./

Learning Report – PYTHON PROGRAMMING



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be approved By** | **Remarks/Revision Details** |
| 1 |  | Name/PS No | Name/PS No | Module Owner Name | Comments |
| 2 | 22/03/21 | 99003779 |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

Table of Contents

ACTIVITY 1……………………………………………………………………………………………………………………………………………………………………4

STATE OF ART…………………………………………………………………………………………………………………………………………………………….4

COST AND FEATURES…………………………………………………………………………………………………………………………………………………4

DEFINING OUR SYSTEM……………………………………………………………………………………………………………………………………………..5

SWOT ANALYSIS…………………………………………………………………………………………………………………………………………………………5

4W1H………………………………………………………………………………………………………………………………………………………………………..6

UML DIAGRAMS………………………………………………………………………………………………………………………………………………………..6

TEST PLANS……………………………………………………………………………………………………………………………………………………………….9

GITHUB……………………………………………………………………………………………………………………………………………………………………11

GIT ISSUES……………………………………………………………………………………………………………………………………………………………….11

MAIN PROJECT ……………………………………………………………………………………………………………………………………………………….12

GIT COMMITS………………………………………………………………………………………………………………………………………………………….15

[Table of Figures 3](#_Toc64289204)

[Table of Tables](#_Toc64289205) 3

## Table of Figures

[Figure 1 COST VS FEATURES](#_Toc52177314) 5

[Figure 2 CLASS DIAGRAM (HIGH LEVEL)](#_Toc52177315) 6

[Figure 3 ACTIVITY DIAGRAM (HIGH LEVEL)](#_Toc52177316) 6

Figure 4 INTERACTION OVERVIEW (HIGH LEVEL)……………………………………………………………………………………………..7

[Figure 5 OBJECT DIAGRAM (LOW LEVEL)](#_Toc52177317) 7

[Figure 6 SEQUENCE DIAGRAM (LOW LEVEL)](#_Toc52177318) 8

[Figure 7 STATE DIAGRAM (LOW LEVEL)](#_Toc52177319) 8

[Figure 8 GITHUB LINK](#_Toc52177324) 12

[Figure 9 GIT ISSUES](#_Toc52177326) 13

[Figure 10 GIT COMMITS](#_Toc52177327) 22

## Table of Tables

[Table 1 SWOT ANALYSIS](#_Toc52177304) 5

[Table 2 TEST PLAN OF HLR](#_Toc52177305) 9

[Table 3 ADDITION TEST PLAN (LOW LEVEL)](#_Toc52177306) 9

[Table 4 SUBTRACTION TEST PLAN (LOW LEVEL)](#_Toc52177307) 10

[Table 5 MULTIPLICATION TEST PLAN (LOW LEVEL) 1](#_Toc52177308)0

[Table 6 DIVISION TEST PLAN (LOW LEVEL) 1](#_Toc52177309)0

[Table 7 REMAINDER TEST PLAN (LOW LEVEL) 1](#_Toc52177309)1

[Table 8 POWER TEST PLAN (LOW LEVEL) 1](#_Toc52177309)1

[Table 9 PERCENTAGE TEST PLAN (LOW LEVEL) 1](#_Toc52177309)1

[Table 10 GCD TEST PLAN (LOW LEVEL) 1](#_Toc52177309)2

**PYTHON PROGRAMMING**

**Introduction to Python**

**Beginning with python :**

Python was released in 1991 and was created by Guido van Rossum. Python is the most easy and one of the most popular programming language in todays world.

It is commonly used for many purposes which includes the following:

* system scripting,
* mathematics,
* software development,
* web development (server-side),

Why Python is very popular and used by beginners as well professionals:

* python runs on almost all the platforms like Raspberry Pi, Linux, Mac, Windows etc.
* It is basically a both procedural as well as imperative language.
* It also supports object oriented programming.
* It’s syntax is similar to that of English Language
* It runs on a system called interpreter which means that the execution of the code occurs as soon as the code is implemented. And hence the prototype is really very quick.

The very basic program that all the beginners start with is the Hello Program which is written below:

Print(“Hello! World”)

OUTPUT:



In python indentation plays a very important role indication the beginning and end of a block. However other programming languages are having curly braces for blocks.

In python Hash Symbol “#” is used for commenting purpose.

**Variables:**

Python does not have a concept of variable declaration however using the word **Type** we can get the data type of variable.

Following are the built in data types that python supports:

Text Type: str

Numeric Types: int, float, complex

Sequence Types: list, tuple,range

Mapping Type: dict

Set Types: set, frozenset

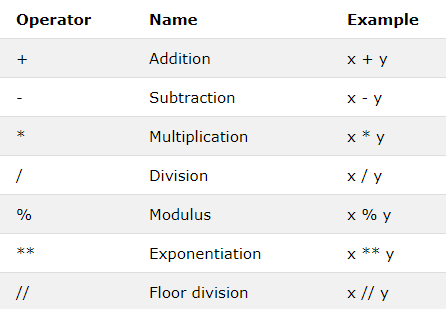
Boolean Type: bool

Binary Types: bytes, byrearray, memoryview.

