Practical Unit Tests

(A /\ B) \/ (B \/ C)
"A and B" or "B or C"

A	B	C	(A /\ B)	(B \/ C)	(A /\ B) \/ (B \/ C)
T	T	T	T	T	T
T	T	F	T	T	T
T	F	T	F	T	T
T	F	F	F	F	F
F	T	T	F	T	T
F	T	F	F	T	T
F	F	T	F	T	T
F	F	F	F	F	F

(A /\ -C) \/ -(B /\ C)
"A and NOT C" or NOT "B and C"

A	B	C	-C	(A /\ -C)	-(B /\ C)	(A /\ -C) \/ -(B /\ C)
T	T	T	F	F	F	F
T	T	F	T	T	T	T
T	F	T	F	F	T	T
T	F	F	T	T	T	T
F	T	T	F	F	F	F
F	T	F	T	F	T	T
F	F	T	F	F	T	T
F	F	F	T	F	T	T

Code

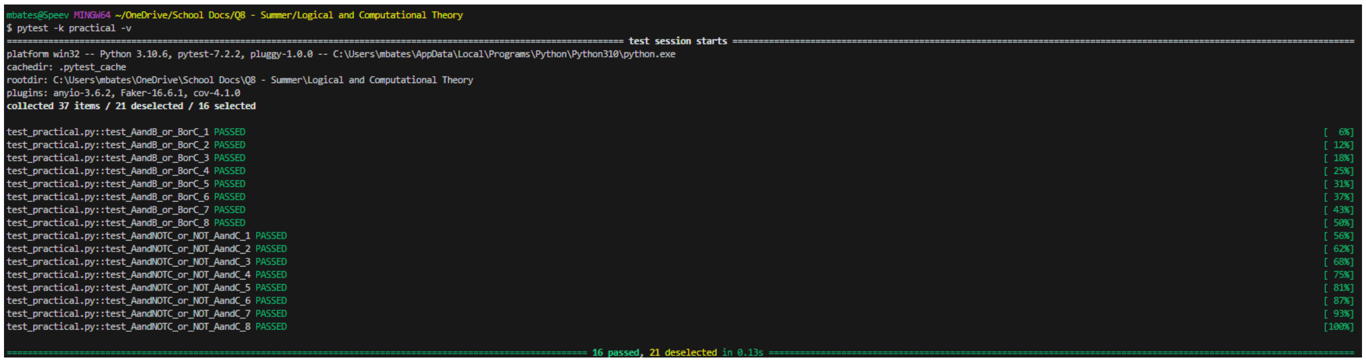
```
class PracticalUnitTests:
    # In retrospect I do not know why I made functions for these instead of just
using the operators
    def Not(self, a: bool):
        return not a

    def And(self, a: bool, b: bool):
        return a and b

    def Or(self, a: bool, b: bool):
        return a or b

    def AandB_or_BorC(self, a: bool, b: bool, c: bool):
        return self.Or(self.And(a, b), self.Or(b, c))

    def AandNOTC_or_NOT_AandC(self, a: bool, b: bool, c: bool):
        return self.Or(self.And(a, self.Not(c)), self.Not(self.And(b, c)))
```

Tests

```
from unitTestLogic.thingsBeingTested import PracticalUnitTests

def test_AandB_or_BorC_1():
    practical = PracticalUnitTests()
    assert practical.AandB_or_BorC(True, True, True) == True

def test_AandB_or_BorC_2():
    practical = PracticalUnitTests()
    assert practical.AandB_or_BorC(True, True, False) == True

def test_AandB_or_BorC_3():
    practical = PracticalUnitTests()
    assert practical.AandB_or_BorC(True, False, True) == True

def test_AandB_or_BorC_4():
    practical = PracticalUnitTests()
    assert practical.AandB_or_BorC(True, False, False) == False

def test_AandB_or_BorC_5():
    practical = PracticalUnitTests()
    assert practical.AandB_or_BorC(False, True, True) == True

def test_AandB_or_BorC_6():
    practical = PracticalUnitTests()
    assert practical.AandB_or_BorC(False, True, False) == True

def test_AandB_or_BorC_7():
    practical = PracticalUnitTests()
    assert practical.AandB_or_BorC(False, False, True) == True
```

