|  |  |
| --- | --- |
|  | C PROJECT |

|  |
| --- |
| **PROJECT TOPIC: PYTHON** **ERROR DETECTOR USING** **C LANGUAGE** |
|  |

* **CONCEPT OF PROJECT: PYTHON ERROR DETECTOR**

**INTRODUCTION:**

Python is a popular programming language known for its simplicity and readability.

However, even experienced programmers make syntax errors while writing Python code.

The "Python Error Detector" is a project designed to help programmers identify and rectify common syntax errors in their Python code.

**OBJECTIVE:**

The main goal of this project is to develop a tool that automates the detection of common Python syntax errors.

The tool analyzes Python code line by line and provides feedback on potential errors to improve code quality and prevent runtime errors.

**KEY FEATURES:**

**AUTOMATED ERROR DETECTION:**

The Python Error Detector scans Python code and identifies common syntax errors automatically.

It provides developers with a list of errors and their respective line numbers for easy debugging.

**COMMON ERROR CHECKS:**

The tool includes checks for common Python syntax errors such as missing colons, indentation issues, invalid variable names, and more.

These checks cover a wide range of common programming mistakes to enhance code reliability.

**CUSTOM ERROR CHECKS:**

Developers can extend the tool by adding custom error checks tailored to their project's requirements.

This flexibility allows the tool to adapt to specific coding standards and project needs.

**USER FRIENDLY INTERFACE:**

The tool offers a user-friendly command-line interface, making it accessible to both beginners and experienced developers.

It provides clear error messages to assist in the debugging process.

**HOW IT WORKS:**

**INPUT:**

The user provides the Python program file as input to the Python Error Detector via the command line.

**ERROR DETECTION:**

The tool reads the Python program line by line.

It checks each line for common syntax errors using predefined and custom error checks.

**ERROR REPORTING:**

If an error is detected, the tool records the error message and the line number in an error log.

**OUTPUT:**

After analyzing the entire program, the Python Error Detector displays a summary of errors found.

The user can review the errors and make necessary corrections.

**BENEFITS OF USING C DETECTOR:**

**IMPROVED CODE QUALITY:**

By catching common syntax errors early in the development process, developers can write more reliable Python code.

**TIME SAVINGS:**

The tool accelerates the debugging process by pinpointing errors quickly.

**LEARNING AID:**

It serves as an educational tool for programmers learning Python, helping them understand common pitfalls.

**FUTURE ENHANCEMENTS:**

Future enhancements could include more advanced error detection techniques, integration with code editors and IDEs, and support for a wider range of Python versions**.**

* **CODE WITH IT’S DESCRIPTION:**

1. **Header Files and Definitions(code):**

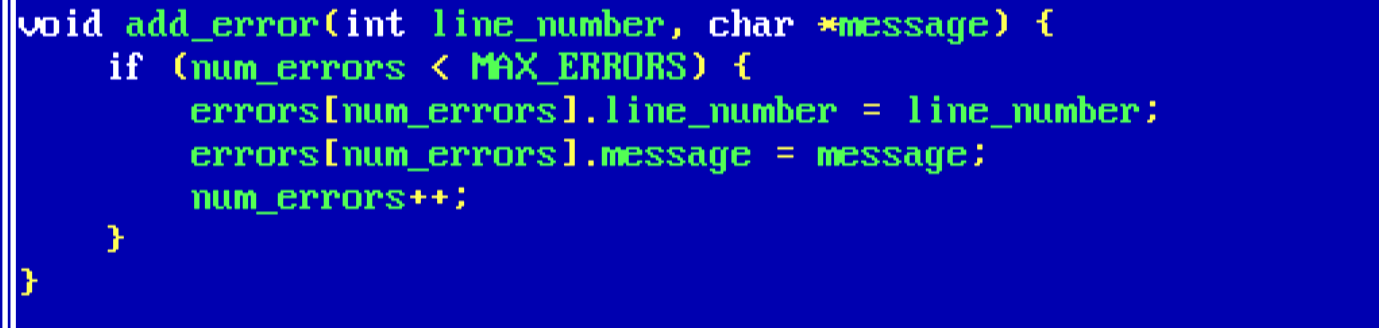
A computer screen shot of a blue screen

Description automatically generated

**Description:**

* **The program includes standard C libraries (stdio.h, stdlib.h, string.h).**
* **It defines a constant ‘MAX\_ERRORS’ and a structure ‘Error’ to store error information.**
* **It declares an array of ‘Error’ structs named ‘errors’ and an integer ‘num\_errors’ to keep track of the number of errors.**