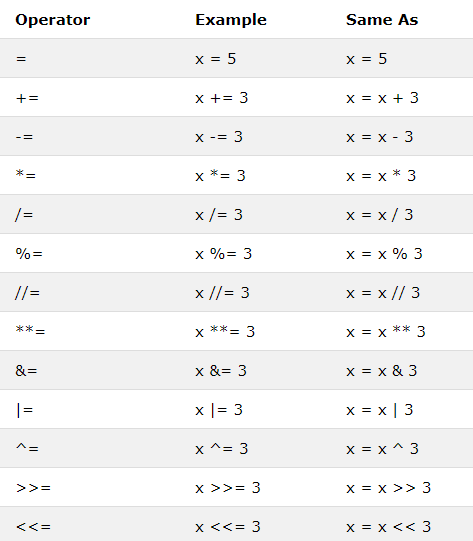
**Operators**

Python supports various operators to perform on various variables and values. Following are the list:

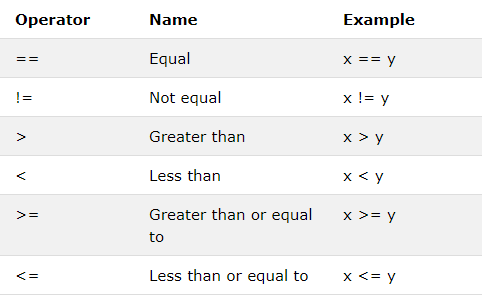
* 1. Arithmetic Operators:



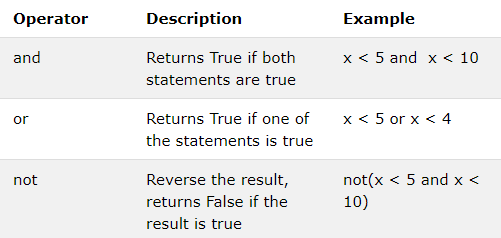
* 1. Assignment operators:



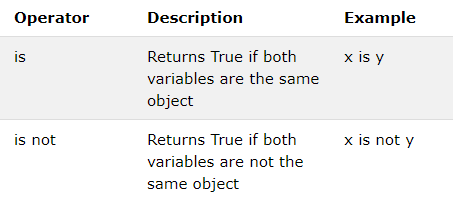
* 1. Comparison Operators:



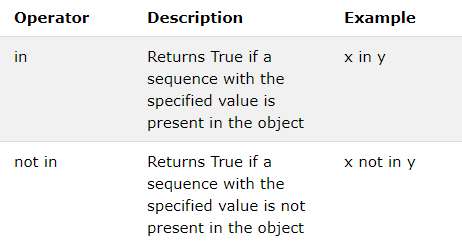
* 1. Logical Operators:



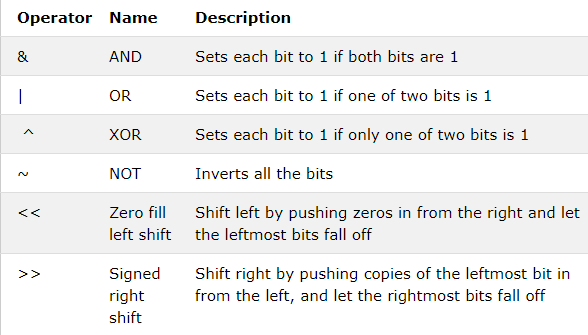
* 1. Identity Operator:



* 1. Membership operator:



* 1. Bitwise Operator:



**List, Tuple, Dictionary and Set:**

**List:**  These are basically used as a single variable to store multiple data. It is an ordered and changeable collection of non repetitive members.

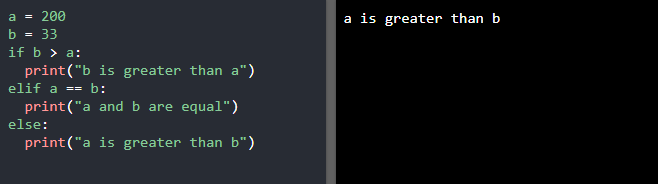
**Tuple:** It is an unchangeable ad ordered collection of repetitive members.

**Dictionary:** It is an unordered and changeable collection of duplicate members.

**Set:** It is an unordered and unindexed collection of repetitive data.

**Conditional Statements:**

1. **If…Else:** If statement is used with an “if” keyword followed by a condition.
2. **Elif:** This statement is executed if the condition in “if” statement is false.
3. **Else:** It is executed when all the previous statements are false.



1. **While statement:** This is executed as long as the condition is true. **Break** and **continue** statements are basically used with while statements.

**Break:** It stops the loop.

**Continue :** It stops the current iteration and continue with the next.

**Functions in Python:**

As like other languages, functions in python are executed only when they are called. Parameters can be passed into the functions which in turn return result. Functions are created using the keyword **def**. Arguments are passed into the function as information.

**\*\*kwargs:** If number of keyword arguments that needs to be passed in a function is undefined then we use the symbol “\*\*” Here the function receives a dictionary of arguments.

**\*args:** If number of arguments that needs to be passed in a function is undefined then we use the symbol “\*” Here the function receives a tuple of arguments.

**Lambda:** It is a small anonymous function**.** It can take any number of arguments but can have only one expression. Its syntax is shown below:

Lamda arguments : expression

**Object oriented Programming :**

As pthon is an object oriented programming language so it is having the concept of class and objects:

**Class:** It is a user defined blueprint or prototype from which objects are created. It basically represents the set of properties or method that are common to all the objects.

**Objects:** It is a basic unit of OOP and represents the real life entities. A python program creates many objects which interact by creating methods.

All classes have a function called \_\_init\_\_() which is executed when the class is being initiated.

**Inheritance:** It is one of the most important concept that an OOP represents.

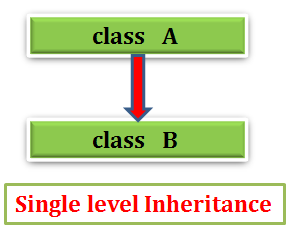
In inheritance parent class and child class are created.

