```
 ____   ___ __  _______    ___ _____
/ __/___ /()/_ /_ __//_ __  / _ )()//___
\_\V_\V//_/ ///_\V_\ /_ / ///__/
_/////// /_ /////_/ /_/ //(_ )
/___/ ___/_/_\___/ /_///_\___/ /___/_/_/___/
   /_/
```

```
> cd /Jungahahn_t1a3/src
> ruby main.rb
```

- > Overview
- > Development Process
- > Logic
- > Features
- > (END)

OVERVIEW

WAYS TO SPLIT $99 / THREE PEOPLE

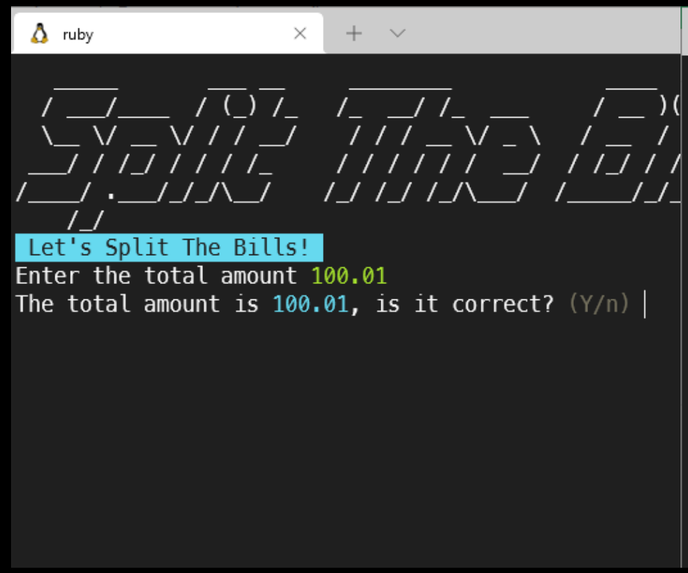
> Common option: $33, $33, $33

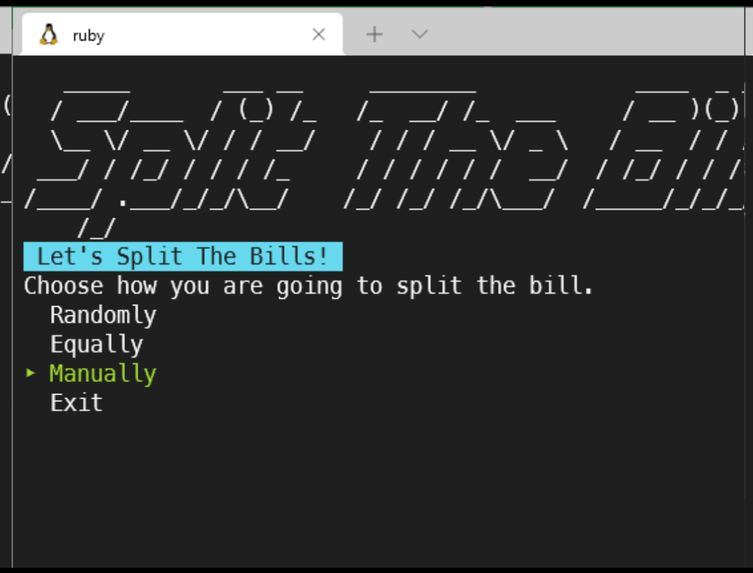
> What if there were other options?

