#11_Walk-through:Refactoring_Calculator, 21/05/2025

Adding a distinctive prompt

- It's a good idea to implement a function when wanting to repetitively include paticular characters in a string.

```
%python calculator.py
Welcome to calculator!
What's the first number?
5
What's the second number?
7
```

→

```
=> Welcome to calculator!
=> What's the first number?
5
=> What's the second number?
7
```

To achieve the above, you could prepend => to the front of each string that's passed to print.

Even better, you could implement a function named prompt

```
def prompt(message):
    print(f"==> {message}")
```

This function will prepend ==> to the front of the string that it prints.

i.e., all print calls can be replaced with prompt calls.

21/05/2025

calculator.py

```
def prompt(message):
    print(f"==>{message}")

prompt('Welcome to the Calculator!')
```

NB: replaced print with prompt

Match Case Statement

22/05/2025

Current program uses if/else statement to compare the user's choice of an operator. This scenario is typical to implement a match case statement instead.

- tidier and less code to write.

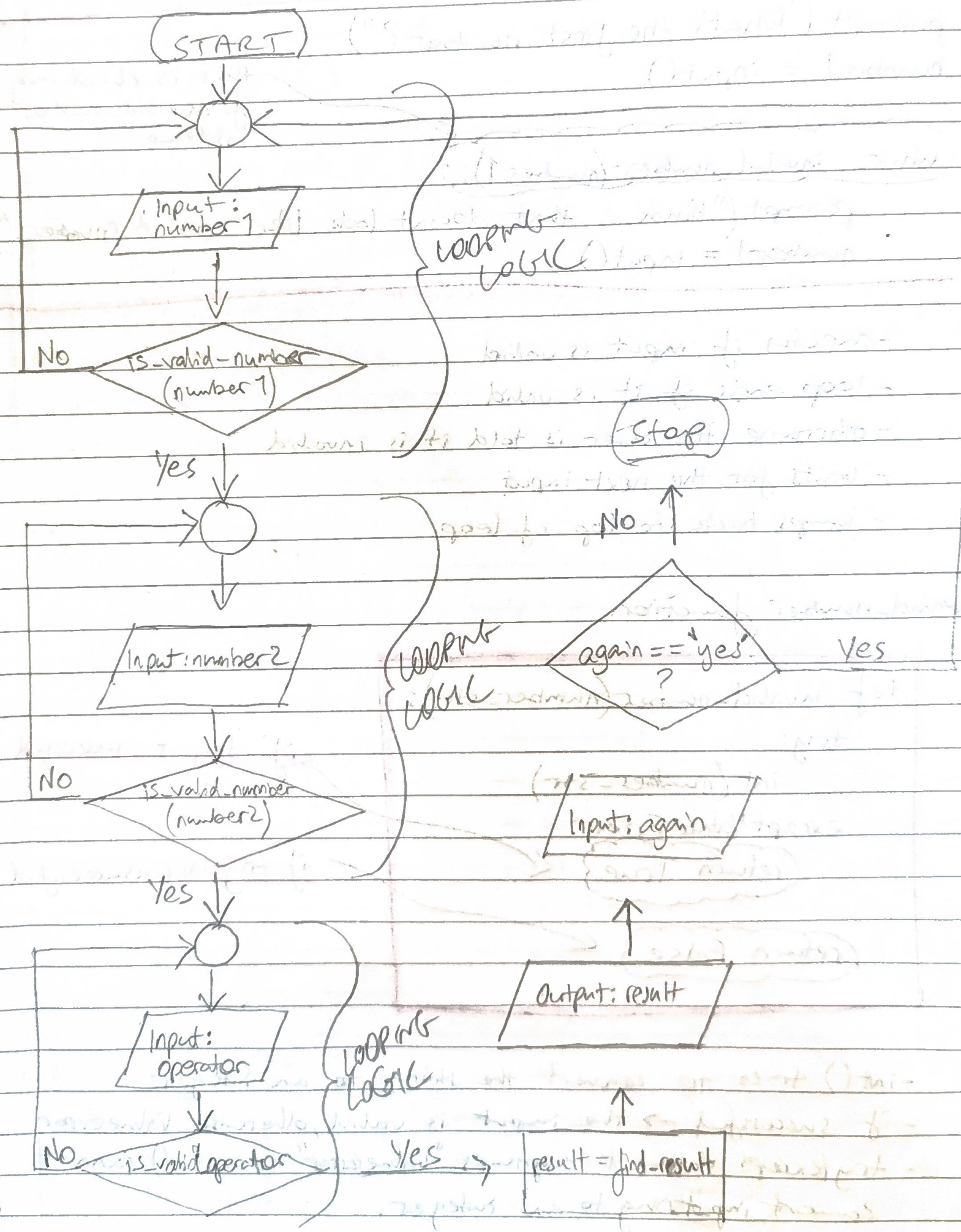
```
if operation == '1':
    output = int(number1) + int(number2)
elif operation == '2':
    output = int(number1) - int(number2)
elif operation == '3':
    output = int(number1) * int(number2)
elif operation == '4':
    output = int(number1) / int(number2)
```

```
match operation:
    case '1':
        output = int(number1) + int(number2)
    case 2:
        output = int(number1) - int(number2)
    case 3:
        output = int(number1) * int(number2)
    case 4:
        output = int(number1) / int(number2)
```

22/05/2025

Validating User Input

- Handy to use a flowchart diagram to represent the behaviour of the program.



22/05/2025

- To implement the looping logic into code, can do a number of things but for this example a while loop is suitable.

```
prompt ("What's the first number?")
number1 = input()

while invalid_number(number1):
    prompt ("Hmm... that doesn't look like a valid number.")
    number1 = input()
```

this is checking if invalid_number is true

- checks if input is valid
- loop ends if it is valid
- otherwise the user is told it is invalid
- waits for the next input
- jumps back to top of loop

- invalid_number function

```
def invalid_number(number_str):
    try:
        int(number_str)
    except ValueError:
        return True

    return False
```

if try is successful

if try is unsuccessful

- int() tries to convert the string to an integer
- if successful → the input is valid, otherwise ValueError
- try/except statement captures "ValueError" when int() cannot convert string to an integer.

22/01/2025

- Also need to validate the operation the user requested.
  - check if one of the four strings has been input by the user

```
prompt("What operation would you like to perform? \n1) Add 2) Subtract
operation = input()                                  3) Multiply 4) Divide

while operation not in ["1", "2", "3", "4"]:
    prompt('You must choose 1, 2, 3, or 4')
    operation = input()
```

Good use of A LIST