CSE 6234 - ADVANCED TOPICS IN SOFTWARE ENGINEERING

ISSUE – SLITHER SUPPORT FOR IMPORTS WITH ALIAS

FINAL ITERATION

SLITHER

PRESENTED BY : TEAM 4

Preeti Singh	- 1002013566
Sripal Thorupunoori	- 1001969001
Mohit Singhi	- 1002004892
Siddhartha Reddy Sungomula	- 1001969005

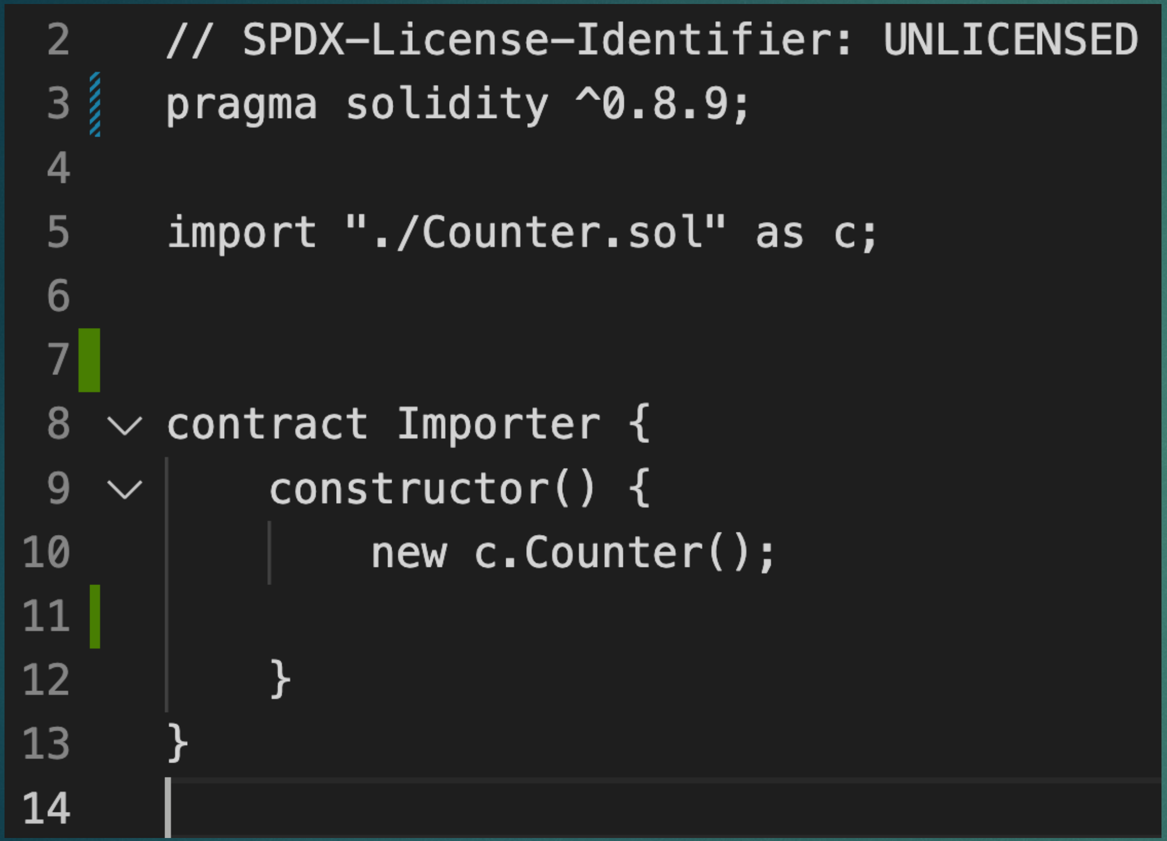
Project Plan

Iteration	Goal Targeted	Goal Achieved
Final Iteration	❖ Ensure that all deliverables are completed on time. ❖ Test the important code elements with the necessary test cases. ❖ Get feedback from users on the fix.	❖ Tried to improve our solution ❖ Able to test few more test cases ❖ Testing important code elements and modules ❖ Tried getting feedback from users ❖ Successfully mitigated possible risks ❖ Successfully reduced risk exposure

Slither Comparision with Competitors :

