Guida Ruby on Rails: il linguaggio Ruby - Parte 1

Benvenuto a questa nuova lezione di Ruby on Rails. Quest'oggi ci perderemo nei meandri del linguaggio Ruby imparandone la sintassi, i costrutti, le convenzioni e tutto ciò che ci renderà più facile scrivere codice per le nostre applicazioni.

Come prima cosa, devi aver ben chiaro che Ruby è un linguaggio Object-Oriented (nel caso non sapessi cos'è, informati in rete), o come diremo noi italiani orientato agli oggetti. Tuttavia Ruby è stato creato con l'intento di essere ancora più orientato agli oggetti di linguaggi quali Perl; questo significa che, tutto quello che manipoli attraverso questo linguaggio è un oggetto, così come lo sono anche i risultati di queste manipolazioni e praticamente qualsiasi altro elemento.

Gli oggetti possono essere creati attraverso un costruttore, ovvero uno specifico metodo associato a una classe. Il nome del costruttore standard è *new()*. Se volessimo creare un oggetto denominato *person*, potremmo scrivere:

Ovvero *luca* è un nuovo oggetto *person*. La distruzione degli stessi oggetti, invece, avviene in maniera meno consueta attraverso la Garbage Collection, meccanismo che avremo modo di approfondire più avanti.

Nomi e convenzioni

In Ruby il modo di scrivere i nomi è parte fondamentale della sintassi.

- Le variabili locali, i nomi di metodi così come i loro parametri dovrebbero iniziare tutti con la lettera minuscola o con l'underscore, per esempio: *product*, *product_id*, *_order*, *product33*.
- Le variabili d'istanza, ovvero quelle variabili associate a una classe di oggetti che rappresentano una proprietà dell'oggetto stesso, iniziano con il simbolo 'at', ovvero @, come @*total*, @*quantity*, @*pet_age*.
- Le variabili globali seguono gli stessi standard di quelle d'istanza con l'eccezione che, in sostituzione del simbolo @, viene utilizzato il simbolo dollaro $.
- Le classi, i moduli e le costanti devono iniziare con la lettera maiuscola e, contrariamente alle precedenti due categorie, invece di preferire l'underscore per separare i nomi composti da più parole, utilizzano nuovamente la maiuscola, per esempio: *Object*, *LineItem*, *OrderNumber*.
- I commenti sono preceduti dal simbolo #, per esempio: # => *Questo è un commento*.

Metodi

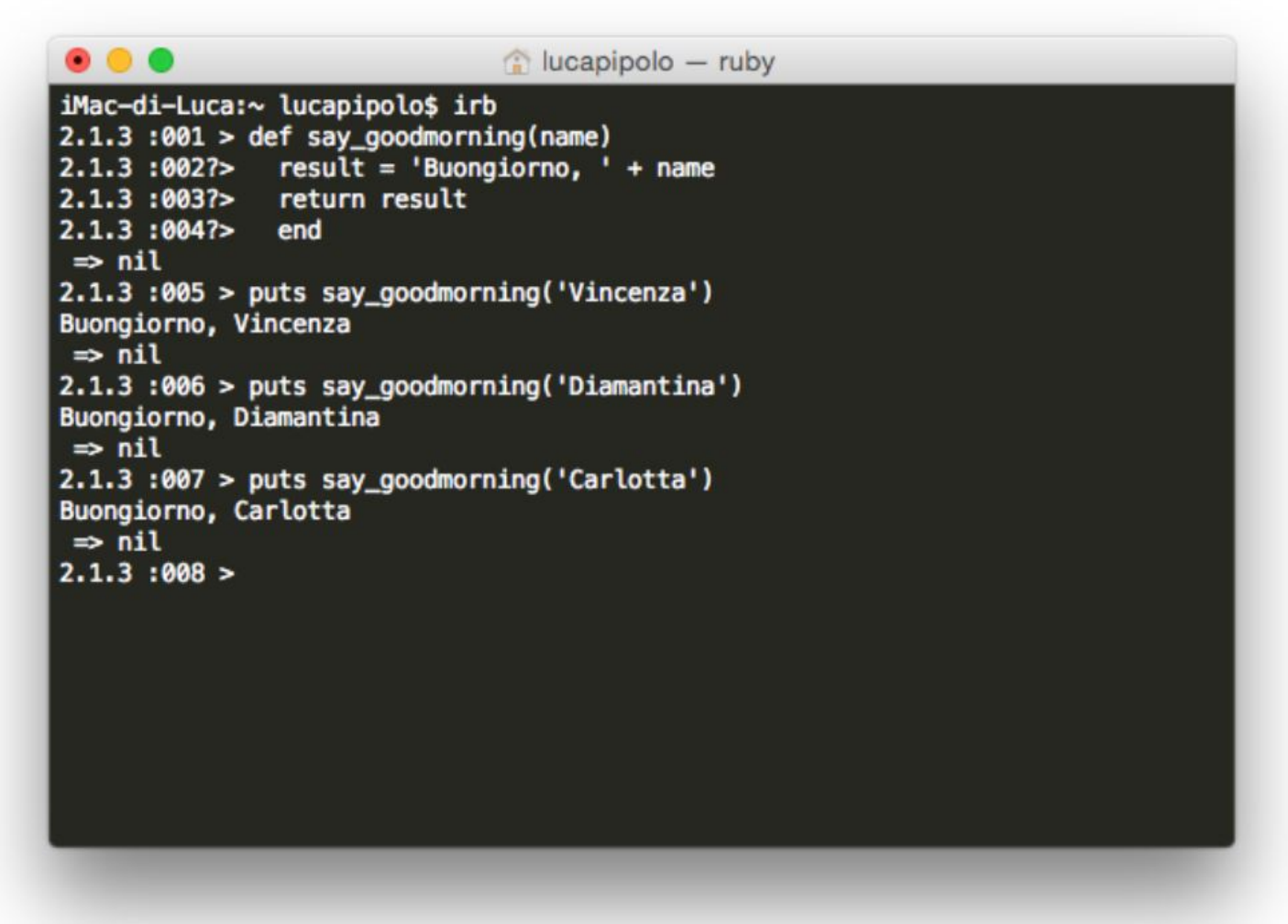
Come creiamo, quindi, un metodo? Proviamo con un semplice esempio pratico, scrivendo un

