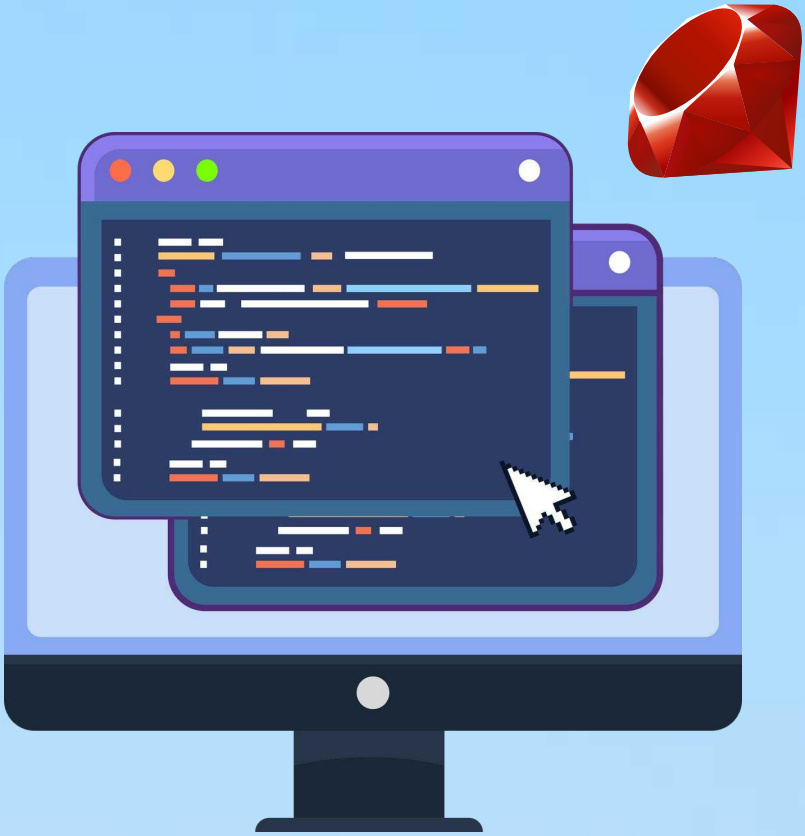


آموزش زبان برنامه نویسی روبی

https://iracode.com/ruby/

۰۹۱۲۴۸۱۴۰۳۵

آموزش زبان برنامه نویسی روبی

زبان برنامه نویسی روبی چیست ؟

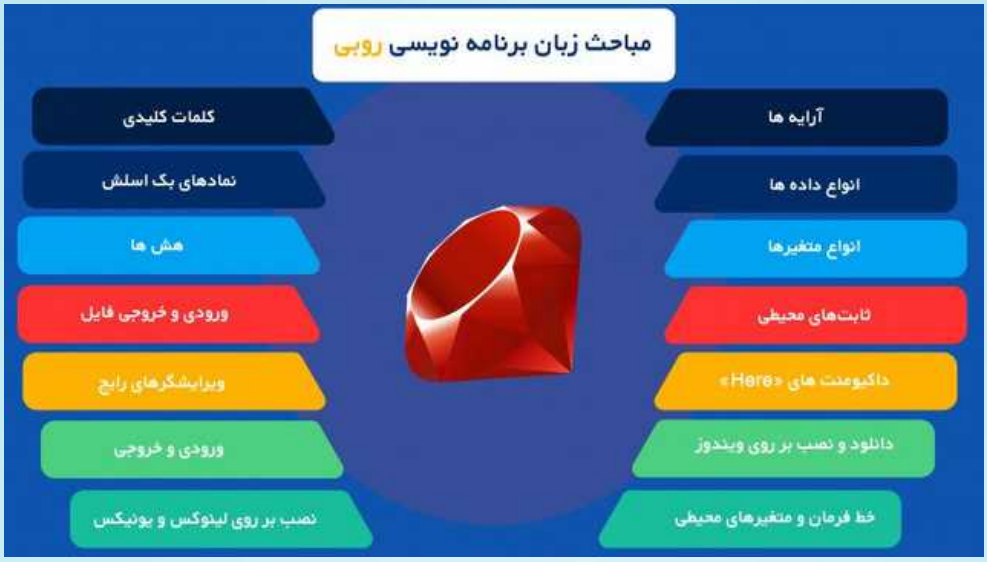
زبان برنامه نویسی Ruby ، یک زبان برنامه نویسی تماما شی گرا می باشد .
زبان برنامه نویسی روبی در سال ۱۹۹۳ توسط یوکیهیرو ماتسوموتو ایجاد شد .
زبان برنامه نویسی Ruby ، یک زبان برنامه نویسی همه منظوره و تفسیر شده همانند زبان برنامه نویسی پرل (Perl) و زبان برنامه نویسی پایتون (python) می باشد.
با آموزش الفبای برنامه نویسی روبی همراه ما باشید.

مشخصات زبان برنامه نویسی روبی

زبان برنامه نویسی روبی ، یک زبان متن باز بوده و بطور رایگان در وب در دسترس می باشد ، اما دارای کپی رایت است .
زبان برنامه نویسی Ruby ، یک زبان برنامه نویسی همه منظوره و تفسیر شده می باشد .
زبان برنامه نویسی روبی ، یک زبان برنامه نویسی شی گرا می باشد .
زبان برنامه نویسی Ruby ، یک زبان اسکریپت نویسی سمت سرور همانند زبان پرل و زبان پایتون می باشد .
از زبان برنامه نویسی روبی می توان برای نوشتن اسکریپت های " رابط دروازه عمومی " (CGI) استفاده نمود .
زبان برنامه نویسی Ruby را می توان در زبان نشانه گذاری فرا متنی (HTML) استفاده نمود .
زبان برنامه نویسی روبی ، سینتکس بی نقص و آسانی دارد که امکان یادگیری بسیار سریع و آسان زبان برنامه نویسی Ruby. را برای یک توسعه دهنده جدید فراهم می آورد.

