

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



Let's Split The Bills!

Enter names one by one.

Please type 'done' when you finish.

Name: |

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

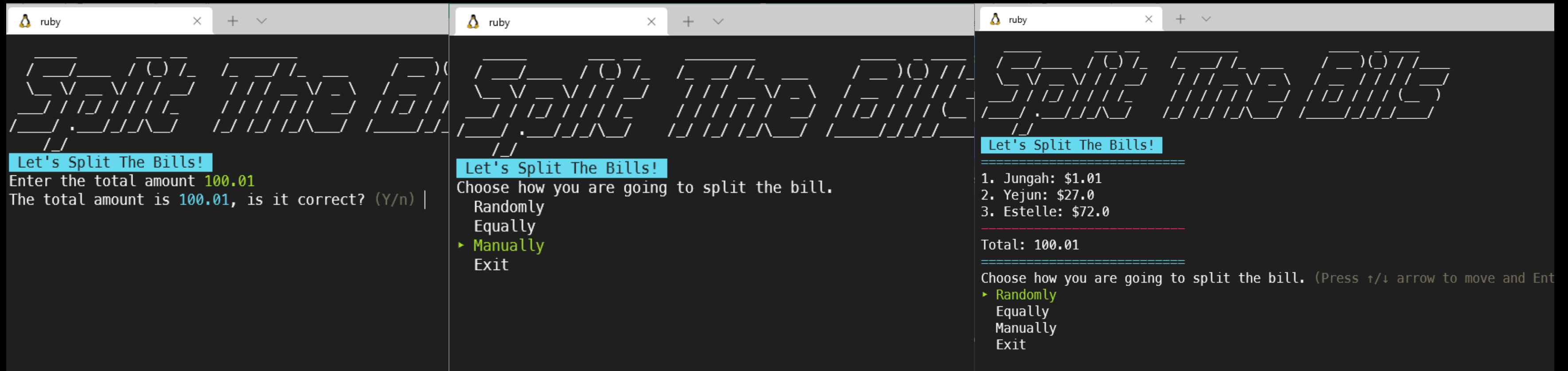
OVERVIEW

WAYS TO SPLIT \$99

- > 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



The image shows three terminal windows, each with a title bar that says 'ruby'. The first window displays a ASCII art logo and the prompt 'Let's Split The Bills!'. It shows the user entering '100.01' for the total amount, and the program confirming 'The total amount is 100.01, is it correct? (Y/n) |'. The second window shows the same prompt, but the user has chosen to split the bill. The program asks 'Choose how you are going to split the bill.' and lists options: 'Randomly', 'Equally', 'Manually' (which is highlighted with a green arrow), and 'Exit'. The third window shows the results of the split. It lists the amounts for three people: '1. Jungah: \$1.01', '2. Yejun: \$27.0', and '3. Estelle: \$72.0'. It also shows the 'Total: 100.01' and a prompt to 'Choose how you are going to split the bill. (Press ↑/↓ arrow to move and Enter)' with 'Randomly' highlighted.

```
ruby
Let's Split The Bills!
Enter the total amount 100.01
The total amount is 100.01, is it correct? (Y/n) |

ruby
Let's Split The Bills!
Choose how you are going to split the bill.
Randomly
Equally
▶ Manually
Exit

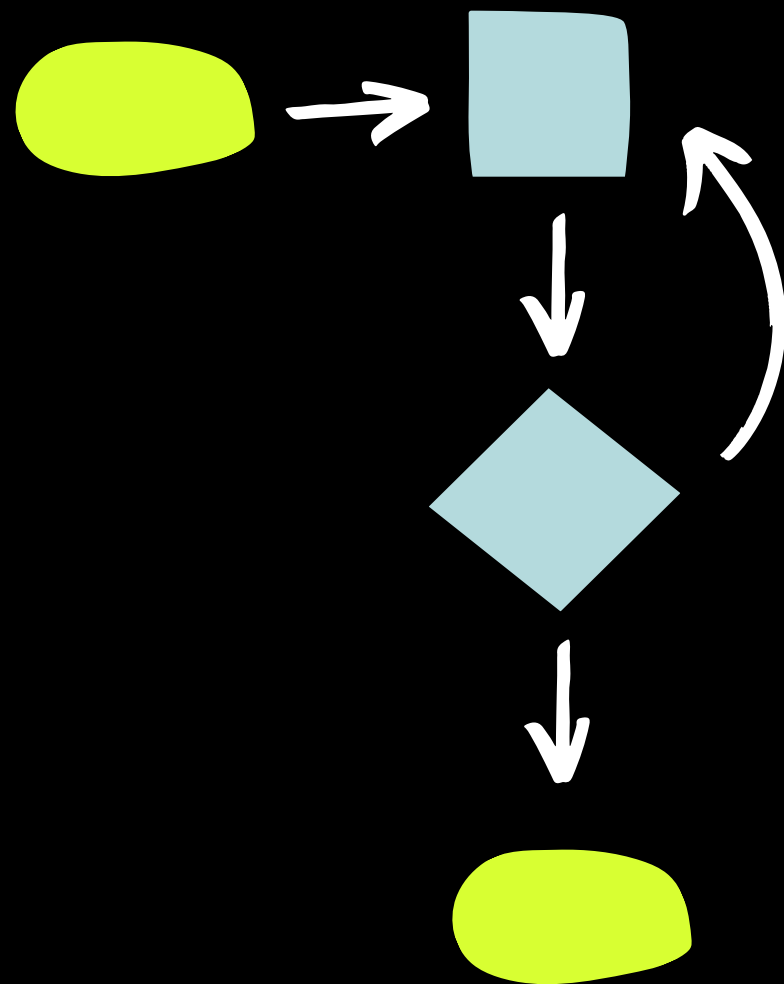
ruby
Let's Split The Bills!
=====
1. Jungah: $1.01
2. Yejun: $27.0
3. Estelle: $72.0
=====
Total: 100.01
=====
Choose how you are going to split the bill. (Press ↑/↓ arrow to move and Enter)
▶ Randomly
Equally
Manually
Exit
```

