Code Understanding Report

Generated: 2025-05-06 17:05:17

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

File: pasted_code.py

Summary

- The code defines a decorator "log_calls" which logs the name and arguments of the function it decorates. The `wrapper` function executes the original function and logs its results. This decorator is useful for debugging by recording the calls to the decorated function.

Here is an example of usage:

```` ```python @log_calls def add(x, y): return x + y ````

result = add(5, 7)

print(result) # Should print "Calling add with arguments: (5, 7), {} returned: - This code defines a function called `add` which takes two parameters `a` and `b` and returns the sum of these two parameters.

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

The `log_calls` decorator is useful for debugging by recording the calls to the decorated function. '''