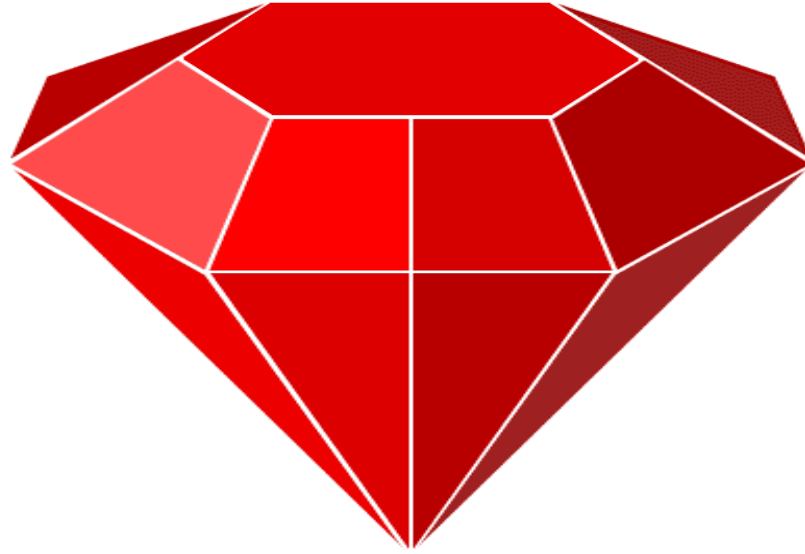


# Ruby Programlama Dili



Pınar Yanardağ

[pinar@comu.edu](mailto:pinar@comu.edu.tr)  
.tr

# İçerik

- ◆ Ruby'nin Tarihçesi
- ◆ Temel Felsefesi
- ◆ Genel Özellikleri
- ◆ Kullanım Alanları
- ◆ Diğer Dillerle Karşılaştırmalar
- ◆ Basit Kod Örnekleri

# Ruby'nin Kısa Tarihçesi

- ◆ Yukihiro Matsumoto (a.k.a *Matz*) tarafından yaratıldı.
- ◆ 24 Şubat 1993 yılında geliştirilmeye başlandı.  
  
Aralık 1995'de, Ruby 0.95 sürümü Japon haber listelerine duyuruldu.
- ◆ Son kararlı sürümü, 1.8.4 (Aralık 2005)

# Matz, Ruby'yi Niçin Yarattı?

- ◆ Üniversite yıllarındaki en büyük hayali, kendi programlama dilini tasarlamaktı.
- ◆ Nesneye yönelik programlamanın avatajlarını, betikleme alanında göstermek istiyordu.
- ◆ Aradığı özelliklere uygun bir dil yoktu. Bu yüzden tasarlayacağı dilin;

Perl'den daha güçlü,

Python'dan daha nesneye yönelik olmasını istiyordu.

# Ruby Adı, Nereden Geliyor?

- ◆ Ruby dilinin tasarım amaçları arasında, Perl'den daha güçlü bir dil olması yatıyor.

- ◆ Perl (pearl), Haziran ayını simgeleyen bir burç taşı.

Ruby ise, Temmuz ayını simgeliyor.

- ◆ İyi bir isim, dizaynın %80'inin bitmiş olduğu anlamına geliyor.

İyi bir isim, motivasyonu sağlıyor, ve çalışmaya teşvik ediyor.

*Programlama dillerinde, en önemli şey isimdir. İyi bir isme sahip olmayan bir programlama dili, başarıya ulaşamaz. Geçenlerde çok iyi bir isim buldum ve şimdi uygun bir dil arıyorum.*

*Donald Knuth*

# Ruby'nin Temel Felsefesi

## - 1

### ◆ *İyi Arayüz Prensibi;*

Programlama dilleri, arayüz olarak kabul edilebilir. İyi arayüzün özellikleri;

- Uyumluluk,
- Esneklik,
- Özlülük

### ◆ *Özlülük Prensibi; (we are the masters they're the slaves)*

Ruby'nin ana felsefesine göre, programcı makinaya odaklanmak yerine, programlamanın nasıl yapıldığına odaklanmalıdır.

