Git Repository Cloner and Code Analysis Tool Document

Author: Aman Bhatt

Date: January 24, 2023

Description:

This script allows you to clone a Git repository from a specified URL and perform static code analysis using Flake8 on the cloned files. It is designed to help developers quickly assess the code quality of a remote repository by analyzing its Python files.

Usage:

1. Ensure that you have the 'git' command-line tool and Python 3.10 installed on your system.

2. Run this script using Python 3.10 or a later version.

Dependencies:

- git (command-line tool)

- Python 3.10

- Flake8 (Python linting tool)

Modules Used:

- git: Provides functionality to interact with Git repositories.

- os: Allows interaction with the operating system, file paths, and directories.

- subprocess: Enables running shell commands from Python.

#Code:

import git

import os

import subprocess

# Define the URL of the Git repository you want to clone

repo\_url = 'https://github.com/geekcomputers/Python.git'

# Get the path to your desktop

desktop\_path = os.path.expanduser('~/Desktop')

# Define the local path where you want to store the cloned repository on your desktop

local\_path = os.path.join(desktop\_path, 'repository')

try:

# Clone the repository

git.Repo.clone\_from(repo\_url, local\_path)

print('Repository cloned successfully!')

except git.exc.GitCommandError as e:

if 'destination path' in str(e):

print('Repository already exists. Parsing through existing files...')

# Iterate over the files in the cloned repository

for root, dirs, files in os.walk(local\_path):

for file in files:

file\_path = os.path.join(root, file)

# Get the relative path of the file

relative\_path = os.path.relpath(file\_path, local\_path)

# Perform your desired operations on the file

# For example, print the relative path

print("Relative Path:", relative\_path)

try:

# Perform static code analysis using Flake8

flake8\_output = subprocess.check\_output(

['/Library/Frameworks/Python.framework/Versions/3.10/bin/python3', '-m', 'flake8', file\_path],

stderr=subprocess.STDOUT

).decode('utf-8')

print("Flake8 Output:")

print(flake8\_output)

except subprocess.CalledProcessError as e:

# Handle the non-zero exit status error

print(f"Error analyzing file: {file\_path}")

print(f"Error message: {e.output.decode('utf-8')}")

#End of script.