> Takes
valid
inputs

> Calculates
them
depending
on user's
choice

> Displays
the result

DEVELOPMENT PROCESS



Control
flow

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

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

Test

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

MVP

DEVELOPMENT PROCESS

- CHALLENGES

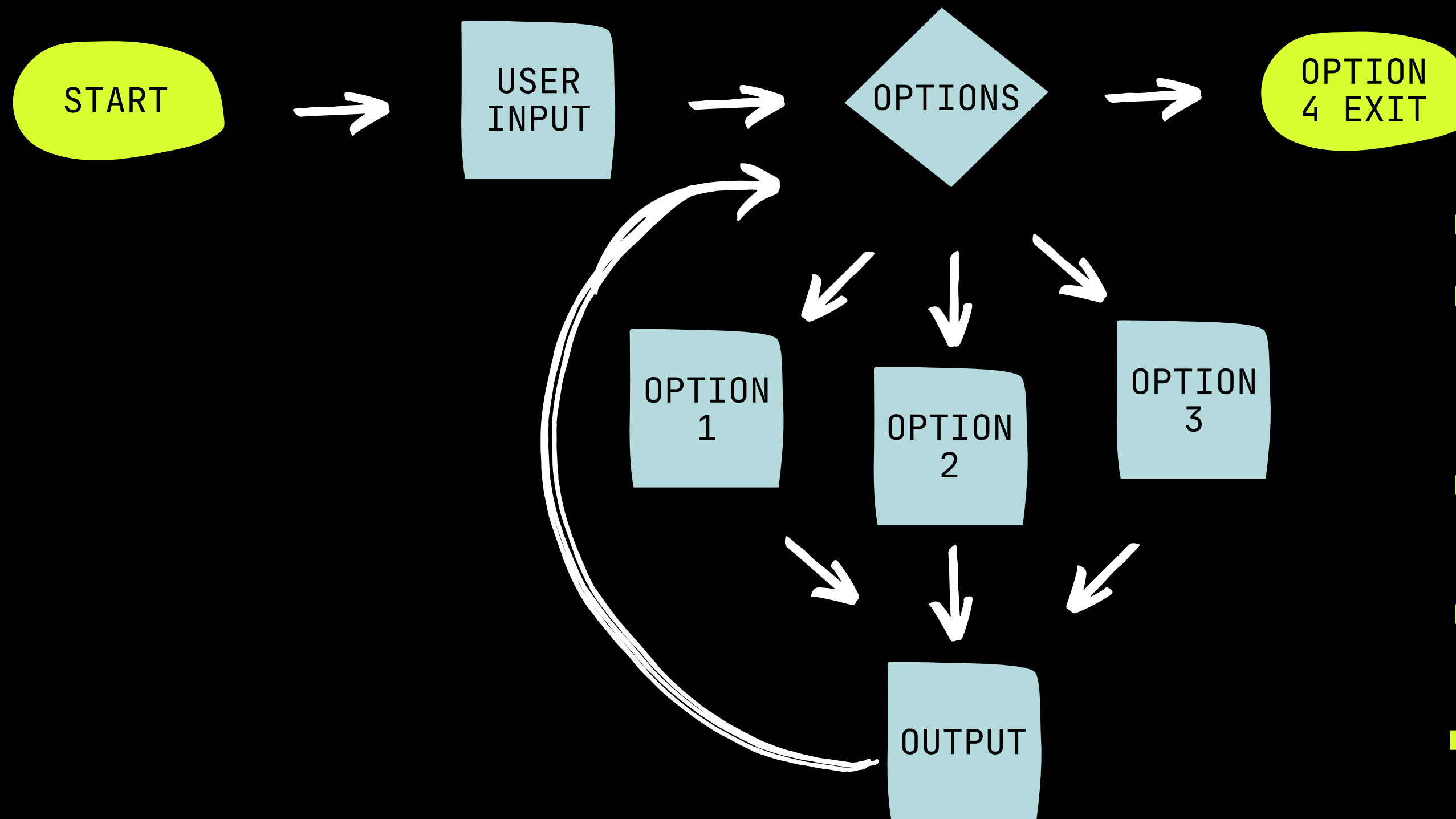
- > Alternative way of **global variables** to use them across the files

```
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

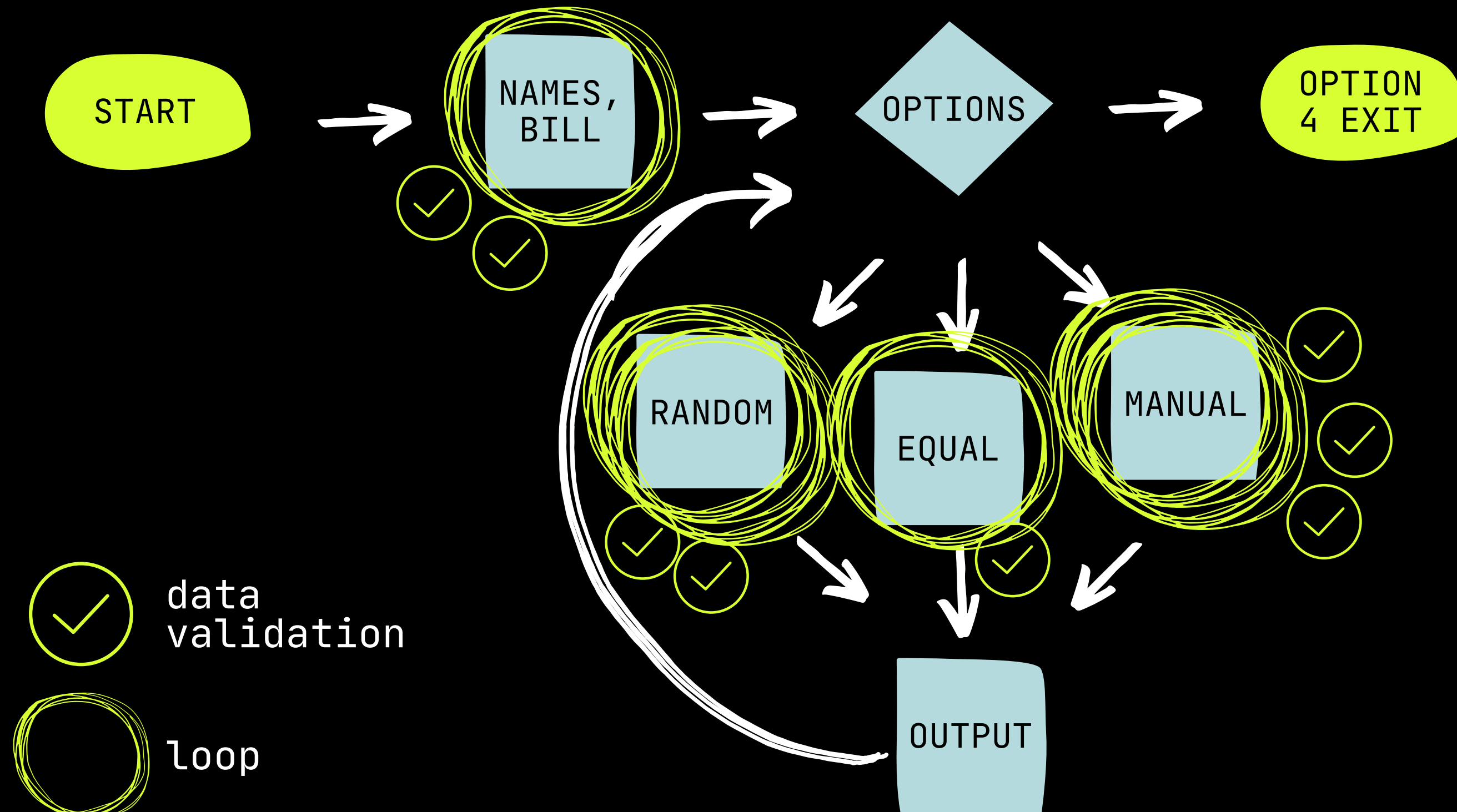
```
70 heading(title)
71 # Generates an instance of Calculator class
72 calculator_instance = Calculator.new(name_array, bill, title)
73
74 while true
75   choices = [
76     {name: 'Randomly', value: 1},
77     {name: 'Equally', value: 2},
78     {name: 'Manually', value: 3},
79     {name: 'Exit', value: 4}
80   ]
81   user_input = prompt.select("Choose how you are going to split the bill.", choices)
82
83   case user_input
84   when 1
85     amount_array = calculator_instance.split_randomly
86     calculator_instance.display(amount_array)
87   when 2
88     result_array = calculator_instance.split_equally
89     calculator_instance.display(result_array)
90   when 3
91     manual_return = calculator_instance.split_manually
92     calculator_instance.display(manual_return)
93   when 4
94     heading(title)
95     puts "Bye for now!"
96     exit
97   end
98 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

