PES UNIVERSITY

PES UNIVERSITY RR CAMPUS

SOFTWARE ENGINEERING

UE20CS303

CASE STUDY : 4

SECTION : ‘F’

PROJECT TEAM : PR_6

PROJECT TITLE : ADVANCED TEXT EDITOR IN PYTHON

TOPIC : TASK 1, 2 & 3

TEAM MEMBERS :

S/N	NAME	SRN
1	RAHUL ROSHAN G	PES1UG20CS320
2	ROHAN C	PES1UG20CS345
3	ROHIT ROSHAN	PES1UG20CS355
4	S M SUTHARSAN RAJ	PES1UG20CS362

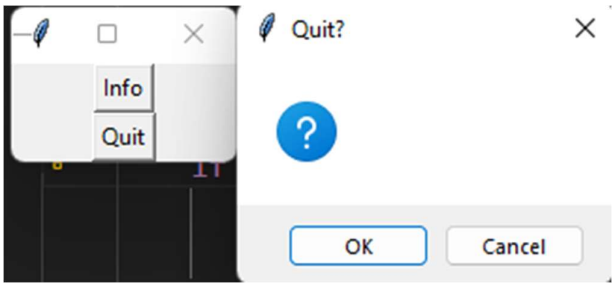
Problem Statement – 1: Mutation Testing

Test 1 and 2

```
import tkinter
from tkinter import font, messagebox, PhotoImage
from tkinter import colorchooser, filedialog
import unittest


class MyGUI(tkinter.Frame):
    def __init__(self, master, **kw):
        tkinter.Frame.__init__(self, master, **kw)
        self.info_button = tkinter.Button(self,
command=self.info_cmd, text='Info')
        self.info_button.pack()
        self.quit_button = tkinter.Button(self,
command=self.quit_cmd, text='Quit')
        self.quit_button.pack()

    def find_sum(self):
        nums = self.find_integers()
        if nums:
            sum_total = sum(int(num) for num in nums)
            messagebox.showinfo('Coastline', f'Sum of
{len(nums)} numbers\nfound is {sum_total}')
        else:
            messagebox.showinfo('Coastline', 'No number
found')



    def info_cmd(self):
        messagebox.showinfo('Info', master=self)

    def quit_cmd(self):
        confirm = messagebox.askokcancel('Quit?',
master=self)
```

```
        if confirm:
            self.destroy()
```

```
----------------------------------------------------------------------
Ran 1 test in 14.219s

FAILED (failures=1)
PS C:\Users\sutha> python -u "c:\Users\sutha\Documents\PESU\SE\PROJECT\CODE\PES_BOIS-TEXT_EDITOR\Coastline - Text Editor\test.py"
.
----------------------------------------------------------------------
Ran 1 test in 4.579s

OK
PS C:\Users\sutha>
```



Test 3

```
import unittest

def addition(x, y):
     return x + y
def subtraction(x, y):
     return x-y

class TestCalc(unittest.TestCase):
    def test_addition(self):
        self.assertEqual(addition(5, 5), 10)
        self.assertEqual(addition(-3, 3), 0)
        self.assertEqual(addition(-5,-5),-10)
    def test_subtraction(self):
        self.assertEqual(subtraction(15, 5), 10)
        self.assertEqual(subtraction(-1, 2),-3)
        self.assertEqual(subtraction(-1,-1), 0)

if __name__ == '__main__':
   unittest.main()
```

```
C:\Users\sutha\Documents\PESU\SE\PROJECT\CODE\PES_BOIS-TEXT_EDITOR\Coastline - Text Editor>py test2.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
```

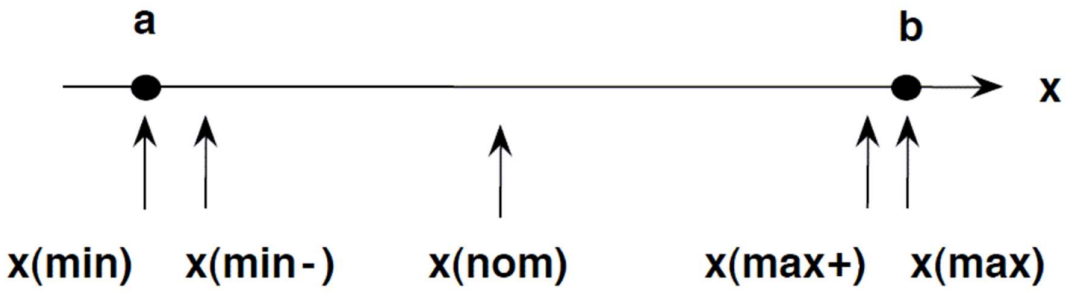
Problem Statement – 2.a: Boundary Value Analysis

When it comes to finding errors in your code base, they are often found at locations where a condition is being tested. Due to this, developers often use Boundary Value tests to reduce defect density.