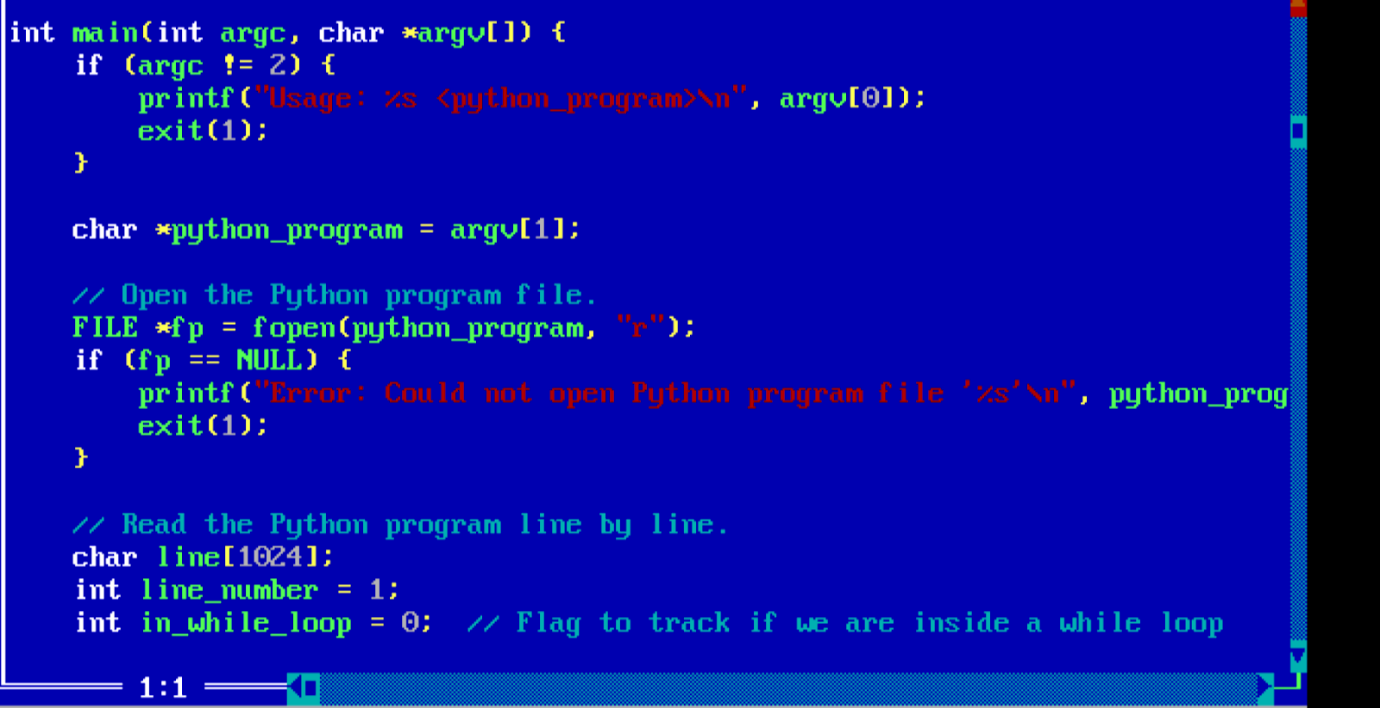
**2) Add\_error fuction(code):**



**Description:**

* **This function takes a line number and an error message as arguments and adds an error to the errors array.**
* **It checks if there is still space in the errors array (num\_errors is less than MAX\_ERRORS) and adds the error if there is space.**

**3)Main Function(code):**



**Description:**

* The main function is the entry point of the program.It checks if the program is called with exactly one command-line argument (the name of the Python program file)and prints a usage message if not.
* It opens the specified Python program file for reading and checks if it's successfully opened.
* It reads the Python program line by line and performs various error checks on each line.

