

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:
 - A brief description of the function
 - Parameters with data types
 - Return values
 - 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

The screenshot shows the OnlineGDB web-based IDE interface. The top navigation bar includes links for Run, Debug, Stop, Share, Save, and Beautify. The language is set to Python 3. The main workspace displays Python code in a file named 'main.py'. The code defines two functions: 'add_numbers' which adds two integers and returns their sum, and 'is_even' which checks if a number is even. Below the code, the terminal output shows the program finished with exit code 0 and prompts the user to press ENTER to exit. The left sidebar contains links for Welcome, Create New Project, My Projects, Classroom, Learn Programming, Programming Questions, Upgrade, and Logout.

```
1 def add_numbers(a: int, b: int) -> int:
2     """
3         Adds two numbers and returns the result.
4     """
5     Args:
6         a (int): The first number.
7         b (int): The second number.
8
9     Returns:
10    int: The sum of a and b.
11
12    Example:
13    >>> add_numbers(5, 10)
14    15
15    """
16    return a + b
17
18
19 def is_even(n: int) -> bool:
20     """
21         Checks whether a given number is even.
22     """
23     Args:
24         n (int): The number to check.
...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

The screenshot shows the OnlineGDB interface. The left sidebar includes links for 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area displays a Python script 'main.py' with code examples and a terminal output.

```
main.py
24     n (int): The number to check.
25
26     Returns:
27         bool: True if the number is even, otherwise False.
28
29     Example:
30         >>> is_even(4)
31         True
32     """
33     return n % 2 == 0
34
35
36 def factorial(num: int) -> int:
37     """
38         Computes the factorial of a given non-negative integer using recursion.
39
40     Args:
41         num (int): The number whose factorial is to be calculated.
42
43     Returns:
44         int: The factorial value of the number.
45
46     Example:
47         >>> factorial(5)

```

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

The screenshot shows the OnlineGDB interface. On the left, there's a sidebar with navigation links like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area has tabs for 'main.py' and 'input'. The 'main.py' tab contains the following Python code:

```
46 -     Example:
47 -         >>> factorial(5)
48 -         120
49 -         """
50 -     if num == 0:
51 -         return 1
52 -     return num * factorial(num - 1)
53 -
54
55 - def reverse_string(text: str) -> str:
56 -     """
57 -     Reverses the given string and returns the reversed version.
58 -
59 -     Args:
60 -         text (str): The string to be reversed.
61 -
62 -     Returns:
63 -         str: The reversed string.
64 -
65 -     Example:
66 -         >>> reverse_string("hello")
67 -         'olleh'
68 -         """
69
70
71
72
73
74
```

The 'input' tab is empty. At the bottom, the terminal window shows the output of the program:

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

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

The screenshot shows the OnlineGDB interface with a Python script named `main.py`. The code defines two functions: `fibonacci` and `binary_search`.

```
1 def fibonacci(n):
2     sequence = []
3     a, b = 0, 1
4
5     for _ in range(n):
6         # Store the current Fibonacci number before updating values
7         sequence.append(a)
8
9         # Update (a, b) together so the sequence progresses correctly
10        a, b = b, a + b
11
12    return sequence
13
14
15 def binary_search(arr, target):
16     low, high = 0, len(arr) - 1
17
18     while low <= high:
19         # Midpoint is recalculated each iteration to shrink the search space
20         mid = (low + high) // 2
21
22         if arr[mid] == target:
23             return mid
24
25
26
27
28
29
30
31
32
33
34
35
```

The output window shows the program finished with exit code 0.

The screenshot shows the OnlineGDB interface with the same Python script `main.py`. The `binary_search` function has been modified to handle cases where the target value is not found.

```
12    return sequence
13
14
15 def binary_search(arr, target):
16     low, high = 0, len(arr) - 1
17
18     while low <= high:
19         # Midpoint is recalculated each iteration to shrink the search space
20         mid = (low + high) // 2
21
22         if arr[mid] == target:
23             return mid
24
25         elif arr[mid] < target:
26             # Target must lie in the right half since middle value is too small
27             low = mid + 1
28
29         else:
30             # Target must lie in the left half since middle value is too large
31             high = mid - 1
32
33
34     # Target was not found after exhausting the search range
35     return -1
```

The output window shows the program finished with exit code 0.

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 ▾

```
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