سایر مشخصات زبان برنامه نویسی روبی

سینتکس زبان برنامه نویسی Ruby شبیه سینتکس بسیاری از زبان های برنامه نویسی از قبیل زبان برنامه نویسی C++ و پرل می باشد.
زبان برنامه نویسی روبی ، بسیار مقیاس پذیر بوده و برنامه های بزرگ نوشته شده در زبان برنامه نویسی روبی به آسانی قابل نگهداری می باشند . از زبان برنامه نویسی Ruby می توان برای توسعه برنامه های کاربردی اینترنت و اینترانت استفاده نمود .زبان برنامه نویسی روبی را می تواند در محیط های ویندوز و POSIX نصب نمود .
زبان برنامه نویسی Ruby از بسیاری از ابزارهای GUI از قبیل GTK ، Tcl / TK و OpenGL پشتیبانی می کند .زبان برنامه نویسی روبی می تواند به سادگی به Oracle ، MySQL ، DB و Sybase متصل شود . زبان برنامه نویسی Ruby دارای مجموعه ای غنی از توابع داخلی بوده .که می توانند مستقیما در اسکریپت های زبان برنامه نویسی روبی مورد استفاده قرار گیرند .



آرایه ها

ثابت های آرایه زبان برنامه نویسی روبی با قرار دادن مجموعه ای ازمنابع(Reference) به اشیائ جدا شده توسط ویرگول در بین کروشه ایجاد می شود . ویرگول انتهایی نادیده گرفته می شود .

اندیس آرایه ها همانند زبان برنامه نویسی C و زبان برنامه نویسی Java از ۰ شروع می شود.

```
usr/bin/ruby/!#
ary = [ "Ali", ۱۰, ۳/۱۴, "This is a string", "last element", ]
lary.each do li
puts i
end
```

این باعث حصول نتیجه ی زیر خواهد شد.

Ali
۱۰
۳/۱۴
This is a string
last element

روش های مختلفی برای ایجاد یک آرایه وجود دارد که یکی از آن ها استفاده از متد new کلاس می باشد .

```
names = Array.new
```

می توان سایز آرایه را نیز به شکل زیر مشخص کرد .

```
names = Array.new(۲۰)
```

آرایه names دارای سایز یا طول ۲۰ واحد است.

```
usr/bin/ruby/!#
names = Array.new(۲۰)
puts names.size # This returns 20
puts names.length # This also returns 20
```

این باعث حصول نتیجه ی زیر خواهد شد:

۲۰

۲۰

می توانید به هر المان آرایه به شکل زیر مقادیری انتساب دهید .

```
usr/bin/ruby/!#
names = Array.new(۴, "mac")
"puts "#{names}
```

این باعث حصول نتیجه ی زیر خواهد شد.

macmacmacmac

شما می توانید از بلاک کدی همراه با new استفاده کنید که هر المان بلاک به یکی از مقادیر آرایه انتساب داده خواهد شد .

```
usr/bin/ruby/!#
nums = Array.new(۱۰) { |e| e = e * ۲ }
"puts "#{nums}
```

این باعث حصول نتیجه ی زیر خواهد شد

۰۲۴۶۸۱۰۱۲۱۴۱۶۱۸

شکل دیگری از ایجاد آرایه به صورت زیر کار میکند

```
nums = Array.[](۱, ۲, ۳, ۴,۵)
```

و حالت ساده تری به شکل زیر نیز معتبر است .

```
nums = Array[۱, ۲, ۳, ۴,۵]
```

می توان یک بازه از اعداد را نیز برای ایجاد آرایه به شکل زیر مشخص کرد .

```
usr/bin/ruby/!#
digits = Array(۰..۹)
"puts "#{digits}
```

این باعث حصول نتیجه ی زیر خواهد شد.

انواع داده ها

انواع اصلی داده ها شامل اعداد ، رشته ها ، دامنه ها ، آرایه ها ، و هش ها می باشد .
در این قسمت انواع اعداد و رشته ها معرفی می شوند و در قسمت های بعدی دامنه ها ، آرایه ها ، و هش ها معرفی خواهند شد .

اعداد صحیح

اعداد صحیح در زبان برنامه نویسی Ruby اشیائ کلاس Fixnum یا Bignum هستند و در بازه ۲۳۰- تا ۲۳۰-۱ یا ۲۶۲- تا ۲۶۲-۱ قرار دارند .
مثال زیر نحوه نوشتن اعداد باینری مبنای ۲ و دهدهی مبنای ۱۰ و اکتال مبنای ۸ و هگزادسیمال مبنای ۱۶ را نشان می دهد .
با قراردادن یک کاراکتر یا بک اسلش کد در داخل کوتیشن می توانید به کد اسکی ASCII آن دست پیدا کنید .

```
Fixnum decimal # ۱۲۳
Fixnum decimal with underline # ۶۸۸۹_۱
Negative Fixnum # ۵۰۰۰-
octal # ۰۳۷۷
۰o377 # octal
۰xee # hexadecimal
۰b1011011 # binary
’b # character code for ‘b?
n # code for a newline (0x0a)\?
Bignum # ۱۲۳۴۵۶۷۸۹۰۱۲۳۴۵۶۷۸۹۰
```

اعداد اعشاری ممیز شناور

اعداد اعشاری اشیایی از کلاس Float هستند .

```
floating point value # ۱۲۳/۴
۱/۰e6 # scientific notation
۴E20 # dot not required
۴e+۲۰ # sign before exponential
```

رشته های زبان برنامه نویسی Ruby به سادگی ، توالی بایتهای ۸ بیتی و اشیائ کلاس String می باشند .
رشته های داخل علامت دابل کوتیشن اجازه جایگذاری و اعمال بک اسلش کد ها را نیز می دهند ولی رشته های داخل تک کوتیشن تنها اجازه اعمال بک اسلش کد به دو کد \\ و \’ را می دهند.