# Ruby'nin Temel Felsefesi -

## 2

### ♦ *Mükemmel Programlama Dili Yoktur;*

Programlama dillerine iki tür bakış vardır,

- Programlama dili ile neler yapabileceğiniz,
- Programlama yaparken kendinizi nasıl hissettiğiniz.

### ♦ *Özgürlük ve Rahatlık;*

Birşeyi yapmanın birden çok yolu vardır.

Ruby, size seçme özgürlüğü verir.

# Ruby'nin Temel Felsefesi - 3

## ♦ *Ruby'den Haz Almak;*

Programlama dilinin sihirli kurallarına dikkat etmek yerine, çözeceğiniz probleme odaklanın.

Eğer, yazdığınız pseudo kodlar bilgisayarınızda direkt çalışırsa, ne hissederdiniz?

## ♦ *En Az Sürpriz Prensipleri (Principle of Matz's Least Surprise);*

Ruby'nin amacı, programlamada harcanan eforu en aza indirmektir.

Belirsiz durumlarla karşılaştığınızda, çözüm sizi sürprize uğratmayacak olanıdır.

# Ruby'nin Genel Özellikleri

## - 1 Nesneye yönelik ;

- Ruby, yüzde yüz saf nesneye yönelik bir dildir (Smalltalk gibi, bu konuda hiçbir istisna kabul etmez).

- Yönlendirdiğiniz herşey ve bunların döndürdüğü sonuçlar birer nesnedir,

- Metotlar, nesneye bir mesaj yollayarak uyandırılırlar,

- `number = Math.abs(number)` // Java

- `number = number.abs` // Ruby

1 + 2

+ ( **2 argümanı ile birlikte**), 1 nesnesine mesaj gönderiliyor.

# Ruby'nin Genel Özellikleri

- 2

## ◆ *Özel Nesneye Yönelik Özellikler;*

- Ruby tekil mirası destekler, ancak çoklu miras gerçekleştirimini şık bir yolla sağlar,

- Mix-in tekniği,

- Tekil metotlar ...

# Ruby'nin Genel Özellikleri

- 3

♦ *Betikleme Dili;*

*Ruby'nin Betikleme Özellikleri;*

- Çoğu betikleme dili gibi, yorumlayıcı sayesinde , edit- run- edit tekniğiyle hızlı geliştirme olanağı sağlar. Derlemeye ihtiyaç duymaz.

- Daha az kod yazmanız gerektiği için, hızlı programlama yaparsınız.

# Ruby'nin Genel Özellikleri

## - 4 ♦ *Betikleme Mi!*

- Betik dilleri; sıradüzensel interaktif komutlar bütününden, tam anlamıyla donatılmış programlama dillerine dönüştü.
- Yakın bir zamana kadar, kimse betik dilleriyle *gerçek* uygulamalar yazmayı düşünmüyordu.
- Ancak bugün betikleme dilleri, çeşitli işlere hizmet ediyorlar;
  - Kişisel yönetici araçları,
  - Web uygulama framework'leri,
  - Gerçek dünya uygulamaları,
  - Devasa verilerin analizleri ...

# Ruby'nin Genel Özellikleri

- 5

## ♦ *Yorumlanan (interpreted) Bir Dil;*

*Avantaj, ya da Dezavantaj?*

- Betik dilleri için hızlı geliştirme,
- Kodlar tek başına çalıştırılabilir uygulama haline dönüştürülemez.
- Performans düşüklüğü/yavaşlık ...?

# Ruby'nin Genel Özellikleri

## – 6 ♦ *Türemiş Bir Dil;*

- Ruby, bir çok programlama dilinden özellikler almıştır;
  - Smalltalk, CLU, Lisp, C, C++, Perl, Kornshell...
- Tekerleği yeniden keşfetmeye gerek yok,
- Bozulmamış olanı düzeltmeye gerek yok,
- İnsanların varolan tecrübelerinden faydalanmak her zaman iyidir.