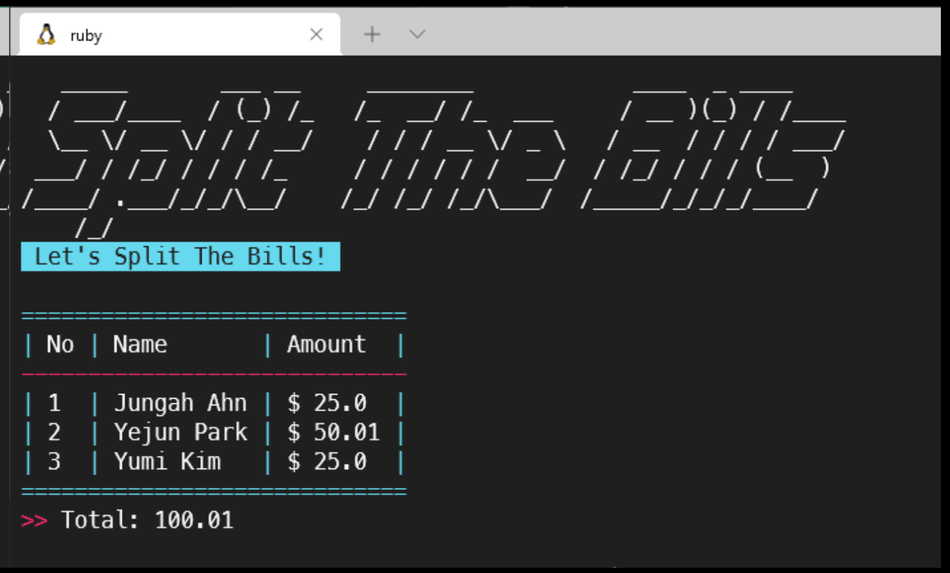
- Random option: $90, $9, $0
- Manual option: $50, ?, ?

To avoid needing change,
you want to pay $50 and then
your friends will share the rest

OVERVIEW





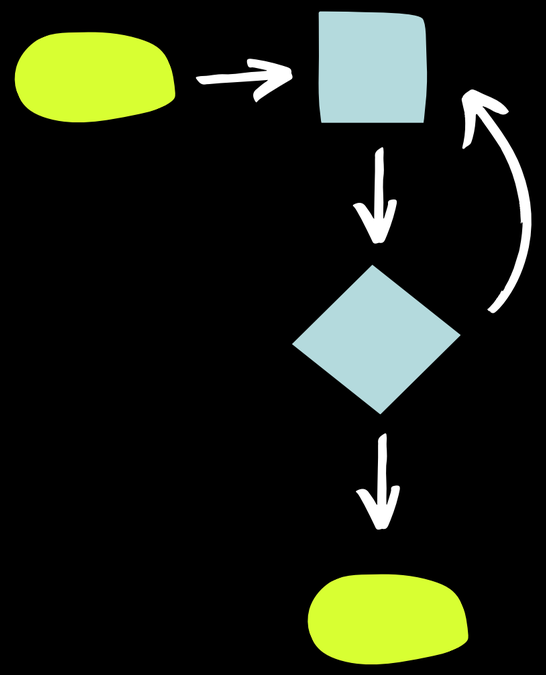


> Takes valid inputs

> Calculates them depending on user's choice

> Displays the result

DEVELOPMENT PROCESS



```
Calculator
  should be an instance of a Calculator
  .split_equally
    should be difined
  .pick_random_num
    should be difined
  .split_randomly
    should be difined
  .split_manually
    should be difined

Finished in 0.01 seconds (files took 0.18604 seconds to load)
5 examples, 0 failures
```

```
ruby

Split The Bill !
Enter names one by one.
Please type 'done' when you finish.
Name:
```

