

Z02DT – Software & Databases

Module Reference: Z02DT 2526 JANFEB

Name: Ali Akli

Student ID: 16876868

Date: 16/01/26

Stage 1 (6 Marks Available)

Artefact – Calculator Functions (3 Marks)

The client will need to perform some basic numeric calculations and have requested an online calculator as part of the system. You must complete this task first, demonstrating your ability to produce professionally commented and tested code and supporting documentation. You must ensure that your design is tolerant of incorrect input from operators.

The required deliverable is a calculator module containing basic mathematical functions accessed by the user entering their equation parts and the result is calculated.

Example:

- the user enters 36
- the user selects / for divide
- the user enters 4
- the result 36 / 4 = 9 is returned

Reflective Report – Calculator Function (150 words / 3 Marks)

Include your requirements specification, test specification, test results and design documents for this Stage in the Appendix and write a short report reflecting on your design, your tests, and describe the operation of your calculator functions. Support your reflection with academic references.

Submission Instructions: Deadline 16/01/2026 Friday Week 1 @ 6pm

- **Artefact** – Submit as a single ZIP file (no other formats accepted)
 - Programming project file
- **Reflective Report** - Report
 - (as a Word document using this provided template - MANDATORY)
 - One sentence Introduction
 - Reflective report (150 Words +/- 10%)
 - Design documents copied into the appendix
 - Code copied as text into the appendix
 - Screenshots of your tests in the appendix
 - Screenshot of quiz results in the appendix

Introduction

Reflection

References

- Coghlan, A., Rossum, G. R., Warsaw, B. (2001). *PEP 8 – Style Guide for Python Code*. Python.org
- Marsic, I. (2012). *Software Engineering*. Rutgers University
- Python Software Foundation. (2024). *Python language reference, version 3.12*.
<https://docs.python.org/3/>

Appendix

Appendix 1 - Module Design

Requirements

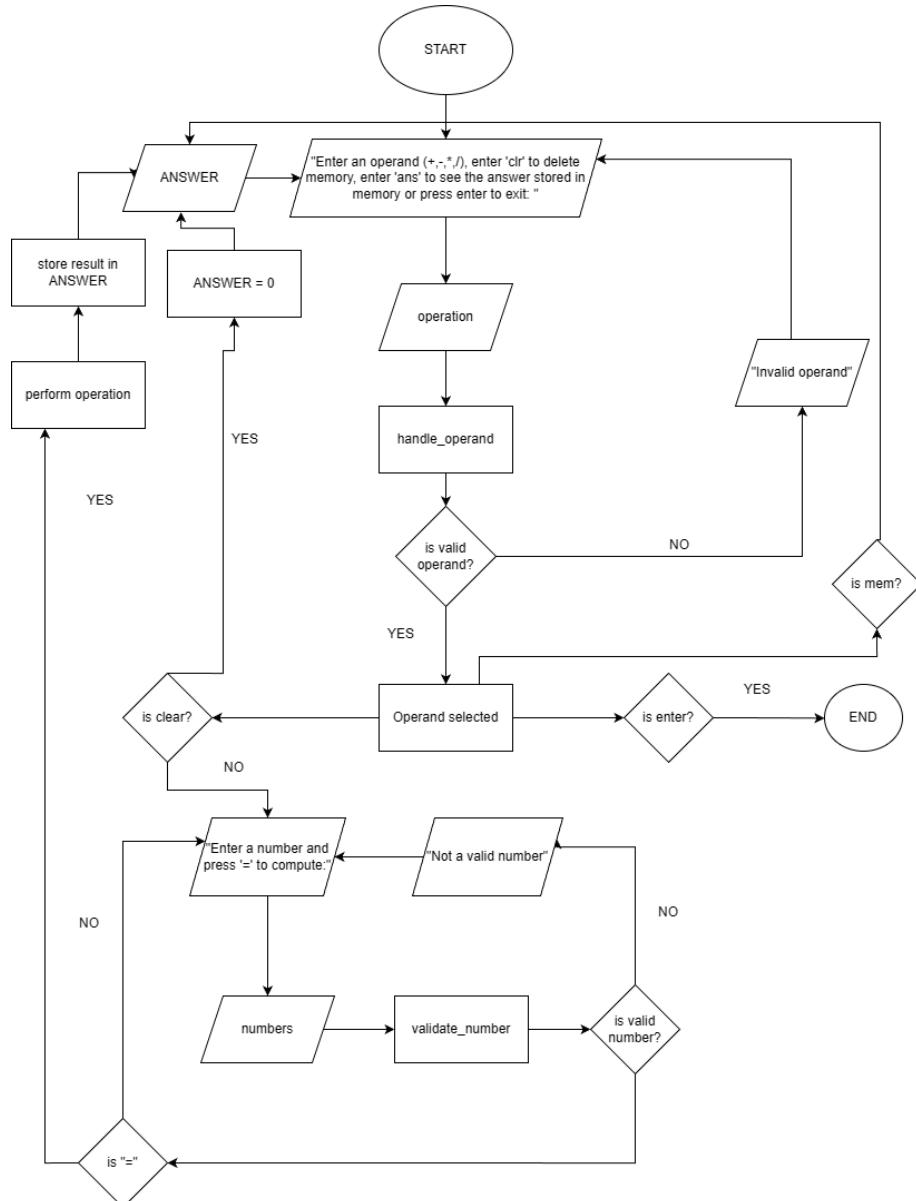
Minimum requirements:

- Significant functionality fully documented with comments
- Include arithmetic operations [+,-,*,/]

Additional requirements:

- Make the calculator accept more than 2 arguments
- Answers stored in memory
- Accept words and symbols for the operand (e.g. ‘add’, ‘sub’, ‘subtract’ etc.)
- Prevent the program from crashing due to unexpected input
- Remove whitespace from user input
- Support exponentiation and modulo
- Keep code in PEP 8 style formatting (Coghlan et al., 2001)
- Output whole numbers when decimals are not used

Flow Chart



Test specifications

Operator input validation

ID	Description	Input	Expected result
Calc-1	Valid operator	+	Addition operator selected
Calc-2	Valid operator using words	"aDd"	Addition operator selected
Calc-3	Invalid operator	"xyz"	"Invalid operand"
Calc-4	Exit program	Enter	"Goodbye" and program close

Number input validation

ID	Description	Input	Expected result
Calc-5	Valid integer input	5	Accepted
Calc-6	Valid float input	2.5	Accepted
Calc-7	Invalid number entry	"abc"	Prompt again and display "Not a valid number"
Calc-8	End number entry	"="	Calculation performed

Arithmetic operations

ID	Description	Input	Expected result
Calc-9	Addition	"+", "2", "3", "="	"5"
Calc-10	Subtraction	"-", "3", "10", "="	"-7"
Calc-11	Division	"/", "20", "4", "="	"5"
Calc-12	Multiplication	"mult", "4", "5", "="	"20"
Calc-13	Multiple numbers	"+", "1", "2", "3", "="	"6"
Calc-14	Different operations	"+" , "5" , "5" , "=" , "-" , "5"	"5"

Memory Functionality

ID	Description	Input	Expected result
Calc-15	Store results in memory	Perform calculation	Result stored in ANSWER
Calc-16	Apply additional operations to memory	(1+2)/6	0.5
Calc-17	View memory	"ans"	Display stored answer
Calc-18	Clear memory	"clr"	ANSWER reset to None or 0

Error Handling

ID	Description	Input	Expected result
Calc-19	Division by zero	"/", "5", "0"	Division by zero, ANSWER reset

Appendix 2 – Module Code

Software function

1. User is asked for arithmetic operator
2. User is then repeatedly asked for operands (the numbers)
3. Program displays requested expression and result
4. Result is stored
5. User can then clear this result or continue choosing another operator and adding onto the result