*Eğer birçoklarından daha ileriye görebilmişsem, bu devlerin omuzlarında durup ileriye bakmamdan ötürüdür olmuştur.*

*Isaac Newton*

# Ruby'nin Genel Özellikleri

- 7

## ◆ *Dinamik Tanımlama;*

- Değişken bildirimleri gereksiz;
- Değişkenlerin yaşam sınırları basit kurallarla belirleniyor;
  - “degisken” = yerel değişken
  - “@degisken” = örnek değişken
  - “\$degisken” = global değişken
- Değişkenlerin türü yoktur.

# Ruby'nin Genel Özellikleri

- 8 *Söz dizimi basit ve tutarlı;*

```
moon@debian:~$ irb
irb(main):001:0> dizi = [1.5, "Ruby", 6]
=> [1.5, "Ruby", 6]
irb(main):002:0> dizi[1]
=> "Ruby"
irb(main):003:0> dizi[-1]
=> 6
irb(main):004:0> dizi << "Rails"
=> [1.5, "Ruby", 6, "Rails"]
irb(main):005:0> dizi2 = dizi [1,2]
=> ["Ruby", 6]
irb(main):006:0> dizi2
=> ["Ruby", 6]
```

# Ruby 'nin Genel Özellikleri-

## 9

### ◆ *Kolay ve Hızlı;*

- Güçlü metin işleme ve düzenli ifadelerle sahiptir,
- Bellek yönetimi otomatiktir,
- Hata yakalama mekanizmaları bulunur,
- Otomatik çöp toplayıcısına sahiptir.

# Ruby'nin Genel Özellikleri- 10

## ◆ *Doğrudan Sistem Çağruları Gönderebilme;*

- UNIX'teki tüm sistem çağrılarına erişebiliyor,
- Win32 API'si sayesinde, tüm sistem çağrılarına erişebiliyor.

## ◆ *Yüksek Taşınabilirlik;*

- Linux üzerinde geliştirilse de diğer işletim sistemleri üzerinde de çalışabilir,
- UNIX, DOS, Windows 95/98/Me/NT/2000/XP, MacOS, BeOS, OS/2 ...

# Ruby, Hangi Uygulamalar İçin Uygun?

- ◆ Metin işleme,
- ◆ CGI programlama,
- ◆ Web programlama,
- ◆ XML programlama,
- ◆ GUI uygulamaları,
- ◆ Yapay zeka ve keşifsel matematik,
- ◆ Genel programlama,
- ◆ Programlama eğitimleri,
- ◆ Extreme programlama ...

# Ruby, Hangi Uygulamalar İçin Uygun Değil?

- ◆ Yüksek trafikli web uygulamalarında,
- ◆ İşletim sistemi gerçekleştirmelerinde,
- ◆ Derleyici gerçekleştirmelerinde ...

# Karşılaştırmalar: Ruby vs. Java -1

## ◆ *Typing;*

Ruby;       dinamik yazımlı,  
Java;       statik yazımlı

## ◆ *Miras;*

Ruby;       mix-in,  
Java;       tekli miras (interface -> mix-in)

## ◆ *Aktif nesne;*

Ruby;       self,

Java;       this

# Karşılaştırmalar: Ruby vs. Java -2

## ◆ *İşleyiş;*

Ruby;      Yorumlanan,  
Java;      Bytecode

## ◆ *Saflık;*

Ruby;      Her şey birer nesne,

Yapılan her işlem, nesneye mesaj olarak geçiyor,  
Java;      Nesne olmayan kısımlar var



Java'nın sanal makinası daha hızlı.