Control flow

Test

MVP

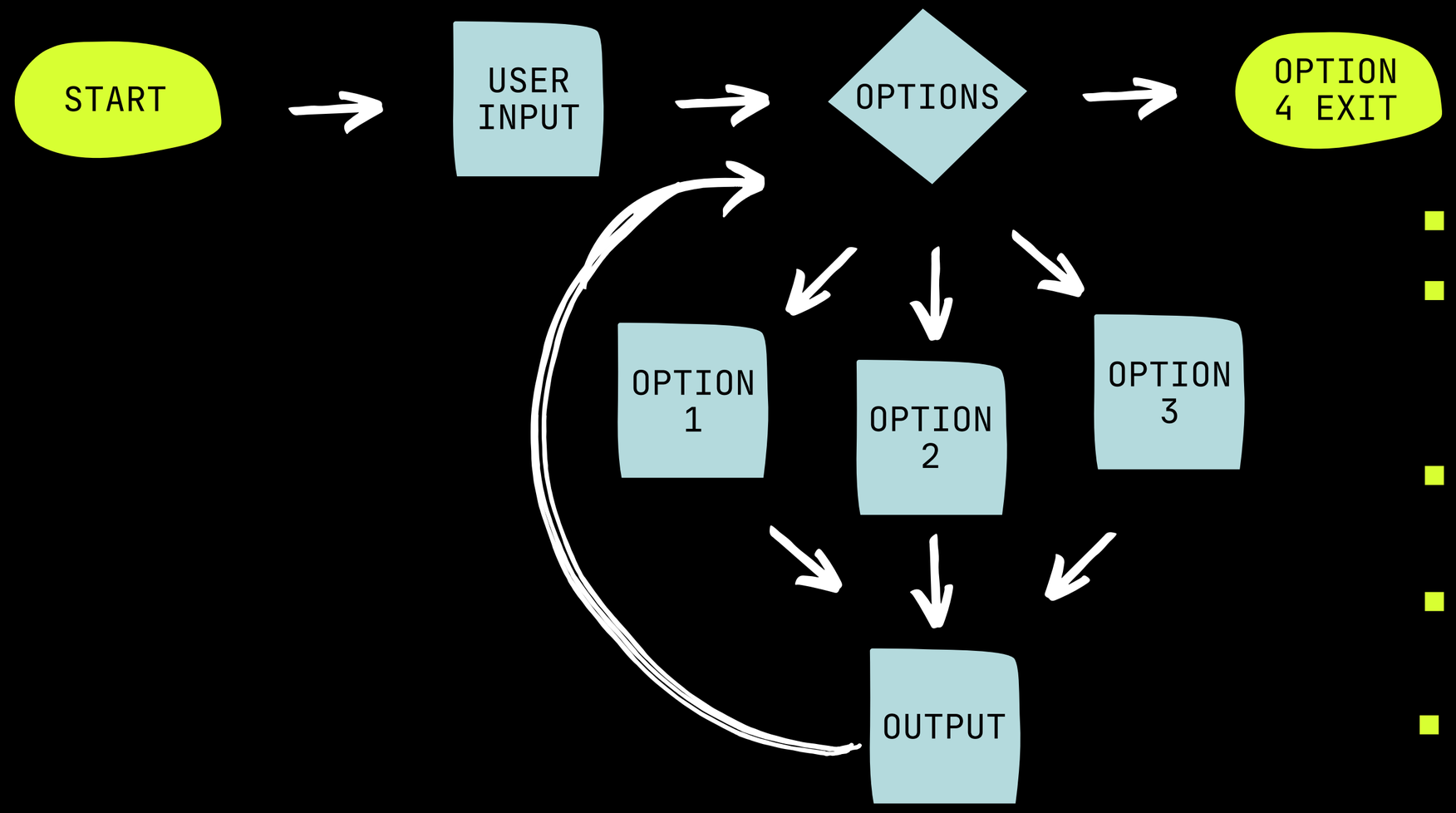
DEVELOPMENT PROCESS – CHALLENGES

> Alternative way of global variables to use them across the files

```ruby
class Variables
    attr_reader :instruction, :highlight, :err_msg

    def initialize
        @instruction = :light_blue
        @highlight = :light_red
        @err_msg = ">>".colorize(@highlight)
    end
end
```

Colour values for the colorize gem that will be used in different files

HIGH-LEVEL LOGIC



- Gets user's input
- Gives the user three calculation options
- Displays the outputs
- Goes back to the options
- Loops until the user selects the exit

