# آموزش زبان برنامه نویسی Ruby

**محمدرضا حقیری**

**۱۸ بهمن ۱۳۹۳**

# فهرست مطالب

# ۱ مقدمه

زبان روبی ، یک زبان اسکریپتی، شی گرا، تابعی و مدرن است. این زبان توسط یوکیهیرو ماتسوموتو [۱] ایجاد شده و هم اکنون توسعه دهندگانی از سراسر جهان، آن را توسعه میدهند. این زبان، نحو [۲] ساده ای داشته، و می توان در عرض چند روز آن را به خوبی فرا گرفت. این زبان، برای توسعه برنامه های کاربردی دسکتاپ، وب و حتی نوشتن کتابخانه و سرویس های مختلف، کاربرد دارد. شما میتوانید به سادگی و با استفاده از این کتاب، این زبان را بیاموزید.

## ۱.۱ این کتاب برای چه کسانی است؟

به اختصار، باید گفت همه علاقمندان به یادگیری این زبان ساده و جذاب. جواب طولانی تر، اینست که این کتاب، برای کسانی است که واقعا میخواهند به سراغ روبی بروند، و چون روبی مرجع فارسی مطمئنی ندارد، پشیمان می شوند. پس اگر از هر دو دسته هستید، توصیه میکنم این کتاب را از دست ندهید.

## ۲.۱ آیا میتوانیم این کتاب را به اشتراک بگذاریم؟



## ۳.۱ چگونه این کتاب را بخوانیم؟

برای خواندن این کتاب، دو راه دارید، یا صرفا مطالعه روزنامه وار انجام دهید تا به آشنایی اجمالی با نحو زبان روبی برسید، یا اینکه تک تک نرم افزارها و مراحل گفته شده در کتاب را مرحله به مرحله انجام دهید، تا کاملا به روبی مسلط شوید.

[۱] Yukihiro Matsumoto
[۲] Syntax

# ۲ پایه ها و مقدمات یادگیری روبی

**در این فصل، چه فراخواهید گرفت؟**

- **نصب روبی در سیستم عامل**
- **آشنایی با محیط اینتراکتیو روبی**
- **ایجاد چند مثال ساده با روبی**
- **اجرای اسکریپت نوشته شده در روبی**

**این موارد، اولین قدم های شما در یادگیری زبان روبی هستند. در فصول بعدی، با ساختارها و کدهای بیشتری آشنا خواهید شد، و طبیعتا آن زمان است که میتوانید از روبی، استفاده کاربردی کنید.**

## ۱.۲ نصب روبی

**برای نصب روبی، شما ابتدا باید نام و نسخه سیستم عامل خود را بدانید. در این کتاب، فرض بر آنست که شما از گنو/لینوکس و توزیع اوبونتو استفاده میکنید.** [۳]

### ۱.۱.۲ وبسایت روبی

**شما میتوانید با مراجعه به وبسایت روبی** [۴] **نسخه مورد نظر روبی را دانلود نموده، سپس آن را کامپایل کنید.** [۵]

### ۲.۱.۲ نصب از مخازن

**برای استفاده از مخازن اوبونتو، کافیست تا یک پنجره ترمینال باز کنید، و سپس دستورات زیر را اجرا کنید :**

```
sudo apt-get install ruby
```

**همچنین اگر مفسر** [۶] **دیگری در نظر دارید، میتوانید در صورت موجود بودن در مخازن، آن را نصب کنید.**

---

[۳] **تفاوتی میانی استفاده از روبی در ویندوز و لینوکس نیست، طبیعتا میتوانید همین دستوراتی که در این کتابچه موجودند را به کار ببرید**

[۴] http://ruby-lang.org

[۵] **وقتی روبی را از سورس کامپایل میکنید، باید پیشاپیش هر چه پیش نیاز آن است را دستی نصب کنید. چنانچه دانش کافی در زمینه کامپایل ندارید، توصیه میکنم از مخازن استفاده کنید.**

[۶] Interpreter

## ۲.۲ استفاده از محیط تعاملی

**محیط تعاملی [۷] روبی، به شما این امکان را میدهد تا بدون نیاز به نوشتن کد در یک ویرایشگر متن، آن را مستقیما در ترمینال تایپ نموده، سپس نتیجه برنامه را در ترمینال ببینید. برای دسترسی به محیط تعاملی، یک پنجره ترمینال باز کرده و مانند دستور زیر عمل کنید :**

```
prp-e@prp-e ~ $ irb
```

**پس از اجرای دستور فوق، باید یک پرامپت [۸] به این شکل مشاهده کنید :**

```
irb(main):001:0>
```

**درون این محیط، میتوانید دستورات مورد نظر را اجرا و تست کنید.**

## ۳.۲ چند دستور در محیط تعاملی

**مثال معروفی به نام «سلام دنیا» در تمامی زبان های برنامه نویسی وجود دارد. بیایید این مثال، اولین مثال ما نیز در زبان روبی باشد. اکنون، پنجره ترمینال را باز کرده، و همانند شکل زیر دستور را درون محیط تعاملی روبی، وارد کنید :**

```
print "Hello, World\n"
```

