Table of Contents

python # Example of adding two positive numbers sum\_result = add\_numbers(3, 5) print(f”The sum of 3 and 5 is: {sum\_result}“) # Expected output: The sum of 3 and 5 is: 8

# 1 Example of adding a positive and a negative number

sum\_result\_negative = add\_numbers(10, -4) print(f”The sum of 10 and -4 is: {sum\_result\_negative}“) # Expected output: The sum of 10 and -4 is: 6

python difference = sub\_numbers(5, 3) print(difference) # Expected output: 2

difference\_with\_negative = sub\_numbers(10, 15) print(difference\_with\_negative) # Expected output: -5

python # Assuming the function is defined as print\_results(results) # and click/colorama are imported import click from colorama import Fore, Style

def print\_results(results): “““Prints the analysis results in a readable format.”“” if not results: click.echo(Fore.GREEN + “No relevant code changes detected.”) return

for file\_path, data in results.items():  
 status\_color = {  
 'A': Fore.GREEN, 'M': Fore.YELLOW, 'D': Fore.RED,  
 }.get(data['status'], Fore.WHITE)  
 click.echo(status\_color + f"\n--- Changes in {file\_path} [{data['status']}] ---")  
  
 if data['added']:  
 click.echo(Fore.GREEN + " [+] Added:")  
 for item in data['added']:  
 click.echo(f" - {item['type']}: {item['name']} (lines {item['start\_line']}-{item['end\_line']})")  
   
 if data['removed']:  
 click.echo(Fore.RED + " [-] Removed:")  
 for item in data['removed']:  
 click.echo(f" - {item['type']}: {item['name']}")  
  
 if data['modified']:  
 click.echo(Fore.YELLOW + " [\*] Modified:")  
 for item in data['modified']:  
 click.echo(f" - {item['type']}: {item['name']} (lines {item['start\_line']}-{item['end\_line']})")  
  
summary\_msg = f"\nSummary: Analyzed {len(results)} files."  
click.echo(Style.BRIGHT + summary\_msg)

# 2 Example usage:

analysis\_results = { “src/utils.py”: { “status”: “M”, “added”: [], “removed”: [ {“type”: “function”, “name”: “old\_helper”} ], “modified”: [ {“type”: “function”, “name”: “calculate\_metrics”, “start\_line”: 25, “end\_line”: 50} ] }, “src/new\_feature.py”: { “status”: “A”, “added”: [ {“type”: “class”, “name”: “FeatureHandler”, “start\_line”: 10, “end\_line”: 45}, {“type”: “function”, “name”: “initialize\_feature”, “start\_line”: 48, “end\_line”: 60} ], “removed”: [], “modified”: [] } }

print\_results(analysis\_results)

# 3 Example with no results

# 4 print\_results({})

# 5 Expected Output: No relevant code changes detected.

python from code\_monitor.main import analyze import os

# 6 Assume ‘/path/to/my-project’ is a Git repository with changes.

project\_path = “/path/to/my-project”

if os.path.exists(project\_path): # Example 1: Analyze all uncommitted changes (staged and unstaged) print(“— Analyzing all uncommitted changes —”) try: analyze(path=project\_path, staged\_only=False) except SystemExit as e: print(f”Analysis exited with code {e.code}“)

# Example 2: Analyze only staged changes (useful for pre-commit hooks)  
print("\n--- Analyzing only staged changes ---")  
try:  
 analyze(path=project\_path, staged\_only=True)  
except SystemExit as e:  
 print(f"Analysis exited with code {e.code}")

else: print(f”Repository path does not exist: {project\_path}“)