**Software Engineering Tools Lab Assignment No-7**

(Module 5- Source code testing using tools) Due **date-31/03/2023**

Q 1. What is Source code analysis? What is its importance?

1. Source code analysis (also known as static code analysis) is the process of examining the code of a software application without actually executing it. This analysis is typically done using specialized software tools that can analyze the code for syntax errors, coding standards violations, security vulnerabilities, and other potential issues.
2. The importance of source code analysis lies in its ability to identify potential issues early in the software development lifecycle. By catching these issues early on, developers can fix them before they become more difficult and costly to resolve later on in the development process, or worse, in production environments. Additionally, source code analysis can help improve the overall quality of the software by identifying areas where code can be optimized or refactored.

Q 2. Below are the some important open source tools used in testing the source code, provide the information of below tools with respect to

1. Owner/ developer
2. Developed in which language
3. Brief information/introduction
4. Language support (applicable for source code written in language)
5. Advantages
6. Disadvantages

**Source code analysis tools-** VisualCodeGrepper:

* 1. Owner/developer: The VisualCodeGrepper tool is an open-source tool that is maintained by a team of developers.
  2. Developed in which language: The tool is developed in C# and is available for Windows, Linux, and macOS.
  3. Brief information/introduction: VisualCodeGrepper is a static code analysis tool that can help identify security vulnerabilities and coding issues in source code. It can scan a wide variety of programming languages, including C, C++, C#, Java, Python, and others.
  4. Language support: The tool supports a wide variety of programming languages, including C, C++, C#, Java, Python, Ruby, and more. e. Advantages: VisualCodeGrepper is easy to use, and its graphical user interface makes it simple to navigate the results of the analysis. It also supports a wide variety of programming languages and is available for multiple operating systems. f. Disadvantages: Some users have reported issues with false positives in the analysis results.

1. RIPS:

a. Owner/developer: RIPS is an open-source tool developed by the RIPS Technologies team.

b. Developed in which language: The tool is developed in PHP and is available for Linux and Windows operating systems.

c. Brief information/introduction: RIPS is a static code analysis tool that can detect security vulnerabilities in PHP applications. It can analyze source code to identify potential security issues, such as SQL injection, cross-site scripting (XSS), and file inclusion vulnerabilities.

d. Language support: The tool is specifically designed for PHP applications.

e. Advantages: RIPS can quickly identify security vulnerabilities in PHP applications and provides detailed information about each issue. It is easy to use and can be integrated into the development process with tools such as Jenkins or GitLab.

f. Disadvantages: The tool is only designed for PHP applications, so it may not be useful for organizations that use multiple programming languages.

1. Brakeman:

a. Owner/developer: Brakeman is an open-source tool developed by the Brakeman team.

b. Developed in which language: The tool is developed in Ruby and is available for Linux and macOS operating systems.

c. Brief information/introduction: Brakeman is a static code analysis tool that can detect security vulnerabilities in Ruby on Rails applications. It can analyze source code to identify potential security issues, such as SQL injection, cross-site scripting (XSS), and more.

d. Language support: The tool is specifically designed for Ruby on Rails applications.

e. Advantages: Brakeman can quickly identify security vulnerabilities in Ruby on Rails applications and provides detailed information about each issue. It is easy to use and can be integrated into the development process with tools such as Jenkins or GitLab.

f. Disadvantages: The tool is only designed for Ruby on Rails applications, so it may not be useful for organizations that use multiple programming languages.

1. Flawfinder:

a. Owner/developer: Flawfinder is an open-source tool developed by David A. Wheeler.

b. Developed in which language: The tool is developed in Python and is available for Linux and macOS operating systems.

c. Brief information/introduction: Flawfinder is a static code analysis tool that can detect security vulnerabilities in C and C++ applications. It can analyze source code to identify potential security issues, such as buffer overflows, format string vulnerabilities, and more.