**در صورتی که کد را درست وارد کرده باشید، باید نتیجه به این شکل گرفته باشید :**

```
irb(main):001:0> print "Hello, World\n"
Hello, World
=> nil
irb(main):002:0>
```

**اما نکته اینجاست، در زبان روبی، اصولا نیازی به دستوری برای ایجاد خط جدید نداریم. دستور مناسبتری وجود دارد که جایگزین print است و می توان به کمک آن، به خوبی داده ها را آنگونه که میخواهیم، چاپ کنیم. برای انجام کار عملی، دستورات زیر را در محیط تعاملی اجرا کنید :**

```
puts "Hello, World"
```

**اکنون باید نتیجه ای مشابه این دریافت کنید :**

```
irb(main):002:0> puts "Hello, World"
Hello, World
=> nil
irb(main):003:0>
```

**دستور puts جایگزین خوبی برای print است. علی الخصوص وقتی نتیجه مورد نظر، تنها یک خط باشد. اگرچه در آینده، از هر دو دستور استفاده خواهیم کرد.**

---

[۷] Interactive
[۸] Prompt

## ۴.۲ نوشتن اسکریپت در یک فایل

روبی یک زبان اسکریپتی است [۹] پس میتوانیم یک اسکریپت بنویسیم و با استفاده از مفسر، آن را اجرا کنیم. یا حتی می توانیم به آن یک دسترسی اجرایی بدهیم، سپس مستقیما اجرایش کنیم.

### ۱.۴.۲ انتخاب ویرایشگر

شما برای آن که بتوانید کد خود را درون یک فایل روبی بنویسید، طبیعتا به یک ویرایشگر متن [۱۰] نیاز دارید. ویرایشگر های متن که همراه سیستم عامل ها نصب می شوند، گزینه های خوبی هستند. اگر میخواهید بدانید من از چه استفاده میکنم، از **nano** به عنوان ویرایگشر تحت ترمینال، و از **gedit** به عنوان ویرایشگر گرافیکی استفاده میکنم. [۱۱]

### ۲.۴.۲ پسوند فایل های روبی

سورس کدهای روبی، عموما با پسوند های .rb و یا .ruby ساخته می شوند. توجه کنید که کامپیوتر به پسوند ها کاری ندارد، و این پسوند ها صرفا برای این ساخته شده اند که انسان بتواند با استفاده از آنها، نوع فایل را تشخیص دهد.

### ۳.۴.۲ نوشتن و اجرای اسکریپت از طریق مفسر

مفسر روبی، از طریق خط فرمان با نام **ruby** در دسترس است، پس میتوانیم به سادگی یک قطعه کد بنویسیم، آن را روی دسکتاپ قرار داده و سپس دستورات مورد نظر را درونش وارد کنیم. اکنون، یک فایل خالی روی دسکتاپ اجرا کنید و این دستور را درونش بنویسید :

```
puts "Hello, World"
```

با فرض آن که این برنامه، روی دسکتاپ شما با نام **HelloWorld.rb** ذخیره شده است، آن را اینگونه اجرا میکنیم :

```
prp-e@prp-e ~ $ ruby ~/Desktop/HelloWorld.rb
```

سپس باید روی ترمینال چنین نتیجه ای را مشاهده کنیم :

```
prp-e@prp-e ~ $ ruby ~/Desktop/HelloWorld.rb
Hello, World
prp-e@prp-e ~ $
```

این اسکریپت، از طریق مفسر روبی، اجرا شد. در قسمت بعدی، از طریق اجازه ها [۱۲] اسکریپت را اجرا خواهیم نمود.

---

[۹] زبان های اسکریپتی، خط به خط اجرا می شوند. برخلاف زبانهای کامپایل شده، که ابتدا کل برنامه به یک زبان قابل فهم برای سیستم عامل یا ماشین ترجمه می شود، سپس اجرا میگردد.

[۱۰] Text Editor

[۱۱] در نظر داشته باشید چنانچه میزکار **MATE** را نصب دارید، ادیتور **gedit** با نام **pluma** در دسترس است.

[۱۲] Permissions

### ۴.۴.۲ اجرای اسکریپت با استفاده از اجازه ها

**این بار، از خاصیت اجازه دسترسی در یونیکس، استفاده خواهیم کرد و سپس اسکریپت قسمت قبل را مستقلا اجرا خواهیم کرد. اکنون، نام HelloWorld.rb را به HelloWorld تغییر دهید و سپس با مفسر اجرایش کنید :**

```
prp-e@prp-e ~ $ ruby ~/Desktop/HelloWorld
Hello, World
prp-e@prp-e ~ $
```

**نتیجه تفاوتی نکرد، نباید هم بکند. مفسر دستورات خودش را تشخیص میدهد. حالا میخواهیم مفسر را درون خود برنامه جای دهیم!**
**کد برنامه را به این شکل ویرایش کنید :**

```
#!/usr/bin/ruby

puts "Hello, World"
```

