# Ruby Riferimento Rapido

Oggetti, blocchi, iteratori, regex, I/O su file essenziali

## Basi

### Hello World

```
puts "Hello, World!"
print "senza newline"
p [1, 2, 3]   # output inspect: [1, 2, 3]
```

### Eseguire Ruby

```
ruby script.rb          # esegui un file
ruby -e 'puts "hi"'     # esegui inline
irb                     # REPL interattivo
```

### Variabili

| | |
|---|---|
| **name** | Variabile locale |
| **@name** | Variabile di istanza |
| **@@count** | Variabile di classe |
| **$debug** | Variabile globale |
| **MAX_SIZE** | Costante (maiuscolo per convenzione) |

### Tipi

```
42.class         # Integer
3.14.class       # Float
"hello".class    # String
true.class       # TrueClass
nil.class        # NilClass
:symbol.class    # Symbol
```

## Stringhe

### Basi delle Stringhe

```
name = "World"
puts "Hello, #{name}!"        # interpolazione (doppi apici)
puts 'No #{interpolation}'    # letterale (singoli apici)
multi = <<~HEREDOC
  heredoc indentato
HEREDOC
```

### Metodi sulle Stringhe

| | |
|---|---|
| **.length / .size** | Conteggio caratteri |
| **.upcase / .downcase** | Conversione maiuscolo/minuscolo |
| **.strip** | Rimuovi spazi iniziali/finali |
| **.split(',')** | Divide in array |
| **.gsub(/pat/, 'rep')** | Sostituzione globale |
| **.include?('sub')** | Verifica se contiene sottostringa |
| **.start_with?('pre')** | Verifica prefisso |
| **.chars / .bytes** | Array di caratteri / byte |
| **.to_i / .to_f** | Converti in intero / float |
| **.freeze** | Rende la stringa immutabile |

## Array e Hash

### Array

```
arr = [1, "two", :three]
arr << 4                  # push (aggiunge in fondo)
arr[0]                    # 1
arr[-1]                   # 4 (ultimo elemento)
arr[1..2]                 # ["two", :three] (fetta)
```

### Metodi sugli Array

| | |
|---|---|
| **.push / .pop** | Aggiunge/rimuove dalla fine |
| **.shift / .unshift** | Rimuove/aggiunge dall'inizio |
| **.flatten** | Appiattisce array annidati |
| **.compact** | Rimuove i valori nil |
| **.uniq** | Rimuove duplicati |
| **.sort / .reverse** | Ordina / ordine inverso |
| **.map { \|x\| x * 2 }** | Trasforma ogni elemento |
| **.select { \|x\| x > 0 }** | Filtra gli elementi |
| **.reduce(0) { \|sum, x\| sum + x }** | Accumula in un singolo valore |

### Hash

```
user = { name: "Alice", age: 30 }   # chiavi symbol
old = { "key" => "value" }           # chiavi stringa
user[:name]                          # "Alice"
user[:email] = "a@b.com"             # aggiunge coppia
user.fetch(:name, "default")         # con default
```

### Metodi sugli Hash

| | |
|---|---|
| **.keys / .values** | Array di chiavi / valori |
| **.each { \|k, v\| }** | Itera coppie chiave-valore |
| **.merge(other)** | Unisce due hash |
| **.key?(k) / .value?(v)** | Verifica esistenza |
| **.select { \|k, v\| }** | Filtra coppie |
| **.transform_values { \|v\| }** | Trasforma tutti i valori |

## Flusso di Controllo

### Condizionali

```
if score >= 90 then "A"
elsif score >= 80 then "B"
else "C"
end
puts "adulto" if age >= 18     # if inline
puts "minore" unless age >= 18 # unless inline
```

### Case / When

```
case status
when :ok then puts "successo"
when :error then puts "fallito"
when 400..499 then puts "errore client"
else puts "sconosciuto"
end
```

### Cicli

```
5.times { |i| puts i }
(1..10).each { |n| puts n }
while condition do end
until condition do end
loop { break if done }
```

### Ternario e Logici

```
status = age >= 18 ? "adulto" : "minore"
name = input || "default"     # or-assign
name ||= "fallback"           # stesso effetto
```

## Metodi

### Definire Metodi

```
def greet(name, greeting = "Hello")
  "#{greeting}, #{name}!"
end
greet("Alice")          # "Hello, Alice!"
greet("Bob", "Hi")      # "Hi, Bob!"
```

### Valori di Ritorno

```
def add(a, b)
  a + b    # ultima espressione è il return implicito
end
def divide(a, b)
  return nil if b == 0
  a.to_f / b
end
```

### Argomenti Keyword e Splat

```
def connect(host:, port: 80, **opts)
  puts "#{host}:#{port} #{opts}"
end
def log(*messages)
  messages.each { |m| puts m }
end
```