**4) Check for Missing Colon (:) After 'if', ‘while’ statements and after function definition header**

**(CODE):**

A blue screen with red and blue text

Description automatically generated

**DESCRIPTION:**

**Check for Missing Colon (:) After 'if' Statement:**

* if (strstr(line, "if True:") && !strstr(line, ":")): This checks if the current line contains the string "if True:" (indicating the start of an 'if' statement) and does not contain ":" (indicating a missing colon).
* If this condition is met, it adds an error to the errors array with the message "Expected colon (:) after if statement."

**Check for Missing Colon (:) After 'while' Statement(code):**

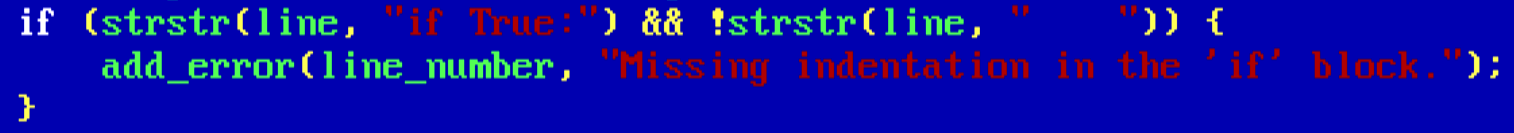
* This checks if the current line contains the string "while True:" (indicating the start of a 'while' loop) and does not contain ":" (indicating a missing colon).
* If this condition is met, it adds an error to the errors array with the message "Expected colon (:) after while statement."

**Check for Missing Colon (:) After Function Definition Header:**

* “else if (strstr(line, "def f(") && !strstr(line, "):")) ”This checks if the current line contains the string "def f(" (indicating the start of a function definition) and does not contain "):" (indicating a missing colon).
* If this condition is met, it adds an error to the errors array with the message "Expected colon (:) after function definition header."

The loop then increments the line\_number variable to keep track of the line being processed.

**5)Check for Missing Indentation in 'if' Block(CODE):**

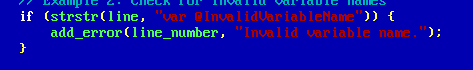


**DESCRIPTION:**

* This code checks if the current line contains the string "if True:", indicating the start of an 'if' block.
* It then checks if the same line does not contain " ", which represents four spaces of indentation.
* If both conditions are met, it adds an error to the errors array with the message "Missing indentation in the 'if' block."

**6) Check for Invalid Variable Names:**

**(CODE):**



**DESCRIPTION:**

* This code checks if the current line contains the string "var @InvalidVariableName".
* If it finds this string, it adds an error to the errors array with the message "Invalid variable name."

**7) Check for Incorrect Function Call:**

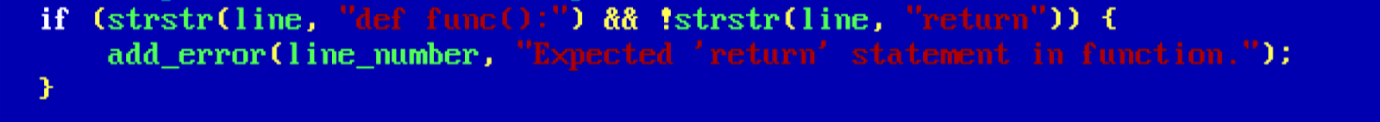
**(CODE):**



**DESCRIPTION:**

* This code checks if the current line contains the string "foo(" (indicating the start of a function call) and does not contain ")" (indicating a missing closing parenthesis).
* If both conditions are met, it adds an error to the errors array with the message "Expected closing parenthesis in function call."

**8) Check for Missing 'return' Statement in Function(code):**



**DESCRIPTION:**

* This code checks if the current line contains the string "def func():" (indicating the start of a function definition) and does not contain "return" (indicating a missing 'return' statement).
* If both conditions are met, it adds an error to the errors array with the message "Expected 'return' statement in function."