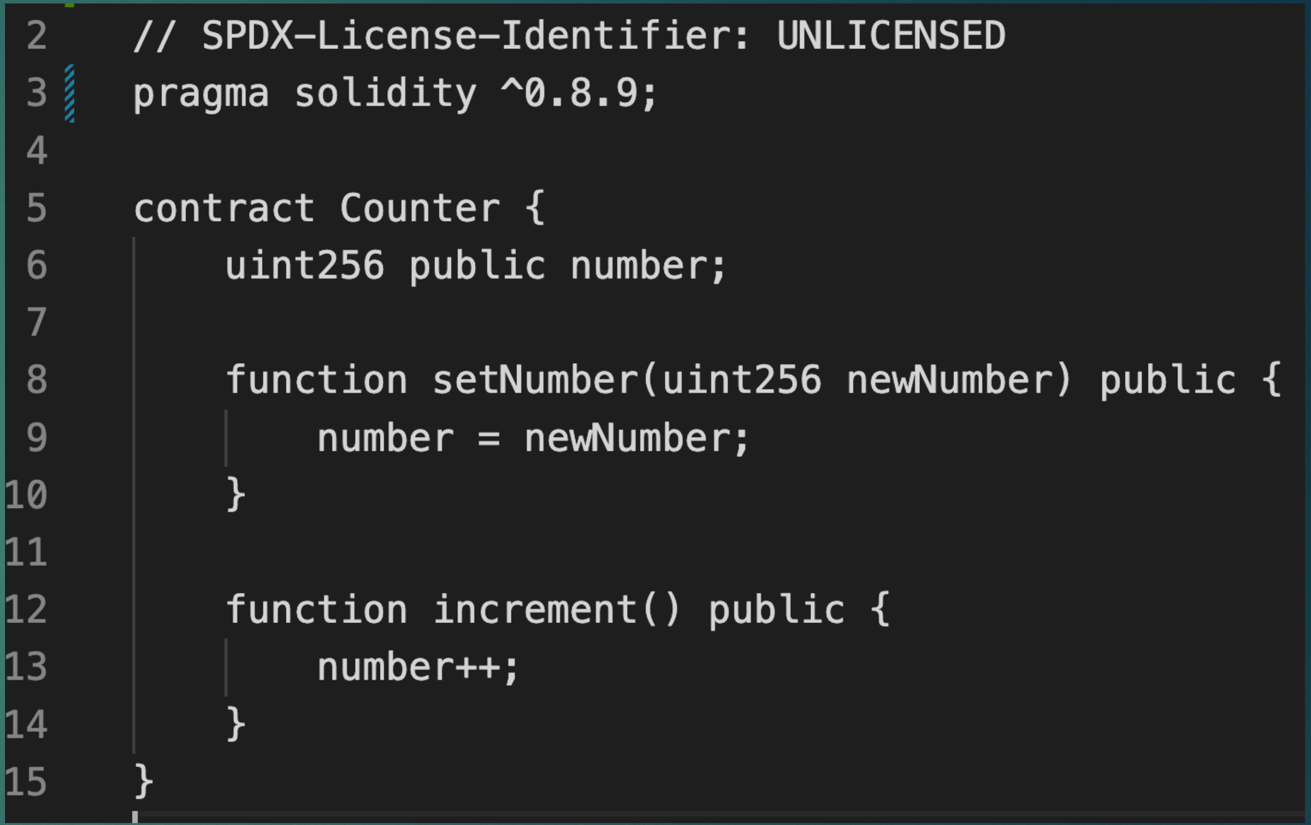
- Mythril:
  - Detection of contract-to-contract interaction issues
  - the ability to generate exploits.
- Oyente:
  - wider range of checks for vulnerabilities
  - Can detect issues that may not be detected by Slither.
- Manticore
  - less user-friendly than Slither
  - Requires some knowledge of symbolic execution
- Securify
  - Securify can detect a wider range of vulnerabilities
  - Including some that are difficult to detect with static analysis alone.
- Remix
  - More focused on the development process than on security analysis
  - Its static analysis tool is less advanced than Slither's.

Issue #1452[4]



```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.9;

import "./Counter.sol" as c;


contract Importer {
    constructor() {
        new c.Counter();

    }
}

```

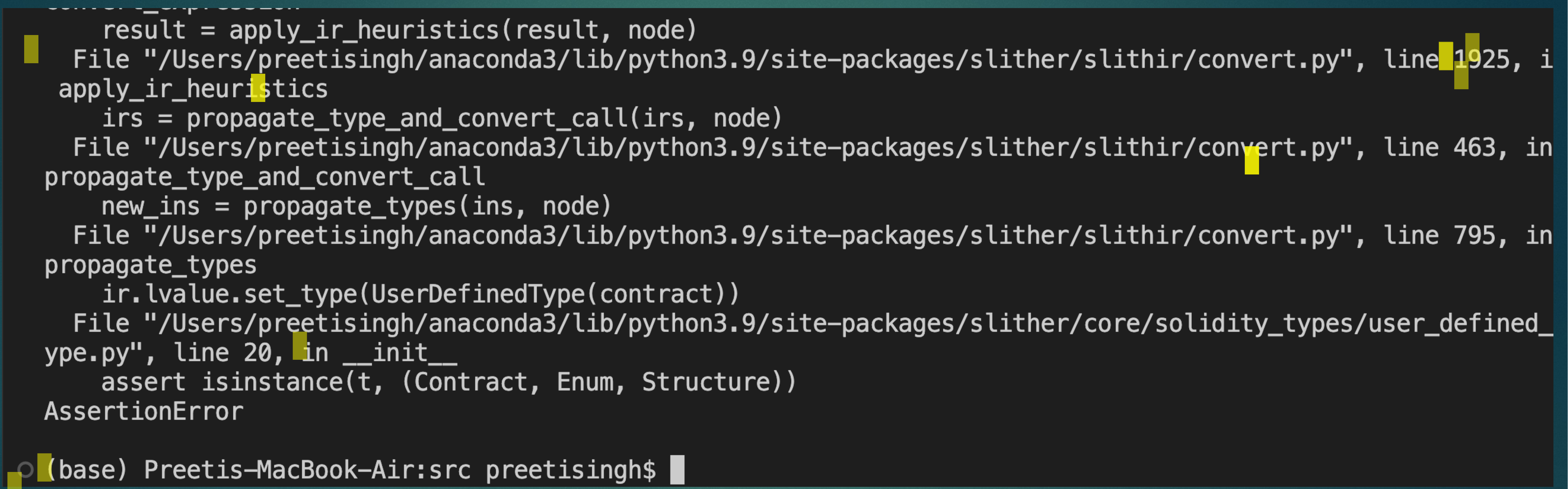
Importer.sol



```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.9;

contract Counter {
    uint256 public number;

    function setNumber(uint256 newNumber) public {
        number = newNumber;
    }

    function increment() public {
        number++;
    }
}
```

