



<http://technotz.info/stuffs/Oct12/Article14/img/img.png>

Ben Borchard

# Basic Program

```
#helloworld program  
  
puts "hello world"  
print "I am "  
print "a ruby program\n"
```



```
C:\Users\Ben Borchard\Desktop\rubyprograms\Presentation>ruby HelloWorld.rb  
hello world  
I am a ruby program
```

# Running Ruby Programs

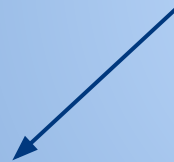
`ruby <filename>`

```
C:\Users\Ben Borchard\Desktop\rubyprograms\Presentation>ruby Conditionals.rb
```



# Strings

```
#Strings
puts "Conca" + "tena" << "tion"
puts 'single quotes'
puts 'don\'t end the quote at the apostraphe'
puts 'line \n space, maybe not...'
name = "Ben"
puts "my name is #{name}"
"abcde".each_char{|x| print x + " "}
puts
puts "abcde".index('d')
puts 5
```



```
Concatenation
Single quotes
don't end the quote at the apostraphe
line \n space, maybe not...
my name is Ben
a b c d e
3
5
```

# Functions

```
def simple
  a=5
end

puts simple

puts '-----'

def function(a)
  puts a
end

function("Hey there")

def function()
  puts "Snake Charmer"
end
```

```
function()

#code below doesn't work, too few arguments
#function("Charm a snake")

puts "-----"

def function2(*nums)
  nums
end

print function2(1)
puts
print function2(1,2,3,4)
puts

puts "-----"
def song1(a, b)
  print "I like to "
  a.call(b)
end

song2 = Proc.new {|a| puts "eat, eat, eat" + a}
song1(song2,"apples and bananas")
```

```
5
-----
Hey there
Snake Charmer
-----
[1]
[1, 2, 3, 4]
-----
I like to eat, eat, eatapples and bananas
```

# Loops

```
4.times{|i| print i}
puts

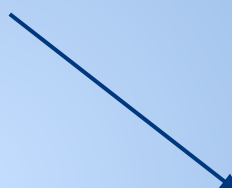
for i in 0..4
  print i
end
puts

(0..5).each{|i| print i}
puts

i=0
while i < 7 do
  print i
  i += 1
end
puts

i=0
until i == 8 do
  print i
  i+=1
end
puts

a = false
9.times do |i|
  print i
  if i == 4 && a == false
    print "\nredoing\n"
    a = true
    redo
  end
end
puts|
```



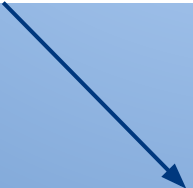
```
0123
01234
012345
0123456
01234567
01234
redoing
45678
```

# Conditionals

```
a = 4
if a = 4
  puts "a=4"
else
  puts "a!=4"
end

unless a == 5
  puts "a!=5"
end

case
|when a == 1
  puts "a=1"
when a == 2
  puts "a=2"
when a == 3
  puts "a=3"
else
  puts "a does not equal 1, 2, or 3"
end
```



```
a=4
a!=5
a does not equal 1, 2, or 3
```



# Arrays

```
#declaration
array1 = [1,2,"hello", 3,"world"]
array2 = Array.new(4){|indx| indx+1}

print array2, "\n"

#adding to array
array1 << 4
array1[7] = "insert"


print array1, "\n"

#removing from array
a = array1.pop(1)
array1.delete("hello")

print array1, "\n"

#mapping
print array2.collect{|x| x*3}, "\n"

#concatenation
print array1+array2, "\n"
```



```
[1, 2, 3, 4]
[1, 2, "hello", 3, "world", 4, nil, "insert"]
[1, 2, 3, "world", 4, nil]
[3, 6, 9, 12]
[1, 2, 3, 4]
[1, 2, 3, "world", 4, nil, 1, 2, 3, 4]
```