metodo che restituisca un buongiorno personalizzato.

Come puoi vedere, contrariamente ad altri linguaggi come JS, non sono previsti punto e virgola al termine della riga, così come non sono previste parentesi graffe che racchiudono il metodo che termina, in modo intuitivo, con la parola *end*.

Proviamo ora a richiamare il metodo appena scritto.

Il comando *puts* ci permette di stampare a video il risultato, in questo caso il risultato del metodo *say_goodmorning* con il parametro *name*. Puoi provare questi comandi semplicemente dal terminale del tuo sistema operativo: digitando *irb* per OS X o Linux oppure *fxri* per Windows, avrai accesso alla shell di Ruby. Il risultato sarà il seguente.



Tipi di dati

Poiché abbiamo detto che in Ruby praticamente tutto è considerabile come un oggetto, i tipi di dati godono di una particolare sintassi, specialmente per definire valori letterali.

Stringhe

Le stringhe vengono racchiuse tra virgolette (‘) o doppie virgolette (“). La differenza sta nel lavoro svolto dall’interprete Ruby a seconda dei due differenti casi. Se racchiuso tra virgolette singole, il contenuto di una stringa diviene il valore della stringa stessa; se, invece, vengono utilizzare le virgolette doppie, l’interprete agisce su di essa come nel caso delle sostituzioni.

Per esempio:

una volta richiamato con:

restituirà: *Buongiorno, Carlotta* poiché all’interno della stringa il valore compreso all’interno del costrutto *#{...}* sarà sostituito, in questo caso, con il parametro *name* e reso maiuscolo dal metodo *capitalize()*.

Array e hash

Gli array e gli hash rappresentano collezioni di oggetti accessibili mediante un indice. Negli array l’indice è indicato da un integer mentre negli hash l’indice può essere qualsiasi oggetto. Per creare un array digitiamo:

Come puoi notare un array accetta senza problemi differenti tipologie di dati, dalle stringhe agli integer passando per i numeri con virgola mobile. Per accedere a un elemento dell’array, in questo caso il primo elemento, digitiamo:

considerando che la numerazione degli indici inizia dal valore 0. Proviamo, infine, ad attribuire un valore ad un elemento:

ovvero attribuiamo all’elemento con indice 1 il valore *nil*, ovvero ‘vuoto’ dove anche *nil* rappresenta un oggetto, un oggetto vuoto. Nel caso in cui necessitassimo di un array di parole, potremmo preferire la seguente sintassi semplificata.