HIGH-LEVEL LOGIC > CODE

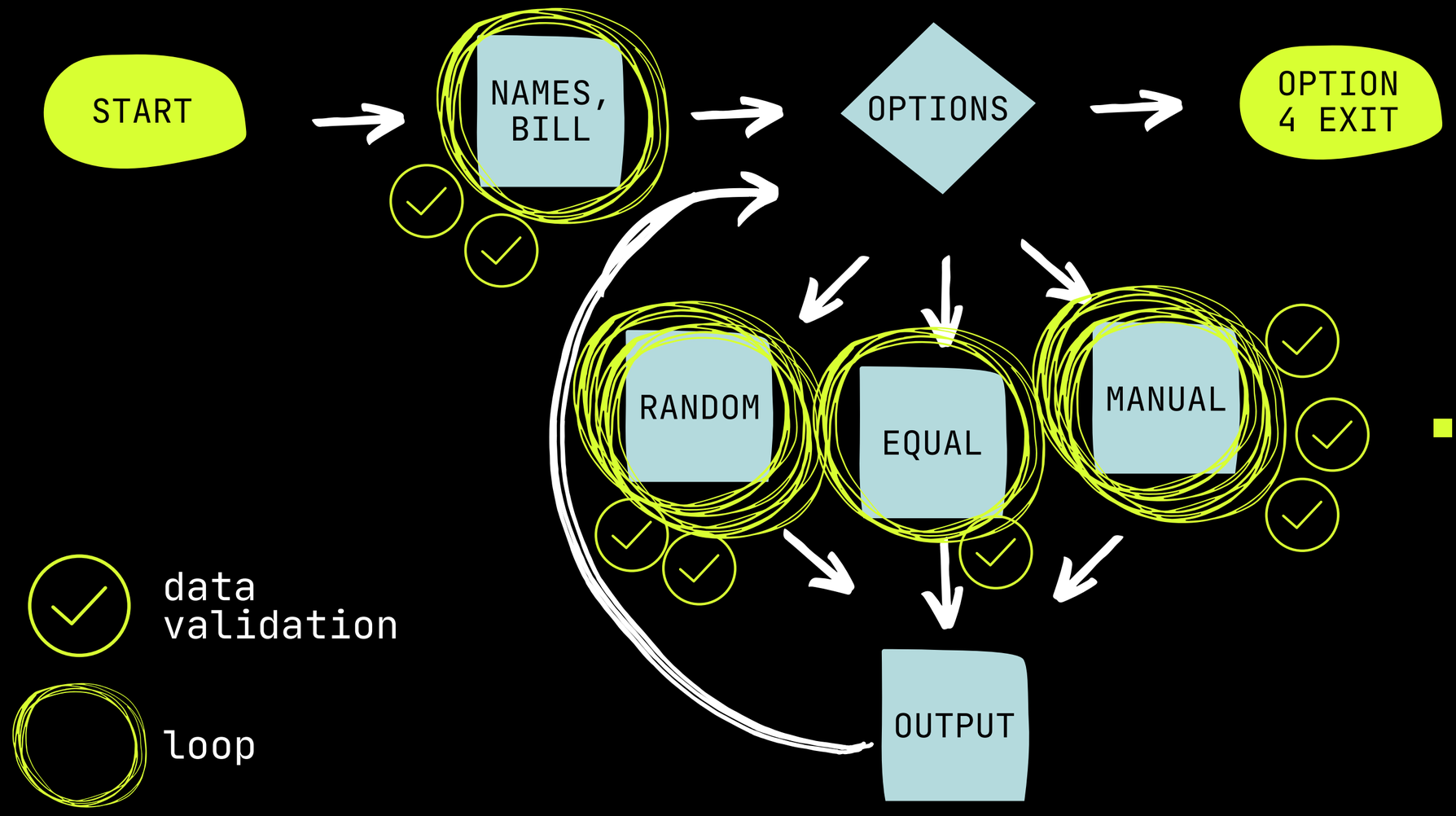
```ruby
heading(title)
# Generates an instance of Calculator class
calculator_instance = Calculator.new(name_array, bill, title)

while true
    choices = [
            {name: 'Randomly', value: 1},
            {name: 'Equally', value: 2},
            {name: 'Manually', value: 3},
            {name: 'Exit', value: 4}
        ]
    user_input = prompt.select("Choose how you are going to split the bill.", choices)

    case user_input
    when 1
        amount_array = calculator_instance.split_randomly
        calculator_instance.display(amount_array)
    when 2
        result_array = calculator_instance.split_equally
        calculator_instance.display(result_array)
    when 3
        manual_return = calculator_instance.split_manually
        calculator_instance.display(manual_return)
    when 4
        heading(title)
        puts "Bye for now!"
        exit
    end
end
```

> Creates an instance of the Calculator class

Based on a user's selection,

- Calls each calculation method
- Stores the return value in a variable
- Invokes the display method to show the output

HIGH-LEVEL LOGIC



- ‘tty-prompt’ gem handles most of the input validation, but there’s still a need to verify some other data, that returned from methods.