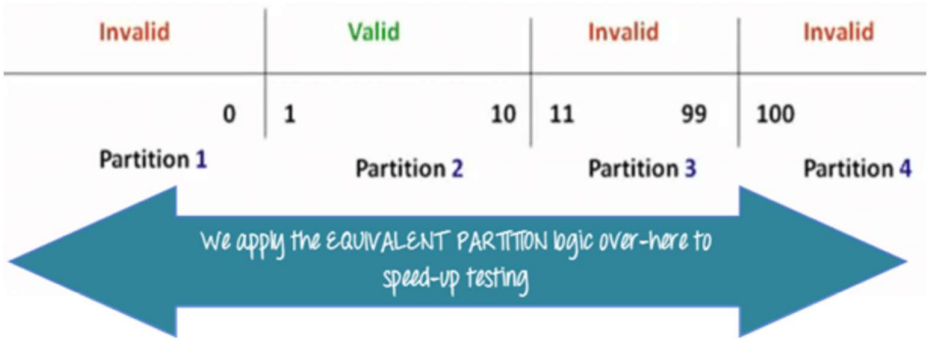
• How would you define a boundary test?

Boundary Value Analysis is based on testing the boundary values of valid and invalid partitions. The behaviour at the edge of the equivalence partition is more likely to be incorrect than the behaviour within the partition, so boundaries are an area where testing is likely to yield defects.

It checks for the input values near the boundary that have a higher chance of error. Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.



Text Editor is built for python language. Python is a strongly typed and dynamic language. This means that the different data types are clearly defined. Also, these datatypes have very well-defined range of values. These values will form the boundary test cases.



Let us define and check the boundary values for different data types in python:

Float:

Minimum Value: -1.8 x 10308

Maximum Value: 1.8 x 10308

If any value greater than the maximum value is given then the editor will automatically set this value to inf (Infinity). Same is applicable to the lower limit.

This will allow for user experience to go ahead smoothly without any error interruptions.

Int:

There is no theoretical maximum limit on the value of the int. User can enjoy an error free experience.

Datetime:

YEAR – MIN = 0000

YEAR – MAX = 9999

TIME – MIN = 00:00:00

TIME – MAX = 23:59:59

DATE – MIN = 1

DATE – MAX = 31

MONTH – MIN = 1

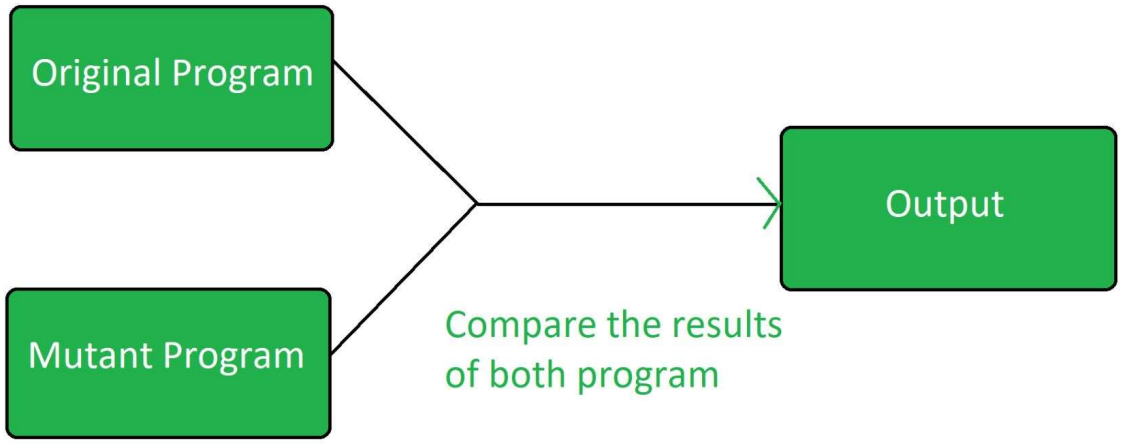
MONTH - MAX = 12

All the values in datetime are strictly ROLLOVER.

