White-box testing and Unit test.

Quality Assurance and Software testing.

Doc. Mustafa Assaf

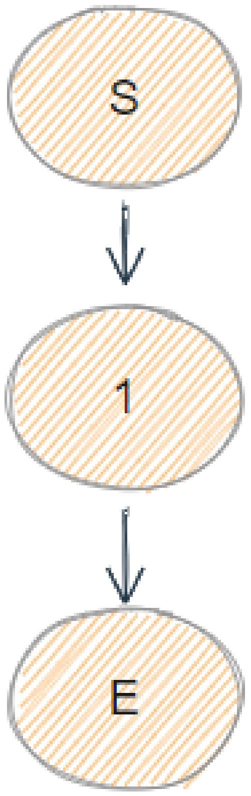
Zubaida Sadder - 11925174

GitHub Repository

Table of Contents

Add method

Flow Graph Analysis



List of Independent paths

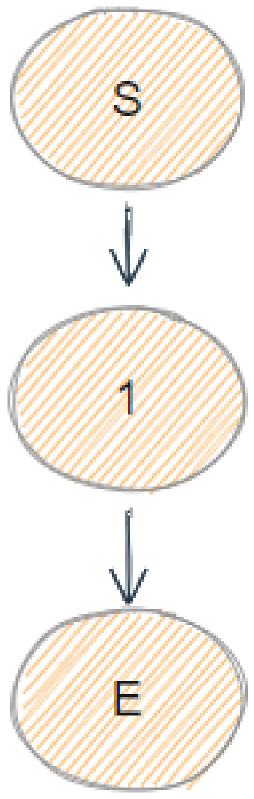
S is the start node, E is the end node.
P1 : [S, 1, E]

Test Case for each path

Test case ID	Test case title	Test data	Status	Path Id
1	Add positives	10 + 10 = 20	Success	P1
2	Add negatives	-10 + -10 = -20	Success	P1
3	Add positive + negative	10 + -20 = -10	Success	P1

Subtract method

Flow Graph Analysis



List of Independent paths

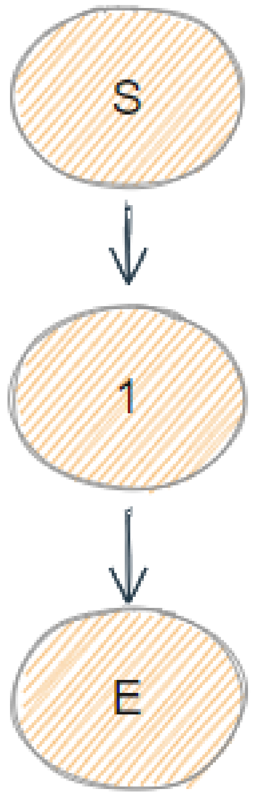
S is the start node, E is the end node.
P1 : [S, 1, E]

Test Case for each path

Test case ID	Test case title	Test data	Status	Path Id
1	Subtract positives	20 - 10 = 10	Success	P1
2	Subtract negatives	-10 - -10 = 0	Success	P1
3	Subtract positive - negative	30 - -20 = 50	Success	P1
4	Subtract negative - positive	-20 - 30 = -50	Success	P1