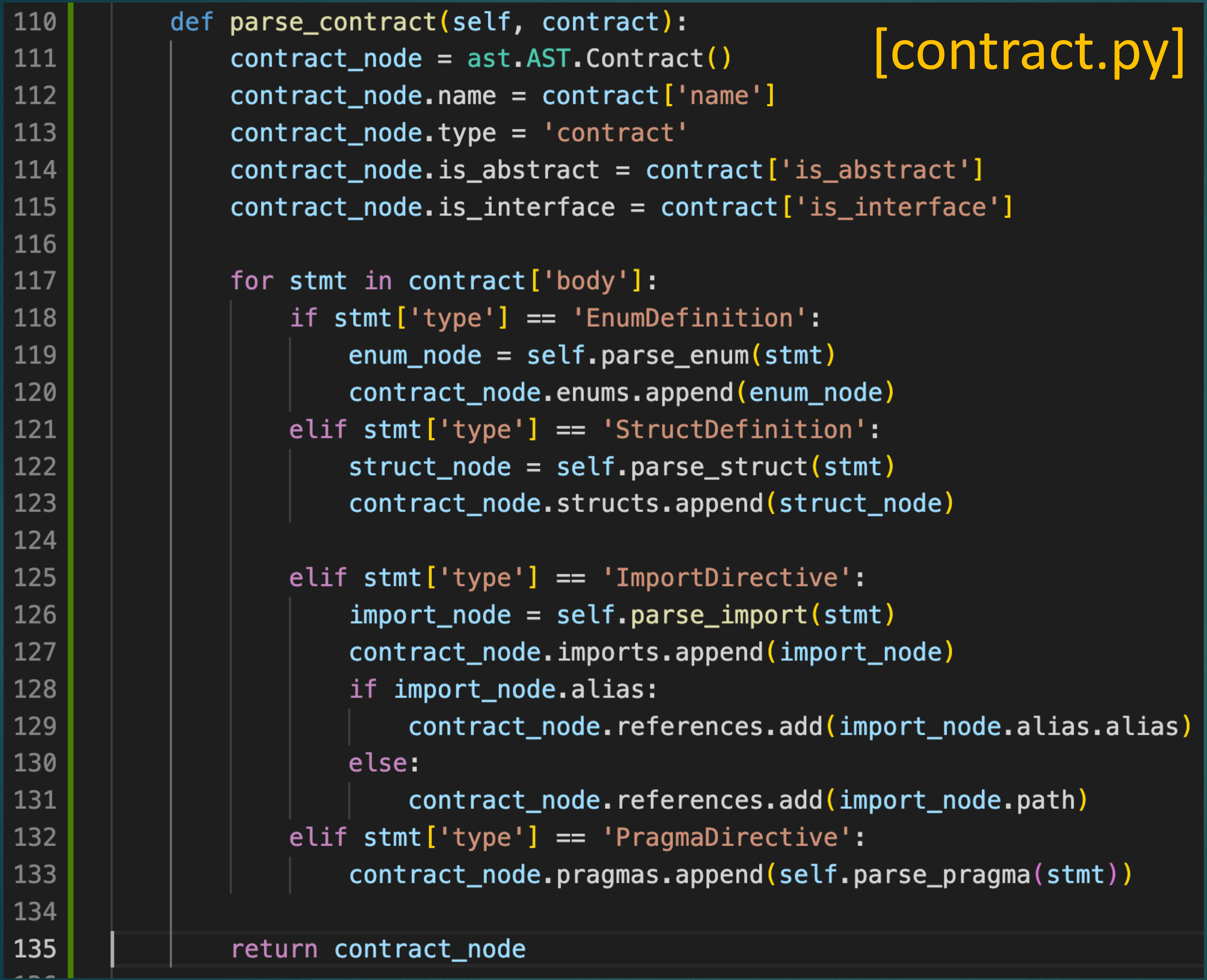
Counter.sol

Error Stack



```
    result = apply_ir_heuristics(result, node)
  File "/Users/preetisingh/anaconda3/lib/python3.9/site-packages/slither/slithir/convert.py", line 1925, i
 apply_ir_heuristics
    irs = propagate_type_and_convert_call(irs, node)
  File "/Users/preetisingh/anaconda3/lib/python3.9/site-packages/slither/slithir/convert.py", line 463, in
propagate_type_and_convert_call
    new_ins = propagate_types(ins, node)
  File "/Users/preetisingh/anaconda3/lib/python3.9/site-packages/slither/slithir/convert.py", line 795, in
propagate_types
    ir.lvalue.set_type(UserDefinedType(contract))
  File "/Users/preetisingh/anaconda3/lib/python3.9/site-packages/slither/core/solidity_types/user_defined_
ype.py", line 20, in __init__
    assert isinstance(t, (Contract, Enum, Structure))
AssertionError

(base) Preetis-MacBook-Air:src preetisingh$
```