Parent class is basically the super class from which all the child classes are inherited.

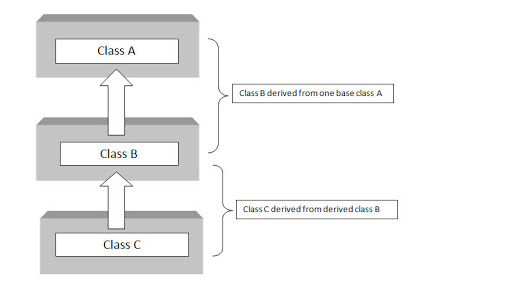
Python exhibit 3 types of inheritance:

1. Single Level Inheritance
2. Multilevel Inheritance
3. Multiple Inheritance

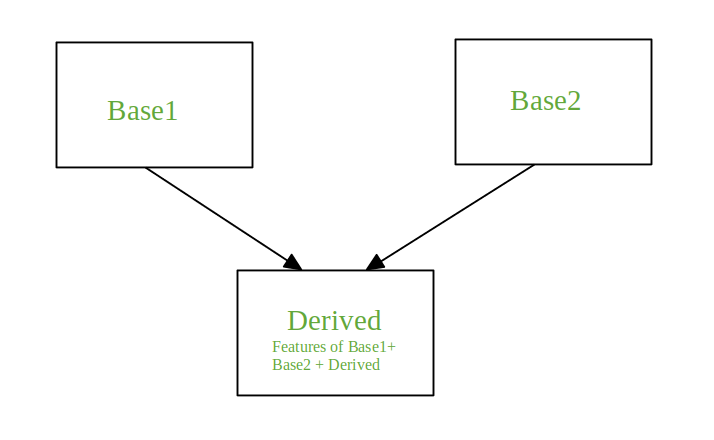
Single Level Inheritance: Here one child class are derived from one parent class.



MultiLevel Inheritance : Here one child class inherits a parent class which in turn is inherited from another parent class.



Multiple Inheritance : Here one child class is inherited by more than one parent class.



Parent class is also known as base class and child class is also known as derived class.

Polymorphism : It is a concept by which we can perform single action by various method. Here poly means many and morphism means method.

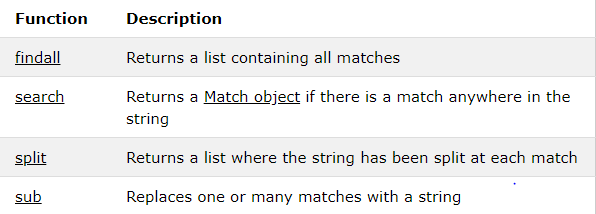
Encapsulation: It is defined as wrapping up of data under single unit It is a mechanism that binds together code and data it manipulates.

Abstraction: It is a method which hides the details and display only the necessary information.

**RegEx In Python :**

Regular expression can be used by importing re module.

RegEx Functions:



**Exception Handling:** It is a concept in which if error occurs, it throws the error and make the program to run smoothly. It has three blocks:

Try block helps to test a block of code for errors.

Except block helps to handle the error.

Finally block executes everytime code is executed.

Else block is executed when no error occurs in try block.

**ACTIVITY 3**

**Mastersheet should print the data from sub-sheets depending on user input, master file should summarize / count the data across rows and publish the data in a new sheet**

**Introduction:**

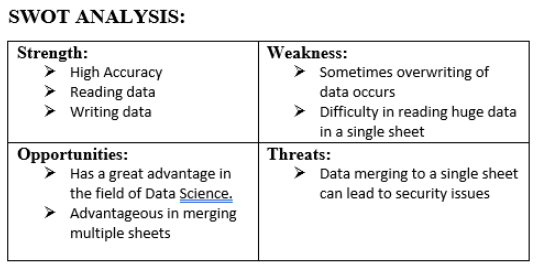
The code implemented, has considered the basic requirement of reading and writing data with user friendly environment.

**Defining Our System:**

The designed code is used to read and write data. An excel sheet has been made manually which consists of 5 sheets and 1 master sheet. Here we are searching details of an individual corresponding to a particular name, registration number and email ID in all the 5 sub-sheets. Once the data has been fetched from the sub sheets then it will be printed to the master sheet. The excel sheet also consists of a summary sheet which indicates the count number of data fetched from each sheet. The whole implementation is used to read a file for better searching and writing. The code makes the study easier in the field of data science where lots and lots of data needs extraction.

**SWOT Analysis:**

**Table 1: SWOT Analysis**



# 4W's and 1'H :

# Who: Basically used in research field and data science fields where large data needs manipulation and extraction.What: It is a highend, affordable calculator.

## **When:** Used when large data needs to manipulated and extracted.

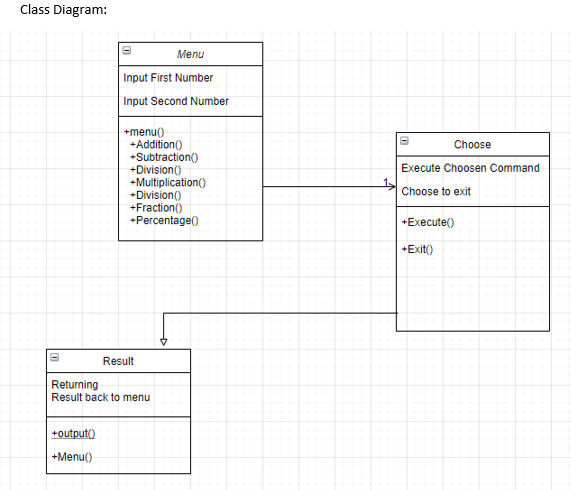
## **Where:** In Research and technical fields