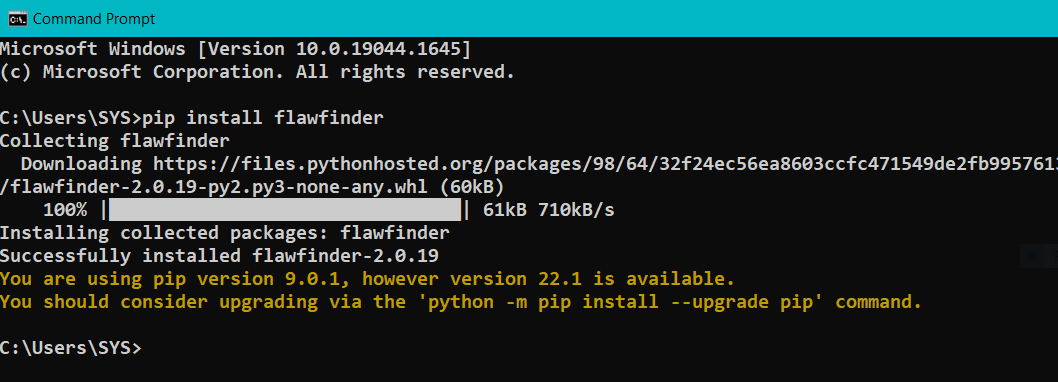
d. Language support: The tool is specifically designed for C and C++ applications.

e. Advantages: Flawfinder can quickly identify security vulnerabilities in C and C++ applications and provides detailed information about each issue. It is easy to use and can be integrated into the development process with tools such as Jenkins or GitLab.

Q 3. Perform source code testing using **Flawfinder** for the code written in ‘c’ and

‘cpp’ language given below

**Installing Flawfinder:**

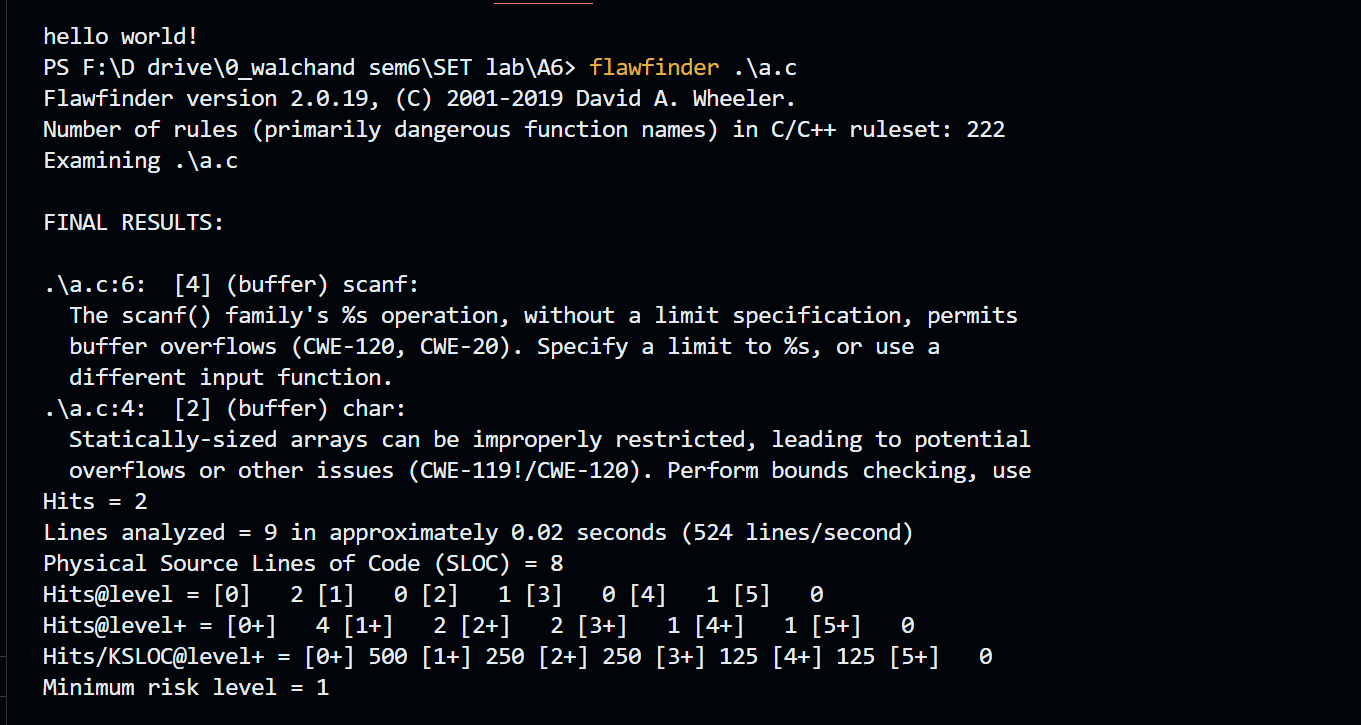
****

**Program1.c file:**

**a. Number of hits: 2**

**b. Potential risks:**

1. First one is use of strcpy function. It does not check for buffer overflows when copying to destination.
2. Another vulnerability is the use of a char array. Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues. Instead, functions can be used to check the limit length and ensure that size is larger than the maximum possible length.

