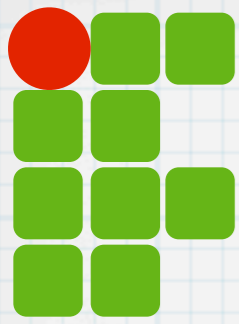


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação de Computadores

Primeiro programa

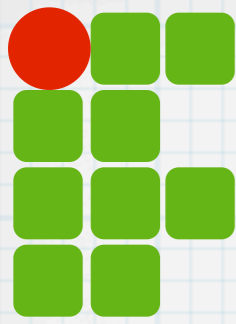
Copyright © 2012 IFRN



O que veremos hoje?

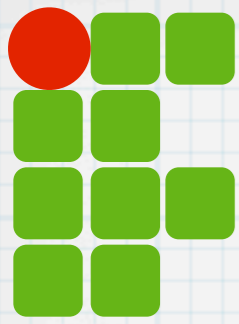
- * Sequenciamento de instruções
- * Mostrando valores
- * Criação de um programa
- * Execução
- * Lendo valores
 - * a instrução gets
- * Mais strings
 - * Expressões em string
- * Formatação de números





Instruções

- * Um computador é capaz de entender/calcular uma instrução/expressão
- * O resultado pode ser associado a uma variável
- * As expressões/instruções podem manipular valores
 - * Valores possuem um tipo
 - * Vimos inteiros, reais e textos (strings)
- * Expressões complexas podem ser divididas e sequencializadas para facilitar o entendimento



Resolução de problemas

* Equação do segundo grau

- * Qual(is) o(s) valor(es) de x ?
- * Precisamos dos valores de a , b e c

* Passos

1. Saber os valores de a , b e c
2. Calcular delta ($b^2 - 4 * a * c$)
3. Calcular raiz de delta (r_delta)
4. Calcular $x1 = (-b + r_delta) / (2 * a)$
5. Calcular $x2 = (-b - r_delta) / (2 * a)$

$$a=2$$

$$b=5$$

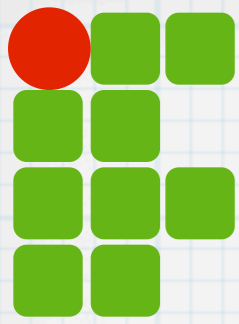
$$c=2$$

$$delta = 9$$

$$r_delta = 3$$

$$x1 = (-5 + 3) / (2 * 2)$$

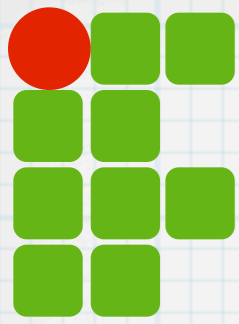
$$x2 = (-5 - 3) / (2 * 2)$$



Programa de computador

- * Sequência de instruções
- * Arquivo texto contendo as instruções
- * Para ruby sugestão de usar extensão .rb
calcula_media.rb

```
nota1 = 8.3
nota2 = 9.2
media = (nota1*2+nota2*3)/5
puts "A media e" + media.to_s
```

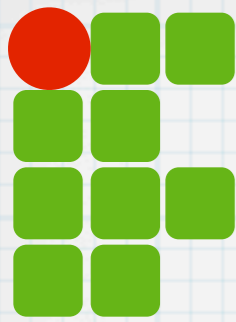


Programa de computador

- * Seqüência de instruções
- * Arquivo texto contendo as instruções
- * Para ruby sugestão de usar extensão .rb
calcula_media.rb

```
nota1 = 8.3  
nota2 = 9.2  
media = (nota1*2+nota2*3)/5  
puts "A media e" + media.to_s
```

Será visto a seguir



Editor de textos

* Arquivo contendo apenas texto

* NÃO a: rtf, doc, docs, pages, odf, etc

* O que usar?