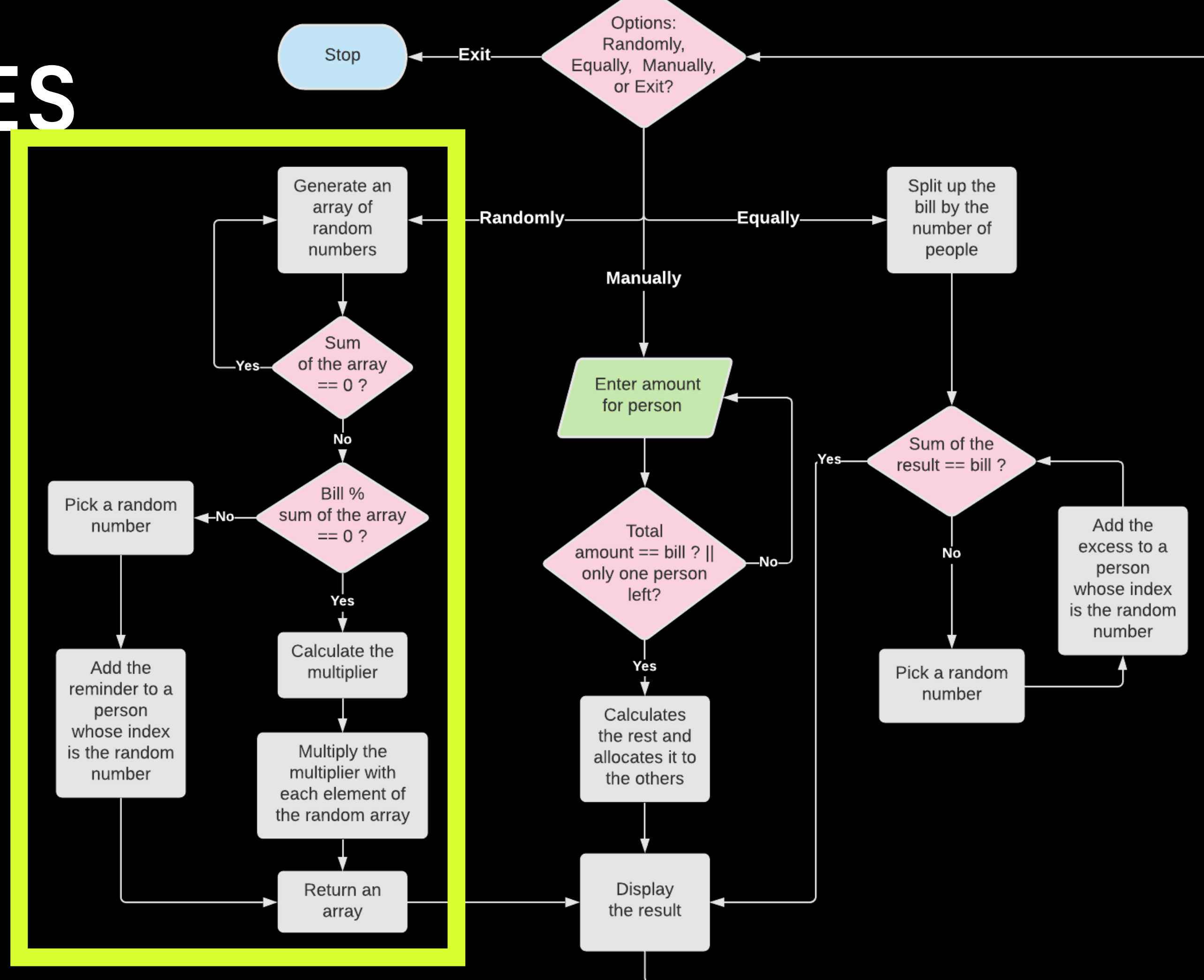
Random

> 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