مثال:

```
usr/bin/ruby -w/!#
;'"\\" puts 'escape using
puts 'That\'s right
```

این خروجی زیر را تولید خواهد کرد.

```
"\" escape using
That's right
```

شما می توانید مقدار هر عبارت زبان برنامه نویسی روبی را با استفاده از کاراکترهای #{ expr } در یک رشته جایگزین کنید .

```
usr/bin/ruby -w/!#
;"puts "Multiplication Value : #{24*60*60}
```

این خروجی زیر را تولید خواهد کرد.

```
Multiplication Value : 86400
```

انواع متغیرها

متغییر ها در برنامه نویسی روبی

در زبان برنامه نویسی روبی پنج تایپ مختلف متغیر وجود دارد که در ادامه معرفی خواهند شد.

- متغیر های عمومی (Global Variables)
- متغیر های شی (Instance Variables)
- متغیر های کلاس (Class Variables)
- متغیر های محلی (Local Variables)
- ثابت ها (Constants)

متغیر های عمومی (Global Variables)

متغیر های عمومی یا گلوبال با $ شروع می شوند و اگر مقداردهی اولیه نشوند دارای مقدار nil هستند.

در مثال زیر $global_variable یک متغیر عمومی است.

در روبی با قراردادن یک کاراکتر # قبل از هر متغیر می توانید به مقدار آن دست پیدا کنید.

```
#!/usr/bin/ruby
$global_variable = ۱۰
class Customer
@@no_of_customers=۰
def initialize(name, addr)
```

```
@@no_of_customers += ۱
@cust_id=@@no_of_customers
@cust_name=name
@cust_addr=addr
end
def display_details()
puts "Customer id: #@cust_id"
puts "Customer name: #@cust_name"
puts "Customer address: #@cust_addr"
puts "Global variable is: #$global_variable"
puts "Total number of customers: #@@no_of_customers"
end
end
# Create Objects
cust1=Customer.new("John", "Wisdom Apartments, Ludhiya")
cust2=Customer.new("Poul", "New Empire road, Khandala")
# Call Methods
cust1.total_no_of_customers()
cust2.total_no_of_customers()
```

این مثال نتیجه زیر را ایجاد خواهد کرد.

```
Customer id: ۱
Customer name: John
Customer address: Wisdom Apartments, Ludhiya
Global variable is: ۱۰
Total number of customers: ۲
Customer id: ۲
Customer name: Poul
Customer address: New Empire road, Khandala
Global variable is: ۱۰
Total number of customers: ۲
```

متغیر های شی

متغیر های شی با @ شروع می شوند و اگر مقداردهی اولیه نشوند دارای مقدار nil هستند.

در مثال قبل @cust_id و @cust_name و @cust_addr متغیر های شی هستند.

متغیر های کلاس

در زبان برنامه نویسی Ruby متغیرهای کلاس متغیرهایی هستند که بین همه اشیایی که از یک کلاس ساخته می شوند مشترک هستند.

متغیر های کلاس با @@ شروع می شوند و اگر مقداردهی اولیه نشوند خطا تولید خواهند کرد.

در مثال قبل @@no_of_customers یک متغیر کلاس است.

متغیر های محلی

متغیر های محلی با حروف کوچک و یا زیرخط (_) شروع می شوند. حوزه تعریف آن ها داخل یک متد، حلقه و یا یک بلاک کد است.

وقتی یک متغیر محلی را بدون مقداردهی اولیه فراخوانی میکنید، مفسر زبان برنامه نویسی روبی فکر خواهد کرد که متدی بدون پارامتر و به همین نام را فراخوانی کرده اید.

در مثال قبل name و addr متغیر های محلی هستند.

ثابت ها

ثابت ها با حروف بزرگ آغاز می شوند. ثابت هایی را که در یک کلاس تعریف کنید در همان کلاس قابل دسترسی هستند و ثابت هایی را که بیرون از کلاس تعریف کنید دارای دسترسی عمومی هستند.

ثابت ها را نمی توانید در داخل یک متد تعریف کنید و باید حتما دارای مقدار اولیه باشند.

مقدار دهی دوباره به ثابت پیغام هشداری (warning) ایجاد خواهد کرد.

در مثال زیر نحوه استفاده از ثابت ها نمایش داده شده است.

```
#!/usr/bin/ruby
class Example
VAR1 = ۱۰۰
VAR2 = ۲۰۰
def show
puts "Value of first Constant is #{VAR1}"
puts "Value of second Constant is #{VAR2}"
end
end
# Create Objects
object=Example.new()
object.show
```

این مثال خروجی زیر را ایجاد خواهد کرد.

Value of first Constant is 100
Value of second Constant is 200

شبه متغیرهای زبان برنامه نویسی Ruby

این متغیرها، متغیرهای ویژه ای هستند که دارای ظاهر متغیرهای محلی بوده اما همانند ثابت ها رفتار می کنند.

شما نمی توانید هیچ مقداری را به این متغیرها اختصاص دهید.

self: شی گیرنده ی تابع جاری.

true: مقدار نشان دهنده ی مقداری صحیح.

false: مقدار نشان دهنده ی مقداری نادرست.

nil: مقدار نشاندهنده ی مقداری تعریف نشده.