**اکنون، مفسر را به اسکریپت خود شناسانده ایم. اکنون وقت آن است که اجازه اجرا به سورس کد خود بدهیم. اکنون فقط کافیست تا این دستور را در ترمینال اجرا کنید :**

```
prp-e@prp-e ~ $ chmod +x Desktop/HelloWorld
```

**سپس میتوان این کد را مستقیما اجرا کرد. برای اجرای مستقیم کد کافیست دستور زیر را اجرا کنید :**

```
prp-e@prp-e ~ $ Desktop/HelloWorld
```

**نتیجه دریافتی روی ترمینال باید به این شکل باشد :**

```
prp-e@prp-e ~ $ Desktop/HelloWorld
Hello, World
prp-e@prp-e ~ $
```

## ۵.۲ جمع بندی

**در اینجا، باید یک جمع بندی نهایی از مباحثی که در فصل مطرح شدند، داشته باشیم. اول اینکه، فراگیری روبی بسیار آسان است، ولی باید بدانید کجا و چگونه، چه دستوری را به کار ببرید. دوم، باید بدانید که چه موقع محیط تعاملی، و چه موقع محیط ویرایشگر را برای نوشتن برنامه انتخاب کنید. در نهایت هم همیشه حواستان به چگونگی نوشتن کد در ویرایشگر باشد، چرا که کوچک ترین اشتباهی، میتواند مانع از اجرای درست برنامه شما باشد.**

## ۶.۲ تمرینات

**همانگونه که در فروم اوبونتو قول داده بودم [۱۳] ، در پایان هر فصل، تعدادی تمرین خواهم گنجاند. با انجام این تمرینات، به سادگی میتوانید دانش و توانایی خود در زمینه روبی را بسنجید.**

۱. **برنامه ای بنویسید که نام خودتان را چاپ کند**

۲. **برنامه ای بنویسید که نام دانش آموزان یک کلاس ۱۵ نفره را چاپ کند، و آن را در قالب یک اسکریپت با اجازه اجرا، اجرا کنید**

[۱۳] **انجمن های فارسی اوبونتو، در آدرس forum.ubuntu.ir در دسترس هستند. قول این کتاب را در تاپیکی با مضمون آموزش روبی، در آن انجمن داده بودم.**

# ۳ انواع داده ها و متغیرها، ثابت ها و عملگرها

در هر زبان برنامه نویسی، چندین نوع داده مختلف وجود دارد. استفاده از انواع مختلف داده، کار شخص برنامه نویس و حتی خود برنامه را آسان میکند. چرا که اینگونه، ماشین میفهمد که باید چگونه عملیاتی را روی داده انجام دهد. در این فصل، قصد داریم چندین نوع داده و همچنین تعریف آنها به عنوان متغیر را در روبی بررسی کنیم.

## ۱.۳ در این فصل چه فرا خواهید گرفت؟

- شناسایی انواع مختلف داده ها
- تعریف انواع مختلف داده در متغیر
- تبدیل انواع داده

## ۲.۳ داده های عددی

به طور کل، این نوع داده ها، اولین داده هایی هستند که در آموزش هر زبان برنامه نویسی، با آنها کار خواهیم داشت. چرا که گاهی با مقایسه و یا کم و زیاد کردن اعداد، کارهای خاصی در روند یک برنامه انجام میشود که میتواند به سرعت اجرای برنامه، کمک کند.

### ۱.۲.۳ اعداد صحیح

اعداد صحیح [۱۴] ، نوعی از اعداد هستند که جزء اعشاری ندارند. دامنه این اعداد در طبیعت از منفی بی نهایت تا مثبت بی نهایت ادامه دارد، با این تفاوت از اعداد حقیقی، که این دسته از اعداد، گسسته هستند. برای مثال در فاصله ۲ تا ۳ در مجموعه اعداد صحیح، هیچ عدد دیگری قرار نمیگیرد، در حالی که در سایر گونه های اعداد، اینگونه نیست. اگرچه اینجا بحث، بحث ریاضیات نیست، ولی بهتر بود توضیح ساده ریاضی اعداد صحیح را در اینجا ذکر میکردم. حالا نوبت آن است تا در محیط تعاملی روبی، چند عدد صحیح را تایپ کنید. برای تایپ کردن کافیست با دستور **irb** وارد محیط تعاملی شده و اعداد زیر را تایپ کنید :

```
2
3
7
```

باید چنین خروجی هم دریافت کنید :

```
irb(main):001:0> 2
=> 2
irb(main):002:0> 3
=> 3
```