****

**c. Suggested alternatives for these risks:**

1. It suggests to use snprintf, strcpy\_s, or strlcpy.
2. Perform bounds checking, usefunctions that limit length, or ensure that the size is larger than themaximum possible length.

**d. Updating the code:**

// C program to demonstrate

// Flawfinder

#include <stdio.h>

#include <string.h>

// Driver code

int main()

{

**char temp[];**

char str[] = "hello";

**strcpy\_s(temp, str);**

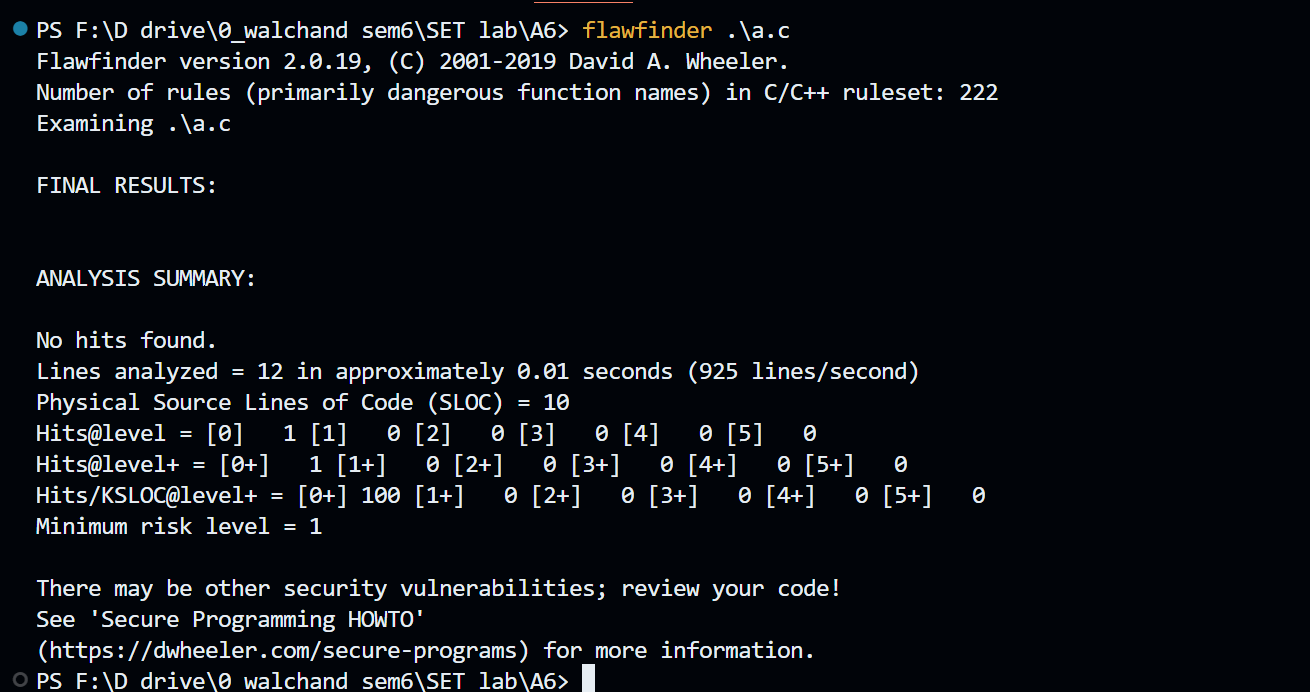
printf("%s", temp);

return 0;

}

**e. Re-execution of code after updating:**

No hits found.