__FILE__: نام سورس فایل جاری.

__LINE__: شماره سطر جاری در فایل سورس.

ثابت‌های محیطی

جدول زیر، بصورت فهرست وار تمام ثابت های محیطی در زبان برنامه نویسی روبی از پیش تعریف شده را نشان می دهد.

تذکرFALSE ، TRUE و NIL به جهت سازگاری با نسخه های قدیمی می باشند. استفاده از false ، true و nil ارجح می باشد.

Constant Name	Description
TRUE	Synonym for true.
FALSE	Synonym for false.
NIL	Synonym for nil.
ARGF	An object providing access to virtual concatenation of files passed as command-line arguments or standard input if there are no command-line arguments. A synonym for $<.
ARGV	An array containing the command-line arguments passed to the program. A synonym for $*.
DATA	An input stream for reading the lines of code following the __END__ directive. Not defined if __END__ isn't present in code.
ENV	A hash-like object containing the program's environment variables. ENV can be handled as a hash.
RUBY_PLATFORM	A string indicating the platform of the Ruby interpreter.
RUBY_RELEASE_DATE	A string indicating the release date of the Ruby interpreter.
RUBY_VERSION	A string indicating the version of the Ruby interpreter.
STDERR	Standard error output stream. Default value of $stderr.
STDIN	Standard input stream. Default value of $stdin.
STDOUT	Standard output stream. Default value of $stdout.
TOPLEVEL_BINDING	A Binding object at Ruby's top level.

داکیومنت های «Here»

داکیومنت های " Here " به رشته هایی اشاره میکند که از چند خط تشکیل شده اند . در این حالت پس از عبارت << می توان شناسه ای را مشخص کرد که رشته با آن پایان می یابد اگر شناسه داخل نقل قول یا کوتیشن قرار گرفته باشد نوع کوتیشن ، نوع رشته های بعدی را مشخص میکند . در اینجا ، مثال های مختلفی بیان شده است.

```
#!/usr/bin/ruby -w
print <<EOF
This is the first way of creating
here document ie. multiple line string.
EOF
print <<"EOF"; # same as above
This is the second way of creating
here document ie. multiple line string.
EOF
print <<`EOC` # execute commands
echo hi there
echo lo there
EOC
print <<"foo", <<"bar" # you can stackthem
I said foo.
foo
I said bar.
bar
```

که این خروجی زیر را تولید خواهد کرد .

```
This is the first way of creating
.her document ie. multiple line string
This is the second way of creating
.her document ie. multiple line string
hi there
lo there
.I said foo
.I said bar
```

دامنه ها

یک دامنه، نشان دهنده یک فاصله یعنی مجموعه ای از مقادیر با یک آغاز و یک پایان می باشد.
دامنه ها ممکن است با استفاده از لیترال های .. یا ... و یا با Range.new ایجاد شده باشند.
دامنه ها در زبان برنامه نویسی روبی به سه منظور استفاده می شوند:

- به عنوان یک توالی
- به عنوان شرط
- به عنوان فاصله یا دامنه

دامنه به عنوان یک توالی

دامنه های ایجاد شده با استفاده از .. از ابتدا تا انتها را شامل می شوند.

دامنه های ایجاد شده با استفاده از ... مقادیر پایانی را در بر نمی گیرند.

هرگاه دامنه ها به عنوان یک iterator بکار روند، تمام مقادیر داخل دنباله را به ترتیب بر میگردانند

دامنه (۱//۵) بدین معنی است که این دامنه، مقادیر ۱، ۲، ۳، ۴، ۵ را در بر گرفته و دامنه (۵...۱) بدین معنی است که این دامنه، مقادیر ۲، ۳، ۴ را در بر گرفته است.

```
usr/bin/ruby/!#
leach do ln.(۱۰//۱۵)
'' ,print n
end
```

این باعث حصول نتیجه ی زیر خواهد شد.

۱۰ ۱۱ ۱۲ ۱۳ ۱۴ ۱۵

با استفاده از متد to_a می توان یک دامنه را به یک آرایه تبدیل کرد.

```
#!/usr/bin/ruby
$, =", " # Array value separator
range1 = (1..10).to_a
range2 = ('bar'..'bat').to_a
puts "#{range1}"
puts "#{range2}"
```

که نتیجه زیر را تولید خواهد کرد.

۱, ۲, ۳, ۴, ۵, ۶, ۷, ۸, ۹, ۱۰

bar, bas, bat

دامنه دارای متد هایی است که به شما اجازه می دهد بر روی مقادیر آن حرکت کنید و مقادیر آن را به شیوه های مختلفی تست کنید.

```
#!/usr/bin/ruby
# Assume a range
digits = ۰..۹
puts digits.include?(۵)
ret = digits.min
puts "Min value is #{ret}"
```

```
ret = digits.max
puts "Max value is #{ret}"
ret = digits.reject {|i| i < ۵ }
puts "Rejected values are #{ret}"
digits.each do |digit|
puts "In Loop #{digit}"
end
```

که خروجی زیر را تولید خواهد کرد.

```
true
Min value is 0
Max value is 9
Rejected values are 5, 6, 7, 8, 9
In Loop 0
In Loop 1
In Loop 2
In Loop 3
In Loop 4
In Loop 5
In Loop 6
In Loop 7
In Loop 8
In Loop 9
```

دامنه ها به عنوان شرط

دامنه ها می توانند به صورت شرط نیز بکار روند.