Per creare, invece, un hash dobbiamo specificare anche il valore dell'indice. La sintassi segue lo schema del seguente esempio:

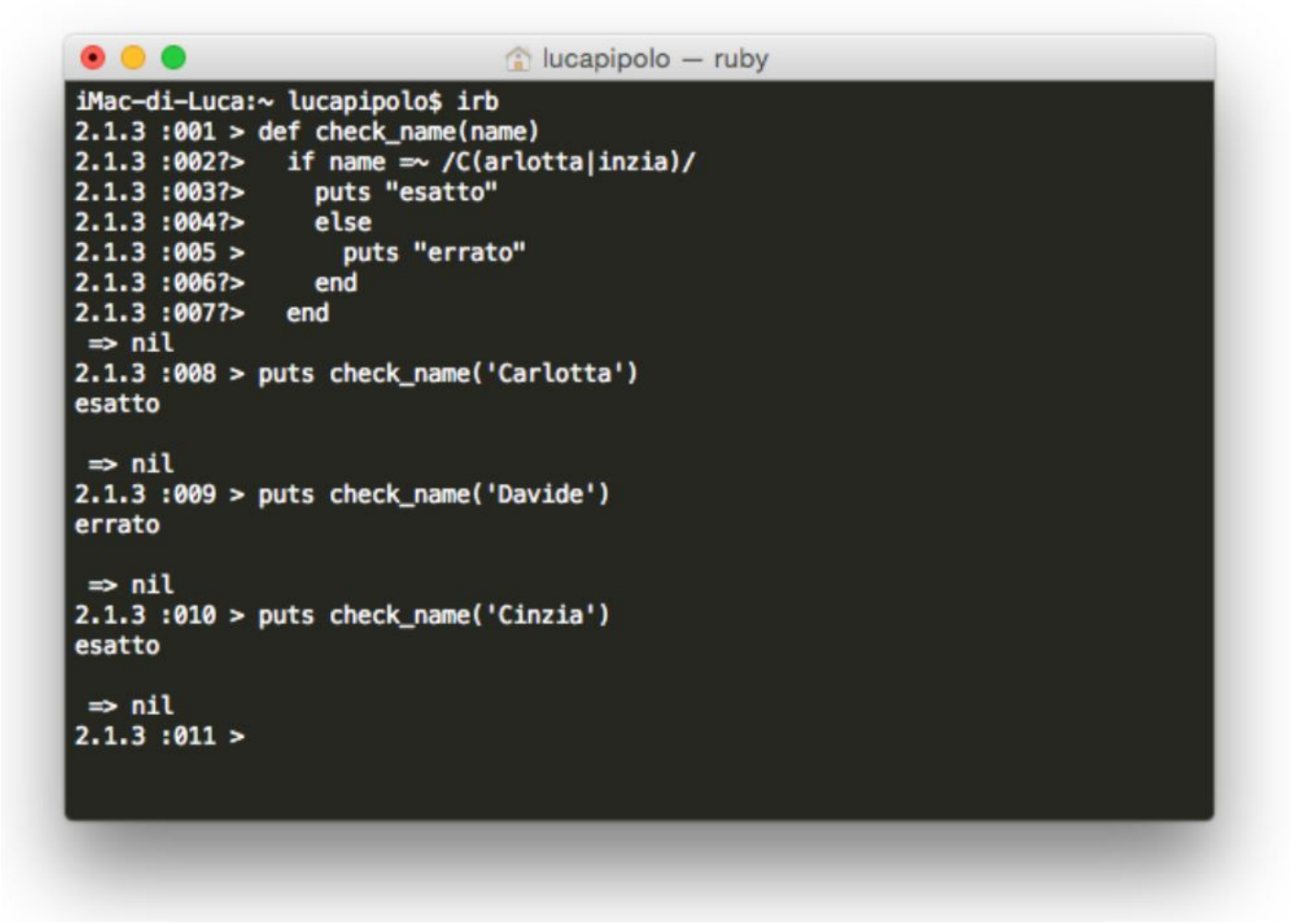
Il meccanismo di richiamo, invece, è lo stesso di un array:

e restituirà *female*.

Regular Expression

Le regular expression ti permettono di confrontare il contenuto di una stringa con un insieme specifico di caratteri. In Ruby esiste una duplice sintassi per far ciò: */pattern/* oppure *%r{pattern}*.

Il meccanismo delle regular expression è complesso e potrebbe addirittura richiedere una guida a sé stante, per questo motivo mi limiterò a farti un banale esempio e a rimandarti a due risorse online: questa documentazione e questo strumento. Con il seguente codice controlliamo se la variabile *name* assume il valore *Carlotta* o *Cinzia* paragonandolo a una regular expression e restituiamo una riposta.



Conclusioni

Nel corso del prossimo articolo continueremo la disquisizione sul linguaggio Ruby parlando di operatori logici, classi, moduli ed espressioni idiomatiche. Appuntamento alla prossima lezione, non mancare!

GUIDA RUBY ON RAILS: INDICE LEZIONI

9) Ruby on Rails: page layout
10) Guida Ruby on Rails: creiamo il carrello