## **How:** User friendly and easily accessible.

**UML Diagram:**

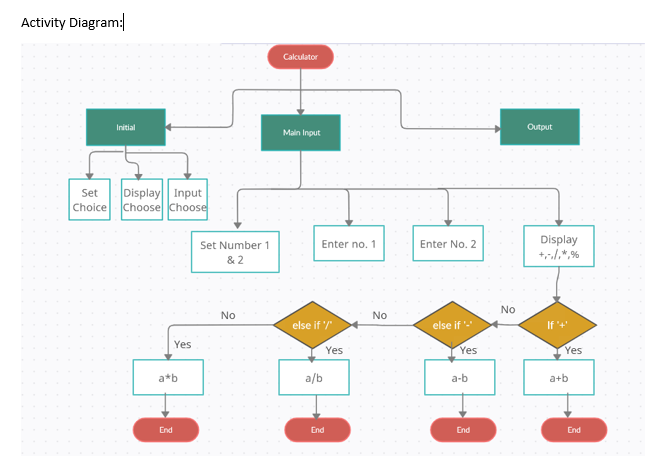
High Level Design:

Structural Diagram

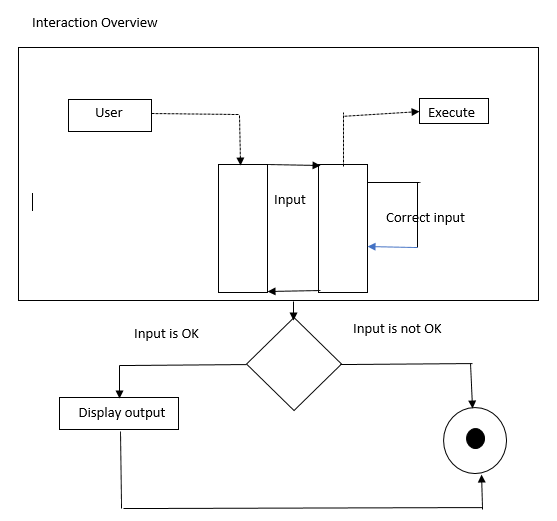


**Figure 2: Class Diagram of arithmetic calculator**

Behavioural Diagram:



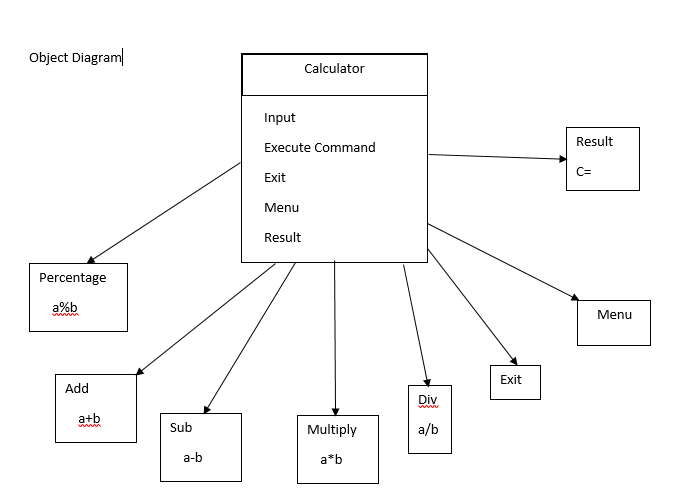
**Figure 3: Activity Diagram of arithmetic calculator**



**Figure 4: Interaction overview of Arithmetic calculator**

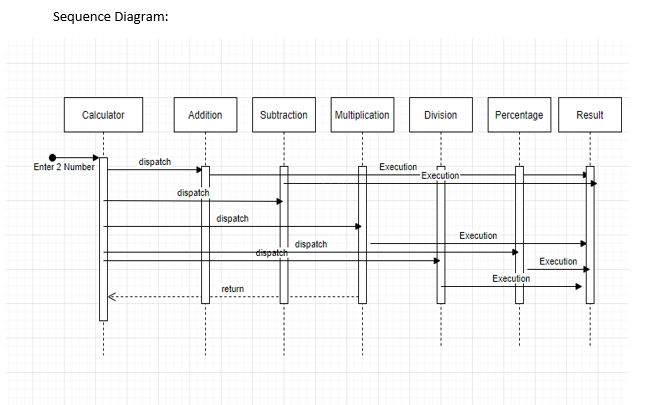
Low Level Design:

Structural Diagram:

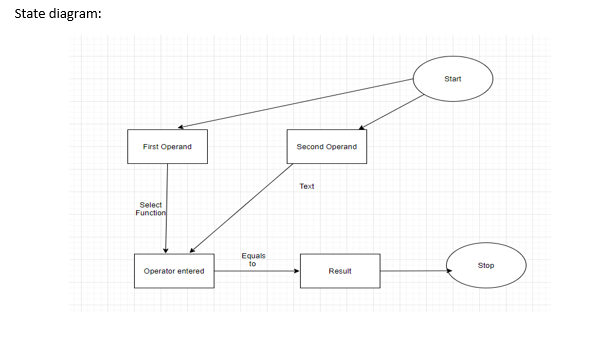


**Figure 5: Object diagram of arithmetic table**

Behavioural Diagram:



**Figure 6: Sequence Diagram of Arithmetic Calculator**

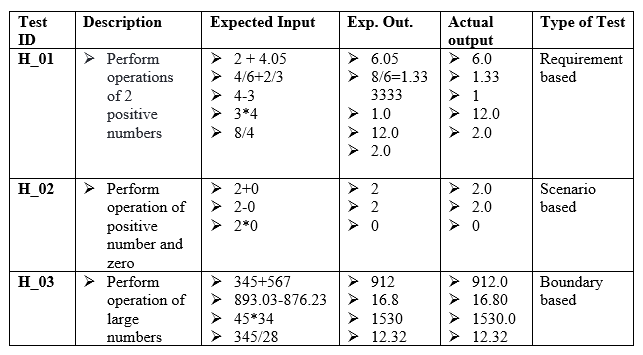


**Figure 7: State Diagram of Arithmetic Calculator**

**Test Plan:-**

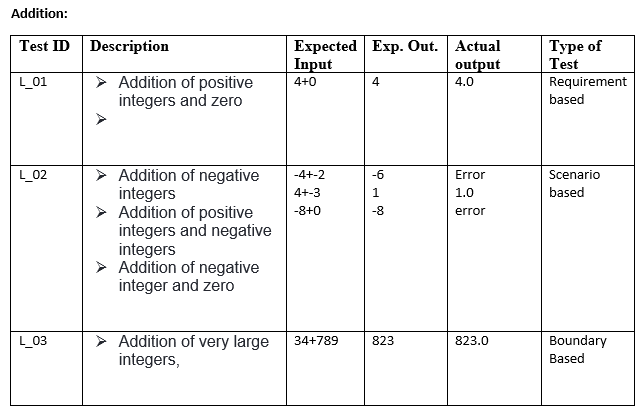
**HLR:**

**Table 2: Test Plan of HLR**

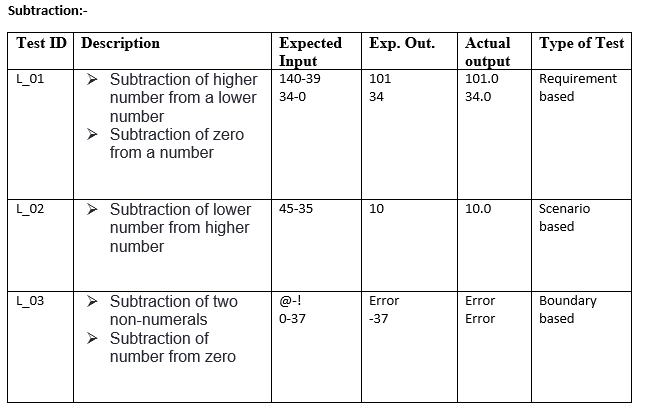


**LLR:**

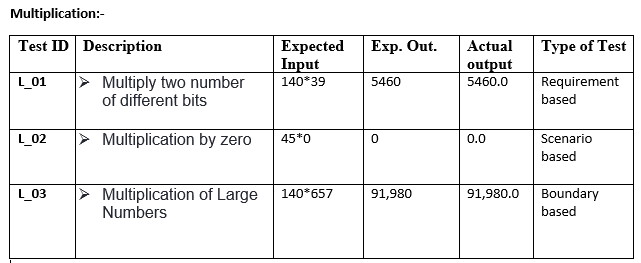
**Table 3: Addition test plan**



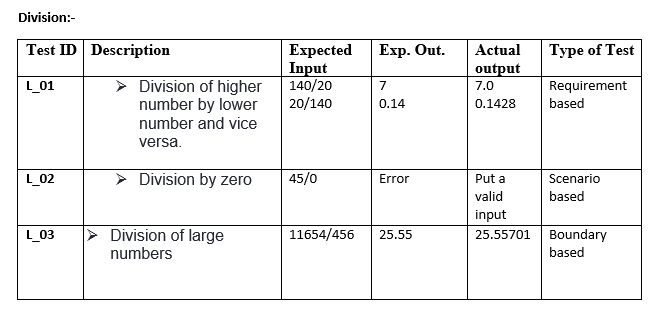
**Table 4: Subtraction Test Plan**

****

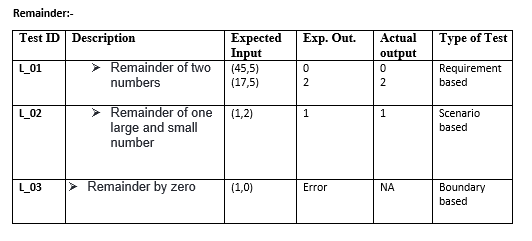
**Table 5: Multiplication Test Plan**



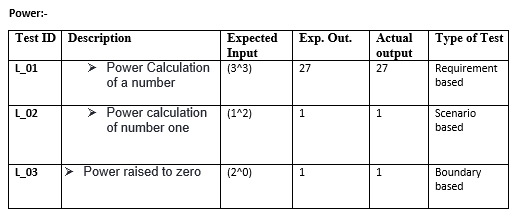
**Table 6: Division Test Plan**



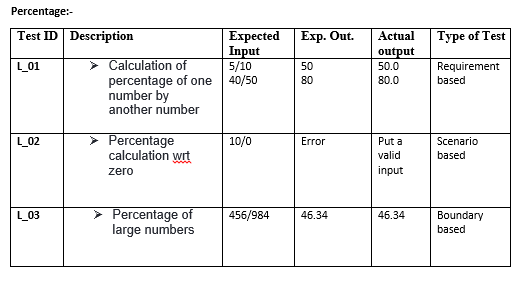
**Table 7: Remainder Test Plan**

****

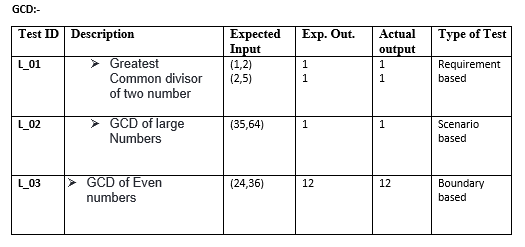
**Table 8: Power Test Plan**

****

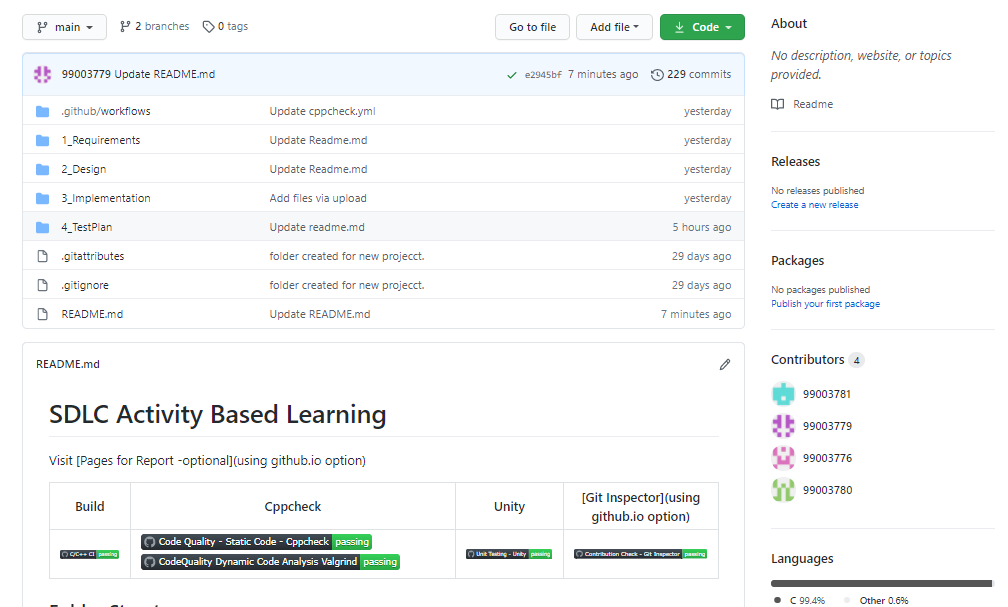
**Table 9: Percentage Test Plan**

****

**Table 10: GCD Test Plan**

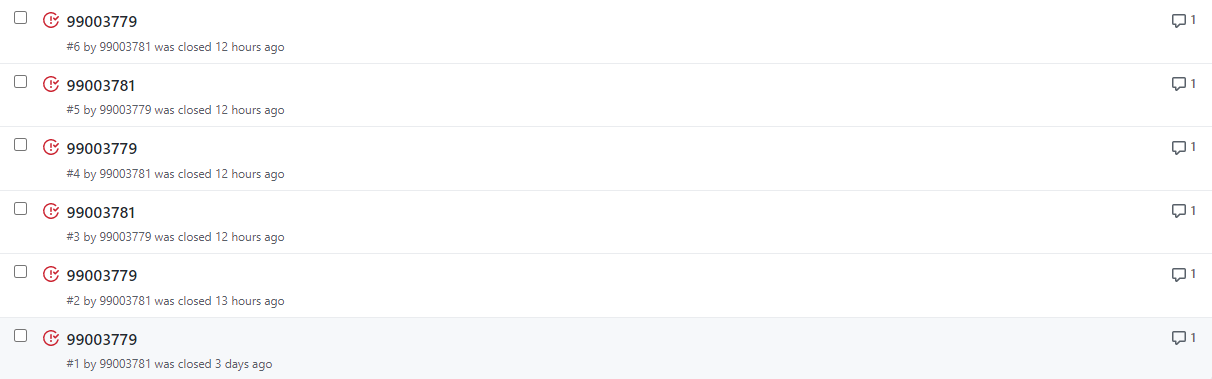
****

**Git :**



**Figure 8:** [**https://github.com/99003781/N8-Calculator.git**](https://github.com/99003781/N8-Calculator.git)

**Git Issues:**



**Figure 9: Screenshots of Git issues raised and closed**

**Main Project:**

#include "unity.h"

#include "main.h"

#include "calculator\_operations.h"

/\* Prototypes for all the test functions \*/

void test\_add(void);

void test\_add\_testcase2(void);