[۱۴] Integer

```
irb(main):003:0> 7
=> 7
irb(main):004:0>
```

**البته میتوانید هر کدام از اعداد را به دلخواه، منفی کنید. مهم اینست که در این قسمت، شما با نوع داده صحیح آشنا شدید.**

### ۲.۲.۳ اعداد ممیز شناور

**اعداد ممیز شناور [۱۵] ، نوعی از اعداد هستند که بخش اعشاری دارند. در واقع، این اعداد در برنامه نویسی، همان اعداد حقیقی هستند. اگرچه ، اعداد گویا، اصم و گنگ را نیز میتوان انواع دیگری از داده حساب نمود، اما در برنامه نویسی، همه اعداد غیر صحیح در مبنای ده را به نوعی میتوان ممیز شناور به حساب آورد.**

**برای تعریف این اعداد، کافیست به سادگی یک ممیز و یک صفر به اعداد معرفی شده در قسمت بالاتر، اضافه نمایید :**

```
2.0
3.0
7.0
```

**و شاهد چنین خروجی باشید :**

```
2irb(main):005:0> 2.0
=> 2.0
irb(main):006:0> 3.0
=> 3.0
irb(main):007:0> 7.0
=> 7.0
irb(main):008:0>
```

**همانگونه که ملاحظه میفرمایید، مفسر ممیز را لحاظ کرده است. پس طبیعتا بین ۲.۰ و ۲ تفاوت قائل می شود. اکنون با دو نوع اصلی و عمده داده های عددی آشنا شده اید، بیاییم با دو نوع داده عددی دیگر نیز آشنا شویم.**

### ۳.۲.۳ اعداد اوکتال

**اعداد اوکتال [۱۶] اعدادی هستند که از مبنای ده، به مبنای هشت برده شده اند. روبی، زبانی است که قابلیت تشخیص مبنای هشت و شانزده را دارد. برای این که این اعداد را به محیط تعاملی روبی بدهیم، کافیست با دستور ورود آنها، و رنجی از اعداد که پشتیبانی میکنند، آشنا باشیم. [۱۷] اعداد اوکتال در روبی، با صفر آغاز می شوند [۱۸] پس اکنون کافیست محیط تعاملی روبی را باز کرده و اعداد**

[۱۵] Floating Point

[۱۶] Octal

[۱۷] اعداد اوکتال یا مبنای هشت، اعدادی هستند که از ارقام صفر تا هفت تشکیل شده اند

[۱۸] این یک قرار داد است که این اعداد، با صفر آغاز شوند.

زیر را وارد کنید :

```
012
02
013457
```

خب اکنون باید نتیجه زیر را شاهد باشید :

```
irb(main):001:0> 012
=> 10
irb(main):002:0> 02
=> 2
irb(main):003:0> 013457
=> 5935
irb(main):004:0>
```

احتمالا بعد از دیدن این نتایج تعجب کرده اید، و پیش خودتان حدس هایی زده اید. باید بگویم درست حدس زده اید! روبی این اعداد را به مبنای ده برده و برای شما، نشان داده است. روبی نوع دیگری از داده های در مبنای غیر ده نیز دارد که در قسمت بعدی با آن آشنا خواهید شد.

### ۴.۲.۳ اعداد هگزادسیمال

اعداد هگزادسیمال [۱۹] در مبنای ۱۶ هستند. این اعداد، در علوم کامپیوتر بسیار پر استفاده اند و در دروسی مانند مبانی برنامه نویسی و ساختمان داده، بسیار به آنها پرداخته می شود. [۲۰] برای تعریف این اعداد در روبی، عدد مورد نظر را پس از صفر و x در محیط تعاملی وارد میکنیم. برای امتحان، اعداد زیر را در محیط تعاملی وارد کنید :

```
0x12
0x2
0x13457
```

سپس باید شاهد چنین نتیجه ای باشید :

```
irb(main):004:0> 0x12
=> 18
irb(main):005:0> 0x2
=> 2
irb(main):006:0> 0x13457
=> 78935
irb(main):007:0>
```

[۱۹] Hexadecimal

[۲۰] در اینجا، ارقام ۰ تا ۹ را به علاوه ارقام A ، B ، C ، D و E داریم که نمایانگر ۱۰ تا ۱۵ هستند.

این بار هم شاهد این بودید که مفسر روبی، اعداد را از مبنای شانزده به ده برده و به شما نمایش داده است. در ادامه، با انواع دیگری از متغیر ها آشنا خواهید شد، که دیگر عددی نیستند.

### ۳.۳ متغیرهای بولین

متغیر های بولین [۲۱] ، متغیر هایی هستند که تنها دو مقدار درست (**true**) و یا غلط (**false**) دریافت میکنند.

### ۴.۳ متغیر های رشته ای

یک بار دیگر، برنامه زیر را چک کنید :

```
puts "Hello, World"
```

اگر دقت کنید، خواهید دید که عبارت Hello, World را درون علامت نقل قول قرار داده ایم. این نوع داده را، رشته [۲۲] میگویند. هر رشته، از بهم پیوستن چندین کاراکتر [۲۳] تشکیل می شود. یک رشته، میتواند شامل چند خط باشد، یا شامل یک کلمه باشد. در کل، قانون و قاعده خاصی برای تعریف مقادیر درون یک متغیر رشته ای، وجود ندارد.