Multiply method

Flow Graph Analysis



List of Independent paths

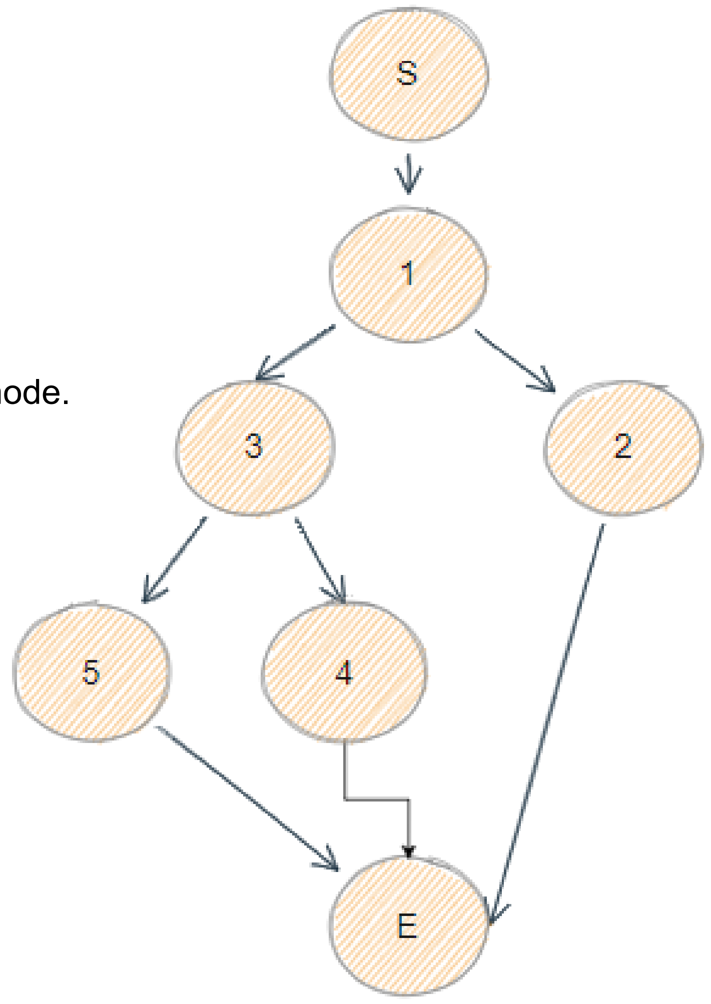
S is the start node, E is the end node.
P1 : [S, 1, E]

Test Case for each path

Test case ID	Test case title	Test data	Status	Path Id
1	Multiply positives	3 * 3 = 9	Success	P1
2	Multiply negatives	-3 * -3 = 9	Success	P1
3	Multiply positive * negative	3 * -3 = -9	Success	P1

Divide method

Flow Graph Analysis



List of Independent paths

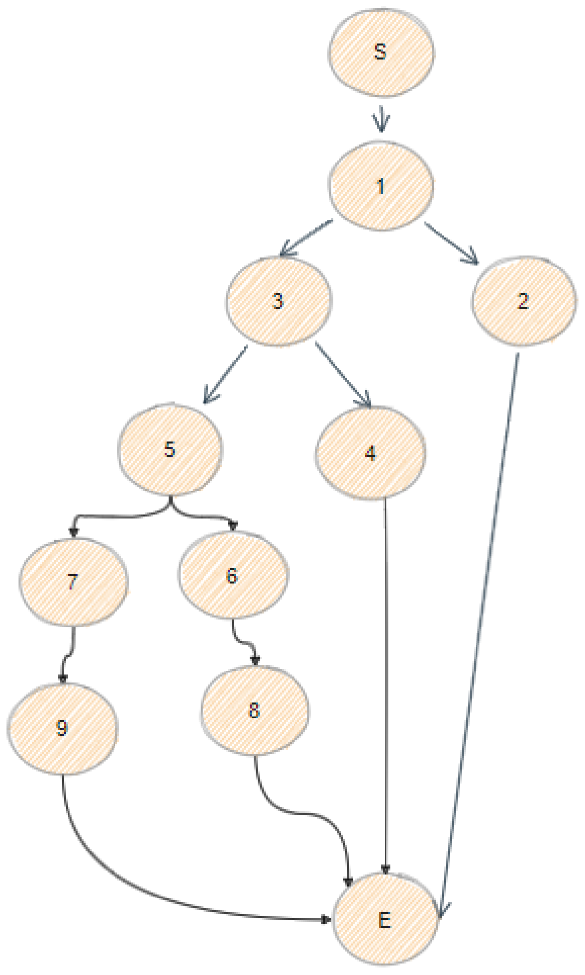
S is the start node, E is the end node.
P1: [S,1, 2, E].
P2: [S, 1, 3, 4, E].
P3: [S, 1, 3, 5, E.]