**9)Check for Missing 'import' Statement(**CODE):



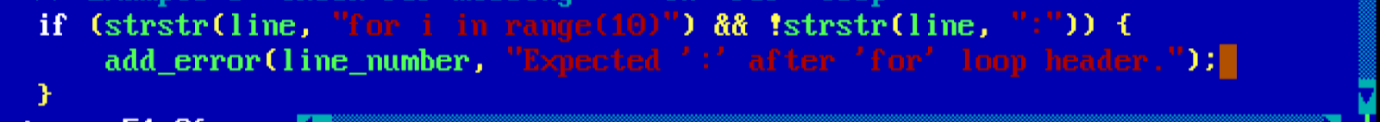
**DESCRIPTION:**

This code checks if the current line contains the string "import module" and does not contain "module" (indicating a missing module name after 'import').

If both conditions are met, it adds an error to the errors array with the message "Missing 'import' statement."

**10) Check for Missing Colon (:) in 'for' Loop:**

**(CODE):**



**DESCRIPTION:**

This code checks if the current line contains the string "for i in range(10)" (indicating the start of a 'for' loop) and does not contain ":" (indicating a missing colon after the loop header).

If both conditions are met, it adds an error to the errors array with the message "Expected ':' after 'for' loop header."

**11) Check for Invalid Syntax:**

**(CODE):**



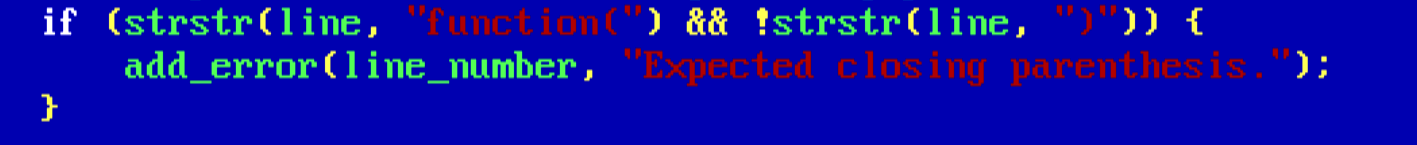
**Check for Invalid Syntax:**

**DESCRIPTION:**

* This code checks if the current line contains the string "invalid syntax here!".
* If it finds this string, it adds an error to the errors array with the message "Invalid syntax detected."

**12) Check for Missing Closing Parenthesis:**

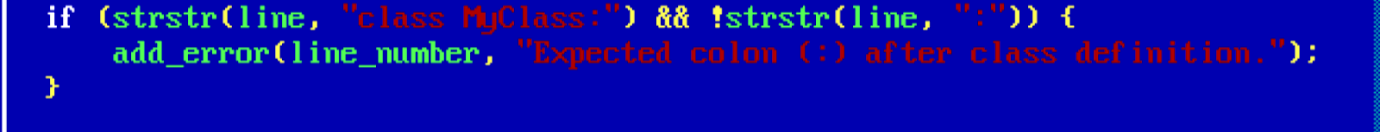
**(CODE):**



**DESCRIPTION:**

* This code checks if the current line contains the string "function(" (indicating the start of a function call) and does not contain ")" (indicating a missing closing parenthesis).
* If both conditions are met, it adds an error to the errors array with the message "Expected closing parenthesis.

**13) Check for Missing 'class' Definition**(CODE):

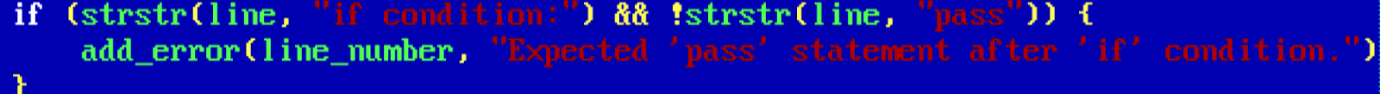


**DESCRIPTION:**

This code checks if the current line contains the string "class MyClass:" (indicating the start of a class definition) and does not contain ":" (indicating a missing colon).