6. User can exit the program when the arithmetic operator is being requested

Screenshots of Code

```

● ● ●
#IA/01
#Imports
import math

#@Global variables
NOT_A_NUMBER = "NaN"
ANSWER = 0

#Main
def main():
    #Sets global ANSWER variable
    global ANSWER
    #Starts program loop
    while True:
        #Asks whether user wants to continue using the calculator, clear memory or exit (strip to remove whitespace and move input to lower case)
        operation = input("Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: ").lower().strip()
        if operation == "":
            print("Goodbye")
            break
        #Stores the operation in the variable 'operation'
        operation = handle_operator(operation)
        if operation == "Not an operator":
            print("Invalid operator")
            continue
        elif operation == "clear":
            print(ANSWER)
            continue
        elif operation == "memory":
            continue
        #Gets user number inputs
        numbers = get_numbers()
        #Asks user to enter their answer then remove it from the first element in the array
        ANSWER += 0 if len(numbers) == 0 else numbers.pop(0)
        #Perform the operation on the numbers stored in 'numbers'
        #Display to output the result
        if operation(numbers) == "Division by zero":
            print("Division by zero")
            ANSWER = 0
        else:
            print(ANSWER)

    #Gets the validated numbers from the user input then stores them in an array
def get_numbers():
    numbers = []
    #Loops until the final input is '='
    while True:
        #Add the validated number to the end of the list
        numbers.append(validate_number())
        #Checks if the last value in the array is '='
        if numbers[-1] == "=":
            #Remove the '='
            numbers.pop()
            break
    return numbers

#Gets number from user input and validates
def get_valid_number():
    num = input("Enter a number and press '=' to compute: ")
    #Loops until valid number is given
    while validate_number(num) == NOT_A_NUMBER:
        num = input("Not a valid number\nEnter a number: ")
    return validate_number(num)

#Function to validate number
def validate_number(num):
    if num == "=":
        return "="
    else:
        try:
            a = float(num)
            return int(a) if a.is_integer() else a
        except ValueError:
            #If a number given is not a valid number, this prevents the program from crashing
            return NOT_A_NUMBER

#Function to decide which operator is being requested
def handle_operator(operator):
    #Using a variety of ways people may ask for an operator to account for the broadest amount of users
    global ANSWER
    match operator:
        case "mem" | "memory" | "ans" | "answer":
            print(ANSWER)
            return "memory"
        case "clr" | "clear" | "del":
            ANSWER = 0
            return "clear"
        case "add" | "addition" | "+" | "+":
            return addition
        case "sub" | "subtraction" | "-" | "-":
            return subtraction
        case "multi" | "mult" | "m" | "*":
            return multiplication
        case "division" | "div" | "d" | "/":
            return division
        case _:
            return "Not an operator"

#Function to add an array of inputs
def addition(lit):
    global ANSWER
    #Output: (n1+n2)+(n3+...)
    output = "(%s(%s)+(%s))..." % ("+".join(["%s" % format(n) for n in lit]))
    ANSWER += sum(lit)

#Function to subtract 2 inputs
def subtraction(lit):
    global ANSWER
    #Output: (n-(n1)-(n2)...)
    output = "(%s-(%s)-(%s))..." % (".".join(["%s" % format(n) for n in lit]))
    ANSWER -= sum(lit)

#Function to divide 2 inputs
def division(lit):
    global ANSWER
    try:
        #Output: (n/(n1)/(n2)...
        print(ANSWER,"/", " / ".join(["%s" % format(n) for n in lit]))
        for number in lit:
            ANSWER /= number
        ANSWER = validate_number(ANSWER)
        #To prevent the program from crashing, this exception catches the ZeroDivisionError and returns the string
        except ZeroDivisionError:
            return "Division by zero"

#Function to multiply 2 inputs
def multiplication(lit):
    global ANSWER
    #Output: (n*(n1)*(n2)...)
    print(ANSWER,"*", " * ".join(["%s" % format(n) for n in lit]))
    for number in lit:
        ANSWER *= number

#Prevents the program from being run when imported (probably will use this with OOP)
if __name__ == "__main__":
    main()

```

Analysis

Artefact 1 is the basis of the calculator. Currently, there are 6 functions: addition, subtraction, multiplication, division, answer recall and clear memory.