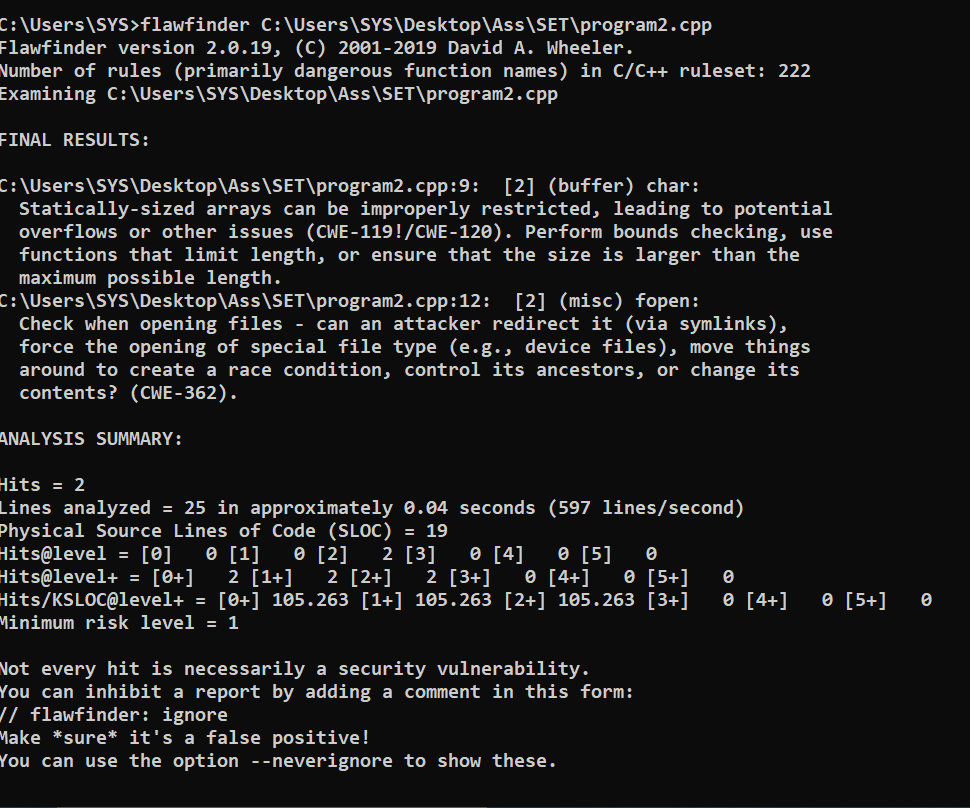
****

**Program2.cpp**

**a. Number of hits: 2**

**b. Potential risks:**

1. Statically-sized arrays can be improperly restricted, leading to potentialoverflows or other issues.
2. Check when opening files - can an attacker redirect it (via symlinks),force the opening of special file type (e.g., device files), move thingsaround to create a race condition, control its ancestors, or change itscontents?

****

**c. Suggested alternatives for these risks:**

* Perform bounds checking, usefunctions that limit length, or ensure that the size is larger than themaximum possible length.

**d. Updating the code:**

#include <iostream>

#include <cstdio>

using namespace std;

int main()

{

int count = 10;

char str[];

FILE \*fp;

**fp = tmpfile();**

fputs("An example file\n", fp);

fputs("Filename is file.txt\n", fp);

rewind(fp);

while(feof(fp) == 0)

{

fgets(str,count,fp);

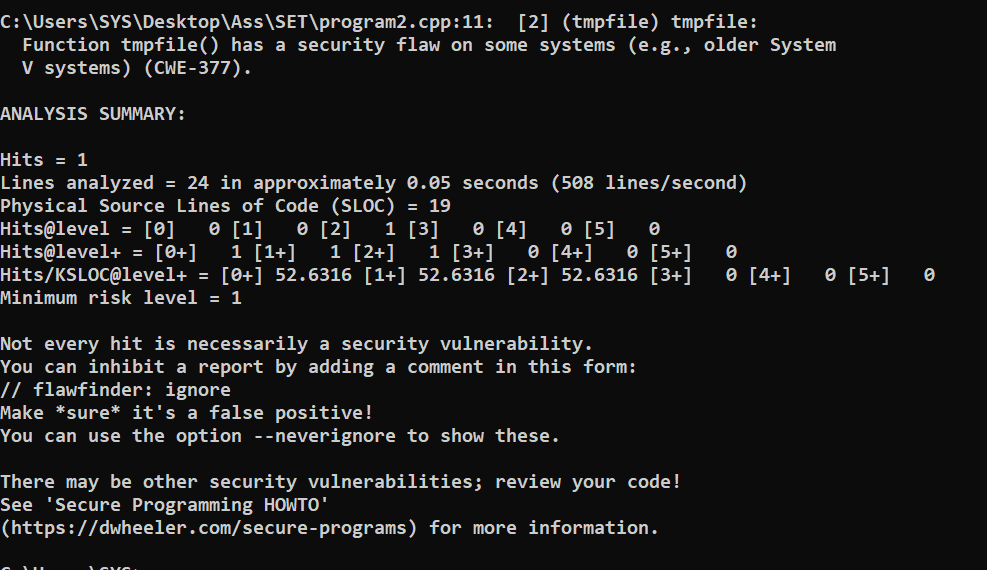
cout<< str <<endl;