# Karşılaştırmalar: Ruby vs. Perl -1

## ◆ *Saflık;*

Ruby; Herşey bir nesne,

Perl; Nesne olmayan şeyler var

## ◆ *Miras;*

Ruby, Mix-in,

Perl, Çoklu miras

# Karşılaştırmalar: Ruby vs. Perl -2

- ◆ Ruby, Perl'e göre öğrenmesi ve kullanması daha kolay bir dil, ve daha basit söz dizimine sahip,
- ◆ Ruby'de `$_` kodları, veri tipleri için değil, değişkenlerin yaşam alanını belirlemek için kullanılır.
- Perl, Ruby'ye göre daha hızlı, ve Unicode desteğine sahip.

# Karşılaştırmalar: Ruby vs. Python -1

## ◆ *Saflık;*

Ruby, Tüm işlemler nesneye mesaj olarak geçer,

`-3.abs`

Python, Metotların haricinde, fonksiyonlar yazmak da mümkündür.

`abs(-3)`

## ◆ *Miras;*

Ruby, Mix-in,

Python, Çoklu miras

# Karşılaştırmalar: Ruby vs. Python -2

- ◆ Ruby, daha doğal bir operator overloading'e sahiptir,

```
def +(x)
  x+5
end
```

- ◆ Ruby, çoğu zaman Python'dan daha hızlı,

# Niçin Ruby?

- ◆ Basit; öğrenmesi ve kodlaması kolay,
- ◆ Güçlü,
- ◆ Zengin kütüphaneler,
- ◆ Hızlı geliştirme,
- ◆ Yardımsever Ruby topluluğu,
- ◆ Açık kaynak kodlu,
- ◆ Eğlenceli :)

# Niçin Ruby Değil?

- ◆ *Performans;*

Perl ya da Python gibi, C kodu ile sarılsa dahi yeterince hızlı değil,

- ◆ *İyi bir VM'ye sahip değil,*

Ruby2, RITE

- ◆ *Varolan dahili standartlar,*

Unicode desteği tümleşik değil,

- ◆ *Deneyim,*

İyi bilinen bir dil değil; belgelendirme çok az,

Çok az, deneyimli coder...

# Ruby Nereelerde Kullanılıyor?

-1

## ♦ *Simülasyonlarda;*

NASA ve Motorola, bazı simülasyonlarını yapmak için Ruby'yi kullanıyor,

## ♦ *Robotbilimde;*

Siemens, bir servis robotunun kontrolünü sağlamak için Ruby'yi kullandı,

## ♦ *Oyunlarda;*

Japonya'da ticari bir oyun firması, Ruby ile geliştirdiği RPG oyununu Haziran 2004'te piyasaya sürdü,

# Ruby Nereelerde Kullanılıyor?

-2

## ◆ *Telefonculukta;*

UCB, kablosuz telefonları ve trafiğin yükünü kontrol etmek için Ruby'yi kullanıyor,

3G kablosuz telefonculuk şirketi, ~150K'lık C++ koduna karşı, ~6K'lık Ruby kodunu kullandı,

## ◆ *Bilimde,*

Yüksek yoğunluklu yıldız sistemlerinin modellenmesi üzerinde çalışan ACS şirketi de projelerinde Ruby kullanıyor...

# Basit Kod Örnekleri 1- Başlangıç

- ◆ `puts "Merhaba Dünya!"`  
Merhaba Dünya!
- ◆ `print "Merhaba Dünya!\n"`  
Merhaba Dünya!
- ◆ `print "Merhaba"+"Dünya!"+"\n"`  
Merhaba Dünya!
- ◆ `print "Merhaba", "Dünya!", "\n"`  
Merhaba Dünya!