به عنوان مثال در زیر خطهایی از ورودی که اولین خط با start شروع می شود و آخرین خط با end پایان می یابد را در خروجی چاپ میکند.

```
while gets
/print if /start/../end
end
```

دامنه ها را همچنین می توان در دستور case بکار برد.

مثال:

اجرای این کد نتیجه زیر را ایجاد خواهد کرد.

Pass with Merit

دامنه ها به عنوان یک بازه

کاربرد دیگر دامنه به عنوان تست یک بازه است، که می توان چک کرد که یک مقدار به یک بازه تعلق دارد یا خیر.

این عمل با اپراتور === که به آن اپراتور تساوی حالت (case equality operator) می گوییم، انجام می شود.

```
usr/bin/ruby/!#
if ((۱..۱۰) === ۵)
"puts "5 lies in (1..10)
end
if (('a'..'j') === 'c')
"puts "c lies in ('a'..'j')
end
if (('a'..'j') === 'z')
"puts "z lies in ('a'..'j')
end
```

اجرای این کد نتیجه زیر را تولید خواهد کرد.

```
lies in (1..10) ۵
c lies in (‘a’..’j’)
```

دانلود و نصب بر روی ویندوز

مراحل نصب زبان برنامه نویسی روبی بر روی سیستم عامل ویندوز در اینجا بیان شده است .

نصب روبی با نرم افزار RubyInstaller

بهترین راه برای نصب زبان برنامه نویسی روبی بر روی ویندوز استفاده از نرم افزار RubyInstaller است که می توانید آخرین نسخه آن را دانلود و نصب کنید .

بسته نرم افزار RubyInstaller همه پیکربندی های لازم برای نصب زبان برنامه نویسی روبی را انجام خواهد داد .

دانلودRuby installer

اگر هدف شما از نصب زبان برنامه نویسی روبی ، استفاده از روبی بر ریل است ، بهتر است از بسته های زیر استفاده کنید .

•بسته نرم افزار RailsInstaller که از RubyInstaller استفاده می کند اما ابزارهای دیگری که برای ریل مورد نیاز است را نیز نصب می کند و از ویندوز و مک اواس ایکس پشتیبانی می کند .
•بسته نرم افزار Bitnami Ruby Stack که محیط کامل توسعه ریل را در خود دارد و از ویندوز ، لینوکس و او اس ایکس پشتیبانی میکند .

نصب روبی از پکیج اصلی

اما اگر می خواهید خودتان زبان برنامه نویسی روبی را نصب کنید.
فایل زیپ آخرین نسخه زبان برنامه نویسی روبی را با استفاده از این لینک دانلود نرم افزار برنامه نویسی روبی بگیرید .
پس از دانلود آرشیو روبی ، آن را باز نموده و به دایرکتوری تازه ایجاد شده بروید.
بر روی فایل Ruby x.y.z.exe دوبار کلیک کنید . ویزارد نصب زبان برنامه نویسی روبی آغاز می شود .
بر روی Next کلیک کرده تا به صفحه اطلاعات مهم ویزارد برسید و تا نصب کامل زبان برنامه نویسی روبی ، ادامه دهید .
اگر نصب شما متغیرهای محیطی را بطور مناسب پیکربندی نکرده باشد ، ممکن است به برخی از آن ها نیاز داشته باشید .
اگر از ویندوز ۹x استفاده می کنید ، خطوط زیر را به فایل c:\autoexec.bat اضافه کنید
"%set PATH="D:\(ruby install directory)\bin;%PATH
کاربران ویندوز NT/2000 و نسخه های جدیدتر باید رجیستری های خود را اصلاح نمایند
برای این منظور ، بر روی Control Panel > System Properties > Environment Variables کلیک نمایید .
در زیر گزینه متغیرهای سیستم System Variables ، گزینه Path را انتخاب نموده و بر روی EDIT کلیک کنید .
دایرکتوری زبان برنامه نویسی روبی خود را به انتهای لیست Variable Value افزوده و بر روی OK کلیک نمایید .
در زیر گزینه System Variables ، گزینه PATHEXT را انتخاب نموده و بر روی EDIT کلیک کنید .
سپس .rb و .rbw را به لیست Variable Value افزوده و بر روی OK کلیک نمایید .
پس از نصب ، با نوشتن دستور زیر در خط فرمان اطمینان حاصل نمایید که همه چیز بدرستی کار می کند .

```
> ruby -v
```

اگر همه چیز صحیح باشد ، نسخه مفسر زبان برنامه نویسی روبی نصب شده باید ruby 1.8.7 بصورت بالا نشان داده شود .

ممکن است نسخه متفاوتی را نصب کرده باشید ، در نتیجه ، نسخه متفاوتی نمایش داده خواهد شد .

دستور BEGIN و END

دستور BEGIN در زبان برنامه نویسی روبی

شکل دستور

```
BEGIN {
code
}
```

مجموعه کدهایی را مشخص میکند که قبل از شروع برنامه باید اجرا شوند .

مثال:

```
#!/usr/bin/ruby
puts "This is main Ruby Program"
BEGIN {
puts "Initializing Ruby Program"
}
```

این کد نتیجه زیر را در پی خواهد داشت .

```
Initializing Ruby Program
This is main Ruby Program
```

دستور END در زبان برنامه نویسی Ruby

شکل دستور

```
END {
code
}
```

مجموعه کدهایی را مشخص میکند که بعد از پایان برنامه باید اجرا شوند .