void test\_add\_testcase3(void);

void test\_add\_testcase4(void);

void test\_add\_testcase5(void);

void test\_subtract(void);

void test\_subtract\_testcase2(void);

void test\_subtract\_testcase3(void);

void test\_subtract\_testcase4(void);

void test\_subtract\_testcase5(void);

void test\_multiply(void);

void test\_multiply\_testcase2(void);

void test\_multiply\_testcase3(void);

void test\_multiply\_testcase4(void);

void test\_multiply\_testcase5(void);

void test\_divide(void);

void test\_divide\_testcase2(void);

void test\_divide\_testcase3(void);

void test\_divide\_testcase4(void);

void test\_divide\_testcase5(void);

void test\_percentage(void);

void test\_percentage\_testcase2(void);

void test\_percentage\_testcase3(void);

void test\_percentage\_testcase4(void);

void test\_percentage\_testcase5(void);

void test\_gcd(void);

void test\_gcd\_testcase2(void);

void test\_gcd\_testcase3(void);

void test\_gcd\_testcase4(void);

void test\_gcd\_testcase5(void);

void test\_rem(void);

void test\_rem\_testcase2(void);

void test\_rem\_testcase3(void);

void test\_rem\_testcase4(void);

void test\_rem\_testcase5(void);

void test\_power(void);

void test\_power\_testcase2(void);

void test\_power\_testcase3(void);

void test\_power\_testcase4(void);

void test\_power\_testcase5(void);

void setUp(){}

void tearDown(){}

int main(void)