# Basit Kod Örnekleri 2 - Dizgeler 1

## ◆ *Tek, çift tırnak?*

```
ifade= "1 2 3 4 5 #{2*3} 7 8 9 10"
```

```
"1 2 3 4 5 6 7 8 9 10"
```

```
ifade= '1 2 3 4 5 #{2*3} 7 8 9 10'
```

```
"1 2 3 4 5 \#{2*3} 7 8 9 10"
```

## ◆ *Birleştirme işlemi;*

```
sozcuk = " ozgur "+" yazilim "
```

```
" ozgur yazilim "
```

## ◆ *Tekrarlatma işlemi;*

```
sozcuk= sozcuk*2
```

```
" ozgur yazilim ozgur yazilim "
```

## Basit Kod Örnekleri 3 - Dizgeler 2

### ♦ *Karakter seçimi;*

```
kelime="ruby"
```

```
kelime[0]
```

```
114      # r harfinin ASCII kodu
```

```
kelime[-1]
```

```
121      # y harfinin ASCII kodu
```

### ♦ *Eşitlik kontrolü;*

```
kelime= "ruby"
```

```
"ruby" == "ruby"
```

```
true
```

```
"ruby" == "perl"
```

```
false
```

# Basit Kod Örnekleri 4 - Dizgeler 3

## ◆ *Aldizge seçimi;*

```
kelime="ruby"
```

```
kelime[0..2]
```

```
"rub"
```

```
kelime[0,1]
```

```
"r"
```

```
kelime[-2,2]
```

```
"by"
```

## Basit Kod Örnekleri 5 - Düzenli İfadeler

- ◆ Düzenli ifadeler, bir dizgenin verilen şablona uyup uymadığını bulmaya yarar,

Ruby'de düzenli ifadeler (*regex*) ters bölü işaretleri arasında yazılır,

```
def hex(s)
  (s =~ /<0(x|X)(\d|[a-f][A-F])+>/) != nil
end
```

< > işaretleri  
arasında, onaltılık  
sayı olup olmadığını  
kontrol edeceğiz

```
hex "Bu değil."
```

```
false
```

```
hex "Belki bu? {0x35}"
```

```
false
```

```
hex "Ya da bu? <0x38z7e>"
```

```
false
```

```
hex "Tamam, bu: <0xfc0004>."
```

```
true
```

## Basit Kod Örnekleri 6 - Diziler 1

- ◆ Ruby'de dizilere farklı türden nesnelere atayabilirsiniz,

```
dizi=[1.9,3,"ruby"]  
[1.9,3 "ruby"]
```

- ◆ Dizileri de , aynı dizgeler gibi birleştirebilir, ya da tekrar ettirebilirsiniz,

```
dizi + ["programlama", "dili"]  
[1.9,3 "ruby", "programlama", "dili"]
```

```
dizi*2  
[1.9,3 "ruby",1.9,3,"ruby"]
```

## Basit Kod Örnekleri 7 - Diziler 2

- ◆ Ruby'de dizileri `to_s` metoduyla dizgeye, dizgeleri de `to_a` metoduyla diziye dönüştürebiliriz;

```
dizi = ["12",21,"ruby"]
```

```
dizi.to_s
```

```
"1221ruby"
```

alternatif; *join*

```
dizge= dizi.join(":")
```

```
"12:21:ruby"
```

```
dizge.to_a
```

```
["12:21:ruby"]
```

alternatif; *split*

```
dizge.split(":")
```

```
["12", "21", "ruby"]
```

## Basit Kod Örnekleri 8 - Denetim Yapıları

### ◆ `case;`

```
i=8
case i
when 1,2..5
    print "1..5\n"
when 6..10
    print "6..10\n"
end
```

*Çıktı;*  
6..10

```
case 'abcdef'
    when 'aaa', 'bbb'
        print "aaa or bbb\n"
    when /def/
        print "/def/ icerir\n"
end
```

*Çıktı;*  
/def/ icerir

## Basit Kod Örnekleri 9 - Denetim Yapıları 2

### ◆ while;

```
sayac = 0

while satir = gets
  if satir =~ /Ruby/
    sayac += 1
  end
end

puts "#{sayac} Ruby satırı var"
```

*Girdi;*

Bu birinci Ruby satırı

Bu ikinci

Bu da üçüncü Ruby satırı

^D

*Çıktı;*

2 Ruby satırı var

### ◆ for;

```
for i in [200,-3.6,"ruby"]
  print "#{i}\t(#{i.type})\n"
end
```

*Çıktı;*

200 (Fixnum)