If both conditions are met, it adds an error to the errors array with the message "Expected colon (:) after class definition."

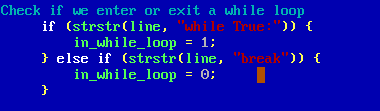
**14)** Check for Missing 'pass' Statement After 'if' Condition (CODE):

****

Description:

* This code checks if the current line contains the string "if condition:" (indicating an 'if' condition) and does not contain "pass" (indicating a missing 'pass' statement).
* If both conditions are met, it adds an error to the errors array with the message "Expected 'pass' statement after 'if' condition."

**15)CODE:**



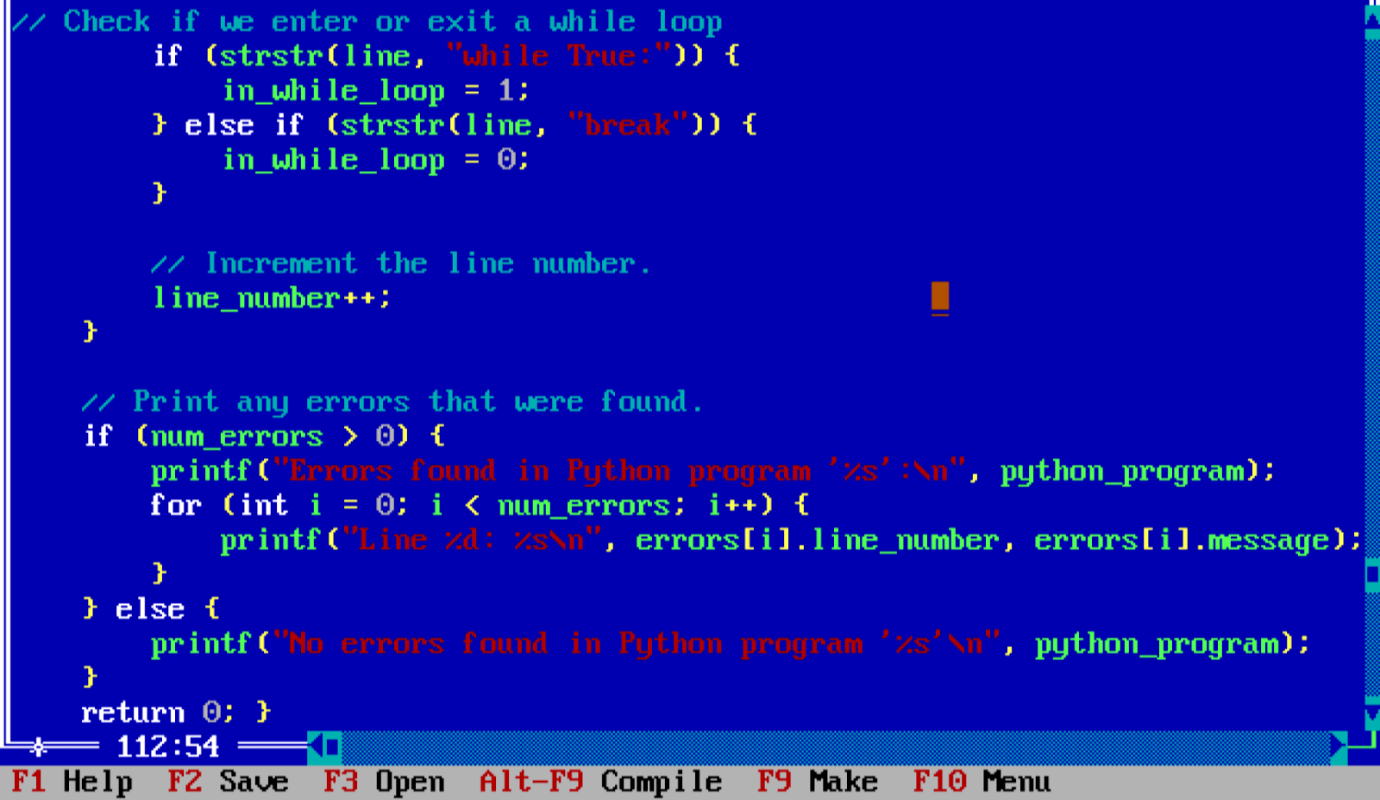
**DESCRIPTION:**

If the line contains "while True:", it sets the in\_while\_loop flag to 1, indicating that it has entered a while loop.