If any user enters a value outside the boundary case, it will roll over and ensure that the value is within the boundary case

Example – 24:00:00 will be rolled over to 00:00:00.

Problem Statement – 2.b: Mutation Testing



- **DECISION MUTATION**

MUTATED CODE :-

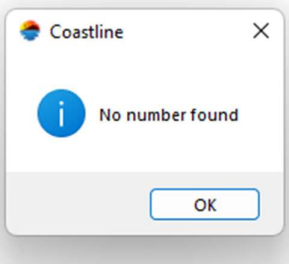
Changing italic to bold and bold to italic

```
def change_font(self, var, indx, mode):
        self.textbox.configure(font=(self.font_family.get(),
self.font_size.get()))

        self.align_text(self.align)

        if self.boldOn:
            self.configure_text('italic')
```

```
        if self.italicOn:
            self.configure_text('bold')
        if self.underlineOn:
            self.configure_text('underline')
        if self.overstrikeOn:
            self.configure_text('strikethrough')
```

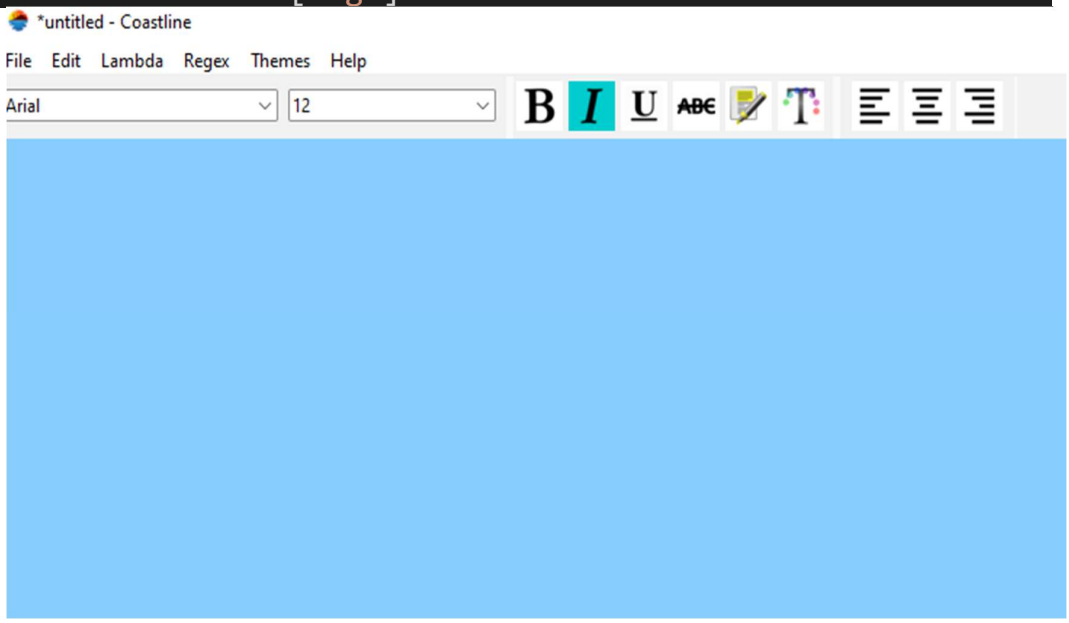
- **STATEMENT MUTATION**

```
def find_sum(self):
        nums = 0
        if nums:
            sum_total = sum(int(num) for num in nums)
            messagebox.showinfo('Coastline', f'Sum of
{len(nums)} numbers\nfound is {sum_total}')
        else:
            messagebox.showinfo('Coastline', 'No number
found')
```