{

UNITY\_BEGIN();

RUN\_TEST(test\_add);

RUN\_TEST(test\_add\_testcase2);

RUN\_TEST(test\_add\_testcase3);

RUN\_TEST(test\_add\_testcase4);

RUN\_TEST(test\_add\_testcase5);

/\*SUBTRACTION\*/

RUN\_TEST(test\_subtract);

RUN\_TEST(test\_subtract\_testcase2);

RUN\_TEST(test\_subtract\_testcase3);

RUN\_TEST(test\_subtract\_testcase4);

RUN\_TEST(test\_subtract\_testcase5);

/\*MULTIPLICATION\*/

RUN\_TEST(test\_multiply);

RUN\_TEST(test\_multiply\_testcase2);

RUN\_TEST(test\_multiply\_testcase3);

RUN\_TEST(test\_multiply\_testcase4);

RUN\_TEST(test\_multiply\_testcase5);

/\*DIVISION\*/

RUN\_TEST(test\_divide);

RUN\_TEST(test\_divide\_testcase2);

RUN\_TEST(test\_divide\_testcase3);

RUN\_TEST(test\_divide\_testcase4);

RUN\_TEST(test\_divide\_testcase5);

/\*PERCENTAGE\*/

RUN\_TEST(test\_percentage);

RUN\_TEST(test\_percentage\_testcase2);

RUN\_TEST(test\_percentage\_testcase3);

RUN\_TEST(test\_percentage\_testcase4);

RUN\_TEST(test\_percentage\_testcase5);

/\*GCD\*/

RUN\_TEST(test\_gcd);

RUN\_TEST(test\_gcd\_testcase2);

RUN\_TEST(test\_gcd\_testcase3);

RUN\_TEST(test\_gcd\_testcase4);

RUN\_TEST(test\_gcd\_testcase5);

/\*REMAINDER\*/

RUN\_TEST(test\_rem);

RUN\_TEST(test\_rem\_testcase2);

RUN\_TEST(test\_rem\_testcase3);

RUN\_TEST(test\_rem\_testcase4);

RUN\_TEST(test\_rem\_testcase5);

/\*POWER\*/

RUN\_TEST(test\_power);

RUN\_TEST(test\_power\_testcase2);

RUN\_TEST(test\_power\_testcase3);

RUN\_TEST(test\_power\_testcase4);

RUN\_TEST(test\_power\_testcase5);

return UNITY\_END();

}

/\*ADDITION\*/

void test\_add(void)

{

TEST\_ASSERT\_EQUAL(30, add(10, 20));

}

void test\_add\_testcase2(void)

{

TEST\_ASSERT\_EQUAL(-10, add(10, -20));

}

void test\_add\_testcase3(void)

{

TEST\_ASSERT\_EQUAL(-20.5, add(-40.5, 20.0));

}

void test\_add\_testcase4(void)

{

TEST\_ASSERT\_EQUAL(41.0, add(15.5, 25.5));

}

void test\_add\_testcase5(void)

{

TEST\_ASSERT\_EQUAL(15000, add(7500, 7500));

}

/\*SUBTRACTION\*/

void test\_subtract(void)

{

TEST\_ASSERT\_EQUAL(-3, subtract(0, 3));

}

void test\_subtract\_testcase2(void)

{

TEST\_ASSERT\_EQUAL(2, subtract(5, 3));

}

void test\_subtract\_testcase3(void)

{

TEST\_ASSERT\_EQUAL(7, subtract(10, 3));

}

void test\_subtract\_testcase4(void)

{

TEST\_ASSERT\_EQUAL(-4, subtract(3, 7));

}

void test\_subtract\_testcase5(void)

{

TEST\_ASSERT\_EQUAL(100, subtract(1000, 900));

}

/\*MULTIPLICATION\*/

void test\_multiply(void)

{

TEST\_ASSERT\_EQUAL(0, multiply(1, 0));

}

void test\_multiply\_testcase2(void)

{

TEST\_ASSERT\_EQUAL(15, multiply(5, 3));

}

void test\_multiply\_testcase3(void)

{

TEST\_ASSERT\_EQUAL(0, multiply(10, 0));

}

void test\_multiply\_testcase4(void)

{

TEST\_ASSERT\_EQUAL(-30, multiply(6, -5));

}

void test\_multiply\_testcase5(void)

{

TEST\_ASSERT\_EQUAL(10, multiply(2, 5));

}

/\*DIVISION\*/

void test\_divide(void)

{

TEST\_ASSERT\_EQUAL(0, divide(1, 0));

}

void test\_divide\_testcase2(void)

{

TEST\_ASSERT\_EQUAL(5, divide(10, 2));

}

void test\_divide\_testcase3(void)

{

TEST\_ASSERT\_EQUAL(5, divide(5, 1));

}

void test\_divide\_testcase4(void)

{

TEST\_ASSERT\_EQUAL(-6, divide(-30, 5));

}

void test\_divide\_testcase5(void)

{

TEST\_ASSERT\_EQUAL(1, divide(2, 2));

}

/\*PERCENTAGE\*/

void test\_percentage(void)

{

TEST\_ASSERT\_EQUAL(100, percentage(1, 1));

}

void test\_percentage\_testcase2(void)

{

TEST\_ASSERT\_EQUAL (40, percentage(2,5));