# Classes

```
#parent class
class Class1

  #class variable
  @@cvar = 3

  #initialization method
  def initialize(a, b, c)

    #object variables
    @string = a
    @int1 = b
    @int2 = c

  end

  #class methods
  def do_something
    @int1+@int2*@@cvar
  end

  def say_something
    @string
  end

end

#child class
class Class2 < Class1

  def say_something
    "overwritten"
  end

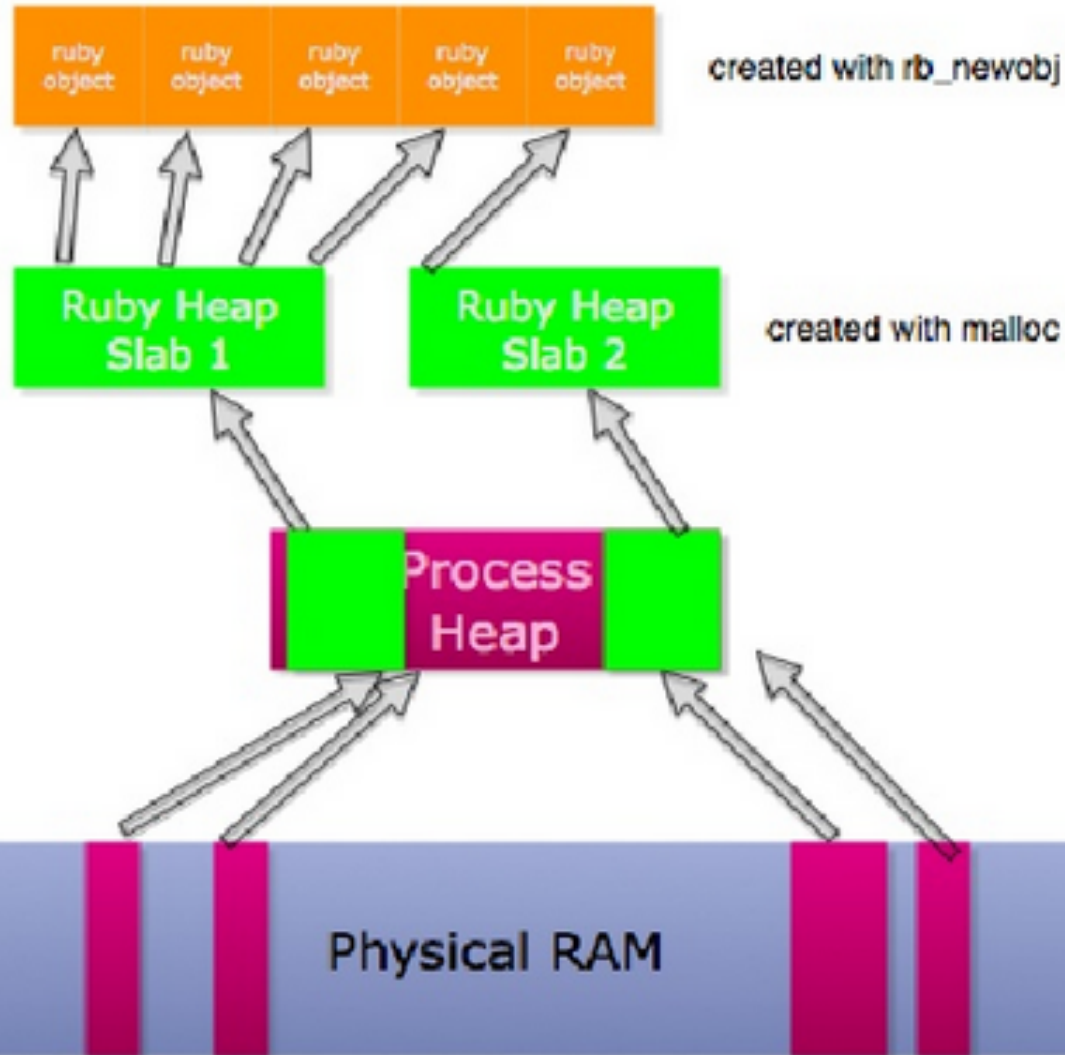
end

parent = Class1.new("Hello world", 1, 2)
child = Class2.new("useless", 4, 5)

puts parent.say_something
puts parent.do_something
puts '-----'
puts child.say_something
puts child.do_something
puts '-----'
puts Class2.superclass
```

```
Hello World
7
-----
Overwritten
19
-----
Class1
```

# Memory Management



# Ruby

vs

# Python

Slower

Many ways to do things

Many different syntactic options

No distinction between data and code

No distinction between statements and expressions

Faster

"One way to do it approach"

Strict syntax rules

Data and code stored separately in memory

Statements and expressions are different

Dynamic typing

Object oriented

Garbage Collected

High level

# Sources

<http://c2.com/cgi/wiki?PythonVsRuby>

<http://ruby.about.com/od/beginningruby/a/vspython.htm>

<http://stackoverflow.com/questions/234721/what-are-the-biggest-differences-between-python-and-ruby-from-a-philosophical-p>  
e

[http://www.tutorialspoint.com/ruby/ruby\\_loops.htm](http://www.tutorialspoint.com/ruby/ruby_loops.htm)

<http://www.ruby-doc.org/core-1.9.3/String.html>

<http://www.ruby-doc.org/core-1.9.3/Hash.html>

<http://www.ruby-doc.org/core-1.9.3/Array.html>

<http://www.ruby-doc.org/core-1.9.3/Class.html>

<http://www.skorks.com/2010/05/ruby-procs-and-lambdas-and-the-difference-between-them/>

<http://rubylearning.com/satishtalim/tutorial.html>

<http://net.tutsplus.com/tutorials/ruby/ruby-for-newbies-conditional-statements-and-loops/>