**Assignment**

**CSA0805 – Python Programming**

|  |  |
| --- | --- |
| **Register Number** | **192324267** |
| **Name** | **VEDVYASS M** |

**Title: File System Watcher**

**Problem Statement:** Design a Python program that monitors a directory for changes, such as file creations, deletions, or modifications, and logs these events in real-time, optionally triggering actions based on predefined rules

**Code:**

**import os**

**import time**

**import hashlib**

**import logging**

**# Configure logging**

**logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(message)s', datefmt='%Y-%m-%d %H:%M:%S')**

**logger = logging.getLogger(\_\_name\_\_)**

**def get\_file\_hash(file\_path):**

**"""Calculate the MD5 hash of a file."""**

**md5\_hash = hashlib.md5()**

**with open(file\_path, 'rb') as f:**

**for chunk in iter(lambda: f.read(4096), b""):**

**md5\_hash.update(chunk)**

**return md5\_hash.hexdigest()**

**def scan\_directory(directory):**

**"""Scan a directory and return a dictionary of file paths and their hashes."""**

**file\_hashes = {}**

**for root, dirs, files in os.walk(directory):**

**for file\_name in files:**

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

**file\_hashes[file\_path] = get\_file\_hash(file\_path)**

**return file\_hashes**

**def monitor\_directory(directory, rules=None, interval=1):**

**"""Monitor the directory for changes."""**

**previous\_scan = scan\_directory(directory)**

**logger.info(f"Starting directory monitoring on: {directory}")**

**try:**

**while True:**

**time.sleep(interval)**

**current\_scan = scan\_directory(directory)**

**# Check for created or modified files**

**for file\_path, file\_hash in current\_scan.items():**

**if file\_path not in previous\_scan:**

**logger.info(f"File created: {file\_path}")**

**apply\_rules(file\_path, 'created', rules)**

**elif previous\_scan[file\_path] != file\_hash:**

**logger.info(f"File modified: {file\_path}")**

**apply\_rules(file\_path, 'modified', rules)**

**# Check for deleted files**

**for file\_path in previous\_scan.keys():**

**if file\_path not in current\_scan:**

**logger.info(f"File deleted: {file\_path}")**

**apply\_rules(file\_path, 'deleted', rules)**

**previous\_scan = current\_scan**

**except KeyboardInterrupt:**

**logger.info("Directory monitoring stopped.")**

**def apply\_rules(file\_path, event\_type, rules):**

**"""Apply predefined rules based on file events."""**

**if rules:**

**for rule in rules:**

**if rule['event\_type'] == event\_type and rule['condition'](file\_path):**

**rule['action'](file\_path)**

**def custom\_action(file\_path):**

**"""Example action to take when a rule is triggered."""**

**logger.info(f"Custom action triggered for {file\_path}")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**# Replace with the directory you want to monitor**

**path\_to\_monitor = "/path/to/monitor"**

**# Define rules for triggering actions**

**rules = [**

**{**

**'event\_type': 'created',**

**'condition': lambda path: path.endswith('.txt'), # Trigger only if a .txt file is created**

**'action': custom\_action**

**},**

**{**

**'event\_type': 'modified',**

**'condition': lambda path: 'important' in path, # Trigger if 'important' is in the file path**

**'action': custom\_action**

**},**

**{**

**'event\_type': 'deleted',**

**'condition': lambda path: True, # Trigger on any file deletion**

**'action': custom\_action**

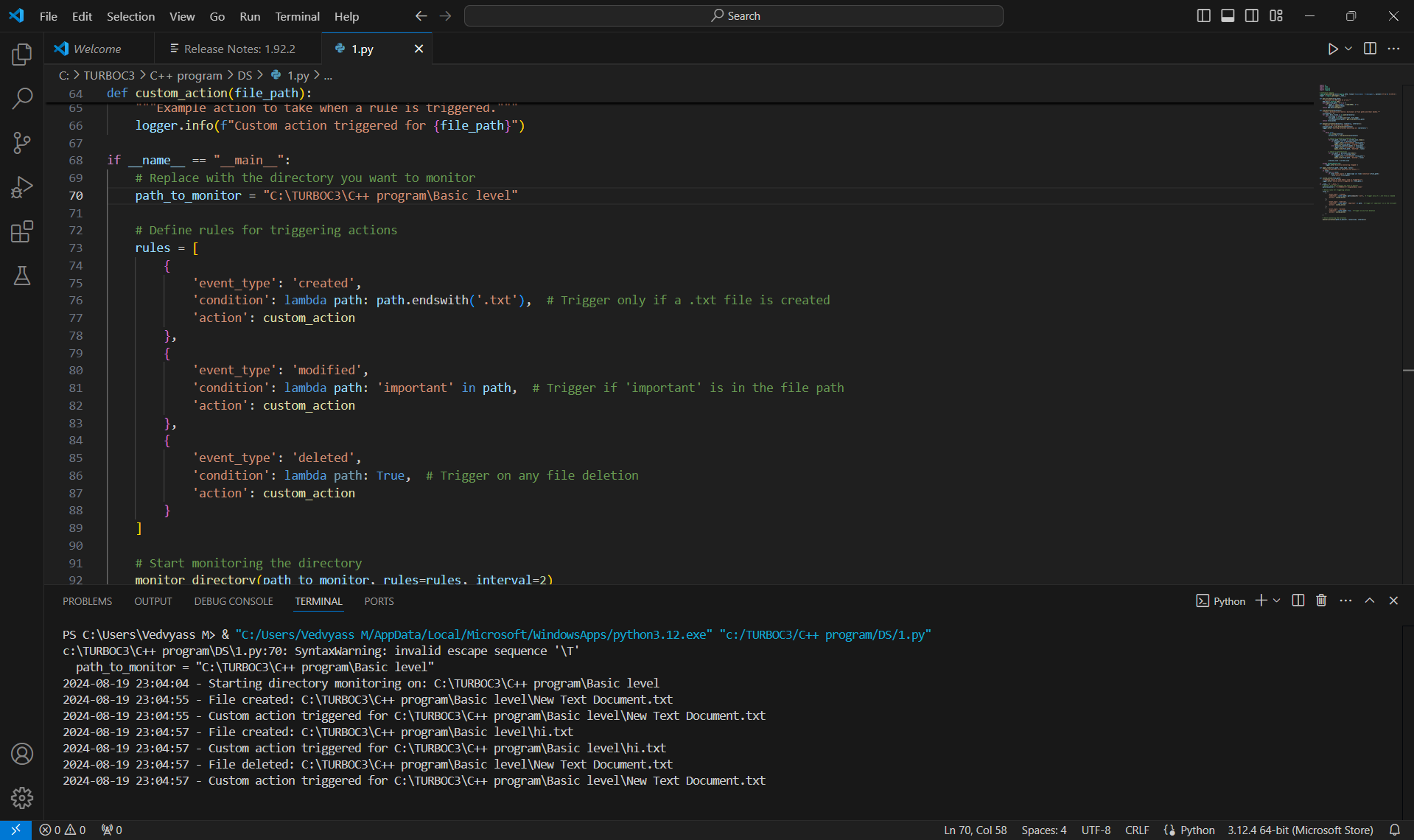
**}**

**]**

**# Start monitoring the directory**

**monitor\_directory(path\_to\_monitor, rules=rules, interval=2)**

**Output Screen Shots:**

****

**Conclusion:**

Python program that monitors a directory for changes, such as file creations, deletions, or modifications, and logs these events in real-time, optionally triggering actions based on predefined rules is successfully created and executed