* Bloco de notas (Win)

* gedit (Linux)

* nano (Linux)

* Pico (Linux)

* Vi, vim (Unix, linux)

* notepad++ (windows)

* SciTE (windows)

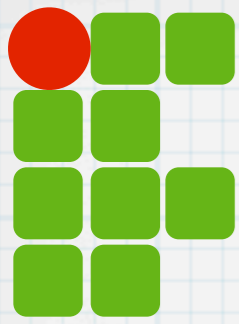
* emacs (win, linux, etc)

* jEdit (win, linux, etc)

* Kate

* etc...





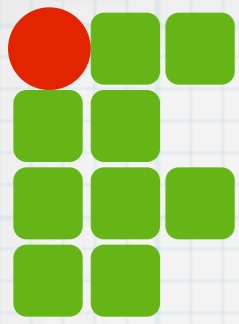
Execução de um programa

* Na linha de comando:

* `ruby NOME_ARQUIVO`

* onde `NOME_ARQUIVO` é o arquivo salvo.

```
tmp — bash — 51x7
Jorgiano:tmp jorgiano$ ruby calcula_media.rb
A media e 8.84
Jorgiano:tmp jorgiano$
```

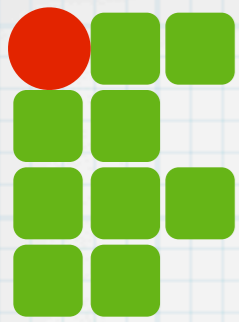
Execução de um programa

* Na linha de comando:

* ruby **NOME_ARQUIVO**

* onde **NOME_ARQUIVO** é o arquivo salvo.

```
tmp — bash — 51x7
Jorgiano:tmp jorgiano$ ruby calcula_media.rb
A media e 8.84
Jorgiano:tmp jorgiano$
```



Observações

* A execução do programa não mostra o resultado das operações

* no irb a linha

`media=(nota1*2+nota2*3)/5` →
mostra o resultado do cálculo

```
tmp -- ruby -- 51x7
irb(main):001:0> nota1=8.3
=> 8.3
irb(main):002:0> nota2=9.2
=> 9.2
irb(main):003:0> media=(nota1*2+nota2*3)/5
=> 8.84
irb(main):004:0>
```

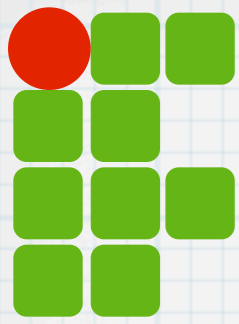
* no programa nada é mostrado

* Para mostrar algo deve-se usar as operações `print` ou `puts`

* A execução do programa só “imprime na tela” o que foi explicitamente pedido

```
tmp -- bash -- 51x7
Jorgiano:tmp jorgiano$ ruby calcula_media.rb
A media e 8.84
Jorgiano:tmp jorgiano$
```

* `puts` ou `print`



Mostrando valores

* Instruções

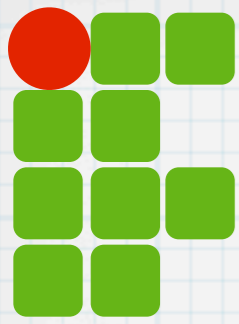
* puts

- * Acrescenta uma quebra de linha no final da impressão

* print

- * Não acrescenta o final de linha

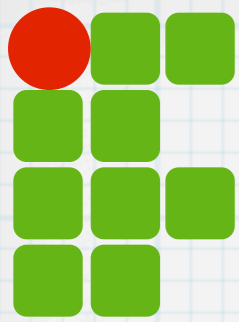
```
jorgiano — ruby — 40x6
irb(main):008:0> puts "Oi"
Oi
=> nil
irb(main):009:0> print "Oi"
Oi=> nil
irb(main):010:0>
```



