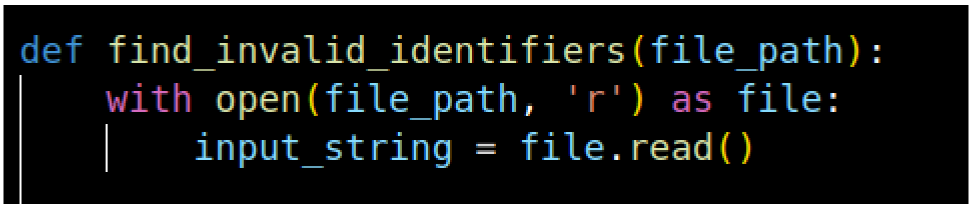
Code Evaluation

1. **File Reading:**

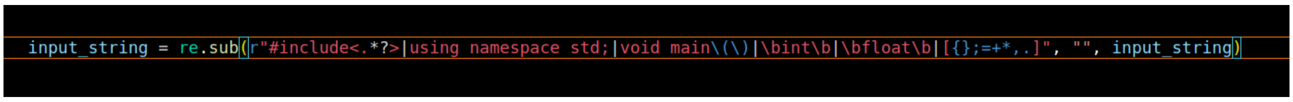
- The code opens the specified file, reads the content, and stores it in a variable for processing. This part works well for loading the input file content.



```python
def find_invalid_identifiers(file_path):
    with open(file_path, 'r') as file:
        input_string = file.read()
```

2. **Cleaning Non-Identifier Parts:**

- The code uses regular expressions to remove unnecessary parts such as keywords (`#include`, `void main()`, `int`, etc.) and symbols (`{}`, `=`, `+`, `*`, `,`, `.`), focusing on identifiers only.

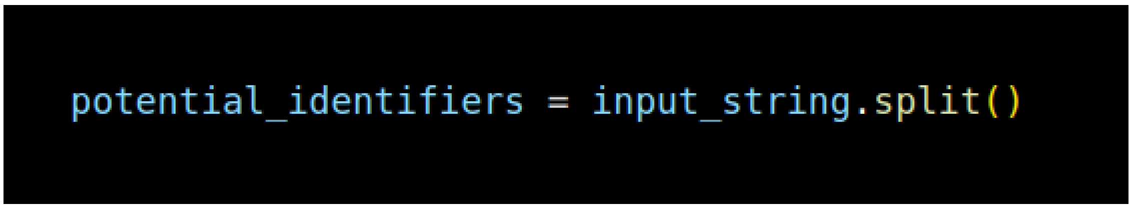


```python
input_string = re.sub(r"#include<.*?>|using namespace std;|void main\(\)|\bint\b|\bfloat\b|[{};=+*,.]", "", input_string)
```

\`\`\`

3. **Splitting and Identifying Potential Identifiers:**

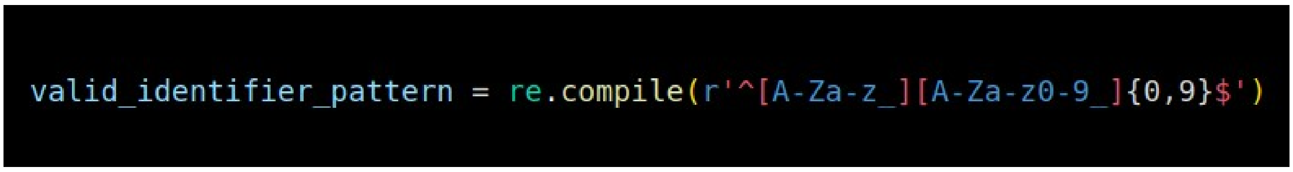
- After removing irrelevant elements, the code splits the content by whitespace to extract potential identifiers for analysis.



```python
potential_identifiers = input_string.split()
```

4. **Defining Valid Identifier Rules:**

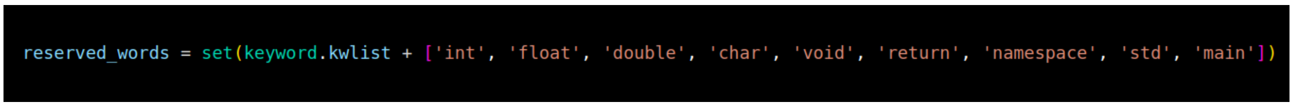
- The code defines a regex pattern to ensure valid identifiers start with a letter or underscore and contain only alphanumeric characters, up to 10 characters in total.



```python
valid_identifier_pattern = re.compile(r'^[A-Za-z_][A-Za-z0-9_]{0,9}$')
```