مثال:

```
#!/usr/bin/ruby
puts "This is main Ruby Program"
END {
puts "Terminating Ruby Program"
}
BEGIN {
puts "Initializing Ruby Program"
}
```

این کد نتیجه زیر را در پی خواهد داشت .

```
Initializing Ruby Program
This is main Ruby Program
Terminating Ruby Program
```

ساختار

فاصله خالی در زبان برنامه نویسی روبی

کاراکترهای فاصله ی سفید مانند فاصله ها (space) و تب (tab) به طور کلی در کد زبان برنامه نویسی Ruby نادیده گرفته می شوند ، مگر زمانی که در رشته ها ظاهر شوند .

```
a + b is interpreted as a+b ( Here a is a local variable)
a +b is interpreted as a(+b) ( Here a is a method call)
```

کارکتر انتهای خط در زبان برنامه نویسی روبی

زبان برنامه نویسی Ruby ، کاراکترهای نقطه – ویرگول ؛ و سطر جدید را به عنوان پای ان یک عبارت تفسیر می کند .

باوجود این ، اگر زبان برنامه نویسی روبی با عملگرهایی مانند + ، – ، یا بک اسلش (backslash) در انتهای یک سطر مواجه شود ، آن ها را به معنای ادامه داشتن عبارت در نظر می گیرد .

شناسه ها در زبان برنامه نویسی Ruby

شناسه ها ، اسامی متغیرها ، ثابت ها و توابع می باشند .

شناسه های زبان برنامه نویسی روبی نسبت به بزرگ یا کوچک بودن حروف ، حساس هستند . این بدین معنی است که Ram و RAM دو شناسه مختلف در زبان برنامه نویسی Ruby می باشند .

شناسه ها در زبان برنامه نویسی روبی شامل حروف کوچک و بزرگ اعداد و زیرخط می باشند .

توضیحات در زبان برنامه نویسی روبی

توضیحات زبان برنامه نویسی Ruby با یک کاراکتر پوند / شارپ # آغاز شده و تا انتهای سطر (EOL) ادامه می یابد .

```
# I am a comment. Just ignore me.
```

و یا

```
name = “Madisetti” # This is again comment
```

خط فرمان و متغیرهای محیطی

گزینه های خط فرمان زبان برنامه نویسی روبی

این فصل ، تمام گزینه های خط فرمان زبان برنامه نویسی Ruby را که می توانید همراه با مفسر زبان برنامه نویسی روبی از آن استفاده نمایید ، فهرست می کند .

مفسر زبان برنامه نویسی Ruby به طور کلی به روش زیر از خط فرمان اجرا می گردد.

$ ruby [options] [.] [programfile] [arguments ...]

مفسر زبان برنامه نویسی روبی می تواند با هر یک از گزینه های زیر مورد استفاده قرار گیرد تا محیط و رفتار آن را کنترل نماید .

Option	Description
-a	Used with -n or -p to split each line. Check -n and -p options.
-c	Checks syntax only, without executing program.
-C dir	Changes directory before executing (equivalent to -X).
-d	Enables debug mode (equivalent to -debug).
-F pat	Specifies pat as the default separator pattern ($;) used by split.
-e prog	Specifies prog as the program from the command line. Specify multiple -e options for multiline programs.
-h	Displays an overview of command-line options.
-i [ext]	Overwrites the file contents with program output. The original file is saved with the extension ext. If ext isn't specified, the original file is deleted.
-I dir	Adds dir as the directory for loading libraries.
-K [kcode]	Specifies the multibyte character set code (e or E for EUC (extended Unix code); s or S for SJIS (Shift-JIS); u or U for UTF-8; and a, A, n, or N for ASCII).
-l	Enables automatic line-end processing. Chops a newline from input lines and appends a newline to output lines.
-n	Places code within an input loop (as in while gets; ... end).
-0 [octal]	Sets default record separator ($/) as an octal. Defaults to \0 if octal not specified.
-p	Places code within an input loop. Writes $_ for each iteration.
-r lib	Uses require to load lib as a library before executing.
-s	Interprets any arguments between the program name and filename arguments fitting the pattern -xxx as a switch and defines the corresponding variable.
-T [level]	Sets the level for tainting checks (1 if level not specified).
-v	Displays version and enables verbose mode
-w	Enables verbose mode. If programfile not specified, reads from STDIN.
-x [dir]	Strips text before #!ruby line. Changes directory to dir before executing if dir is specified.
-X di	Changes directory before executing (equivalent to -C).
-y	Enables parser debug mode.
–copyright	Displays copyright notice
–debug	Enables debug mode (equivalent to -d).
–help	Displays an overview of command-line options (equivalent to -h).
–version	Displays version.
–verbose	Enables verbose mode (equivalent to -v). Sets $VERBOSE to true.
–yydebug	Enables parser debug mode (equivalent to -y).

عبارت های الگودار

عبارت های الگودار یک توالی خاص از کارکترهاست که با یک الگو کمک میکند یک رشته یا یک دسته از رشته های به فرم مشخص را پیدا کنیم.

ساختار

یک عبارت الگودار، الگویی است که بین دو علامت اسلش و یا بعد از کاراکترهای r% بین و بین دو علامت دلخواه قرار می گیرد. به فرم زیر:

```
/pattern/
/pattern/im # option can be specified
%r!/usr/local! # general delimited regular expression
```

مثال:

```
#!/usr/bin/ruby
line1 = "Cats are smarter than dogs";
line2 = "Dogs also like meat";
if ( line1 =~ /Cats(.*)/ )
puts "Line1 contains Cats"
end
if ( line2 =~ /Cats(.*)/ )
puts "Line2 contains Cats"
end
```

این مثال خروجی زیر را تولید خواهد کرد:

```
Line1 contains Cats
```

تغییر دهنده ها

عبارتهای الگودار ممکن است تغییر دهنده های مختلفی داشته باشند که ابعاد مختلف انطباق را تغییر می دهند.تغییر دهنده ها بعد از دومین اسلش قرار میگیرند. فهرست برخی از آن ها در جدول زیر آمده است.

