

## Mutation

```
# Unary functions have a higher precedence than binary operators
- unary_precedence = max(precedence.values()) + 1
+ unary_precedence = max(precedence.values()) - 1
```

## Mutant survived. Adding tests:

```
class UnaryOperatorPrecedenceTestCase(TestCase):

    def test_unary_precedence_multiplication(self):
        result = mathparse.parse('sqrt(4) * 2')
        self.assertEqual(result, 4.0)

    def test_unary_precedence_division(self):
        result = mathparse.parse('sqrt(4) / 2')
        self.assertEqual(result, 1.0)

    def test_unary_precedence_decimal(self):
        result = mathparse.parse('sqrt(4) . 5')
        self.assertEqual(result, 2.5)
```

## Test failed:

```
===== FAILURES =====
----- UnaryOperatorPrecedenceTestCase.test_unary_precedence_decimal -----
self = <tests.test_binary_operations.UnaryOperatorPrecedenceTestCase testMethod=test_unary_precedence_decimal>

    def test_unary_precedence_decimal(self):
        result = mathparse.parse('sqrt(4) . 5')
        self.assertEqual(result, 2.5)
>
E       AssertionError: 2.1213203435596424 != 2.5

tests\test_binary_operations.py:105: AssertionError
===== short test summary info =====
FAILED tests/test_binary_operations.py::UnaryOperatorPrecedenceTestCase::test_unary_precedence_decimal - AssertionError:
2.1213203435596424 != 2.5
===== 1 failed, 168 passed in 0.84s =====
```

## Mutation:

```
if all_prev_non_math and i + 1 < len(tokens):  
-         next_token = tokens[i + 1]  
+         next_token = tokens[i * 1]  
         if (  
             is_int(next_token) or is_float(next_token)  
or  
             is_constant(next_token) or  
is_unary(next_token) or
```

## Mutant survived. Adding test:

```
def test_last_and_operator(self):  
    result = mathparse.extract_expression("Please calculate + 3  
and 5", language='ENG')  
    self.assertEqual(result, "+ 3 and 5")
```

## Mutant killed:

```
===== FAILURES =====  
----- WordBasedDecimalTestCase.test_last_and_operator -----  
self = <tests.test_binary_operations.WordBasedDecimalTestCase testMethod=test_last_and_operator>  
  
def test_last_and_operator(self):  
    result = mathparse.extract_expression("Please calculate + 3 and 5", language='ENG')  
    self.assertEqual(result, "+ 3 and 5")  
+   AssertionError: '3 and 5' != '+ 3 and 5'  
- 3 and 5  
+ + 3 and 5  
? ++  
  
tests\test_binary_operations.py:196: AssertionError  
===== short test summary info =====  
FAILED tests/test_binary_operations.py::WordBasedDecimalTestCase::test_last_and_operator - AssertionError: '3 and 5' !=  
+ 3 and 5'  
===== 1 failed, 170 passed in 0.77s =====
```

### Mutant:

```
if all_prev_non_math and i + 1 < len(tokens):  
-         next_token = tokens[i + 1]  
+         next_token = tokens[i / 1]  
         if (  
             is_int(next_token) or is_float(next_token)  
or  
             is_constant(next_token) or  
is_unary(next_token) or
```

Mutant survived. Mutant was killed with previously added test.