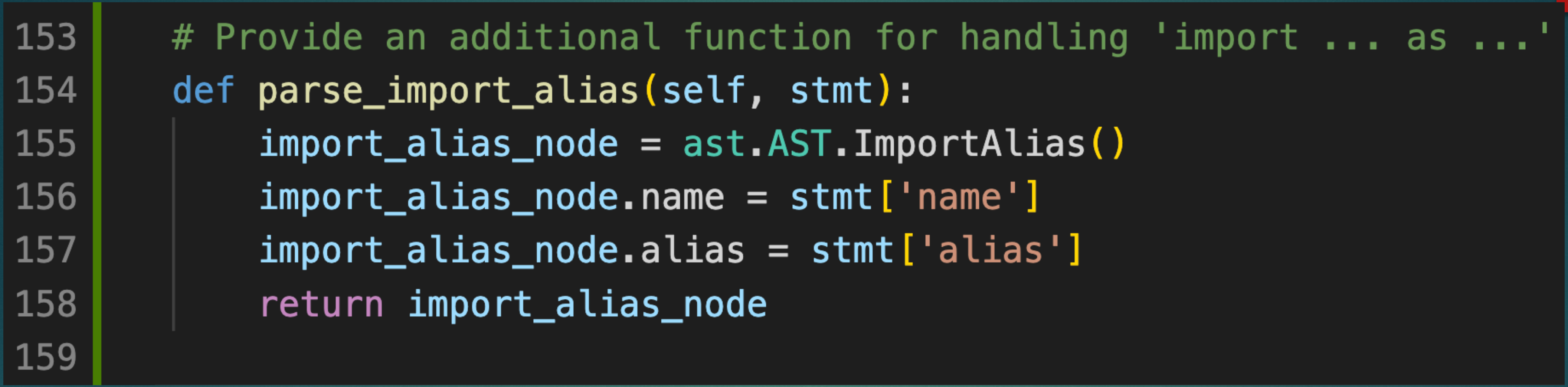
Implementation



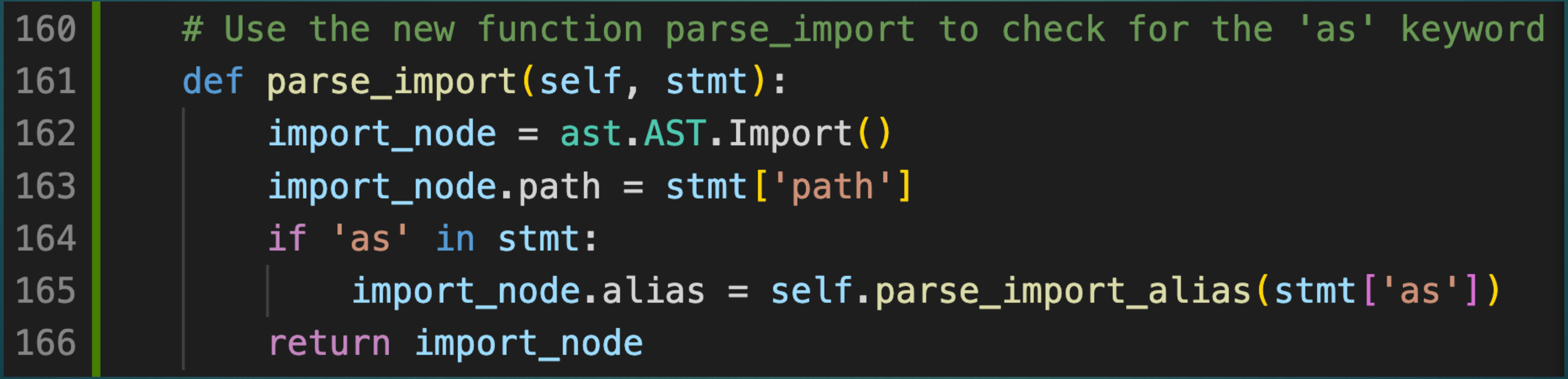
[contract.py]

```
    def parse_contract(self, contract):
        contract_node = ast.AST.Contract()
        contract_node.name = contract['name']
        contract_node.type = 'contract'
        contract_node.is_abstract = contract['is_abstract']
        contract_node.is_interface = contract['is_interface']

        for stmt in contract['body']:
            if stmt['type'] == 'EnumDefinition':
                enum_node = self.parse_enum(stmt)
                contract_node.enums.append(enum_node)
            elif stmt['type'] == 'StructDefinition':
                struct_node = self.parse_struct(stmt)
                contract_node.structs.append(struct_node)

            elif stmt['type'] == 'ImportDirective':
                import_node = self.parse_import(stmt)
                contract_node.imports.append(import_node)
                if import_node.alias:
                    contract_node.references.add(import_node.alias.alias)
                else:
                    contract_node.references.add(import_node.path)
            elif stmt['type'] == 'PragmaDirective':
                contract_node.pragmas.append(self.parse_pragma(stmt))

        return contract_node
```

Implementation



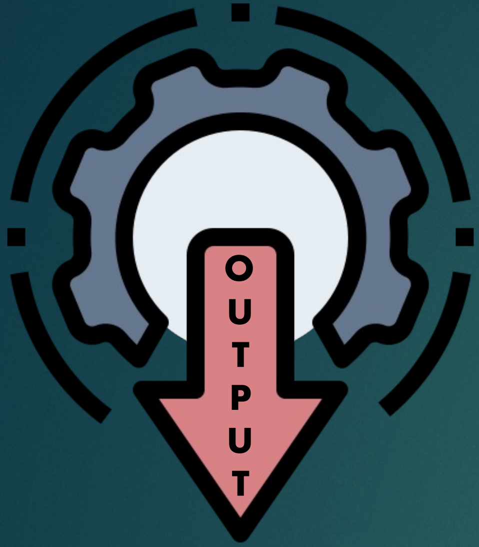
```python
    # Provide an additional function for handling 'import ... as ...'
    def parse_import_alias(self, stmt):
        import_alias_node = ast.AST.ImportAlias()
        import_alias_node.name = stmt['name']
        import_alias_node.alias = stmt['alias']
        return import_alias_node

