## Overview  
  
This script provides a command-line interface (CLI) for analyzing and reporting changes at the function and class level within a Git repository. It can analyze all uncommitted changes or only staged changes, making it suitable for use as a pre-commit hook. The tool uses the `RepoAnalyzer` class to perform the core analysis and `click` to build the CLI. The output is color-coded using `colorama` for better readability.  
  
## Functions  
  
### `add\_numbers()`  
  
A simple utility function that adds two integers and returns their sum. This function primarily serves as an example or a basic utility within the module.  
  
\* \*\*Parameters:\*\*  
 \* `a` (int): The first integer.  
 \* `b` (int): The second integer.  
\* \*\*Returns:\*\*  
 \* `int`: The sum of `a` and `b`.  
  
\* \*\*Usage Example:\*\*  
 ```python  
 >>> add\_numbers(3, 5)  
 8  
 ```  
  
### `print\_analysis()`  
  
Formats and prints the analysis results to the console. It uses colors to distinguish between added, modified, and deleted code elements (functions/classes) and provides a summary of the changes for each file.  
  
\* \*\*Parameters:\*\*  
 \* `results` (dict): A dictionary containing the analysis results. The keys are file paths, and the values are dictionaries detailing the changes in that file. Each inner dictionary has the following structure:  
 \* `status` (str): The Git status of the file ('A' for added, 'M' for modified, 'D' for deleted).  
 \* `added` (list): A list of dictionaries for each added function or class.  
 \* `removed` (list): A list of dictionaries for each removed function or class.  
 \* `modified` (list): A list of dictionaries for each modified function or class.  
\* \*\*Returns:\*\*  
 \* `None`. This function prints directly to standard output.  
  
### `analyze()`  
  
The main entry point for the CLI tool. This function is decorated with `click` options to handle command-line arguments. It initializes the `RepoAnalyzer`, triggers the analysis of the Git repository, and then passes the results to `print\_analysis` for display.  
  
\* \*\*Parameters (as CLI options):\*\*  
 \* `path` (str): The path to the Git repository to be analyzed. Corresponds to the `--path` option. Defaults to the current directory (`.`).  
 \* `staged\_only` (bool): A flag that, when set, restricts the analysis to only staged files (i.e., changes added to the Git index). Corresponds to the `--staged-only` flag. Defaults to `False`.  
  
\* \*\*Returns:\*\*  
 \* `None`.  
  
\* \*\*Usage Example (from the command line):\*\*  
  
 ```bash  
 # Analyze all uncommitted changes in the current directory  
 python -m code\_monitor.main  
  
 # Analyze only staged changes in a specific repository path  
 python -m code\_monitor.main --path /path/to/your/repo --staged-only  
 ```