RUBY PROGRAMMING :

In Ruby, everything is an object. Every bit of information and code can be given their own properties and actions. Object-oriented programming calls properties by the name instance variables and actions are known as methods. Ruby’s pure object-oriented approach is most commonly demonstrated by a bit of code which applies an action to a number.

In many languages, numbers and other primitive types are not objects. Ruby follows the influence of the Smalltalk language by giving methods and instance variables to all of its types. This eases one’s use of Ruby, since rules applying to objects apply to all of Ruby.

Ruby’s Flexibility

Ruby is seen as a flexible language, since it allows its users to freely alter its parts. Essential parts of Ruby can be removed or redefined, at will. Existing parts can be added upon. Ruby tries not to restrict the coder.

For example, addition is performed with the plus (+) operator. But, if you’d rather use the readable word plus, you could add such a method to Ruby’s builtin Numeric class.

### Blocks: a Truly Expressive Feature

Ruby’s block are also seen as a source of great flexibility. A programmer can attach a closure to any method, describing how that method should act. The closure is called a block and has become one of the most popular features for newcomers to Ruby from other imperative languages like PHP or Visual Basic.

Blocks are inspired by functional languages. Matz said, “in Ruby closures, I wanted to respect the Lisp culture[3](https://www.ruby-lang.org/en/about/#fn3).

Ruby and the Mixin

Unlike many object-oriented languages, Ruby features single inheritance only,**on purpose**. But Ruby knows the concept of modules (called Categories in Objective-C). Modules are collections of methods.

Classes can mixin a module and receive all its methods for free. For example, any class which implements the each method can mixin the Enumerablemodule, which adds a pile of methods that use each for looping.

**class** **MyArray.**

Ruby’s Visual Appearance

While Ruby often uses very limited punctuation and usually prefers English keywords, some punctuation is used to decorate Ruby. Ruby needs no variable declarations. It uses simple naming conventions to denote the scope of variables.

* var could be a local variable.
* @var is an instance variable.
* $var is a global variable.

These sigils enhance readability by allowing the programmer to easily identify the roles of each variable. It also becomes unnecessary to use a tiresomeself. prepended to every instance member.