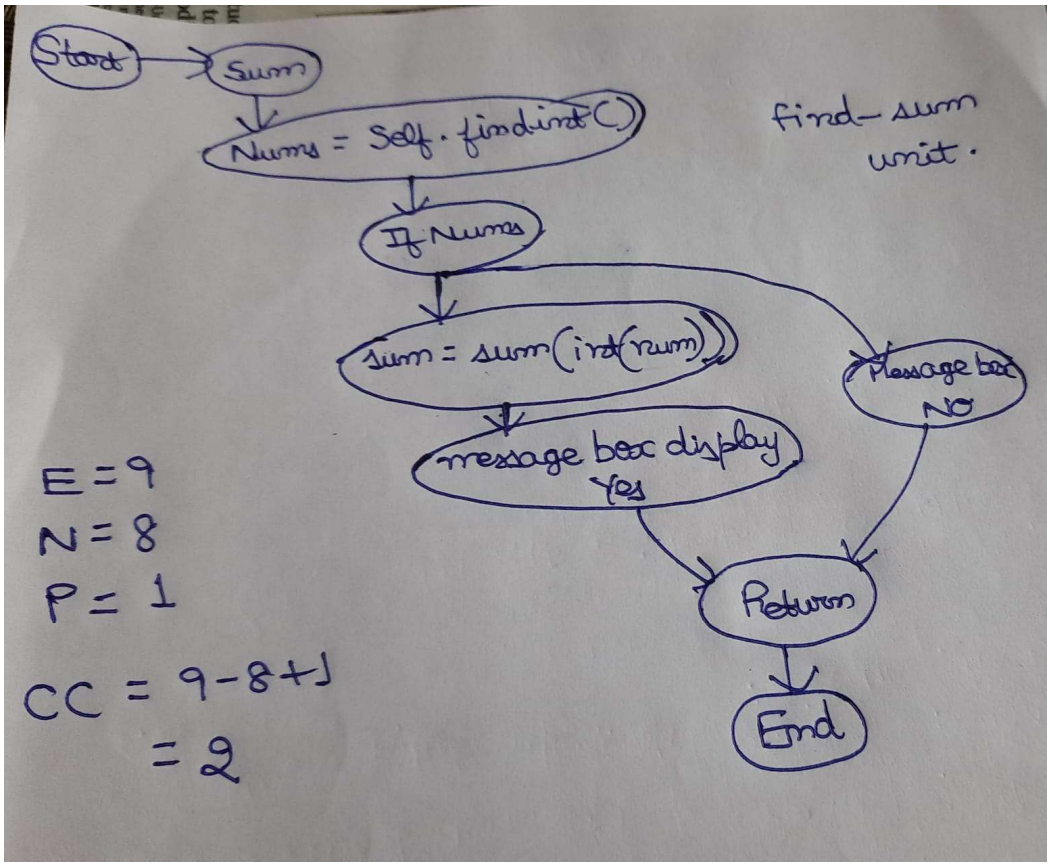
- **VALUE MUTATION**

```
def change_color(self):
        color = colorchooser.askcolor()[1]
        self.textbox['fg'] = color
```