الگوهای مختلف

بغیر از کاراکترهای کنترلی (+ ? . * ^ $ ( ) [ ] { } | \) همه دیگر کاراکترها بر خودشان انطباق می یابند.شما می توانید با یک بک اسلش رفتار یک کاراکتر کنترلی را به مانند کاراکترهای عادی تغییر دهید.فهرست زیر ساختار عبارتهای الگودار قابل استفاده در زبان برنامه نویسی روبی را نشان می دهد.

کلمات کلیدی

فهرست زیر ، کلمات کلیدی در زبان برنامه نویسی روبی را نشان می دهد .
این کلمات کلیدی نباید به عنوان اسامی ثابت یا متغیر در برنامه شما مورد استفاده قرار گیرند .

```
do next then
END else nil true
alias elsif not undif
and end or unless
begin unsure redo until
break false rescue when
case for retry while
class if return while
def in self _FILE_
defined? module super _LINE_
```

نصب بر روی لینوکس و یونیکس

در اینجا ، مراحل نصب زبان برنامه نویسی Ruby بر روی سیستم عامل لینوکس و یونیکس بیان شده است

هشدار: قبل از ادامه ، مطمئن شوید root هستید یا جزو گروه sudoers می باشید .

فایل زیپ آخرین نسخه روبی را با استفاده از این لینک دانلود نرم افزار برنامه نویسی روبی بگیرید .

پس از دانلود نرم افزار برنامه نویسی روبی ، آرشیو آن را باز نموده و به دایرکتوری تازه ایجاد شده بروید.

```
$ tar -xvzf ruby-x.y.z.tar.gz
$ cd ruby-x.y.z
```

بجای x و y و z اعدادی مربوط به نسخه ای که شما دانلود کرده اید قرار میگیرند.

حال ، سورس کد را به شرح زیر ، پیکربندی و کامپایل نمایید.

```
$ ./configure
$ make
```

در نهایت ، مفسر زبان برنامه نویسی روبی را به شرح زیر نصب کنید.

```
$ su -l root # become a root user
$ make install
$ exit     # become the original user again
```

پس از نصب ، با نوشتن دستور زیر در خط فرمان اطمینان حاصل نمایید که همه چیز بدرستی کار می کند .

```
$ ruby -v
ruby 1.8.7 (2011-06-30 patchlevel 352) [i686-linux]
```

اگر همه چیز صحیح باشد ، نسخه مفسر زبان برنامه نویسی Ruby نصب شده باید بصورت بالا نشان داده شود .
ممکن است نسخه متفاوتی را نصب کرده باشید ، در نتیجه ، نسخه متفاوتی نمایش داده خواهد شد
در نسخه اوبونتو Ubuntu لینوکس برای نصب زبان برنامه نویسی روبی کافی است دستور زیر را اجرا کنید .

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

اگر از لینوکس توزیع RedHat یا SUSE استفاده می کنید استفاده از ابزار yum ساده ترین روش نصب روبی یا هر RPM دیگر می باشد .
دستور زیر را در خط فرمان خود تایپ نمایید ؛ خواهید دید که زبان برنامه نویسی روبی بر روی کامپیوتر شما نصب می شود .

```
$ yum install ruby
```

نمادهای بک اسلش

فهرستی از نمادهای بک اسلش (backslash) که توسط زبان برنامه نویسی Ruby پشتیبانی می شوند ، در زیر ارائه شده است.

هش ها

هش در زبان برنامه نویسی روبی با قرار دادن لیستی از جفت های کلید/مقدار key=>value در بین آکولاد ، همراه با یک ویرگول بین هر جفت ایجاد می شود . ویرگول انتهایی نادیده گرفته می شود .
مثال:

```
#!/usr/bin/ruby
hsh = colors = { "red" => 0xf00, "green" => 0x0f0 }
hsh.each do |key, value|
print key, " is ", value, "\n"
end
```

این باعث حصول نتیجه ی زیر خواهد شد.

```
green is 240
red is 3840
```

همانند آرایه ها روش های مختلفی برای ایجاد هش وجود دارد . یک هش را می توان با استفاده از متد new به شکل زیر ایجاد کرد .

```
months = Hash.new
```

شما می توانید در هنگام ایجاد هش یک مقدار پیشفرض نیز بجای nil به آن انتساب دهید .

```
months = Hash.new( "month" )
or
months = Hash.new "month"
```

اگر کلیدی در هش را تقاضا کنید که وجود نداشته باشد ، مقدار nil یا مقدار پیشفرض برگردانده خواهد شد .

```
#!/usr/bin/ruby
months = Hash.new( "month" )
puts "#{months[0]}"
puts "#{months[72]}"
```

این باعث حصول نتیجه ی زیر خواهد شد

```
month
month
```

و مثال دیگر:

```
#!/usr/bin/ruby
H = Hash["a" => ۱۰۰, "b" => ۲۰۰]
puts "#{H['a']}"
puts "#{H['b']}"
```

این باعث حصول نتیجه ی زیر خواهد شد

```
100
200
```

شما می توانید از هر شی زبان برنامه نویسی Ruby به عنوان کلید یا مقدار استفاده کنید بنابراین تعریف زیر معتبر خواهد بود .

```
[۱,"jan"] => "January"
```

ورودی و خروجی فایل

متد File.new

شما می توانید یک شی از نوع File را با استفاده از متد File.new برای خواندن، نوشتن و یا هردو استفاده کنید و در نهایت با استفاده از متد File.close فایل را خواهید بست.

```
aFile = File.new("filename", "mode")
# ... process the file
aFile.close
```

متد File.open

از متد File.open نیز می توانید برای خواندن و نوشتن بر روی یک فایل استفاده کنید.