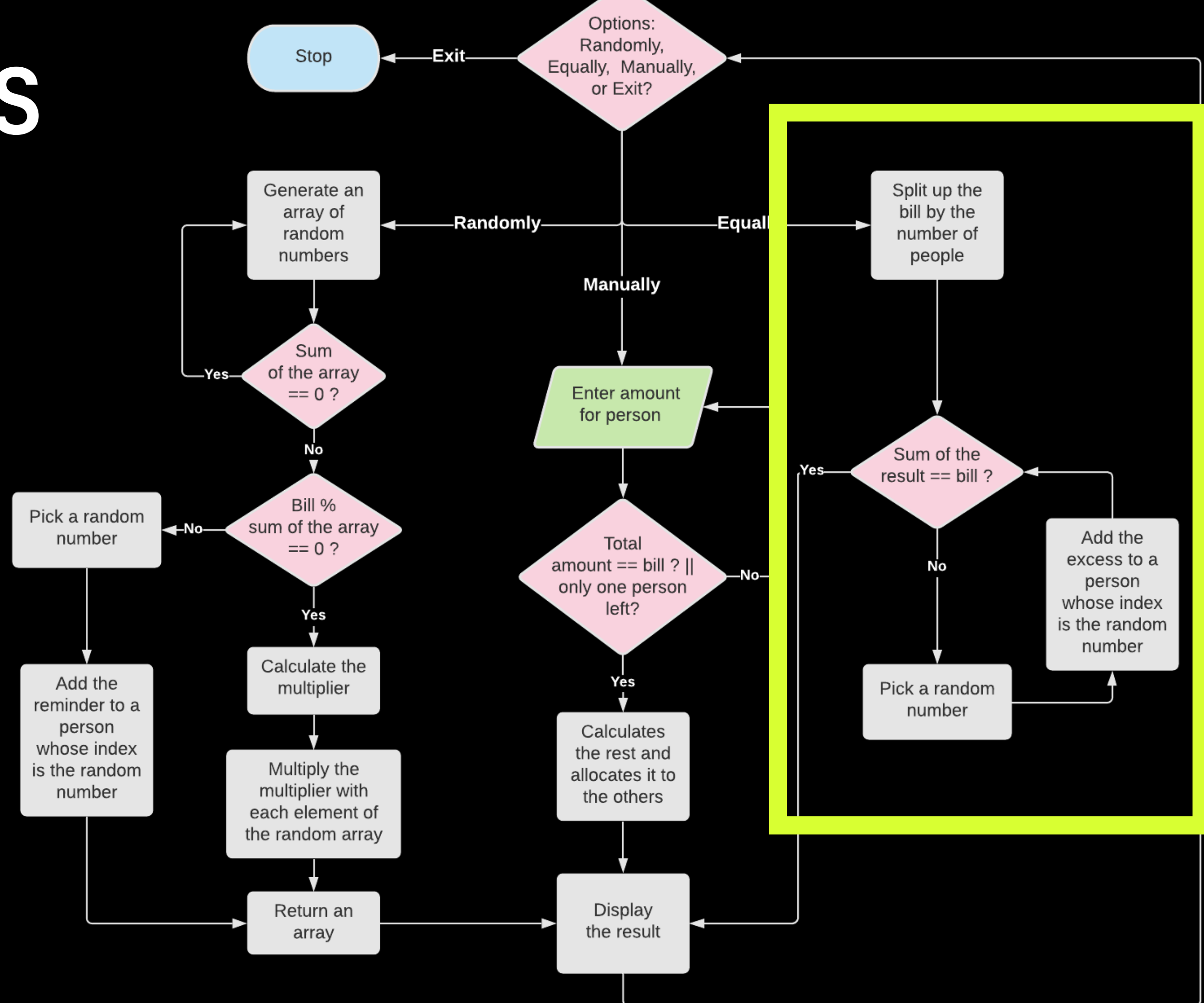
FEATURES

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.