```

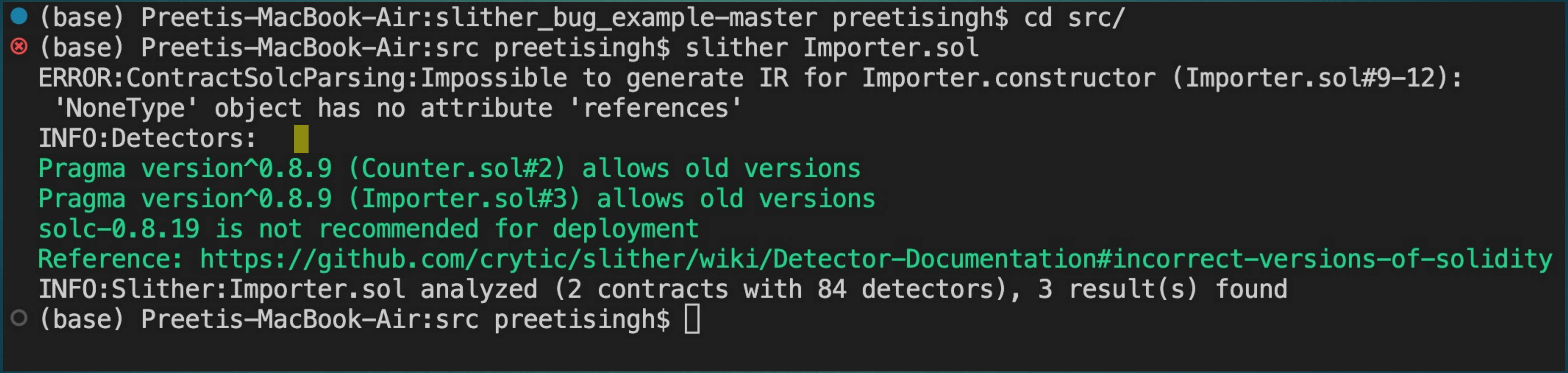


```python
    # Use the new function parse_import to check for the 'as' keyword
    def parse_import(self, stmt):
        import_node = ast.AST.Import()
        import_node.path = stmt['path']
        if 'as' in stmt:
            import_node.alias = self.parse_import_alias(stmt['as'])
        return import_node
```

contract.py

Post Implementation: Slither Analysis Output



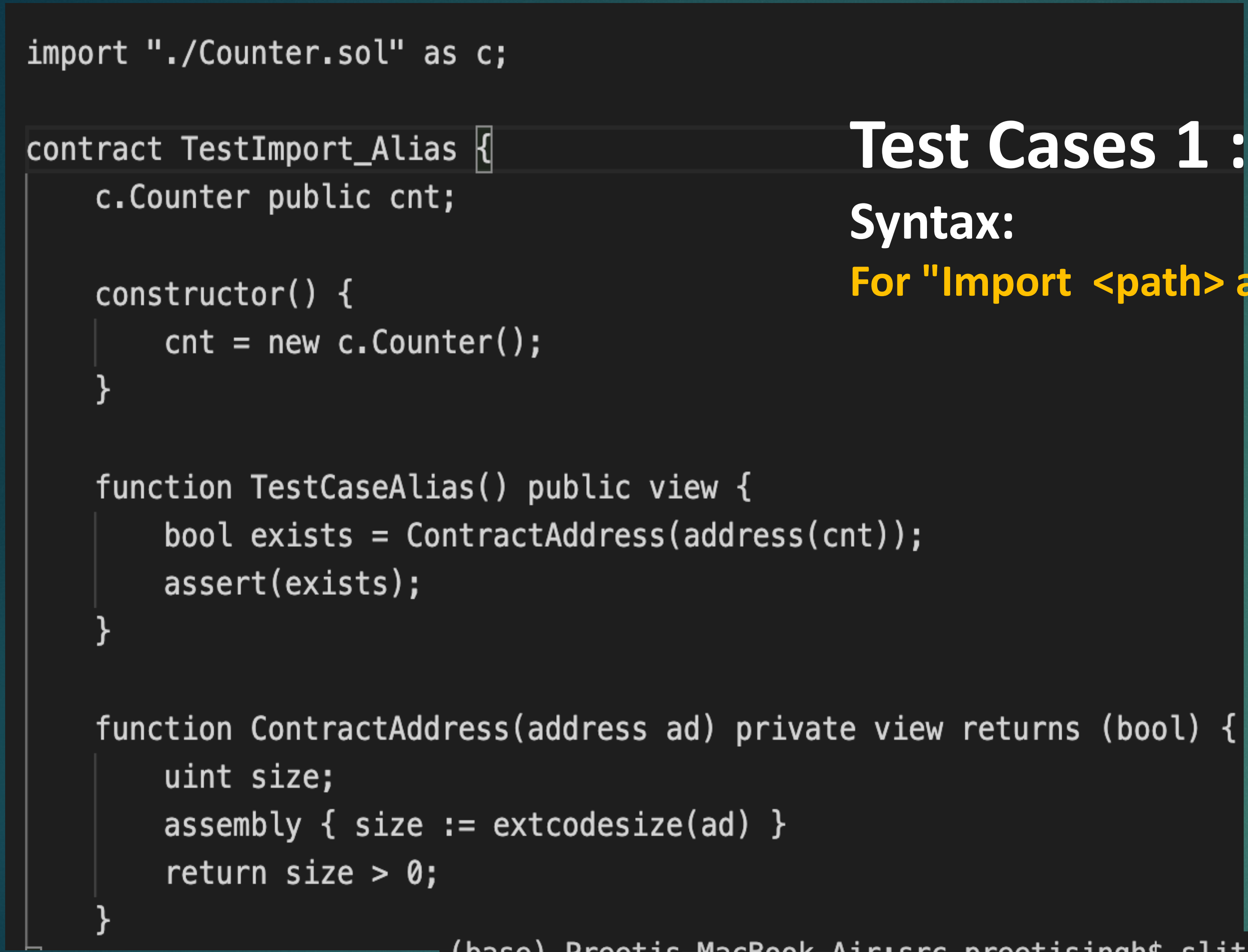


```
● (base) Preetis-MacBook-Air:slither_bug_example-master preetisingh$ cd src/
⊗ (base) Preetis-MacBook-Air:src preetisingh$ slither Importer.sol
  ERROR:ContractSolcParsing:Impossible to generate IR for Importer.constructor (Importer.sol#9-12):
   'NoneType' object has no attribute 'references'
  INFO:Detectors:
  Pragma version^0.8.9 (Counter.sol#2) allows old versions
  Pragma version^0.8.9 (Importer.sol#3) allows old versions
  solc-0.8.19 is not recommended for deployment
  Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
  INFO:Slither:Importer.sol analyzed (2 contracts with 84 detectors), 3 result(s) found
○ (base) Preetis-MacBook-Air:src preetisingh$ 
```