Test Case for each path

Test case ID	Test case title	Test data	Status	Path Id
1	Division of positives	3 / 3 = 1	Success	P3
2	Division of negatives	-3 / -3 = 1	Success	P3
3	Division of positive / negative	3 / -3 = -1	Success	P3
4	Division of negative / positive	-3 / 3 = -1	Success	P3
5	Division of num/zero	3 / 0 = ERROR	Success	P1
6	Division of zero /num	0 / 3 = 0	Success	P2

Check user input method

Flow Graph Analysis



List of Independent paths

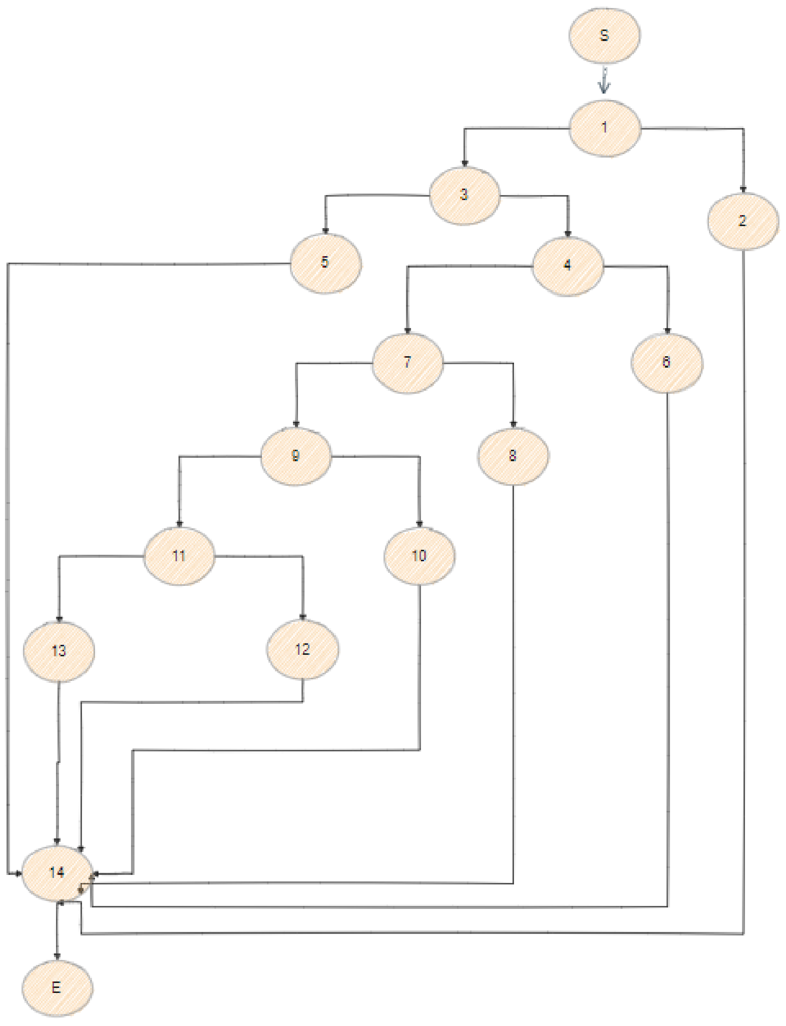
S is the start node, E is the end node.
P1: [S,1, 2, E.]
P2: [S, 1, 3, 4, E].
P3: [S, 1, 3, 5, 6, 8, E].
P4: [S, 1, 3, 5, 7, 9, E].

Test Case for each path

Test case ID	Test case title	Test data	Status	Path Id
1	Empty input	‘ ‘	Success	P1
2	Float input to int input	‘10’	Success	P3
3	Int input to float input	‘10.2’	Success	P2
4	String input	‘hi’	Success	P4

Calculate method

Flow Graph Analysis



List of Independent paths

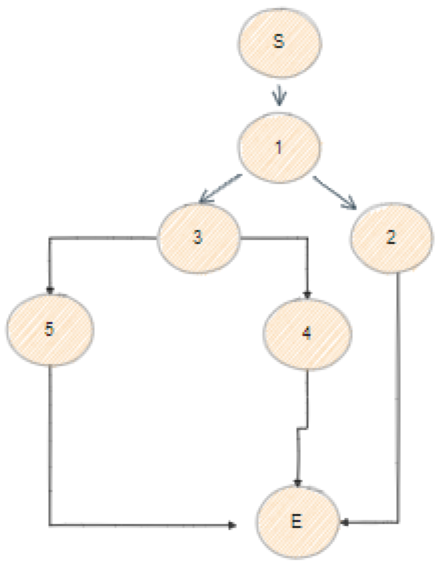
S is the start node
E is the end node.
P1: [S,1, 2, 14, E.]
P2: [S, 1, 3, 4, 6, 14, E].
P3: [S, 1, 3, 4, 7, 8, 14, E].
P4: [S, 1, 3, 4, 7, 9, 10, 14, E].
P5: [S, 1, 3, 4, 7, 9, 11, 12, 14, E].
P6: [S, 1, 3, 4, 7, 9, 11, 13, 14, E].
P7 : [S, 1, 3, 5, 14, E].