اما اختلاف آن با File.new در این است که می توانید از File.open در یک بلاک کد استفاده کنید در حالی که از File.new نمی توان به این صورت استفاده کرد.

```
File.open("filename", "mode") do laFilel
# ... process the file
end
```

در اینجا، فهرستی از حالت های مختلف باز کردن یک فایل ارائه شده است.

خواندن و نوشتن فایل

همه متد هایی که برای ورودی خروجی استاندارد وجود داشت برای ورودی و خروجی فایل نیز وجود دارد.

به این ترتیب همانگونه که gets برای خواندن از ورودی استاندارد بکار رفت aFile.gets نیز برای ورودی از فایل بکار می رود.

اما در فایلها یک سری دستورات ویژه ورودی خروجی فایل نیز وجود دارند که کار را راحت تر میکنند.

متد sysread

از متد sysread برای خواندن یک فایل می توان استفاده کرد

اگر خط زیر محتوای یک فایل متنی باشد:

.This is a simple text file for testing purpose

با استفاده از sysread فایل را به شکل زیر می خوانیم:

```
#!/usr/bin/ruby

aFile = File.new("input.txt", "r")
if aFile
content = aFile.sysread(20)
puts content
else
puts "Unable to open file!"
end
```

این دستور ۲۰ کاراکتر اول فایل را می خواند و اشاره گر فایل را در محل ۲۱ امین کاراکتر قرار می دهد.

متد syswrite

با استفاده از متد syswrite میتوانید اطلاعات را در داخل یک فایل بریزید:

```
#!/usr/bin/ruby
aFile = File.new("input.txt", "r+")
if aFile
aFile.syswrite("ABCDEF")
else
puts "Unable to open file!"
end
```

این کد کاراکترهای ABCDEF را در فایل می نویسد.

ویرایشگرهای رایج

برای نوشتن برنامه های Ruby خود ، به یک ویرایشگر نیاز دارید .

در این قسمت ویرایشگرهای رایج زبان برنامه نویسی روبی را معرفی میکنیم:

اگر بر روی سیستم عامل ویندوز کار می کنید ، می توانید از تمام ویرایشگرهای ساده متنی مانند Notepad++ یا Edit plus استفاده نمایید .

ادیتور VIM ، ویرایشگر بسیار ساده متنی است که تقریبا بر روی تمام سیستم عامل های یونیکس و در حال حاضر ویندوز ، در دسترس می باشد .

همچنین ، می توانید از ویرایشگر مورد علاقه Vi خود برای نوشتن برنامه های زبان برنامه نویسی روبی استفاده نمایید .

نرم افزار RubyWin نیز یک محیط توسعه یکپارچه زبان برنامه نویسی روبی (IDE) برای ویندوز می باشد .

محیط توسعه روبی (RDE) نیز یک IDE بسیار خوب دیگر برای کاربران ویندوز می باشد .

روبی تعاملی IRb چیست ؟

روبی تعاملی (IRb) ، خط فرمانی را برای آزمایش فراهم می آورد .

شما می توانید نتایج عبارات را بلافاصله در داخل خط فرمان IRb بصورت خط به خط مشاهده نمایید .

این ابزار همراه با نصب زبان برنامه نویسی روبی راه اندازی می شود ، بنابراین برای راه اندازی IRb نیاز به کار اضافه ای نیست .

تنها irb را در خط فرمان خود تایپ نمایید تا نشست روبی تعاملی آغاز گردد .

```
$irb
 irb 0.6.1(99/09/16)
 irb(main):001:0> def hello
 irb(main):002:1> out = "Hello World"
 irb(main):003:1> puts out
 irb(main):004:1> end
 nil
 irb(main):005:0> hello
 Hello World
 nil
 irb(main):006:0>
```

ورودی و خروجی

ورودی و خروجی در زبان برنامه نویسی روبی روشهای مختلفی را فراهم می کند.
کلاس IO همه متد های پایه مانند read و write و gets و puts و readline و getc و printf را فراهم آورده است.

دستور puts

دستور puts به مفسر زبان برنامه نویسی روبی فرمان می دهد مقادیر یک متغیر را در خروجی چاپ کند.
در این هنگام یک کاراکتر سطر جدید نیز چاپ خواهد شد.
مثال:

```
#!/usr/bin/ruby
val1 = "This is variable one"
val2 = "This is variable two"
puts val1
puts val2
```

این خروجی زیر را تولید خواهد کرد:

```
This is variable one
This is variable two
```

دستور gets

دستور gets برای گرفتن داده های ورودی از ورودی استاندارد STDIN استفاده می شود.
مثال:

```
#!/usr/bin/ruby
puts "Enter a value :"
val = gets
puts val
```

این خروجی زیر را تولید خواهد کرد:

```
Enter a value :
This is entered value
This is entered value
```

دستور putc

برخلاف دستور puts که کل یک رشته را در خروجی چاپ می کند. دستور putc هر بار یک کاراکتر را چاپ خواهد کرد.

مثال:

```
#!/usr/bin/ruby
str="Hello Ruby!"
putc str
```

این خروجی زیر را تولید خواهد کرد:

```
H
```

دستور print

دستور print مشابه دستور puts می باشد، با این اختلاف که دستور print کاراکتر خط جدید را در انتها اضافه نخواهد کرد.

مثال:

```
#!/usr/bin/ruby
print "Hello World"
print "Good Morning"
```

این خروجی زیر را تولید خواهد کرد:

```
Hello WorldGood Morning
```