Code Understanding Report

Generated: 2025-05-05 23:52:20

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

File: pasted_code.js

Summary

- This decorator is a form of function factory. The @repeat decorator creates a wrapper function that repeats the execution of another function. This concept is frequently used in function decorators to perform certain actions on the wrapped function before or after it is called. Here, num_times is the number of times the function is to be repeated.

Docstring

- ### Code: from functools import wraps

def repeat(num*times): def decorator*repeat(func): @wraps(func) def wrapper(*args, *kwargs): for _ in range(num*times): result = func(*args, **kwargs) return result return wrapper return *decorator*repeat

@repeat(num_times=3) def greet(name): print(f"Hello, {name}!")

greet("Alice"

Docstring:

The `repeat` function is a decorator that takes a number of times to repeat a function. The `@repeat(num_times=3)` syntax is used to apply the decorator to a function.

The `

Code Quality

Tool: `eslint`
Issues: 0`

```
text [ESLint Not Found] [WinError 2] The system cannot find the file specified — assuming valid JS.
```

Conclusion

This code is a simple example of function decorators, where a decorator is a function that wraps another function and adds some functionality to it. In this case, the decorator `@repeat(num_times=3)` repeats the execution of another function. The `repeat` decorator is commonly used in function decorators to perform certain actions on the wrapped function before or after it is called.

In this example, the function `greet` is wrapped by the `@repeat(num_times=3)` decorator, which means that `greet` will be repeated execution 3 times before being called. In the example, this code will print "Hello, Alice!" 3 times.