Lambdas

Lambdas are anonymous functions. *Lambdas* in Ruby are objects of the class *Proc*.
They are useful in most of the situations where you would use a proc.

The simplest lambda takes no argument and returns nothing as shown below:

Example:

```
#Ruby version <= 1.8
  lambda { .... }

  lambda do
    ....
  end

#Ruby version >= 1.9, "stabby lambda" syntax is added
  -> { .... }

  -> do
    ....
  end
```

Ruby version ≥ 1.9 can use both `lambda` and stabby lambda, `->`.

Lambdas can be used as arguments to higher-order functions. They can also be used to construct the result of a higher-order function that needs to return a function.

Example:

(a). Lambda that takes no arguments.

```
def area (l, b)
  -> { l * b }
end

x = 10.0; y = 20.0

area_rectangle = area(x, y).call
area_triangle = 0.5 * area(x, y).()

puts area_rectangle    #200.0
puts area_triangle     #100.0
```

(b). Lambda that takes one or more arguments.

```
area = ->(a, b) { a * b }

x = 10.0; y = 20.0

area_rectangle = area.(x, y)
area_triangle = 0.5 * area.call(x, y)

puts area_rectangle    #200.0
puts area_triangle     #100.0
```

In the above example, we can see that lambdas can be called using both `.call()` and `.()`.

Is there any difference between lambdas and procs?
Yes, there is difference between a proc and a lambda in Ruby.

Task

You are given a partially complete code. Your task is to fill in the blanks (`_____`).
There are 5 variables defined below:

- *square* is a lambda that squares an integer.
- *plus_one* is a lambda that increments an integer by 1.
- *into_2* is a lambda that multiplies an integer by 2.
- *adder* is a lambda that takes two integers and adds them.
- *values_only* is a lambda that takes a hash and returns an array of hash values.