### Convenzioni sui Metodi

| | |
|---|---|
| **method?** | Restituisce booleano (predicato) |
| **method!** | Muta il ricevitore (metodo bang) |
| **self.method** | Definizione di metodo di classe |

## Classi

### Definizione di Classe

```
class User
  attr_accessor :name, :email
  def initialize(name, email)
    @name = name
    @email = email
  end
end
```

### Ereditarietà

```
class Admin < User
  def initialize(name, email, level)
    super(name, email)
    @level = level
  end
end
```

### Controllo degli Accessi

| | |
|---|---|
| **public** | Default; accessibile da qualsiasi punto |
| **private** | Accessibile solo all'interno della classe |
| **protected** | Accessibile nella classe e nelle sottoclassi |
| **attr_reader** | Genera metodo getter |
| **attr_writer** | Genera metodo setter |
| **attr_accessor** | Genera getter e setter |

## Moduli

### Mixin

```
module Greetable
  def greet
    "Ciao, sono #{name}"
  end
end
class User; include Greetable; end
```

### Namespace

```
module Payment
  class Processor
    def charge(amount) end
  end
end
p = Payment::Processor.new
```

# Ruby Riferimento Rapido

### Include vs Extend

| | |
|---|---|
| `include ModName` | Aggiunge come metodi di istanza |
| `extend ModName` | Aggiunge come metodi di classe |
| `prepend ModName` | Inserisce prima della classe nella catena dei metodi |

## Blocchi e Iteratori

### Sintassi dei Blocchi

```
[1, 2, 3].each { |n| puts n }        # blocco su una riga
[1, 2, 3].each do |n|
  puts n                               # blocco multi-riga
end
```

### Yield

```
def with_logging
  puts "inizio"
  result = yield
  puts "fine"
  result
end
with_logging { expensive_operation }
```

### Proc e Lambda

```
square = Proc.new { |x| x ** 2 }
square.call(5)                  # 25
double = ->(x) { x * 2 }        # lambda
double.call(3)                   # 6
[1, 2, 3].map(&square)           # [1, 4, 9]
```

### Iteratori Comuni

| | |
|---|---|
| `.each` | Itera sugli elementi |
| `.map / .collect` | Trasforma ogni elemento |
| `.select / .filter` | Mantiene gli elementi corrispondenti |
| `.reject` | Rimuove gli elementi corrispondenti |
| `.reduce / .inject` | Accumula in un singolo valore |
| `.each_with_index` | Itera con indice |
| `.flat_map` | Map e appiattisce un livello |
| `.any? / .all? / .none?` | Controlli booleani sulla collezione |

## Regex

### Corrispondenza

```
"hello 42" =~ /\d+/         # 6 (posizione corrispondenza)
"hello" =~ /\d+/            # nil (nessuna corrispondenza)
"hello".match?(/ell/)       # true
md = "age: 30".match(/(\d+)/)
md[1]                        # "30"
```

### Pattern Comuni

| | |
|---|---|
| `/^start/` | Ancorato all'inizio |
| `/end$/` | Ancorato alla fine |
| `/\d+/` | Una o più cifre |
| `/\w+/` | Caratteri parola |
| `/\s+/` | Spazi bianchi |
| `/[a-z]+/i` | Case-insensitive |
| `/(group)/` | Gruppo di cattura |

### Sostituzione

```
"hello world".sub(/world/, "Ruby")     # prima corrispondenza
"aabba".gsub(/a/, "x")                 # tutte: "xxbbx"
"foo bar".gsub(/(\w+)/) { $1.upcase }  # "FOO BAR"
```

## I/O su File

### Leggi e Scrivi

```
content = File.read("data.txt")
lines = File.readlines("data.txt", chomp: true)
File.write("out.txt", "hello\n")
File.open("log.txt", "a") { |f| f.puts "entry" }
```

### Operazioni sui File

| | |
|---|---|
| `File.exist?(path)` | Verifica se il file esiste |
| `File.directory?(path)` | Verifica se il percorso è una directory |
| `File.basename(path)` | Nome file senza directory |
| `File.extname(path)` | Estensione del file |
| `File.size(path)` | Dimensione del file in byte |
| `File.delete(path)` | Elimina un file |
| `Dir.glob('*.rb')` | Trova file corrispondenti al pattern |
| `FileUtils.mkdir_p(path)` | Crea directory in modo ricorsivo |

### CSV e JSON

```
require "json"
data = JSON.parse(File.read("data.json"))
File.write("out.json", JSON.pretty_generate(data))
require "csv"
CSV.foreach("data.csv", headers: true) { |row| puts row["name"] }
```