}

fclose(fp);

return 0;

**e. Re-execution of code after updating:**

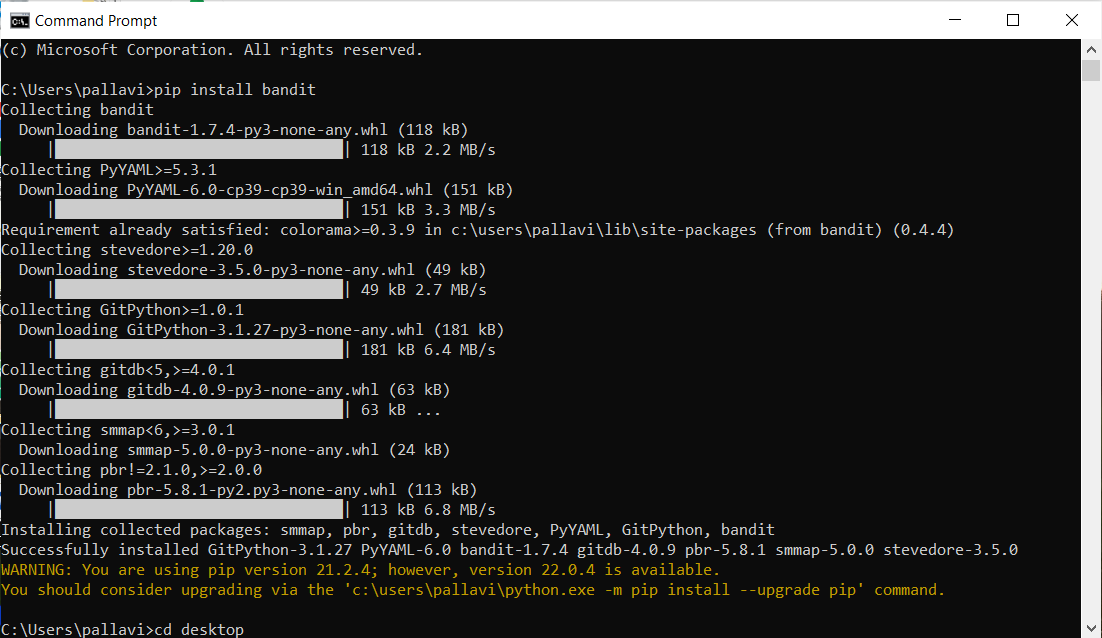
****

**Q 4. Perform source code testing using Bandit for your code written in ‘python’ language (use your previous code)for any security flaws**

**After performing analysis create a report which will contain below points**

1. **Number of hits**
2. **Potential risks**
3. **Suggested alternatives for these risks**
4. **Updating the code as per suggestions**
5. **Re-execution of code after updating the changes.**