### ۵.۳ آرایه ها و هش ها

آرایه ها [۲۴] و هش ها [۲۵] دو نوع متغیر مجموعه ای هستند. آرایه، مجموعه ای از متغیر های یک نوع خاص است، و هش مانند یک دیکشنری عمل می کند. با عملکرد این دو نوع متغیر، در آینده آشنا خواهید شد.

### ۶.۳ تعریف متغیرها

بر خلاف زبان هایی مانند C و C++ که ابتدا نوع، سپس نام و سپس مقدار متغیر تعریف میگردد، در روبی تنها نام متغیر کافیست تا به آن مقدار داده شود. دستورات زیر، چند متغیر عددی و رشته ای را در روبی تعریف میکنند :

```
name = "Muhammadreza"
a = "2"
b = 3
```

متغیر های **name** و **a** از نوع رشته ای هستند (به علامت نقل قول توجه کنید) ، و متغیر **b** از نوع عدد صحیح. نکته اینجاست که متغیرها، امکان تبدیل به یک دیگر را نیز دارند.

---

[۲۱] Boolean
[۲۲] String
[۲۳] Character
[۲۴] Array
[۲۵] Hash

### ۱.۶.۳ تبدیل نوع متغیر ها

متغیر a از نوع رشته ای، و متغیر b از نوع عدد صحیح است. برای تبدیل a به عدد صحیح، و برای تبدیل b به ممیز شناور، از دستورات زیر استفاده میکنیم :

```
a.to_i
b.to_f
```

و برای ذخیره نتایج، کافیست متغیر جدیدی تعریف کرده، و دستورات را درونش بریزید. اگرچه، تغییر نوع های دیگری نیز وجود دارد که در ادامه به آنها خواهیم پرداخت.

## ۷.۳ عملگرها

عملگرها، در ریاضی و همچنین برنامه نویسی از اهمیت ویژه ای برخوردارند. شما اگر بخواهید روی متغیرها عملیاتی انجام دهید، قطعا نیاز خواهید داشت تا از عملگرها استفاده کنید. در روبی، چهار عمل اصلی جمع، تفریق، ضرب و تقسیم توسط چهار عملگر / ,* ,- ,+ انجام می شوند. همچنین، برای توان رسانی نیز از عملگر ** استفاده می شود. برای آزمایش عملگرها، کافیست دستورات زیر را در محیط تعاملی وارد کنید :

```
a + b
a - b
a * b
a / b
a ** b
```

و سپس باید شاهد چنین خروجی باشید :

```
irb(main):003:0> a + b
=> 5.0
irb(main):004:0> a - b
=> -1.0
irb(main):005:0> a * b
=> 6.0
irb(main):006:0> a / b
=> 0.6666666666666666
irb(main):007:0> a ** b
=> 8.0
irb(main):008:0>
```

نکته ای که متوجه آن شده اید، اینست که اگر یکی از متغیرها از نوع ممیزشناور باشد، خود مفسر تبدیل نوع را انجام میدهد.

### ۸.۳ ثابت ها

**ثابت ها نیز همانند متغیرها، میتوانند از نوع عدد صحیح، ممیزشناور، متن و ... باشند. با این تفاوت که متغیر ها را میتواند مقدارشان را تغییر داد ولی ثابت ها را نه. تفاوت تعریف ثابت با متغیر، این است که نام ثابت، همیشه با حرف بزرگ شروع می شود، برای آزمایش، مقدار زیر را به عنوان یک ثابت به محیط تعاملی بدهید :**

```
CONST_INT = 2
CONST_FLOAT = 3.0
CONST_STRING = "Hello, World"
```

**اکنون سعی کنید مقدار یکی از این ثابت ها را تغییر دهید، طبیعتا با مشکل روبرو خواهید شد. چرا که از دید مفسر، این ها یک بار برای همیشه تعریف شده اند و نمیتوان آنها را تغییر داد.**

### ۹.۳ جمع بندی

**در این فصل، آموختید که هر نوع داده ای کجا و چگونه به کار میرود. مهم است که بدانید کجا و چطور، از چه نوع داده ای استفاده کنید. مثلا موقعی که خرده داشتن نتیجه، برایتان مهم نیست، منطقی تر است حتی اگر ورودی ممیز شناور باشد، شما خروجی را به عدد صحیح تبدیل کنید. اما اگر جایی قرار است مثلا تا n رقم اعشار محاسبه شود، منطقی تر آن است که حتی ورودی های صحیح، در خروجی به صورت ممیز شناور نمایش داده شوند. برخی از انواع داده نیز معرفی شدند، اما مثالی از آنها زده نشد. این نوع داده ها، هر کدام خودشان یک فصل جداگانه میطلبیدند، پس نگران نباشید، این ها را در ادامه فراخواهید گرفت.**

### ۱۰.۳ تمرینات

۱. **برنامه ای بنویسید که اعداد یک تا ده را به توان ۲ رسانده و در خروجی نشان دهد**

۲. **برنامه ای بنویسید که نشان دهد شما چند روز عمر کرده اید، و خروجی را به صورت عدد صحیح نشان دهد**

۳. **برنامه ای بنویسید که از یک ثابت PI و یک متغیر radius استفاده کرده، سپس مساحت دایره ای را حساب کند**

# ۴ آرایه ها و هش ها

در فصل پیش، اشاره کوچکی به این دو نوع داده ای شد. در واقع، این دو نوع داده ای، در کنار چندتایی ها [۲۶] نوع داده هایی هستند که با مجموعه ها سرو کار دارند. در این فصل، این نوع داده ها را بررسی کرده، سپس متدهایی که در روبی برای آنها در نظر گرفته است را بررسی میکنیم.

