## Overview

This file serves as the main entry point for the code\_monitor package when it is executed as a module (e.g., python -m code\_monitor). Its primary responsibility is to invoke the main analysis function, analyze, from the .main submodule.

The script is designed to work with the click library, including a mechanism to gracefully handle SystemExit exceptions that click might raise after displaying help messages or upon normal command completion.

## Functions

### main()

This function is the primary entry point for the command-line execution of the code\_monitor package. It calls the analyze function, which contains the core logic for the tool.

The analyze function is called with standalone\_mode=False. This is a convention used by click to manage how command-line applications are invoked, especially when dealing with command groups. It tells the underlying click application that it’s not being run in a standalone mode, which can affect context management.

* **Parameters**: None
* **Returns**: None

## Script Execution

The if \_\_name\_\_ == "\_\_main\_\_": block handles the direct execution of the script. It calls the main() function within a try...except block. This block specifically catches SystemExit exceptions, which are commonly raised by click after processing commands like --help or upon successful completion. The code ensures that only SystemExit exceptions with non-zero exit codes (indicating an error) are re-raised, preventing benign exits from producing traceback messages.

### Usage Example

To run the code monitor tool from your terminal, you would execute the package as a module. This command triggers this \_\_main\_\_.py script.

# Execute the code monitor package from the command line  
python -m code\_monitor [OPTIONS] [PATHS]...