Guida alla Programmazione in Ruby

1. Cos'è Ruby?

Ruby è un linguaggio di programmazione interpretato, orientato agli oggetti, e noto per la sua semplicità e leggibilità. È stato progettato per essere produttivo e piacevole da scrivere. Ruby è comunemente usato per sviluppare applicazioni web, soprattutto grazie al framework Ruby on Rails.

2. Installazione e Ambiente di Sviluppo

Per iniziare con Ruby, è necessario installare Ruby stesso. Su macOS e Linux, Ruby è spesso preinstallato, mentre su Windows puoi installarlo tramite il pacchetto **RubyInstaller**.

Per verificare l'installazione, esegui:

bash
Copia codice
```
ruby -v
```

Per scrivere codice Ruby, puoi usare un qualsiasi editor di testo, come **Visual Studio Code**, oppure ambienti di sviluppo dedicati come **RubyMine**.

3. Il Primo Programma in Ruby

Crea un file chiamato `hello.rb` e aggiungi questo codice:

ruby
Copia codice
```
puts "Hello, World!"
```

Esegui il programma con:

bash
Copia codice
```
ruby hello.rb
```

4. Variabili e Tipi di Dati

In Ruby, le variabili non richiedono una dichiarazione di tipo specifica (è un linguaggio dinamico):

ruby
Copia codice

```ruby
nome = "Alice"      # Stringa
età = 25            # Intero
altezza = 1.75      # Float
is_admin = true     # Booleano
```

Tipi di Dati Principali

- **String**: Sequenza di caratteri, es. `"Ciao"`
- **Integer**: Numeri interi, es. `10`
- **Float**: Numeri a virgola mobile, es. `3.14`
- **Boolean**: Valori booleani `true` o `false`
- **Array**: Lista di valori, es. `[1, 2, 3, "quattro"]`
- **Hash**: Collezione di coppie chiave-valore, es. `{"nome" => "Alice", "età" => 25}`

5. Operatori

Ruby supporta vari tipi di operatori:

- **Aritmetici**: `+`, `-`, `*`, `/`, `%`
- **Confronto**: `==`, `!=`, `<`, `>`, `<=`, `>=`
- **Logici**: `&&`, `||`, `!`
- **Di assegnazione**: `=`, `+=`, `-=`, `*=`, `/=`

6. Condizioni

Ruby utilizza strutture condizionali per il controllo del flusso:

ruby
Copia codice

```ruby
età = 20
if età >= 18
  puts "Sei maggiorenne"
else
  puts "Sei minorenne"
end
```

Elsif consente di aggiungere condizioni aggiuntive:

ruby
Copia codice

```ruby
età = 16
if età >= 18
```

```ruby
  puts "Adulto"
elsif età >= 13
  puts "Adolescente"
else
  puts "Bambino"
end
```

7. Cicli

Ruby supporta diversi cicli per iterare sui dati.

Ciclo `while`

ruby
Copia codice

```ruby
i = 0
while i < 5
  puts i
  i += 1
end
```

Ciclo `for`

ruby
Copia codice

```ruby
for i in 1..5
  puts i
end
```

Ciclo `each` (usato con Array)

ruby
Copia codice

```ruby
[1, 2, 3, 4, 5].each do |n|
  puts n
end
```

8. Array e Hash

Gli **Array** e gli **Hash** sono tra le strutture dati più utilizzate in Ruby.

Array

ruby
Copia codice

```ruby
numeri = [1, 2, 3, 4, 5]
puts numeri[0]         # Accesso all'elemento con indice 0
numeri.push(6)         # Aggiunge 6 all'array
numeri.each { |n| puts n }
```

Hash

ruby
Copia codice

```ruby
persona = { "nome" => "Alice", "età" => 25 }
puts persona["nome"]     # Accesso alla chiave "nome"
persona["altezza"] = 1.70  # Aggiunge una nuova coppia chiave-valore
```

9. Metodi

I metodi definiscono blocchi di codice riutilizzabili.

ruby
Copia codice

```ruby
def saluto(nome)
  puts "Ciao, #{nome}!"
end

saluto("Alice")  # Output: Ciao, Alice!
```

- I metodi restituiscono l'ultimo valore calcolato, ma è anche possibile usare `return`.

10. Programmazione Orientata agli Oggetti

Ruby è un linguaggio orientato agli oggetti e supporta classi e metodi.

Definire una Classe

ruby
Copia codice

```ruby
class Persona
  def initialize(nome, età)
    @nome = nome
    @età = età
  end

  def presentati
    puts "Ciao, sono #{@nome} e ho #{@età} anni."
  end
```

```ruby
end

persona = Persona.new("Mario", 30)
persona.presentati   # Output: Ciao, sono Mario e ho 30 anni.
```

- **@nome** e **@età** sono variabili di istanza.
- Il metodo **initialize** è il costruttore, chiamato automaticamente quando si crea una nuova istanza.

Ereditarietà

ruby
Copia codice

```ruby
class Studente < Persona
  def studia
    puts "#{@nome} sta studiando."
  end
end

studente = Studente.new("Luigi", 20)
studente.presentati
studente.studia
```

11. Gestione delle Eccezioni

Ruby gestisce le eccezioni con `begin`, `rescue` e `ensure`.

ruby
Copia codice

```ruby
begin
  numero = 10 / 0
rescue ZeroDivisionError
  puts "Errore: divisione per zero!"
ensure
  puts "Questa parte viene sempre eseguita."
end
```

12. Moduli

I moduli sono usati per organizzare metodi e costanti. Consentono di raggruppare funzionalità comuni senza creare classi.

ruby

Copia codice
```ruby
module Saluti
  def saluta
    puts "Ciao!"
  end
end

class Persona
  include Saluti
end

persona = Persona.new
persona.saluta   # Output: Ciao!
```

13. Blocchi e Lambdas

I blocchi e le lambdas permettono di passare codice come argomento.

Blocco

ruby
Copia codice
```ruby
def ripeti_tre_volte
  yield
  yield
  yield
end

ripeti_tre_volte { puts "Ciao!" }
```

Lambda

ruby
Copia codice
```ruby
saluto = -> (nome) { puts "Ciao, #{nome}!" }
saluto.call("Marco")
```

14. Input e Output

Per ottenere input dall'utente, usa `gets`:

ruby
Copia codice
```ruby
puts "Inserisci il tuo nome:"
```

```ruby
nome = gets.chomp
puts "Ciao, #{nome}!"
```

15. Gemme (Gems) e Librerie

Ruby ha un sistema di pacchetti chiamato **RubyGems**, che permette di installare librerie esterne.

Esempio di installazione di una gemma:

bash
Copia codice
```bash
gem install nome_gemma
```

Per usarla nel tuo programma:

ruby
Copia codice
```ruby
require 'nome_gemma'
```

16. Esecuzione di Script Ruby

Salva il codice Ruby in un file con estensione `.rb` ed eseguilo con il comando:

bash
Copia codice
```bash
ruby nome_script.rb
```

17. Conclusione

Ruby è un linguaggio potente, versatile e facile da imparare, ideale per sviluppatori principianti e avanzati. Imparando queste basi e sperimentando con esercizi pratici, puoi cominciare a creare applicazioni interessanti e ad esplorare framework come **Ruby on Rails** per lo sviluppo web. Buona programmazione in Ruby!