## ۱.۴ در این فصل چه فراخواهید گرفت؟

- تعریف یک آرایه
- تعریف یک هش
- استفاده از چند تایی ها
- استفاده از اعضای یک آرایه به عنوان متغیر
- استفاده از متدهای در نظر گرفته شده، برای مرتب کردن، کوتاه کردن، برعکس کردن و جایگزین کردن یک عضو در آرایه و هش.

## ۲.۴ آرایه ها

آرایه ها، مجموعه ای از چندین مقدار می باشند. این مقادیر، میتوانند عدد صحیح، ممیز شناور و رشته و کاراکتر باشند. گاهی نیز برای مرتب شدن متغیرهای مورد نیاز، یک سری از مقادیر مورد نیاز را داخل یک آرایه قرار میدهیم، و برای استفاده از آن مقدار، خانه ای از آرایه که آن مقدار درونش قرار داده شده را فراخوانی میکنیم. برای تعریف یک آرایه، دستور زیر را در محیط تعاملی وارد کنید :

```
a = [1, 2, 3, 4, 5]
```

دستور فوق، میگوید یک آرایه که اعداد ۱ تا ۵ درونش قرار دارند ایجاد شود. تا اینجا، آرایه را ایجاد کردیم. حال نوبت متدهاست که یکی یکی، روی آرایه ایجاد شده اعمال میکنیم تا کار با آرایه را فرا بگیریم. [۲۷]

### ۱.۲.۴ متد length

واژه length در زبان انگلیسی، معنای «طول» میدهد. با اجرای این متد، طول آرایه به دست می آید. برای اجرای دستور، در محیط تعاملی به این شکل وارد کنید :

```
a.length
```

نتیجه دریافتی باید به شکل زیر باشد :

---

[۲۶] Tuple

[۲۷] در اینجا همه متدها آورده نشده است، و دلیل این امر، آنست که اگر میخواستیم تک تک متدهای آرایه ها را اینجا بیاوریم، مطلب بسیار طولانی و حوصله سر بر می شود.

```
irb(main):002:0> a.length
=> 5
irb(main):003:0>
```

**مفسر اینجا به شما می گوید که پنج عضو درون این آرایه قرار دارند.**

### ۲.۲.۴ متد reverse

**این متد، آرایه را برعکس میکند. برای آنکه بدانید چگونه کار میکند، دستور زیر را درون محیط تعاملی وارد کنید :**

```
a.reverse
```

**نتیجه باید به شکل زیر باشد :**

```
irb(main):003:0> a.reverse
=> [5, 4, 3, 2, 1]
irb(main):004:0>
```

**استفاده از این متد هم زمانی که یک آرایه که از کوچک به بزرگ (یا برعکس) مرتب شده باشد را نیاز داریم، کار آمد است.**

### ۳.۲.۴ مرتب کردن آرایه

**اگر به زبانهایی مانند C آشنا باشید، میدانید که برای مرتب کردن آرایه ها، نیاز دارید تا از روش هایی مانند حلقه های تکرار و ... استفاده کنید. اما روبی، اینگونه نیست.**
**قبل از اینکه بخواهیم آرایه را مرتب کنیم، دقت کنید که آرایه داده شده در قسمت های قبل، مرتب بود. حالا آن را به هم میریزیم :**

```
a = [1, 3, 2, 4, 5]
```

**حالا تنها کافیست دستور زیر را اجرا کنید، تا آرایه مرتب گردد :**

```
a.sort
```

**نتیجه دریافتی باید به شکل زیر باشد :**

```
irb(main):005:0> a.sort
=> [1, 2, 3, 4, 5]
irb(main):006:0>
```

**اکنون، با دستور زیر، آرایه را از بزرگ به کوچک مرتب میکنیم :**

```
a.sort.reverse
```

**و نتیجه ما چنین خواهد بود :**

```
irb(main):007:0> a.sort.reverse
=> [5, 4, 3, 2, 1]
irb(main):008:0>
```

**تا اینجا، کار با آرایه را یاد گرفتید و با چندین متد آشنا شدید. زین پس، به سراغ این می رویم که بعضی خواص آرایه را بررسی کنیم.**

