# Closures

**Closure** is a function/method that:

► **Can be passed around like an object.**

It can be treated like a variable, which can be assigned to another variable, passed as an argument to a method.

► **Remembers the value of variables no longer in scope.**

It remembers the values of all the variables that were in scope when the function was defined. It is then able to access those variables when it is called even if they are in a different scope.

**Example:**

```
def plus_1(y)
  x = 100
  y.call      #remembers the value of x = 1
end

x = 1
y = -> { x + 1 }
puts plus_1(y)  #2
```

In this example, the variable $x$, which is closed within the lambda $y$, remembers its values. Here, $x$ remembers its value as $1$.

Blocks, Procs and Lambdas are closures in Ruby.

---

**Task**

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

$\rightarrow$ *block_message_printer* prints the message if the block exists.
$\rightarrow$ *proc_message_printer* prints the message inside a Proc.
$\rightarrow$ *lambda_message_printer* prints the message inside a Lambda.