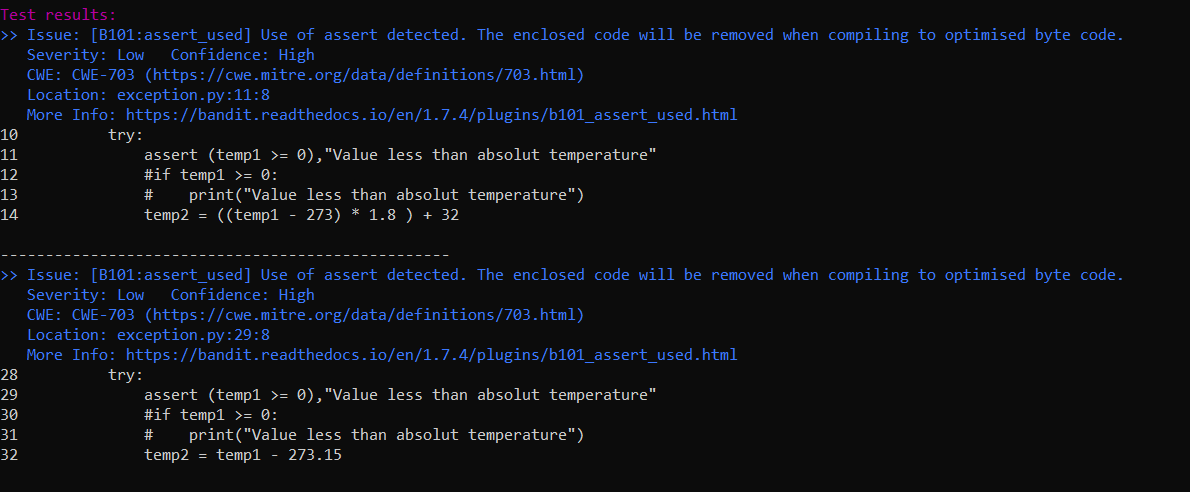
**Install bandit**

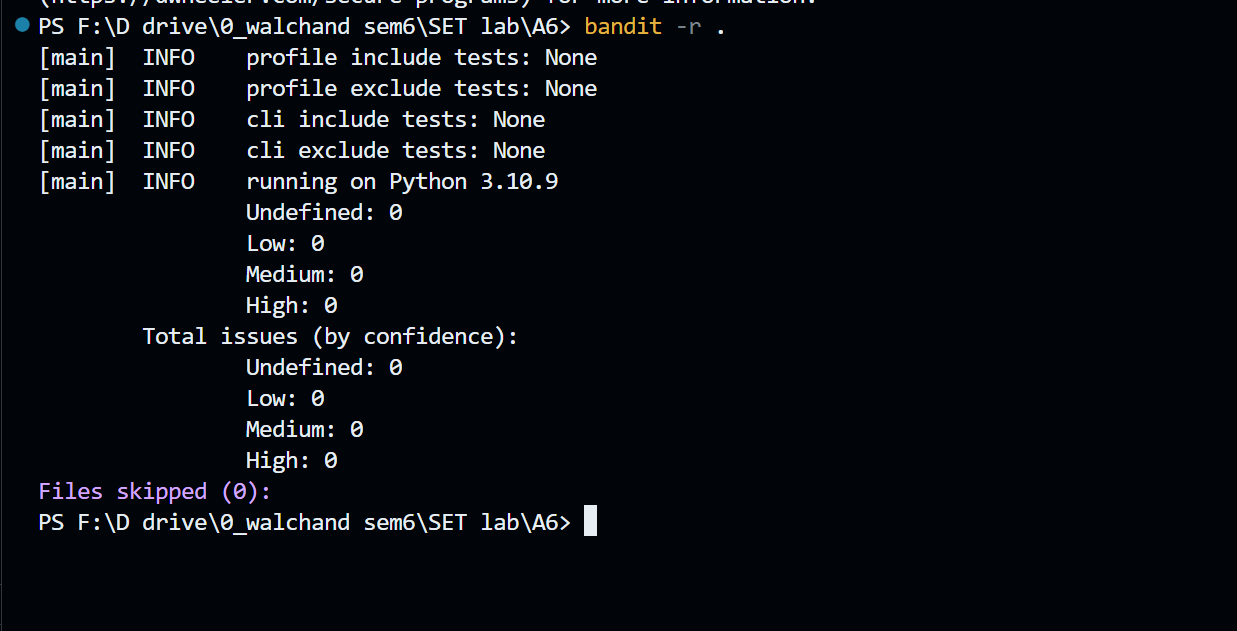
****

**a. Number of hits: 2**

**b. Potential risks:**

Issue: [B101:assert\_used] Use of assert detected. The enclosed code will be removed when compiling to optimized byte code

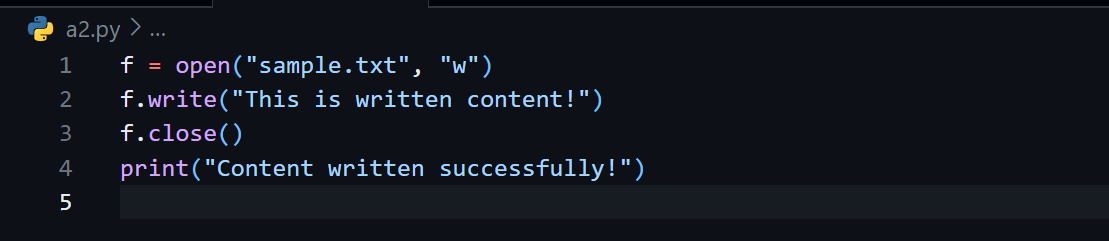
****

****

**c. Suggested alternatives for these risks:**

Use if statements instead of assert.

**d. Updating the code as per suggestions**



**e. Re-execution of code after updating the changes.**

