Code Understanding Report

Generated: 2025-05-06 16:15:30

This report presents automated insights based on large language models and code analysis tools.

File: pasted_code.py

Summary

- - This code defines a decorator `log_calls`. It takes in a function `func`. - The `wrapper` is a function that prints out the name of the function and its arguments, then calls the original function with those arguments, and then prints out the result. - The `return wrapper` statement makes the `log_calls` decorator return the `wrapper` function each time it's called, so you can call it like a function. - Example usage: ``` @log_calls def add(x, y): return x + - This Python code snippet defines a function that adds two numbers.

Docstring

- ### Code: def log_calls(func): def wrapper(*args, *kwargs): print(f"Calling {func.**name**} with arguments: {args}, {kwargs}") result = func(*args, *kwargs) print(f"{func.**name**} returned: {result}") return result return wrapper

Docstring:

This function is a decorator that logs the calls to a function.

Parameters: - `func`: The function to be decorated.

Returns: - `wrapper`: A new function that logs the calls to - ### Code: def add(a, b): return a + b

Docstring:

''' Adds two numbers.

:param a: The first number. :param b: The second number. :return: The sum of a and b. '''

Test Cases:

Code Quality

Tool: `pylint`
Issues: 0`

```
text No issues
```

Conclusion

This Python code is designed to add two numbers. The `log_calls` decorator logs the function name and its arguments before it calls the function, and then it logs the result of the function after it returns.

Example: ``` @log_calls def add(a, b): return a + b

print(add(2, 3))

Output:

Calling add with arguments: (2, 3), {}

add returned: 5

5

```
```