### ۴.۲.۴ نمایه اعضای آرایه

**نمایه [۲۸] به عددی گفته می شود که نماینده اعضای آرایه است. نمایه یا ایندکس، از صفر شروع می شود (یعنی نخستین عضو نمایه صفر میگیرد) و تا طول آرایه، ادامه می یابد. در واقع برای a ، که در بالا تعریف کردیم، اعداد ۰ تا ۴ هستند که نمایه های اعضا به شمار می آیند. برای امتحان کردن این موضوع، دستور زیر را اجرا کنید :**

```
a[0]
```

**و نتیجه باید چنین باشد :**

```
irb(main):009:0> a[0]
=> 1
irb(main):010:0>
```

**و اگر این یکی دستور را اجرا کنید :**

```
a[4]
```

**نتیجه باید این چنین باشد :**

```
irb(main):010:0> a[4]
=> 5
irb(main):011:0>
```

**همچنین، اگر از اعداد منفی استفاده کنید، از آخر به اول مقادیر برگردانده میشوند، یعنی عضو -۱ در این آرایه، همان عضو ۴ است.**

[۲۸] Index

### ۵.۲.۴ اضافه کردن اعضا به آرایه

برای اضافه کردن اعضا به آرایه، چندین راه حل وجود دارد. ساده ترین راه آن، این است که اعضایی که لازم داریم را در قالب یک آرایه جدید، به آرایه پیشین اضافه نماییم. به این شکل:

```
a += [6, 7, 8]
```

و اکنون، طول آرایه نیز عوض شده است. شما تا عدد ۸ را به آرایه افزوده اید، و طبیعتا اگر **length** را اجرا کنید، نتیجه عدد ۸ است. همچنین، میتوانید با استفاده از اندیس ( = نمایه) مناسب، نیز عضو مورد نظر را اضافه کنید. ما اکنون میخواهیم عدد ۹ را نیز به آرایه خود اضافه کنیم و چون میدانیم اندیس گذاری آرایه، از صفر شروع می شود، این دستور را اجرا میکنیم :

```
a[8] = 9
```

اکنون نوبت چاپ آرایه است. برای چاپ آرایه هم دستور زیر را اجرا میکنیم :

```
puts a
```

و چنین نتیجه ای دریافت خواهیم کرد :

```
irb(main):014:0> puts a
1
2
3
4
5
6
7
8
9
=> nil
irb(main):015:0>
```

چون **puts** برای چاپ خط جدید نیز به کار میرود، اینجا میگوید اولین عضو را چاپ کن، برو خط بعدی، دومی را چاپ کن، برو خط بعدی و الی آخر. به این شکل، هر کدام از اعضا درون یک خط جداگانه چاپ می شوند. برای چاپ یک عضو مشخص هم، تنها کافیست تا از اندیس مناسب استفاده کنید.

### ۶.۲.۴ جایگزین کردن اعضای آرایه

برای جایگزین کردن یک عضو، باید از اندیس آن استفاده کنید. مثلا میخواهیم عدد ۱ را با صفر جایگزین کنیم، میدانیم که ۱، اندیس صفر گرفته است پس این دستور را اجرا می کنیم :

```
a[0] = 0
```

اینگونه، عضو مورد نظر ما، با یک مقدار جدید جایگزین می‌شود . راه حل دیگری نیز وجود دارد که بتوانیم به صورت دسته ای، مقادیر را جایگزین کنیم. برای جایگزین کردن دسته ای، کافیست این گونه عمل کنیم :

```
a[0, 3] = 0, 1, 2
```

و با این دستور، سه عضو اول آرایه، تغییر میکنند. حالا دوباره دستور **puts** را اجرا میکنیم :

```
irb(main):019:0> puts a
0
1
2
4
5
6
7
8
9
=> nil
irb(main):020:0>
```

اکنون دیدید که مواردی که میخواستیم، به خوبی اجرا شدند. حالا نوبت آن است که نکاتی در مورد آرایه ها بدانید، و بعد برویم سراغ چندتایی ها و بعد هم هش ها.

### ۷.۲.۴ نکات مهم

- یادتان باشد که آرایه ها، تغییر پذیرند، میتوانید هر گاه که خواستید، اعضای آن را تغییر دهید.
- طول آرایه ها نیز تغییر پذیر است، یعنی میتوانید اعضایی که نیاز است را به آرایه اضافه کنید یا از آن کم کنید
- متدهایی که برای آرایه ها در نظر گرفته شده، برای رشته ها هم قابل استفاده است.

### ۳.۴ چندتایی ها

اگر پایتون کار کرده باشید، با یک شکل عمومی از چندتایی ها آشنا هستید. اما در روبی، چند تایی به آن شکل نداریم. در روبی، چند تایی ها مجموعه ای از نام ها و یک آرایه را دریافت میکنند، سپس تعدادی از اعضای آرایه (یا همه آن ها) را، به متغیرها نسبت میدهد. برای مثال، از آرایه اولیه مان یک چند تایی درست میکنیم :

```
(b, c, d, e, f) = a
```

و در صورت چاپ کردن هر کدام از متغیرهای تعریف شده، عضو نظیر در آرایه چاپ میگردد. در کل، همین قدر در مورد چندتایی ها بدانید، کافیست. اکنون بیایید به سراغ هش برویم.