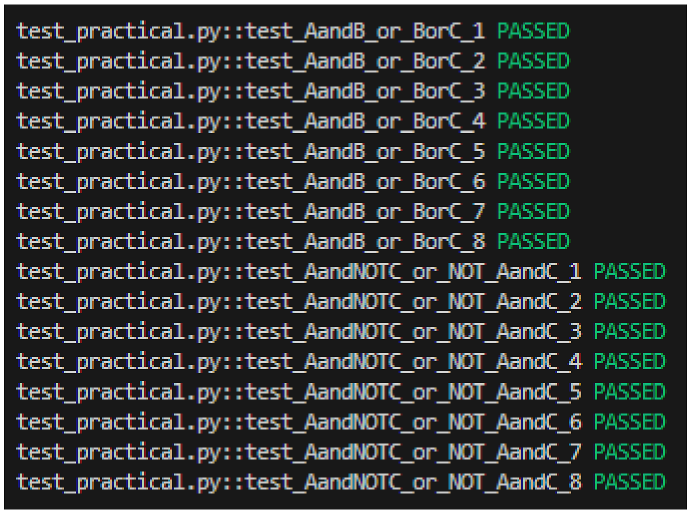
```python
def test_AandB_or_BorC_8():
    practical = PracticalUnitTests()
    assert practical.AandB_or_BorC(False, False, False) == False

# ---------------------------------------------- DELINEATION ----------------------------------------------

def test_AandNOTC_or_NOT_AandC_1():
    practical = PracticalUnitTests()
    assert practical.AandNOTC_or_NOT_AandC(True, True, True) == False

def test_AandNOTC_or_NOT_AandC_2():
    practical = PracticalUnitTests()
    assert practical.AandNOTC_or_NOT_AandC(True, True, False) == True

def test_AandNOTC_or_NOT_AandC_3():
    practical = PracticalUnitTests()
    assert practical.AandNOTC_or_NOT_AandC(True, False, True) == True

def test_AandNOTC_or_NOT_AandC_4():
    practical = PracticalUnitTests()
    assert practical.AandNOTC_or_NOT_AandC(True, False, False) == True

def test_AandNOTC_or_NOT_AandC_5():
    practical = PracticalUnitTests()
    assert practical.AandNOTC_or_NOT_AandC(False, True, True) == False

def test_AandNOTC_or_NOT_AandC_6():
    practical = PracticalUnitTests()
    assert practical.AandNOTC_or_NOT_AandC(False, True, False) == True

def test_AandNOTC_or_NOT_AandC_7():
    practical = PracticalUnitTests()
    assert practical.AandNOTC_or_NOT_AandC(False, False, True) == True

def test_AandNOTC_or_NOT_AandC_8():
    practical = PracticalUnitTests()
    assert practical.AandNOTC_or_NOT_AandC(False, False, False) == True
```

Test Results



```
mbates@Speev MINGW64 ~/OneDrive/School Docs/Q8 - Summer/Logical and Computational Theory
$ pytest -k practical -v
==================== test session starts ====================
platform win32 -- Python 3.10.6, pytest-7.2.2, pluggy-1.0.0 -- C:\Users\mbates\AppData\Local\Programs\Python\Python310\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\mbates\OneDrive\School Docs\Q8 - Summer\Logical and Computational Theory
plugins: anyio-3.6.2, Faker-16.6.1, cov-4.1.0
collected 37 items / 21 deselected / 16 selected

test_practical.py::test_AandB_or_BorC_1 PASSED                    [  6%]
test_practical.py::test_AandB_or_BorC_2 PASSED                    [ 12%]
test_practical.py::test_AandB_or_BorC_3 PASSED                    [ 18%]
test_practical.py::test_AandB_or_BorC_4 PASSED                    [ 25%]
test_practical.py::test_AandB_or_BorC_5 PASSED                    [ 31%]
test_practical.py::test_AandB_or_BorC_6 PASSED                    [ 37%]
test_practical.py::test_AandB_or_BorC_7 PASSED                    [ 43%]
test_practical.py::test_AandB_or_BorC_8 PASSED                    [ 50%]
test_practical.py::test_AandNOTC_or_NOT_AandC_1 PASSED            [ 56%]
test_practical.py::test_AandNOTC_or_NOT_AandC_2 PASSED            [ 62%]
test_practical.py::test_AandNOTC_or_NOT_AandC_3 PASSED            [ 68%]
test_practical.py::test_AandNOTC_or_NOT_AandC_4 PASSED            [ 75%]
test_practical.py::test_AandNOTC_or_NOT_AandC_5 PASSED            [ 81%]
test_practical.py::test_AandNOTC_or_NOT_AandC_6 PASSED            [ 87%]
test_practical.py::test_AandNOTC_or_NOT_AandC_7 PASSED            [ 93%]
test_practical.py::test_AandNOTC_or_NOT_AandC_8 PASSED            [100%]

==================== 16 passed, 21 deselected in 0.13s ====================
```

Test Results Zoomed In



```
test_practical.py::test_AandB_or_BorC_1 PASSED
test_practical.py::test_AandB_or_BorC_2 PASSED
test_practical.py::test_AandB_or_BorC_3 PASSED
test_practical.py::test_AandB_or_BorC_4 PASSED
test_practical.py::test_AandB_or_BorC_5 PASSED
test_practical.py::test_AandB_or_BorC_6 PASSED
test_practical.py::test_AandB_or_BorC_7 PASSED
test_practical.py::test_AandB_or_BorC_8 PASSED
test_practical.py::test_AandNOTC_or_NOT_AandC_1 PASSED
test_practical.py::test_AandNOTC_or_NOT_AandC_2 PASSED
test_practical.py::test_AandNOTC_or_NOT_AandC_3 PASSED
test_practical.py::test_AandNOTC_or_NOT_AandC_4 PASSED
test_practical.py::test_AandNOTC_or_NOT_AandC_5 PASSED
test_practical.py::test_AandNOTC_or_NOT_AandC_6 PASSED
test_practical.py::test_AandNOTC_or_NOT_AandC_7 PASSED
test_practical.py::test_AandNOTC_or_NOT_AandC_8 PASSED
```