I began with handling operator selection. There are a few ways of doing this: if statements, match case and mapping are a few. I decided to use match case statements over if statements due to its cleaner code, and over mapping due to the small amount of actual possible inputs. This approach also improves readability by grouping similar user inputs (such as symbols and text commands) under a single case, allowing the program to account for a wider range of user behaviour without increasing complexity.

main() is the main loop of the program which will repeatedly run until exited. The function handles the inputted operator using handle_operator(), the memory recall and clearing and exiting the program.

The handle_operator() function returns either a reference to the corresponding operation function where it is later called passing the numbers from get_numbers() or a control command, such as clearing or recalling memory.

I also chose to give functions only one task to decrease cohesion (Martin, 2012, pg. 125); this is why there is get_numbers(), get_valid_number() and validate_number(). The function validate_number() checks whether the parameter is a valid number. If it is, the function returns the number again (if it's a decimal, as a float or otherwise return as an integer). If the parameter is "=", the function returns "=", indicating the end of a sequence of numbers for get_numbers(). If it is not, the function returns NOT_A_NUMBER, indicating that the parameter was not a number to any other functions.

get_valid_number() continuously prompts the user for a number and validates if that number is valid using validate_number(). If the number is invalid, the function displays an error message and prompts the user again for a number.

The function get_numbers() creates an array and uses the get_valid_number() function to create an array of valid numbers. If the last number is an "=", the function stops the loop, removes the "=" using .pop() and returns the array.

The arithmetical functions (addition(), subtraction(), multiplication(), division()) all take the array of numbers from get_numbers() to perform their respective calculations and assign their results to the global variable ANSWER.

addition() takes the sum of the array of numbers using sum() and displays the sum performed and adds to the answer stored in ANSWER.

subtraction() takes the sum of the array of numbers using sum(), negates it, displays the result and subtracts the answer stored in ANSWER .

multiplication() loops through the array and multiplies each element to ANSWER

division() loops through the array and divides each element to and validates the number before adding to ANSWER. The ZeroDivisionError is handled in the try, except block and if there is division by zero, the text "Division by zero" is returned.

Appendix 3 – Module Tests

Test1

2026-01-16

Calc-1 (PASS)

```
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: =
0 +
0
```

Calc-2 (PASS)

```
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: aDd
Enter a number and press '=' to compute: =
0 +
0
```

Calc-3 (PASS)

```
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: xyz
Invalid operand
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: |
```

Calc-4 (PASS)

```
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
Goodbye
>>> |
```

Calc-5, Calc-6 (PASS)

```
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: 2.5
Enter a number and press '=' to compute: =
5 + 2.5
7.5
```

Calc-7 (PASS)

```
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: abc
Not a valid number
Enter a number: |
```

Calc-8 (PASS)

```
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: =
1 + 2 + 3
6
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
```

Calc-9 (PASS)

```
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: =
2 + 3
5
```

Calc-10 (PASS)

```
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: -
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: 10
Enter a number and press '=' to compute: =
3 - 10
-7
```

Calc-11 (PASS)

```

Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: /
Enter a number and press '=' to compute: 20
Enter a number and press '=' to compute: 4
Enter a number and press '=' to compute: =
20 / 4
5

```

Calc-12 (PASS)

```

Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: mult
Enter a number and press '=' to compute: 4
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: =
4 * 5
20

```

Calc-13 (PASS)

```

Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: =
1 + 2 + 3
6

```

Calc-14 (FAIL)

```

Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: =
5 + 5
10
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: -
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: =
15 -
15
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: |