Test Case for each path

Test case ID	Test case title	Test data	Status	Path Id
1	Null num1 and num2	‘1’ ,’’,’’	Success	P1
2	Choice not in (1,2,3,4)	‘10’,’1’,’2’	Success	P7
3	Choice 1	‘1’,’1’,’1’	Success	P3
4	Choice 2	‘2’,’3’,’2’	Success	P4
5	Choice 3	‘3’,’3’,’2’	Success	P5
6	Choice 4 and num != 0	‘4’,’1’,’2’	Success	P6
7	Choice 4 and num == 0	‘4’,’1’,’0’	Success	P6
8	Invalid choice	‘x’,’1’,’2’	Success	P7

isExit method

Flow Graph Analysis



List of Independent paths

S is the start node, E is the end node.
P1: [S,1, 2, E.]
P2: [S, 1, 3, 4, E].
P3: [S, 1, 3, 5, E].

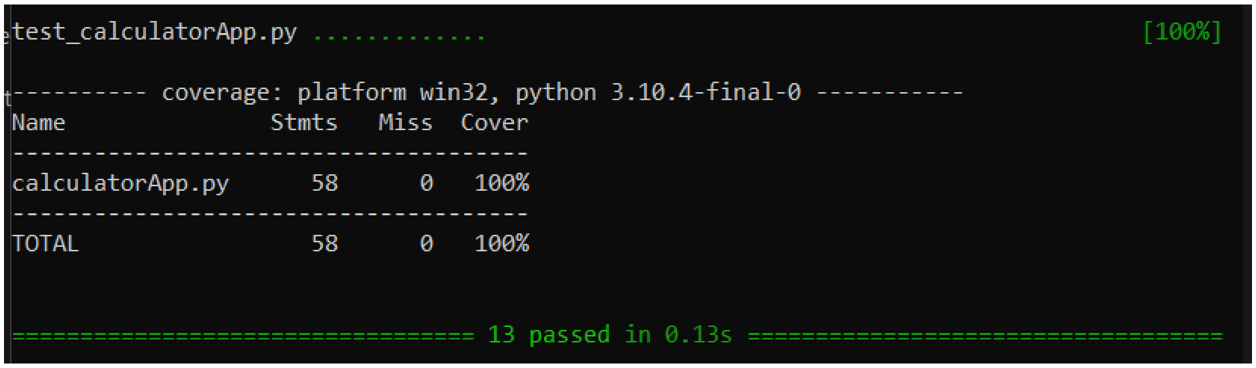
Test Case for each path

Test case ID	Test case title	Test data	Status	Path Id
1	No next calculation	no	Success	P1
2	Yes next calculation	yes	Success	P2
3	Another input	Maybe	Success	P3