-3.6 (Float)

ruby (String)

### ◆ *each? for?*

```
dizi =[200,-3.6,"ruby"]
dizi.each do |i|
  print i
end
```

*Çıktı;*

200-3.6ruby

## Basit Kod Örnekleri 10 - Sınıflar/Miras

### ◆ *Basit bir sınıf tanımı;*

```
class Memeli
  def nefes
    print "Nefes al, Nefes ver\n"
  end
end
```

```
pisi = Kedi.new
```

```
pisi.nefes
Nefes al, Nefes ver
```

```
pisi.konus
Miyauvvvv
```

### ◆ *Miras;*

```
class Kedi<Memeli
  def konus
    print "Miyav\n"
  end
end
```

## Basit Kod Örnekleri 10 - Miras 2

- ◆ Süper sınıf, alt sınıf..?

```
class Kus
  def gagala
    print "Tüylerimi temizliyorum."
  end
  def uc
    print "Uçuyorum."
  end
end
```

```
class Penguen<Kus
  def uc
    fail "Üzgünüm, yüzmeyi tercih ederim."
  end
end
```

```
tux= Penguen.new
```

```
tux.uc
```

RuntimeError: Üzgünüm,  
yüzmeyi tercih ederim.

## Basit Kod Örnekleri 10 - Modüller/Mixin

### *Modüller;*

Modülün örneği, altsınıfı yoktur,  
module ... end şeklinde tanımlanır,

### *Mix-in;*

```
class Geo
  include Trig
  # ....
end
```

```
module Trig
  PI = 3.141592654
  def Trig.sin(x)
    # ..
  end
  def Trig.cos(x)
    # ..
  end
end
```

# İnteraktif Ruby - irb

*irb; Ruby ifadelerini stdin'den okuyarak anlık çalıştırmaya ve sonuçları görmenize yarayan bir araçtır;*

```
moon@debian:~$ irb
irb(main):001:0> "ruby "*3
=> "ruby ruby ruby "
irb(main):002:0> Fixnum.superclass
=> Integer
irb(main):003:0> 10.methods
=> ["%", "upto", "<<", "div", "&", "object_id", ">>", "times", "singleton_methods",
"equal?", "taint", "id2name", "*", "succ", "frozen?", "instance_variable_get", "+",
"kind_of?", "round", "to_a", "respond_to?", "-", "divmod", "integer?", "chr", "/",
"type", "protected_methods", "to_sym", "|", "eql?", "instance_variable_set", "~",
"hash", "is_a?", "truncate", "between?", "to_s", "send", "prec", "modulo",
"singleton_method_added", "class", "size", "zero?", "tainted?", "private_methods",
"__send__", "^", "untaint", "+@", "next", "-@", "id", "**", "step", "to_i", "<",
"inspect", "<=>", "method", "instance_eval", "==", "prec_i", "remainder", ">", "===",
"nonzero?", "clone", "public_methods", "floor", "extend", ">=", "<=", "freeze",
"display", "quo", "downto", "to_f", "__id__", "=~", "methods", "prec_f", "abs", "nil?",
"dup", "to_int", "coerce", "instance_variables", "[]", "instance_of?", "ceil"]
```

# Kaynaklar

◆ Ruby Kullanıcı Kılavuzu

<http://docs.comu.edu.tr/howto/ruby-ug.html>

◆ Programming Ruby

<http://www.ruby-doc.org/docs/ProgrammingRuby/>

◆ Ruby Ana Sayfası

<http://www.ruby-lang.org/en/>

◆ Matz'ın Seminerleri

<http://www.rubyist.net/~matz/slides/>

◆ Ruby Garden

<http://www.rubygarden.com/ruby?RealWorldRuby>

◆ Programlama Dili Kıyaslamaları

<http://www.jvoegele.com/software/langcomp.html>

◆ Ruby FAQ

<http://dev.rubycentral.com/faq/rubyfaq.html>