If the line contains "break", it sets the in\_while\_loop flag to 0, indicating that it has exited a while loop.

**16) ERROR REPORTING:**

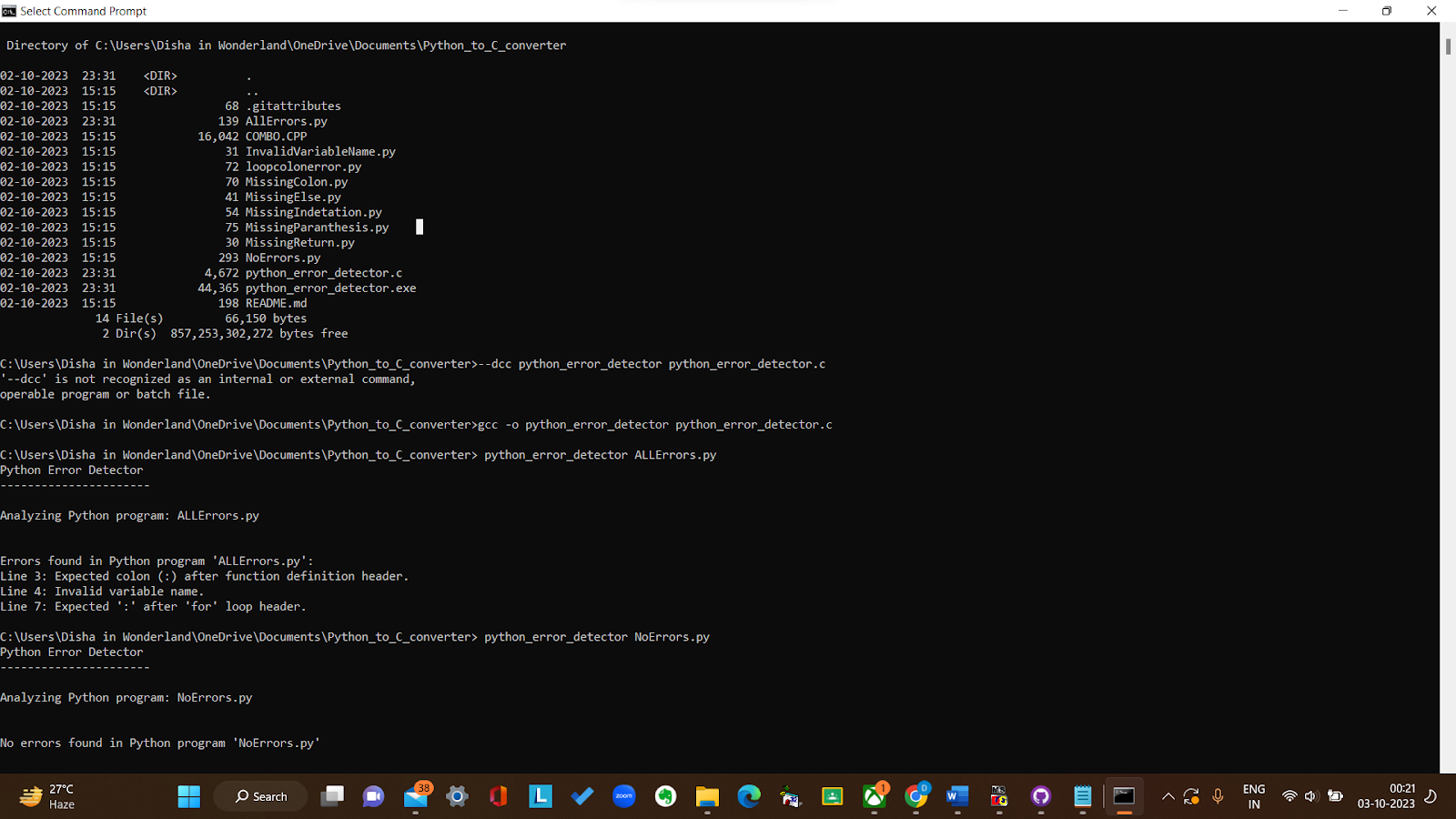
**CODE:**



**DECSRIPTION:**

* After processing all lines, the program prints any errors that were found, including the line number and the error message.
* If no errors are found, it indicates that no errors were detected in the Python program.