Minimal Number of Paths that achieve 100% code coverage

= 20 paths

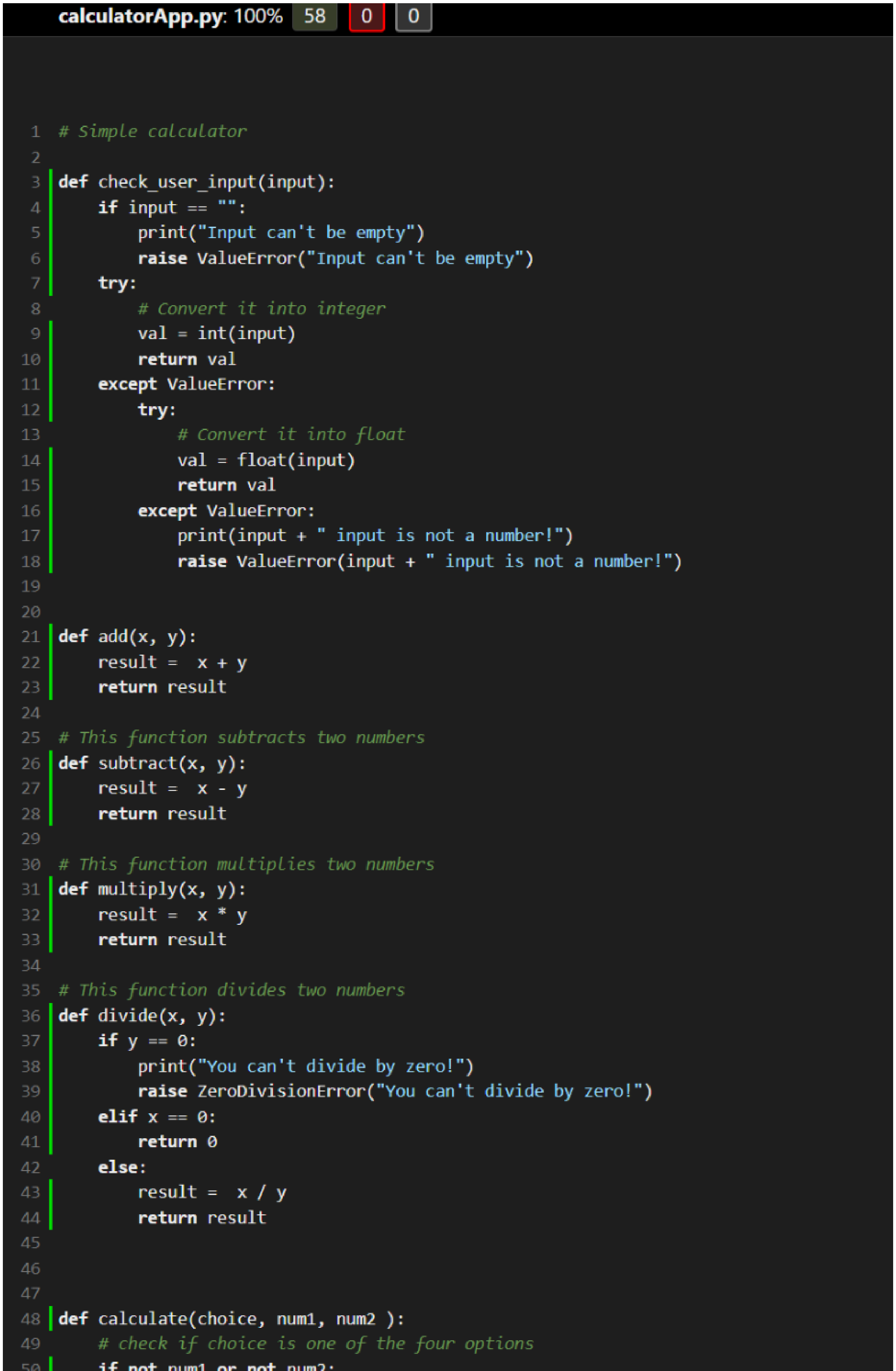
Code coverage run report



```
test_calculatorApp.py .............                                               [100%]

---------- coverage: platform win32, python 3.10.4-final-0 -----------
Name                Stmts   Miss  Cover
---------------------------------------
calculatorApp.py       58      0   100%
---------------------------------------
TOTAL                  58      0   100%


================================== 13 passed in 0.13s ===================================
```


calculatorApp.py: 100% 58 0 0

```python
# Simple calculator

def check_user_input(input):
    if input == "":
        print("Input can't be empty")
        raise ValueError("Input can't be empty")
    try:
        # Convert it into integer
        val = int(input)
        return val
    except ValueError:
        try:
            # Convert it into float
            val = float(input)
            return val
        except ValueError:
            print(input + " input is not a number!")
            raise ValueError(input + " input is not a number!")


def add(x, y):
    result =  x + y
    return result

# This function subtracts two numbers
def subtract(x, y):
    result =  x - y
    return result

# This function multiplies two numbers
def multiply(x, y):
    result =  x * y
    return result

# This function divides two numbers
def divide(x, y):
    if y == 0:
        print("You can't divide by zero!")
        raise ZeroDivisionError("You can't divide by zero!")
    elif x == 0:
        return 0
    else:
        result =  x / y
        return result



def calculate(choice, num1, num2 ):
    # check if choice is one of the four options
    if not num1 or not num2:
```