### ۴.۴ هش ها

هش ها، رفتاری مانند دیکشنری دارند. این نوع داده ها، یک کلید [۲۹] و یک مقدار [۳۰] دریافت میکنند. در واقع، به ازای هر کلیدی، یک خروجی یکتا تولید میکنند. کاربرد هش ها، در ایجاد لیست ها، لغت نامه ها و ... است. هش ها هم همانند آرایه ها، یک سری متد خاص دارند که در این قسمت با آنها آشنا می شوید.

#### ۱.۴.۴ ایجاد یک هش جدید

ساده ترین نوع هش، آنست که یک سری لغت را به زبان های مختلف بخواهیم ترجمه کنیم. اینجا میخواهیم واژه «سلام» را به زبان های انگلیسی، اسپانیایی، اسپرانتو و پرتغالی درون یک هش قرار دهیم :

```
salam = { :en => "Hello" , :es => "Hola", :eo => "Saluton", :pt => "Ola"}
```

بدین گونه، ما یک هش تازه ایجاد کرده ایم، و لغات مورد نظر را درونش قرار داده ایم. اکنون، برای نمایش لغت انگلیسی، کافیست کد زیر را اجرا کنیم :

```
salam[:en]
```

و سپس خروجی مناسبی دریافت خواهید کرد.

#### ۲.۴.۴ ویرایش هش ها

هش ها هم تغییر پذیرند، منتها با تفاوتی بزرگ، که نمیتوان آنها را مانند آرایه ها ویرایش کرد. فرض کنیم شما نیاز دارید تا واژه را به زبان ایتالیایی هم اضافه کنید. اکنون باید کلید ایتالیایی را با مقدار مناسب، استفاده کنید، به این شکل :

```
salam[:it] = "Ciao"
```

و مقدار مورد نظر شما، به هش اضافه می شود.

---

[۲۹] Key
[۳۰] Value

### ۳.۴.۴ مرتب کردن هش

**هش ها نیز مرتب می شوند. منتها، بر اساس کلید ها. اگر کلید ها عددی باشند، از کوچک به بزرگ، و اگر حرف باشند (همانند هش salam که ما ساختیم) به ترتیب حروف الفبا، مرتب می شوند. برای مرتب کردن هش همان دستور sort به کار می رود. دقت کنید که البته، هش ها بعد از مرتب شدن به آرایه تبدیل می شوند.**

### ۴.۴.۴ چاپ کردن اعضای یک هش

**چاپ کردن اعضای هش، بیشتر به ساختار حلقه های تکرار شبیه است، ولی باز با این حال در این قسمت اشاره کوچکی به آن میکنیم، برای چاپ کردن اعضای هش، از متد each استفاده میکنیم :**

```
salam.each{|x, y| puts x + ": " y}
```

**و نتیجه خروجی باید این چنین باشد :**

```
irb(main):025:0> salam.each{|x, y| puts x + ": " + y}
en: Hello
es: Hola
eo: Saluton
it: Ciao
=> {"en"=>"Hello", "es"=>"Hola", "eo"=>"Saluton", "it"=>"Ciao"}
irb(main):026:0>
```

**البته توجه کنید که کلید ها به این شکل تغییر کردند :**

```
{"en"=>"Hello", "es"=>"Hola", "eo"=>"Saluton", "it"=>"Ciao"}
```

**و به رشته تبدیل شدند، تا در چاپ کردن آنها مشکلی پیش نیاید.**

## ۵.۴ جمع بندی

**در این فصل، با آرایه و هش آشنا شدید. همچنین یاد گرفتید که مفهوم چندتایی در روبی کاملا با پایتون متفاوت است، و بنابراین برای یادگیری روبی، قطعا نیاز است تا در آموخته های پیشین خود، تجدید نظر کنید. همچنین یاد گرفتید که هش ها کجا کاربرد دارند (و بعد ها نیز بیشتر با کاربرد آنها آشنا خواهید شد) و همچنین در نکاتی که برای آرایه ها بیان شد، اشاره شد که رشته ها خواص مشابه آرایه ها دارند. [۳۱] در نهایت هم با مرتب کردن، و سایر متد های پرکاربرد درون روبی آشنا شدید.**

[۳۱] فراموش نکنید که یک رشته، خود آرایه ای از کاراکتر هاست.

## ۶.۴ تمرینات

۱. برنامه ای بنویسید که شامل یک آرایه از حروف انگلیسی باشد، سپس توسط یک چندتایی، یک کلمه ایجاد کنید.

۲. برنامه ای بنویسید که لغات پر کاربرد انگلیسی را شامل شود، و کلید گذاری را توسط اعداد انجام دهید.

۳. برنامه ای بنویسید که نام دانش آموزان یک کلاس را درون یک هش شامل حروف الفبا باشد، و هر کلید به یک آرایه از اسامی نسبت داده شود.