}

void test\_percentage\_testcase3(void)

{

TEST\_ASSERT\_EQUAL(20, percentage(20, 100));

}

void test\_percentage\_testcase4(void)

{

TEST\_ASSERT\_EQUAL(40, percentage(2, 5));

}

void test\_percentage\_testcase5(void)

{

TEST\_ASSERT\_EQUAL(20, percentage(2, 10));

}

/\*GCD\*/

void test\_gcd(void)

{

TEST\_ASSERT\_EQUAL(1, gcd(1, 2));

}

void test\_gcd\_testcase2(void)

{

TEST\_ASSERT\_EQUAL (20, gcd(20, 40));

}

void test\_gcd\_testcase3(void)

{

TEST\_ASSERT\_EQUAL (1, gcd(35, 64));

}

void test\_gcd\_testcase4(void)

{

TEST\_ASSERT\_EQUAL (12, gcd(24, 36));

}

void test\_gcd\_testcase5(void)

{

TEST\_ASSERT\_EQUAL(1, gcd(2, 5));

}

/\*REMAINDER\*/

void test\_rem(void)

{

TEST\_ASSERT\_EQUAL(1, rem(1, 2));

}

void test\_rem\_testcase2(void)

{

TEST\_ASSERT\_EQUAL (0, rem(45,5));

}

void test\_rem\_testcase3(void)

{

TEST\_ASSERT\_EQUAL (2, rem(17,5));

}

void test\_rem\_testcase4(void)

{

TEST\_ASSERT\_EQUAL (2, rem(18,4));

}

void test\_rem\_testcase5(void)

{

TEST\_ASSERT\_EQUAL(1, rem(5, 4));

}

/\*POWER\*/

void test\_power(void)

{

TEST\_ASSERT\_EQUAL (1, power(1, 2));

}

void test\_power\_testcase2(void)

{

TEST\_ASSERT\_EQUAL (27, power(3,3));

}

void test\_power\_testcase3(void)

{

TEST\_ASSERT\_EQUAL (1, power(3,0));

}

void test\_power\_testcase4(void)

{

TEST\_ASSERT\_EQUAL (1, power(1,4));

}

void test\_power\_testcase5(void)

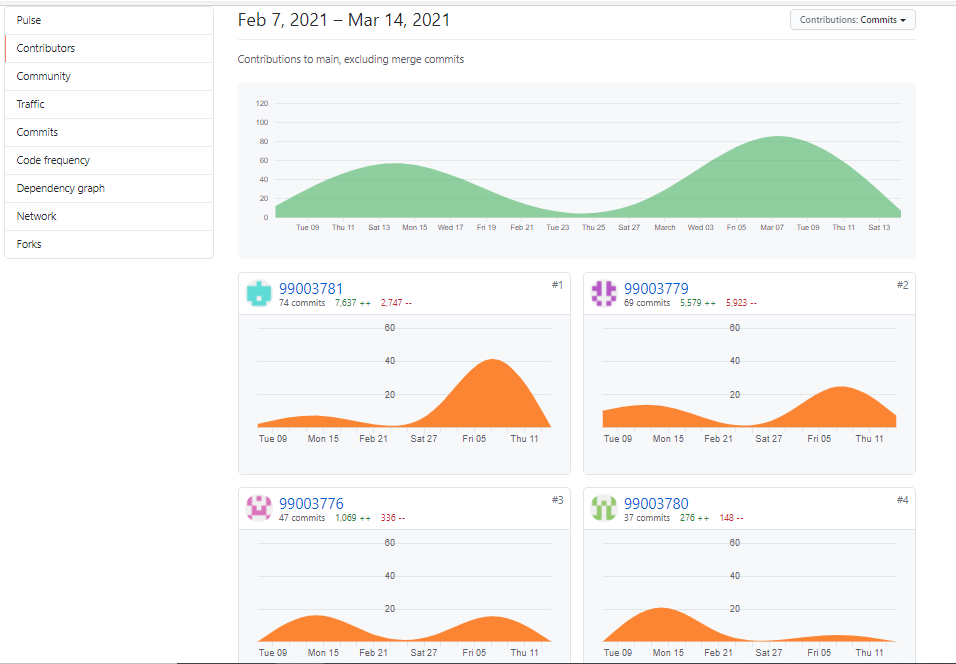
{

TEST\_ASSERT\_EQUAL(25, power(5, 2));

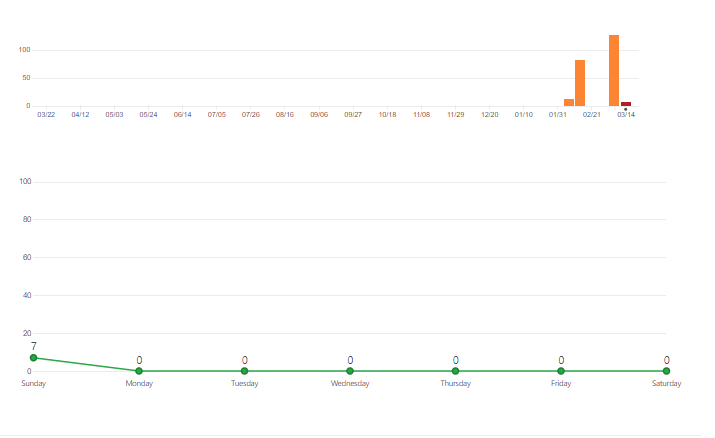
}

}

**Git commits:**



**Figure 10: Graph of commits on Github**



**Figure 11: Plot of Git Commits on github**