```

Calc-15 (PASS)

```

Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
+
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: =
1 + 2 + 3
6
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
Goodbye

```

Name	Value
ANSWER	6
NOT_A_NUMBER	'NaN'
addition	<function addi>
division	<function divis>
get_numbers	<function get_n>
get_valid_number	<function get_v>
handle_operand	<function han>
main	<function mair>
math	<module 'math'>
multiplication	<function mult>
subtraction	<function subt>
validate_number	<function valic>

Calc-16 (FAIL)

```

Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: =
1 + 2
3
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: /
Enter a number and press '=' to compute: 6
Enter a number and press '=' to compute: =
9 /
9

```

Calc-17 (PASS)

```

Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
+
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: =
1 + 2 + 3
6
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
ans
6

```

Calc-18 (PASS)

```

Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
+
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: =
1 + 2 + 3
6
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
ans
6
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
clr
0
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
Goodbye

```

Calc-19 (PASS)

```

Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
/
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: 0
Enter a number and press '=' to compute: =
5 / 0
Division by zero
Enter an operand (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
ans
0

```

17/19 tests passed

Debugging

Test Calc-16 failed. To debug this, I'll create a breakpoint at line 36 where answer is being assigned.

The screenshot shows a debugger interface with two main windows. On the left is the code editor for the `main()` function. The code is as follows:

```

main()
{
    print("Invalid operator")
    continue
elif operation == "clear":
    print(ANSWER)
    continue
elif operation == "memory":
    continue

#Get user number inputs
numbers = get_numbers()
#Assign the first number to answer then remove it from the first element in the array
ANSWER += 6 if len(numbers) == 0 else numbers.pop(0)
#Perform the operation on the numbers stored in numbers
#Display to output the result
if operation(numbers) == "Division by zero":
    print("Division by zero")
    ANSWER = 0
else:
    print(ANSWER)
}

```

The line `ANSWER += 6 if len(numbers) == 0 else numbers.pop(0)` is highlighted in red, indicating it is the current breakpoint. On the right is the Variables window, which displays the current state of variables:

Name	Value
ANSWER	3
NOT_A_NUMBER	'NaN'
addition	<function addition>
division	<function division>
get_numbers	<function get_num>
get_valid_number	<function get_valid>
handle_operator	<function handle_o>
main	<function main at 0>
math	<module 'math' (built-in)>
multiplication	<function multiply>
subtraction	<function subtract>
validate_number	<function validate_i>

One issue stands out, 6 is being added on to the result of 1+2 and then the list is emptied. To fix this, an if statement can be created rather than using a ternary operator.

```
#Assign the first number to 0 if the list is empty
if len(numbers) == 0:
    ANSWER += 0
```

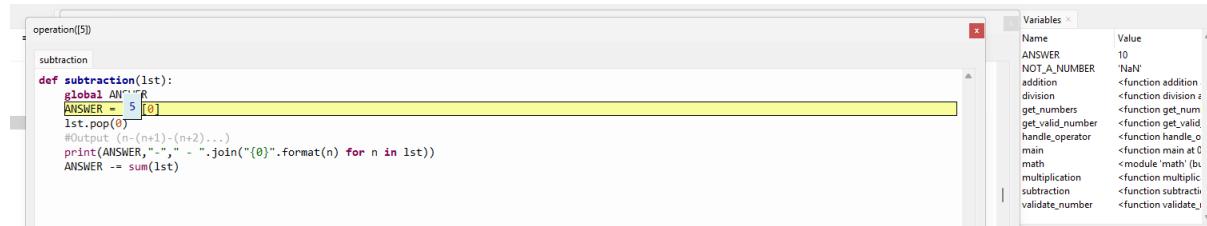
```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: =
0 + 1 + 2
3
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: /
Enter a number and press '=' to compute: 6
Enter a number and press '=' to compute: =
3 / 6
0.5
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: |
```

This does fix the issue but causes another function to break in the process. When doing the first operation, the summary of the expression starts with 0 when it should be with the first operation. To fix this, I can assign ANSWER to the first element of lst in the function operations.