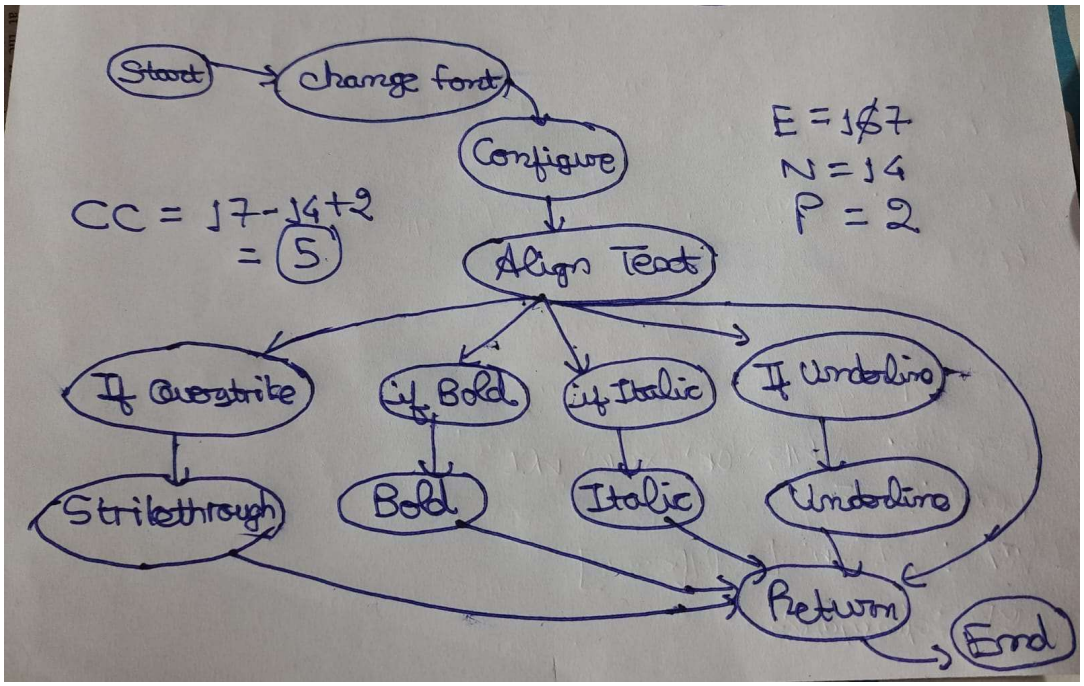


Problem Statement – 3: Static Testing

Selecting **component 1** from Unit test :
CYCLOMETIC COMPLEXITY = 2

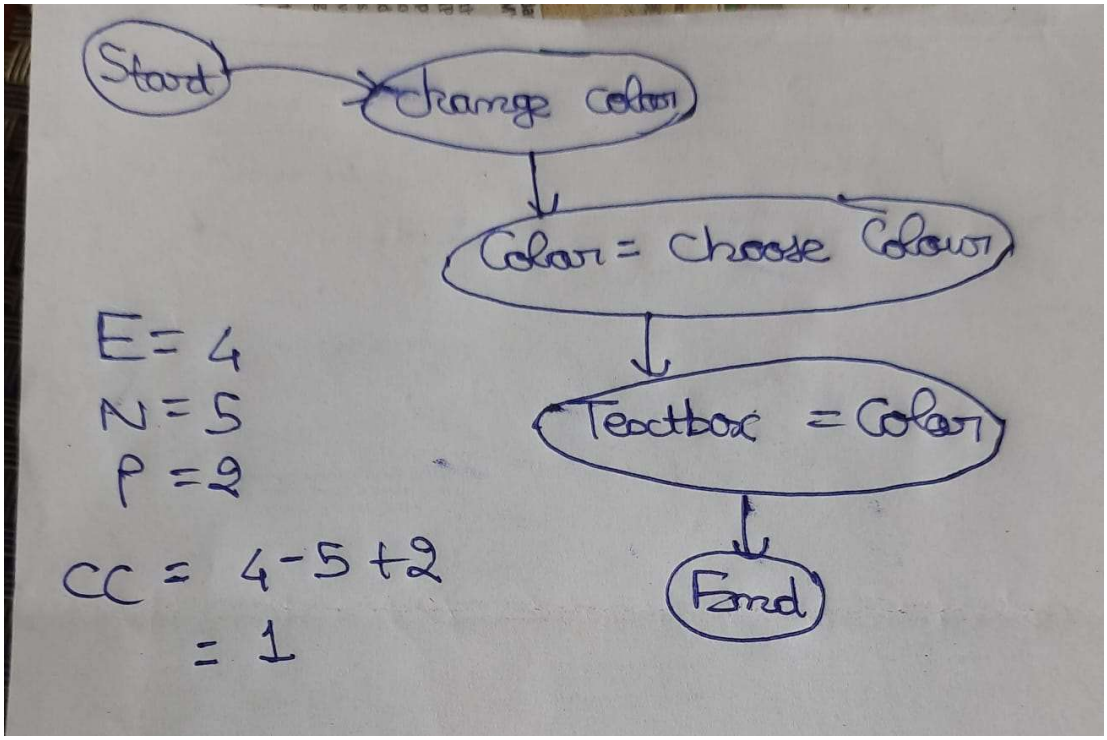


Selecting **component 2** from Mutant test :
CYCLOMETIC COMPLEXITY = 5



Selecting **component 3** from Mutant test :

CYCLOMETIC COMPLEXITY = 1



Since the existing cyclomatic complexity of all the 3 unit codes are less than 10, it is fine to proceed with these snippets of code itself.

THANK YOU