5. **Reserved Words:**

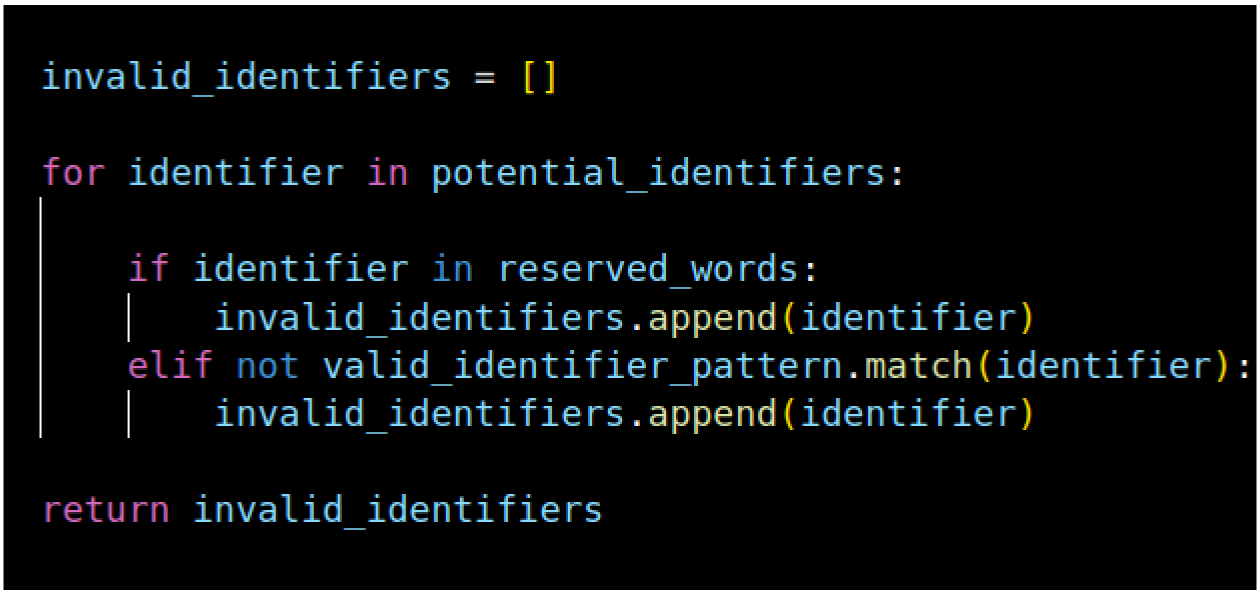
   - A set of reserved words combines Python keywords with some C++ keywords, which are excluded from identifiers.



```python
reserved_words = set(keyword.kwlist + ['int', 'float', 'double', 'char', 'void', 'return', 'namespace', 'std', 'main'])
```

6. **Loop for Detecting Invalid Identifiers:**

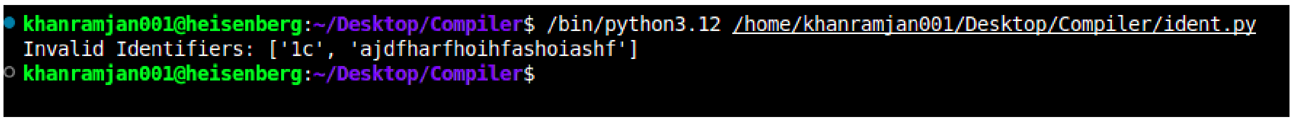
   - Each potential identifier is checked against the reserved words list and the regex pattern. Any identifier that fails these checks is added to `invalid_identifiers`.



```python
invalid_identifiers = []

for identifier in potential_identifiers:

    if identifier in reserved_words:
        invalid_identifiers.append(identifier)
    elif not valid_identifier_pattern.match(identifier):
        invalid_identifiers.append(identifier)

return invalid_identifiers
```

7. **Output**:

- The invalid identifiers are printed for inspection.



```
● khanramjan001@heisenberg:~/Desktop/Compiler$ /bin/python3.12 /home/khanramjan001/Desktop/Compiler/ident.py
  Invalid Identifiers: ['1c', 'ajdfharfhoihfashoiashf']
○ khanramjan001@heisenberg:~/Desktop/Compiler$
```

Challanges Faced:

Making the good sequence string for matching was a major challenge.