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: =
1 + 2
3
```

This fixes the issue.

Test Calc-14 failed. To debug this, I'll create a breakpoint on line 114 which is where the function subtraction() begins. The solution to Calc-16 seems to have solved the issue, however, after further testing, the issue seems to be because of ANSWER being reset each time.



To fix this, I initialised ANSWER as an int with None rather than 0

```
5 #Global variables
6 NOT_A_NUMBER = "NaN"
7 ANSWER: int | None = None
8
```

Then in all the operator functions, the program checked if ANSWER was none. If it was, the program would change to the first value of the lst argument (the first operand) and remove that element from the array.

```
def addition(lst):
    global ANSWER
    if ANSWER is None:
        ANSWER = lst.pop(0)
```

This however did not fix multiplication and division which multiplied or divided the first number by 0 when ANSWER was 0 which broke the clear memory feature there were two ways of fixing this, changing the clear memory feature to make ANSWER = None rather than 0 or making multiplication and division check the length of the list. I chose the latter as it allowed for subtraction to be able to go below 0 and into the negatives.

```
def division(lst):
    global ANSWER
    if ANSWER is None or len(lst) > 1:
        ANSWER = lst.pop(0)
```

Test2

2026-01-20

Calc1 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: =
0 +
0
```

Calc2 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: add
Enter a number and press '=' to compute: =
0 +
0
```

Calc3 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: xyz
Invalid operator
```

Calc4 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit:
Goodbye
```

Calc5, Calc6 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: 2.5
Enter a number and press '=' to compute: =
5 + 2.5
7.5
```

Calc7 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: abc
Not a valid number
```

Calc8 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: =
1 + 2 + 3
6
```

Calc9 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: =
2 + 3
5
```

Calc10 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: -
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: 10
Enter a number and press '=' to compute: =
3 - 10
-7
```

Calc11 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: /
Enter a number and press '=' to compute: 20
Enter a number and press '=' to compute: 4
Enter a number and press '=' to compute: =
20 / 4
5
```

Calc12 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: mult
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: 4
Enter a number and press '=' to compute: =
5 * 4
20
```

Calc13 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: 3
Enter a number and press '=' to compute: =
1 + 2 + 3
6
```

Calc14 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: =
5 + 5
10
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: -
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: =
10 - 5
5
```

Calc15 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: =
5 + 5
10
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: -
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: =
10 - 5
5
```

Name	Value
ANSWER	5
NOT_A_NUMBER	'NaN'
addition	<function addition>
division	<function division>
get_numbers	<function get_num>
get_valid_number	<function get_valid>
handle_operator	<function handle_o>
main	<function main at 0>
math	<module 'math' (built-in)>
multiplication	<function multiplic>
subtraction	<function subtracti>
validate_number	<function validate_n>

Calc16 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: =
1 + 2
3
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: /
Enter a number and press '=' to compute: 6
Enter a number and press '=' to compute: =
3 / 6
0.5
```

Calc17 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: =
1 + 2
3
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: /
Enter a number and press '=' to compute: 6
Enter a number and press '=' to compute: =
3 / 6
0.5
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: ans
0.5
```

Calc18 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: +
Enter a number and press '=' to compute: 1
Enter a number and press '=' to compute: 2
Enter a number and press '=' to compute: =
1 + 2
3
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: /
Enter a number and press '=' to compute: 6
Enter a number and press '=' to compute: =
3 / 6
0.5
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: ans
0.5
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: clr
None
```

Calc19 (PASS)

```
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: /
Enter a number and press '=' to compute: 5
Enter a number and press '=' to compute: 0
Enter a number and press '=' to compute: =
5 / 0
Division by zero
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: ans
None
Enter an operator (+,-,*,/), enter 'clr' to delete memory, enter 'ans' to see the answer stored in memory or press enter to exit: |
```

19/19 Tests passed