Test Cases 1 :

Syntax:

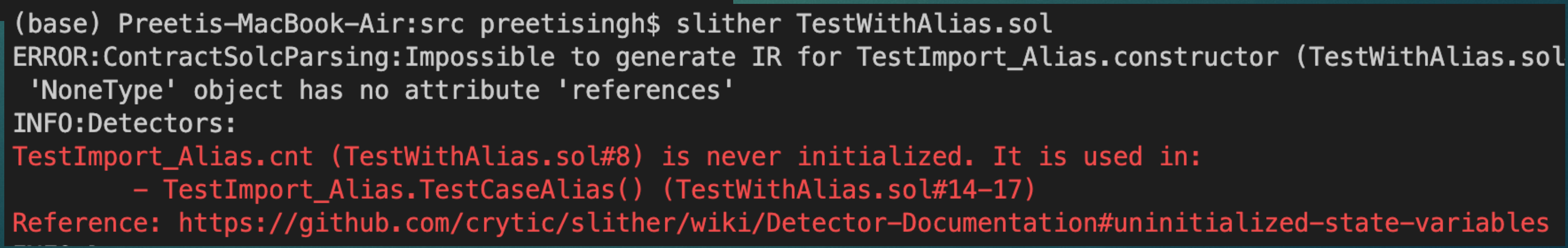
For "Import <path> as <alias>"



```
import "./Counter.sol" as c;

contract TestImport_Alias {
    c.Counter public cnt;

    constructor() {
        cnt = new c.Counter();
    }

    function TestCaseAlias() public view {
        bool exists = ContractAddress(address(cnt));
        assert(exists);
    }

    function ContractAddress(address ad) private view returns (bool) {
        uint size;
        assembly { size := extcodesize(ad) }
        return size > 0;
    }
```

