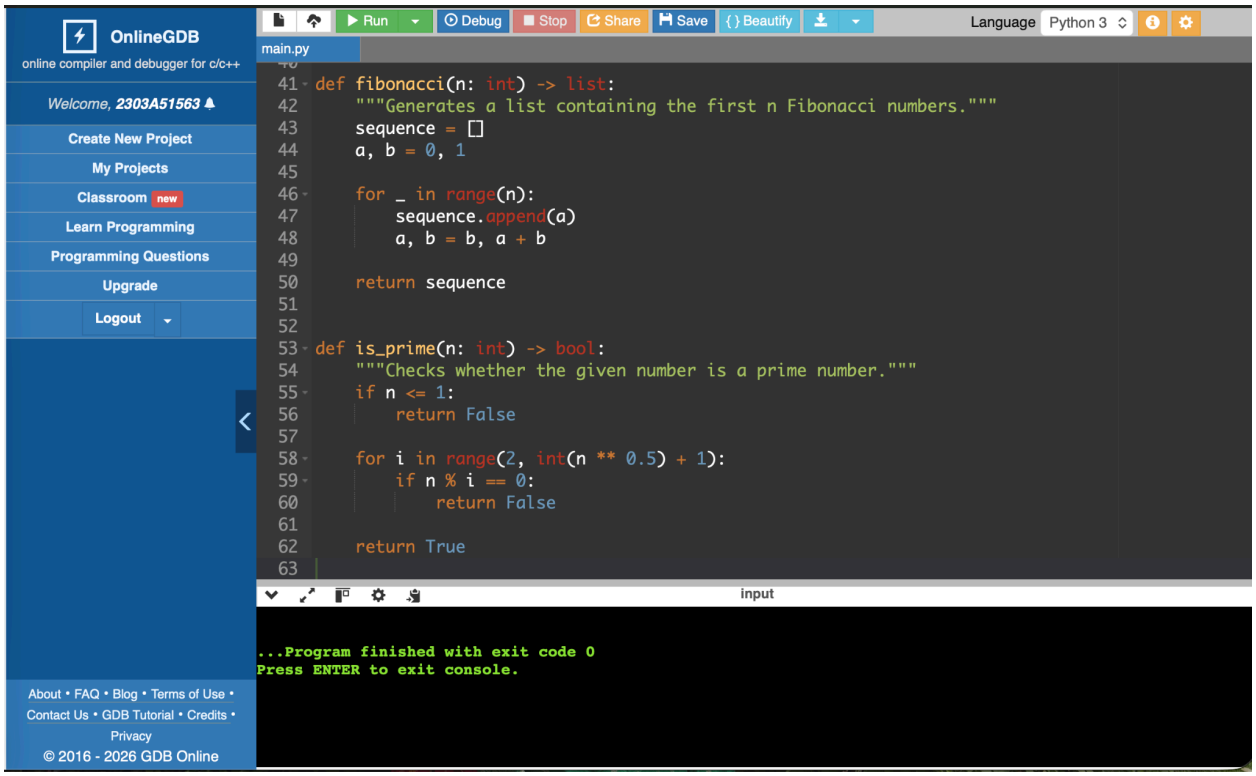
Expected Outcome

• A well-written multi-line module-level docstring
• Clear overview of what the module does and how to use it
• Documentation suitable for real-world projects or repositories

Code

```
"""
math_utils.py

This module provides a collection of commonly used mathematical utility
functions that can be reused across multiple Python projects.

Purpose:
    The goal of this module is to simplify basic mathematical operations
    such as factorial calculation, Fibonacci sequence generation, and
    prime number checking.

Dependencies:
    - No external libraries are required.
    - Works with Python 3.x standard library only.

Key Functions:
    - factorial(n): Computes the factorial of a given integer.
    - fibonacci(n): Generates the first n Fibonacci numbers.
    - is_prime(n): Checks whether a number is prime.

Example Usage:
    >>> from math_utils import factorial, fibonacci, is_prime
    >>> factorial(5)
    120

    >>> fibonacci(6)
    [0, 1, 1, 2, 3, 5]
```

```
    >>> is_prime(11)
    True
"""


def factorial(n: int) -> int:
    """Returns the factorial of a non-negative integer."""
    if n == 0:
        return 1
    return n * factorial(n - 1)


def fibonacci(n: int) -> list:
    """Generates a list containing the first n Fibonacci numbers."""
    sequence = []
    a, b = 0, 1

    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b

    return sequence


def is_prime(n: int) -> bool:
    """Checks whether the given number is a prime number."""
    if n <= 1:
        return False

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

    return True
```


OnlineGDB
online compiler and debugger for c/c++
Welcome, 2303A51563
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Upgrade
Logout
Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
41 def fibonacci(n: int) -> list:
42 """Generates a list containing the first n Fibonacci numbers."""
43 sequence = []
44 a, b = 0, 1
45
46 for _ in range(n):
47 sequence.append(a)
48 a, b = b, a + b
49
50 return sequence
51
52
53 def is_prime(n: int) -> bool:
54 """Checks whether the given number is a prime number."""
55 if n <= 1:
56 return False
57
58 for i in range(2, int(n ** 0.5) + 1):
59 if n % i == 0:
60 return False
61
62 return True
63
input
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2026 GDB Online