

[Wikidot.com](http://Wikidot.com)

.wikidot.com

Share on      

[Edit](#) [History](#) [Tags](#) [Source](#)

[Explore »](#)

# Ruby Tutorial

## ...o como pasar un buen rato programando

- [admin](#)
  - [site manager](#)

[Create account](#) or [Sign in](#)



## Lección 1

- [Introducción](#)
- [Instalación](#)
- [El Primer Programa](#)
- [Números en Ruby](#)
- [Strings y diversión](#)
- [Variables](#)
- [Alcance de las variables](#)

## Lección 2

- [Introduciendo Datos](#)
- [Normas en los nombres](#)

- [Los métodos](#)
- [Los métodos: argumentos](#)
- [Rangos](#)
- [Arrays](#)

## Lección 3

- [Bloques](#)
- [Más malabares con strings](#)
- [Expresiones Regulares](#)
- [Condicionales](#)
- [Bucles](#)
- [Números Aleatorios](#)

## Lección 4

- [Clases y Objetos](#)
- [Accesores](#)
- [Ficheros: lectura/escritura](#)
- [Cargando librerías](#)
- [Herencia de clases](#)
- [Modificando clases](#)
- [Congelando objetos](#)
- [Serializando objetos](#)

## Lección 5

- [Control de acceso](#)
- [Excepciones](#)
- [Módulos](#)
- [Constantes](#)
- [Hashes y Símbolos](#)
- [La clase Time](#)

## Lección 6

- [self](#)
- [Duck Typing](#)
- [Azúcar Sintáctico](#)
- [Test de unidades](#)

## contacto

[e-mail](#)



Control de Acceso y Accesores

En Ruby, la única forma de cambiar el estado de un objeto, es invocando uno de sus métodos: si controlas el acceso a laso métodos, controlarás el acceso a los objetos. Una buena regla, es cerrar el acceso a los métodos que puedan dejar al objeto en un estado no válido.

## Los tres niveles de acceso

- **public** - los métodos públicos (public) pueden ser usados por cualquiera; no hay un control de acceso.
- **protected** - los métodos protegidos (protected) pueden ser usados únicamente por objetos de la misma clase y subclases, a las que pertenece el método; pero nunca por el propio objeto. Por así decirlo, el método sólo lo pueden usar los otros miembro de la familia.

- **private** - los métodos privados (private) sólo pueden ser usado por el propio objeto. Técnicamente, se dice que el receptor del método siempre es el mismo: self.

El control de acceso se determina dinámicamente, a medida que el programa transcurre. Se obtiene una violación de acceso siempre que se intenta ejecutar un método no público.

```
class ControlAcceso
  def m1      # este método es público
  end
  protected
    def m2    # este método es protegido
    end
  private
    def m3    # este método es privado
    end
    def m4
    end
end

ca = ControlAcceso.new
ca.m1
ca.m2
ca.m3
```

La privacidad de los métodos, también se pueden especificar de esta forma:

```
class ControlAcceso
  def m1      # este método es público
  end
  def m2      # este método es protegido
  end
  def m3      # este método es privado
  end
  def m4      # este método es privado
  public :m1
  protected :m2
  private :m3, :m4
end

ca = ControlAcceso.new
ca.m1
ca.m2
ca.m3
```

# protected

Tal vez el nivel de acceso protegido (protected) sea un poco lioso de entender. Es mejor verlo con un ejemplo:

```
class Persona
  def initialize(edad)
    @edad = edad
  end
  def edad
    @edad
  end
  def comparar_edad(op) # op = otra persona
    if op.edad > edad
      'La edad de la otra persona es mayor.'
    else
      'La edad de la otra persona es la misma o menor.'
    end
  end
  protected :edad
end

pedro = Persona.new(15)
almudena = Persona.new(17)
puts Pedro.comparar_edad(almudena) # La edad ... es mayor
```

El objeto que hace la comparación (pedro) necesita preguntar al otro objeto (almudena) su edad, lo que significa que ejecute su método edad. Por eso el nivel de acceso es protegido y no privado: al estar protegido "pedro" puede usar el método de "almudena".

La excepción viene cuando "pedro" se pregunta a sí mismo la edad, por ser un método protegido, esto no será posible. self no puede ser el receptor de un método protegido.

Por ejemplo:

```
puts Pedro.edad #da error
```

page\_revision: 2, last\_edited: 13 Dec 2007, 06:43 GMT-06 (1083 days ago)

[EditTags](#) [History](#) [Files](#) [Print](#) [Site tools+ Options](#)

[Help](#) | [Terms of Service](#) | [Privacy](#) | [Report a bug](#) | [Flag as objectionable](#)

Powered by [Wikidot.com](#)

Unless otherwise stated, the content of this page is licensed under [Creative Commons Attribution-ShareAlike 3.0 License](#)

## Other interesting sites



### [Ammon Allred's Online Classroom](#)

University of Toledo Philosophy Professor



### [Gilchrist 21st CCLC](#)

Gilchrist Co.'s 21st Century After-School Program



## SFU Game Developers Club



## Neo Steam Wiki

I put on my muumuu and pufu hat