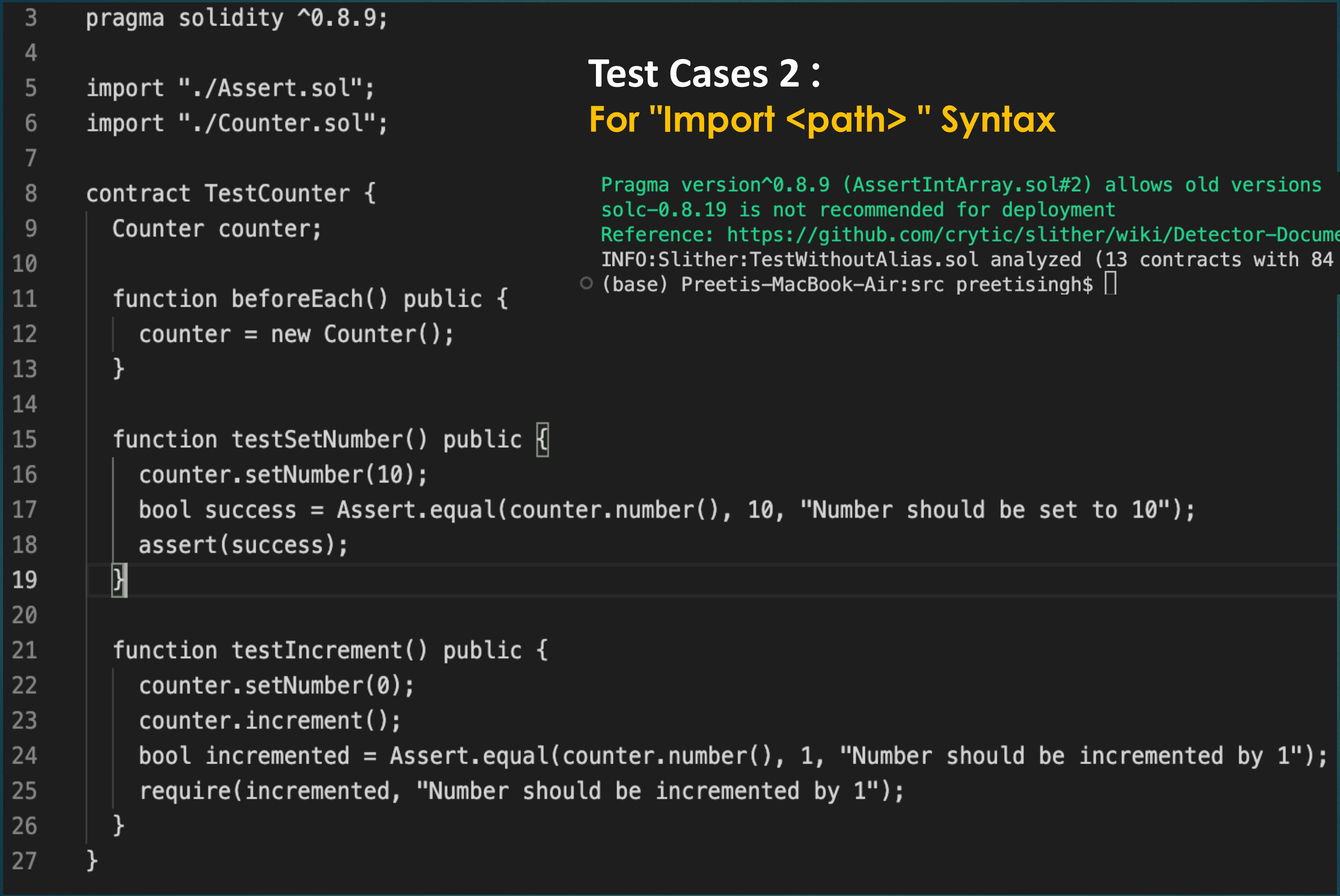
Output

Error : Due to Alias Issue unable to generate IR

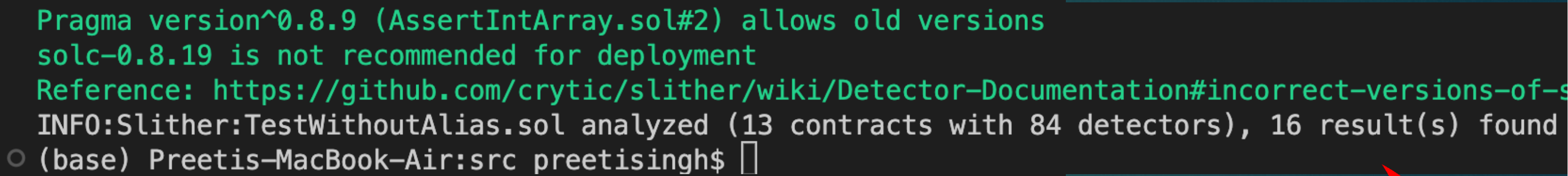


```
(base) Preetis-MacBook-Air:src preetisingh$ slither TestWithAlias.sol
ERROR:ContractSolcParsing:Impossible to generate IR for TestImport_Alias.constructor (TestWithAlias.sol
 'NoneType' object has no attribute 'references'
INFO:Detectors:
TestImport_Alias.cnt (TestWithAlias.sol#8) is never initialized. It is used in:
        - TestImport_Alias.TestCaseAlias() (TestWithAlias.sol#14-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
```

Test Cases 2 :
For "Import <path> " Syntax



```solidity
pragma solidity ^0.8.9;

import "./Assert.sol";
import "./Counter.sol";

contract TestCounter {
  Counter counter;

  function beforeEach() public {
    counter = new Counter();
  }

  function testSetNumber() public {
    counter.setNumber(10);
    bool success = Assert.equal(counter.number(), 10, "Number should be set to 10");
    assert(success);
  }

  function testIncrement() public {
    counter.setNumber(0);
    counter.increment();
    bool incremented = Assert.equal(counter.number(), 1, "Number should be incremented by 1");
    require(incremented, "Number should be incremented by 1");
  }
}
```



```
Pragma version^0.8.9 (AssertIntArray.sol#2) allows old versions
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-
INFO:Slither:TestWithoutAlias.sol analyzed (13 contracts with 84 detectors), 16 result(s) found
(base) Preetis-MacBook-Air:src preetisingh$ 
```

Output :