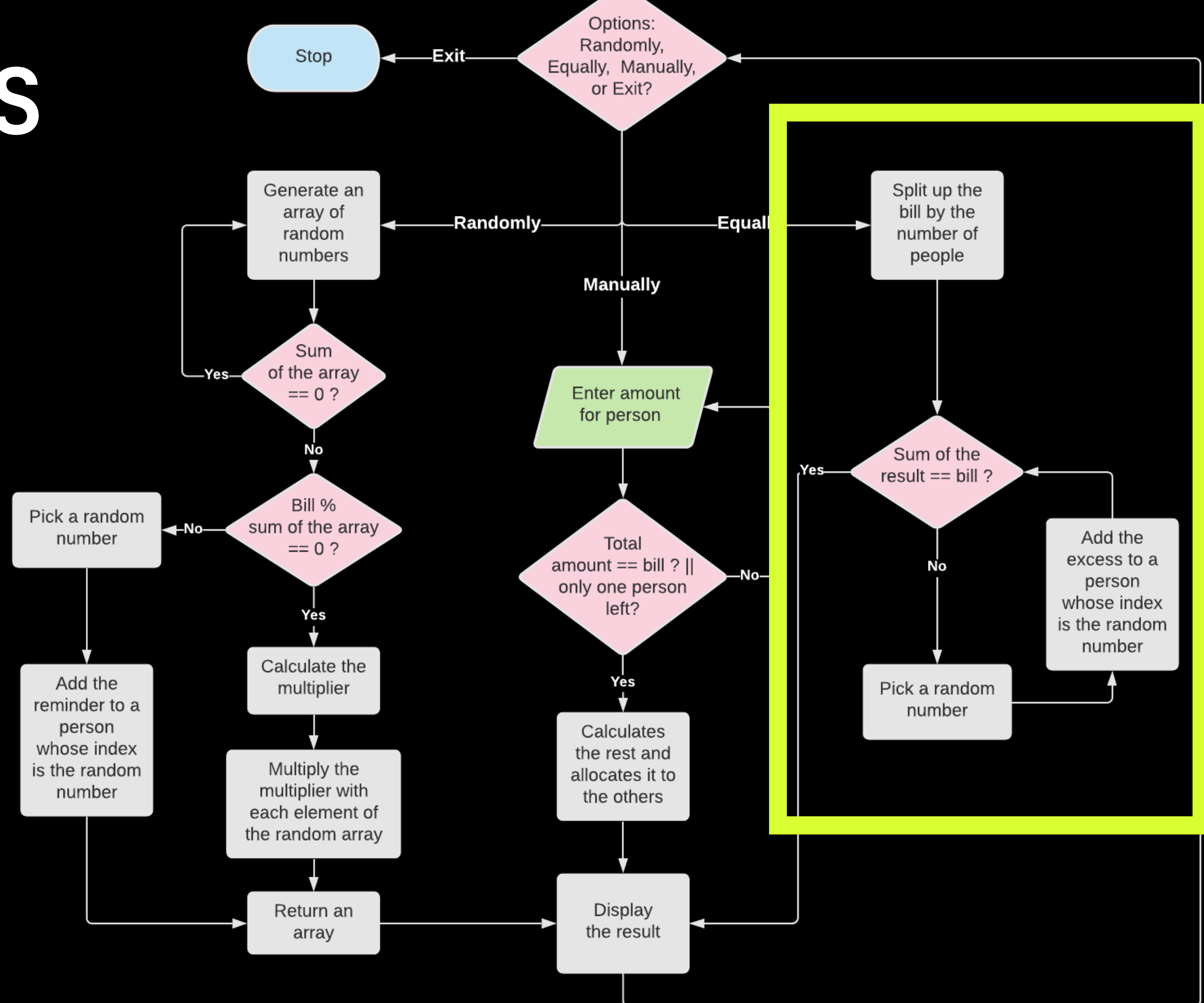
FEATURES

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

