Procs

Passing blocks is one way to pass functions as arguments to other functions.

Blocks are one of the very few exceptions to the "everything is an object" rule in Ruby. Blocks are not objects, and they can't be saved to variables.

Proc objects are blocks of code that can be bound to a set of local variables. You can think of a *proc* object as a "saved" block.

Procs are a great way of keeping your code DRY. DRY stands for *Do not Repeat Yourself*.

Example:

CODE

```
def foo(a, b, my_proc)
    my_proc.call(a, b)
end

add = proc {|x, y| x + y}

puts foo(15, 10, add)
```

OUTPUT

```
25
```

In this example, we have created the proc *add*, which adds two numbers.
The proc *add* is passed as a parameter to the method *foo*.
In the method *foo*, `my_proc.call(a, b)` executes the proc.

Task

You are given a partially complete code. Your task is to fill in the blanks (`______`).

The *square_of_sum* method computes the sum of the elements in an input array and returns the square of the summed elements.

For example:

```
Input array: [1, 2, 3]
Output: 36
```