The Test Case for Smart Contract "Counter.sol" is executed without any error

Risk management

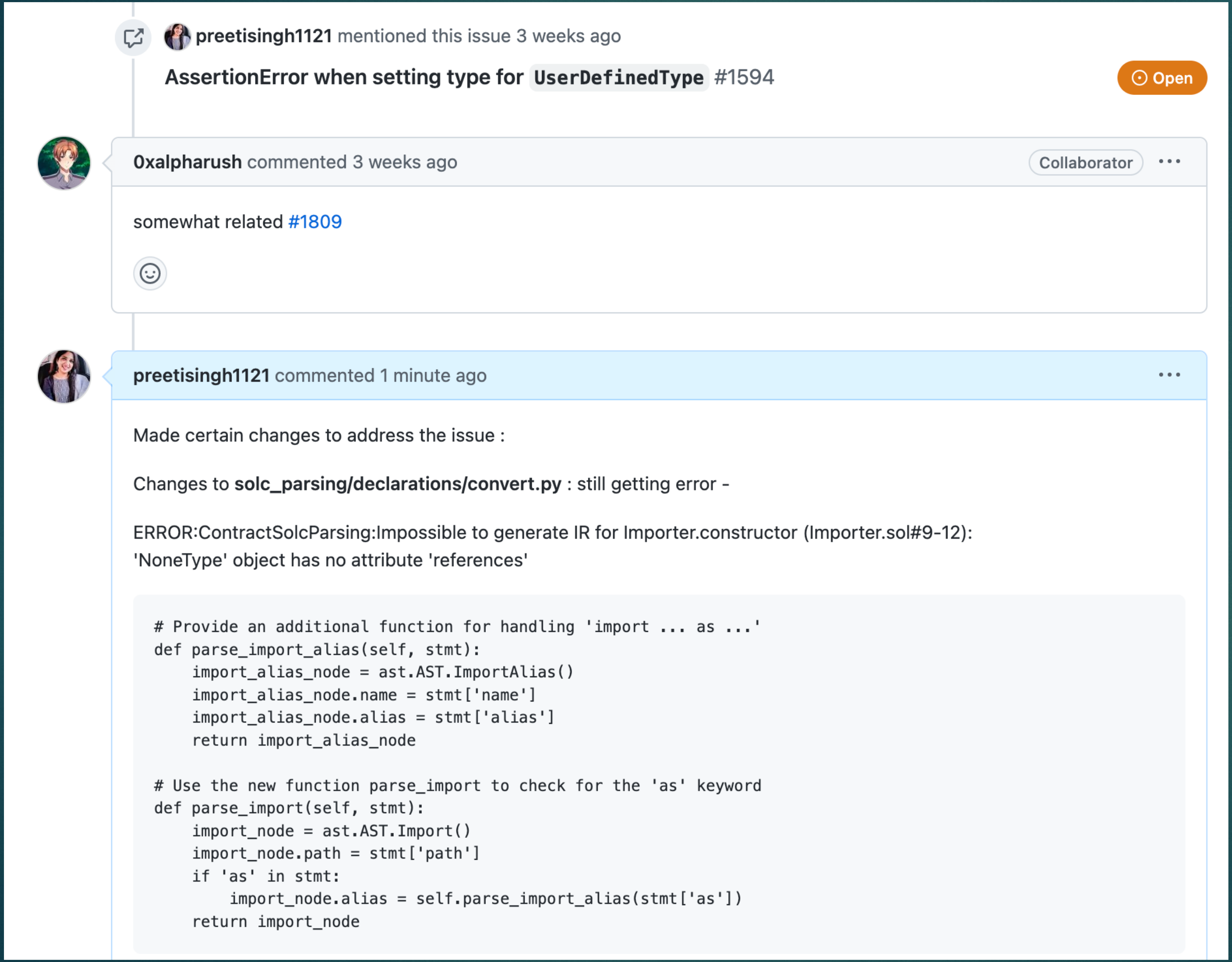
Sr no	Risk	Risk Exposure
1	The project might be delayed if an iteration is finished later than expected.	P= 10% E=10; R.E= 1 hours
2	Unforeseen error may make resolving the original issue more challenging.	P= 10% E=15; R.E= 1.5 hours
3	The team may need to spend more time because they are not aware of possible unknown related issues.	P= 15% E=30, R.E= 4.5 hours
4	If teammates absences occur, the development and timelines may suffer.	P= 5% E=10; R.E= 0.5 hours

Mitigated Risks

- Dividing the total issue into small chunks and working on them helped to reduce the risk exposure.
- Mastered the challenge of virtual machine setup
- Slither tool installation on each team individual's computer.
- Ability to tackle the problem at once, cutting down on time spent and risk exposure

https://cdni.iconscout.com/illustration/premium/thumb/business-risk-management-5540365-4626513.png

Customer Interaction:



preetisingh1121 mentioned this issue 3 weeks ago

AssertionError when setting type for `UserDefinedType` #1594

Open

0xalpharush commented 3 weeks ago

Collaborator

somewhat related #1809

preetisingh1121 commented 1 minute ago

Made certain changes to address the issue :

Changes to **solc_parsing/declarations/convert.py** : still getting error -

ERROR:ContractSolcParsing:Impossible to generate IR for Importer.constructor (Importer.sol#9-12): 'NoneType' object has no attribute 'references'

```
# Provide an additional function for handling 'import ... as ...'
def parse_import_alias(self, stmt):
    import_alias_node = ast.AST.ImportAlias()
    import_alias_node.name = stmt['name']
    import_alias_node.alias = stmt['alias']
    return import_alias_node

# Use the new function parse_import to check for the 'as' keyword
def parse_import(self, stmt):
    import_node = ast.AST.Import()
    import_node.path = stmt['path']
    if 'as' in stmt:
        import_node.alias = self.parse_import_alias(stmt['as'])
    return import_node
```

GitHub Link

https://github.com/preetisingh1121/Slither

References

1. https://arxiv.org/pdf/1908.09878.pdf

2. https://blog.trailofbits.com/2019/05/27/slither-the-leading-static-analyzer-for-smart- contracts/

3. https://github.com/crytic/slither/issues/1452

4. https://github.com/SheldonHolmgren/slither_bug_example/tree/master/src

5. https://github.com/crytic/slither/issues/1594

6. https://remix.ethereum.org/