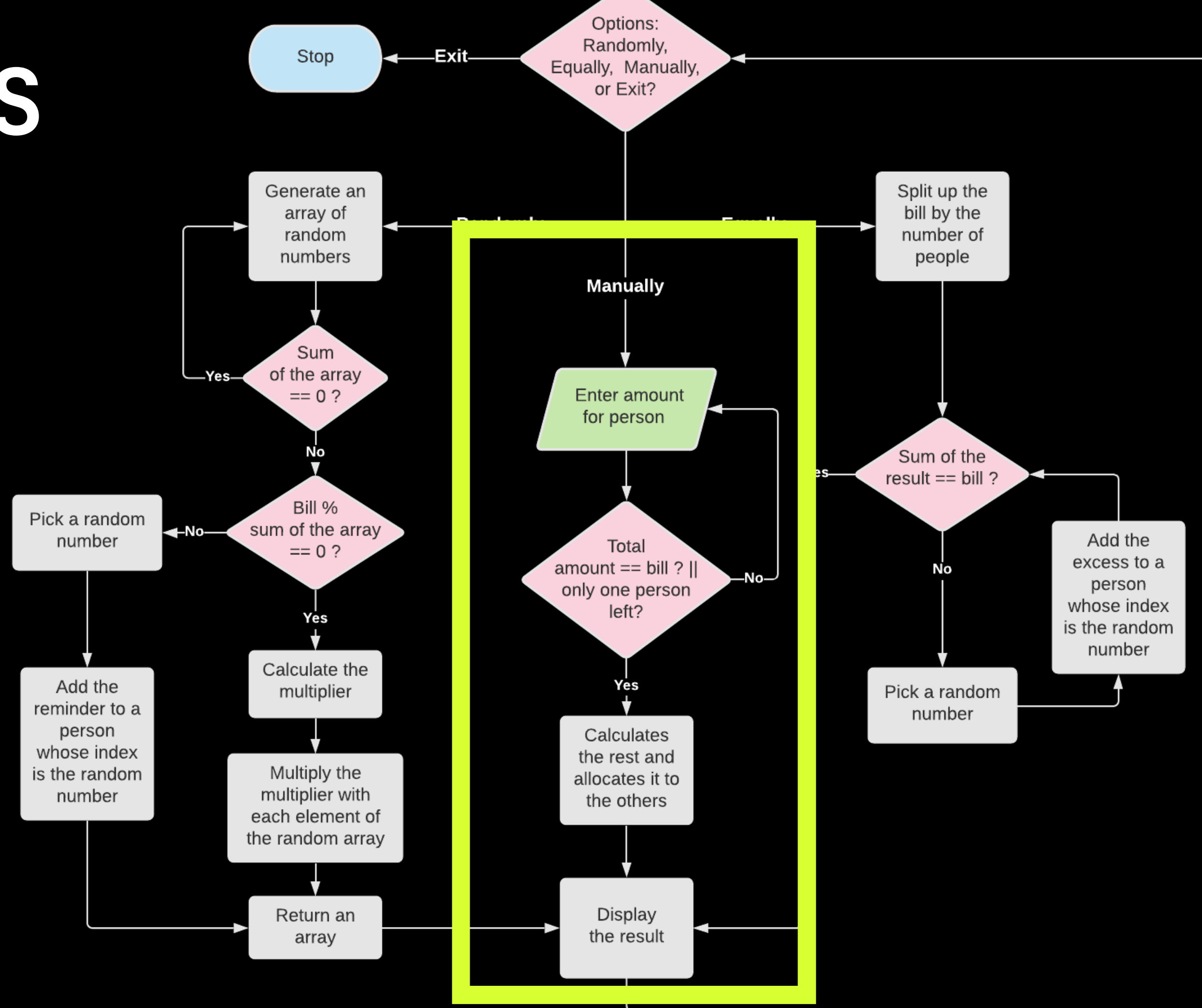
Manual

> 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