**OUTPUT OF THE PROGRAM:**



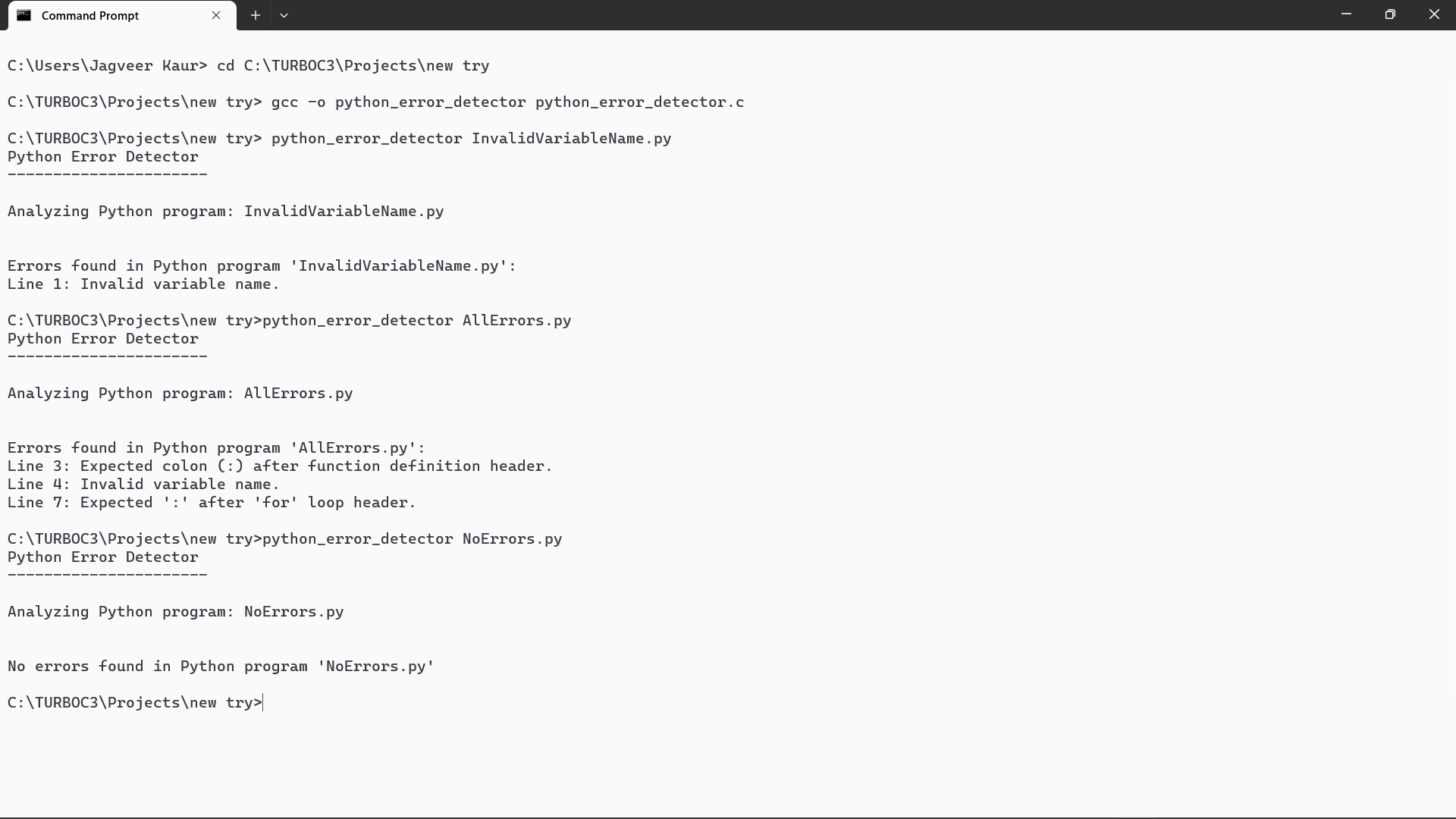
**DESCRIPTION OF OUTPUT:**

**Steps to be followed by user for output:**

1. The output of the python error detector to be executed in the command prompt.
2. Change the directory using cd function to that directory where the source c code and all python files are stored.