FEATURES

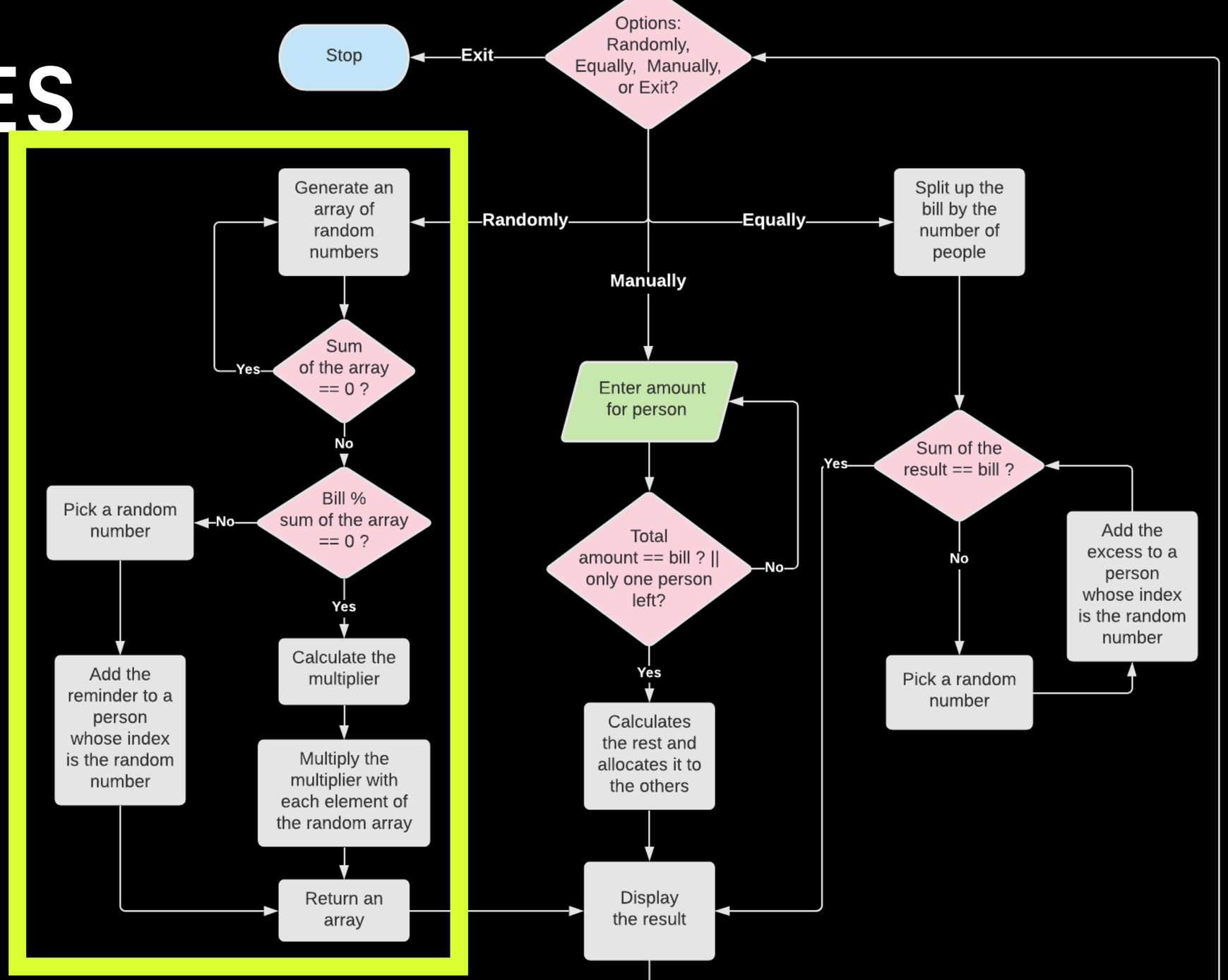
Random

> ZeroDivisionError handling

It keeps retrying to generate a valid array

> Edge case handling

Whether the sum of the elements in the array is not equal to the bill



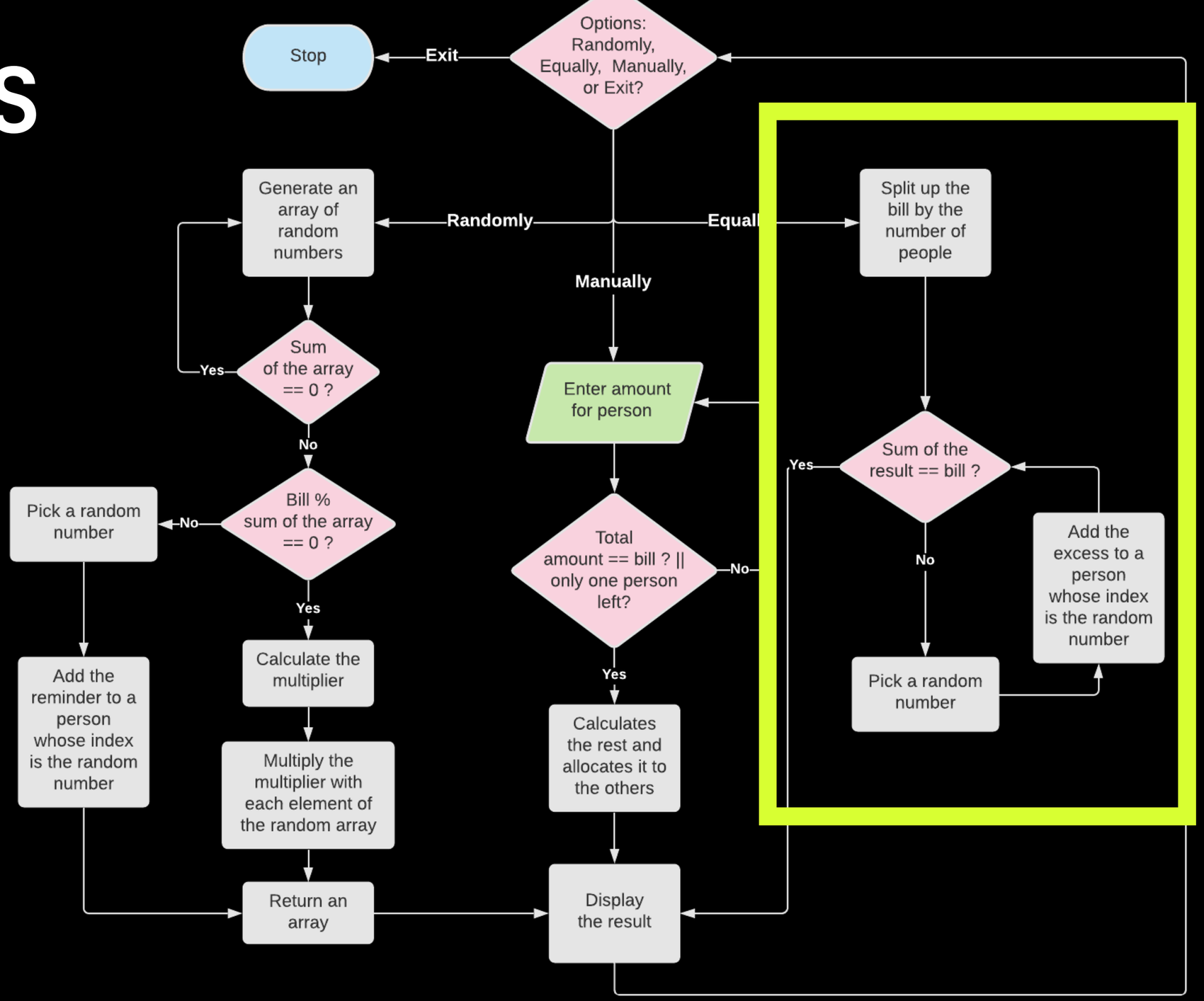
FEATURES

Equal

> Edge case handling

If the bill can't be equally split by the number of people, it randomly chooses one person to take the excess

※ This calculation is done to 2 decimal points so the sum of the 100/3 results 99.99, not 100. So it verifies the input amount and the sum of the result.



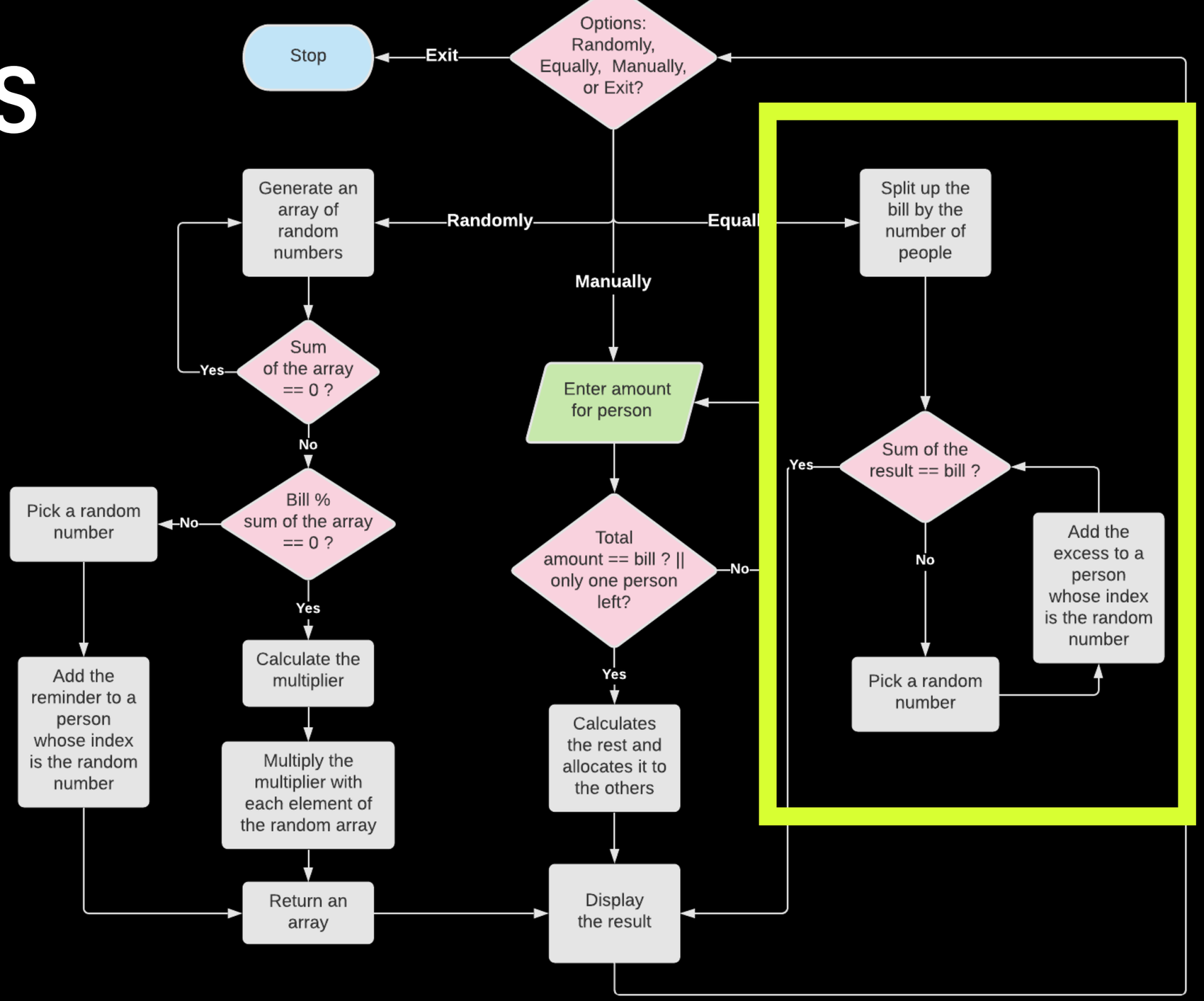
FEATURES

Equal

> Edge case handling

If the bill can't be equally split by the number of people, it randomly chooses one person to take the excess

※ This calculation is done to 2 decimal points so the sum of the 100/3 results 99.99, not 100. So it verifies the input amount and the sum of the result.