(Note: All the program files are supposed to be on one folder/directory .)

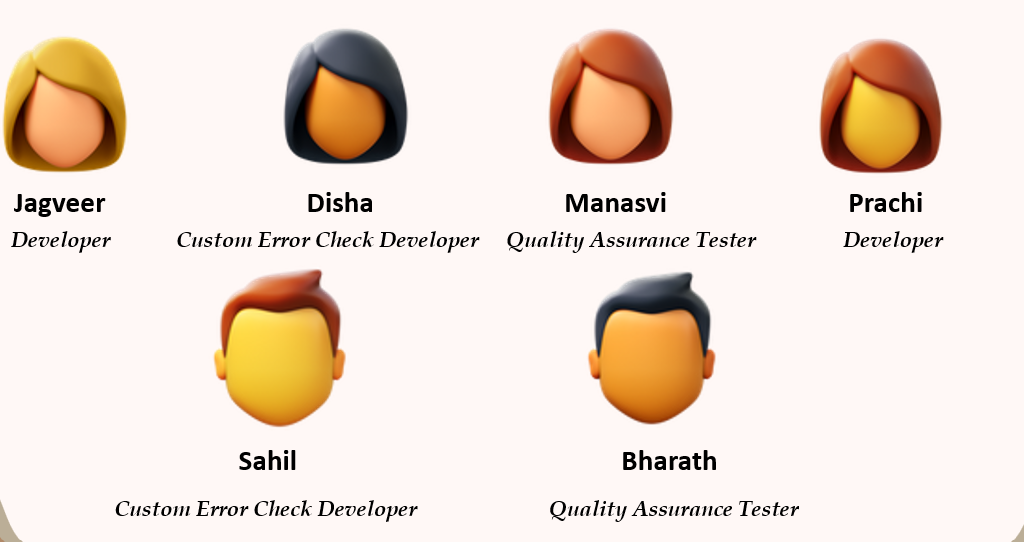
1. The user needs to enter “gcc -o python\_error\_detector python\_error\_detector.c” which would create an extension file.
2. User will then have to input their python file name in the form “python\_error\_detector your\_python\_file\_name.py”
3. Output screen would be displayed.



**GCC COMPILER:**

GCC stands for the GNU Compiler Collection. It is a collection of compilers and tools used for compiling and building software. In the context of the command prompt, "GCC compiler" typically refers to the C and C++ compilers provided by the GNU Compiler Collection. These compilers allow you to compile C and C++ source code into executable programs on various platforms, including Windows.

This C program is designed to analyze Python programs for various syntax errors.

***Team***

**Developer:**

**Role:** *The Developer is responsible for the technical aspects and overall code of the project***.**

**Responsibilities:**

*Developing the core functionality of the Python Error Detector.*

*Overseeing code reviews and ensuring code quality.*

*Guiding other team members in technical aspects***.**

**Quality Assurance (QA) Tester:**

**Role:** *The QA Tester is responsible for ensuring the tool functions correctly and is free of errors.*

**Responsibilities:**

*Creating test cases and test plans to evaluate the Python Error Detector.*

*Conducting rigorous testing and identifying issues.*

*Collaborating with developers to resolve any bugs or errors***.**

**Custom Error Check Developer:**

**Role:** *The Custom Error Check Developer focuses on extending the tool's functionality by adding custom error checks*.

**Responsibilities:**

*Identifying project-specific requirements for error checking.*

*Developing and integrating custom error checks into the Python Error Detector.*

*Collaborating with the Lead Developer to ensure seamless integration.*