FEATURES

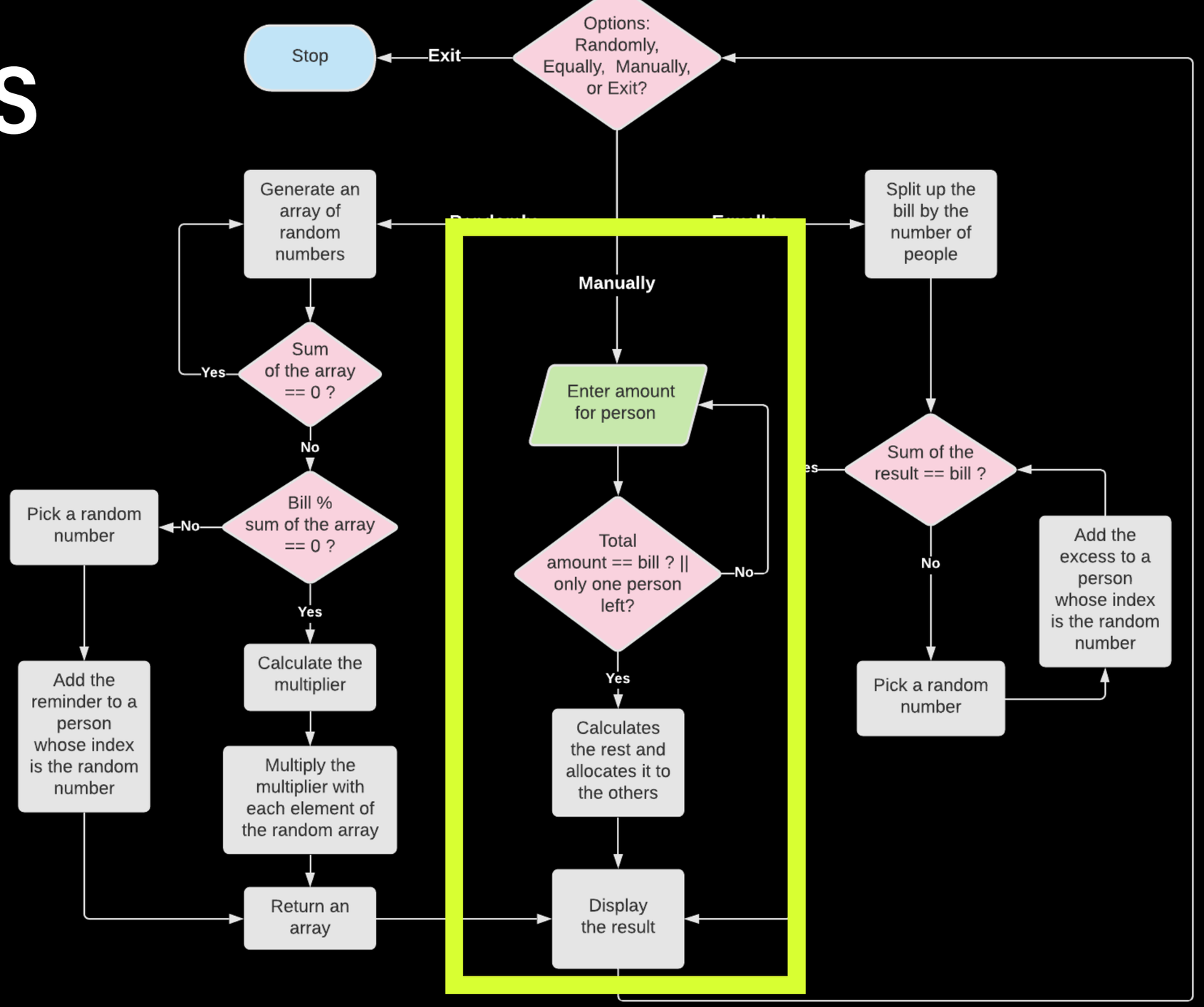
Manual

> **Edge case handling**

Whether the input amount is valid
- input amount <= bill

> **Automatic calculation**

When one person is left, after entering the amount for every other person, it automatically calculates the rest and allocates it to the last person



(END)

> Command line arguments

-help (-h) or -info (-i) are available

> Ethical issues?

This app may be misused as a sort of gambling if the user were not making good use of it.

> Favourite part - using gems!

Gems make the look better and handles input validation

> Preplanning - !important

Creating a flowchart and planning what the MVP is keeps the entire process on track