Mostrando valores

* puts e print podem ser usados para mostrar qualquer tipo de valor

```
jorgiano — ruby — 40x15
irb(main):016:0> puts "Mostrar texto"
Mostrar texto
=> nil
irb(main):017:0> puts 10
10
=> nil
irb(main):018:0> puts 3.3
3.3
=> nil
irb(main):019:0> puts 10/3.3
3.0303030303030303
=> nil
irb(main):020:0> print 10+20+30+y
97=> nil
irb(main):021:0> █
```

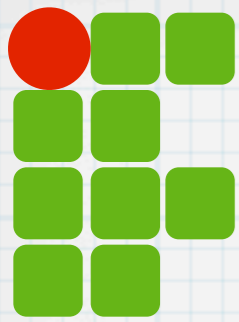


Mostrando valores

* puts e print podem ser usados para mostrar qualquer tipo de valor

```
jorgiano — ruby — 40x15
irb(main):016:0> puts "Mostrar texto"
Mostrar texto
=> nil
irb(main):017:0> puts 10
10
=> nil
irb(main):018:0> puts 3.3
3.3
=> nil
irb(main):019:0> puts 10/3.3
3.0303030303030303
=> nil
irb(main):020:0> print 10+20+30+y
97=> nil
irb(main):021:0> █
```

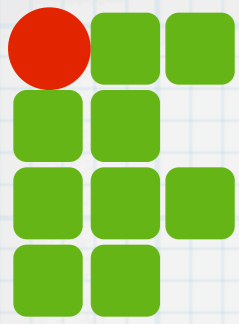
O uso de uma variável equivale ao uso do valor atribuído a ela



Mostrando valores

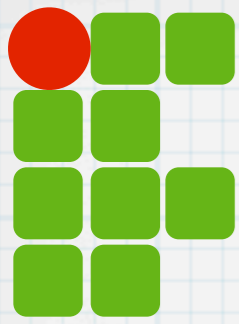
* ~~Conteúdo~~ de variáveis

```
jorgiano — ruby — 60x11
Jorgiano:~ jorgiano$ irb2
irb(main):001:0> nome = "Joaquim Jose da Silva Xavier"
=> "Joaquim Jose da Silva Xavier"
irb(main):002:0> puts nome
Joaquim Jose da Silva Xavier
=> nil
irb(main):003:0> print nome
Joaquim Jose da Silva Xavier=> nil
irb(main):004:0>
```



Entrada de dados

- * Nosso programa deve ser alterado para calcular médias de valores diferentes
 - * O que fazer para usar o mesmo programa para ~~qualquer~~ média de quaisquer notas?
- * Instrução de leitura de dados
 - * gets
 - * Ler uma linha como string
 - * Programa fica “parado” esperando o usuário digitar algo
 - * Programa continuar após usuário digitar [ENTER]

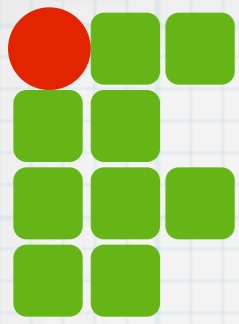


Entrada de dados - exemplo

```
ler_e_imprime.rb
```

```
x = gets  
puts x
```

```
tmp — bash — 51x7  
Jorgiano:tmp jorgiano$ ruby ler_e_imprime.rb  
Bom dia  
Bom dia  
Jorgiano:tmp jorgiano$ _
```

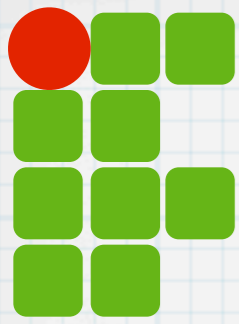
Entrada de dados - exemplo

```
ler_e_imprime.rb
```

```
x = gets
```

```
puts x
```

```
tmp -- bash -- 51x7
Jorgiano:tmp jorgiano$ ruby ler_e_imprime.rb
Bom dia
Bom dia
Jorgiano:tmp jorgiano$ _
```

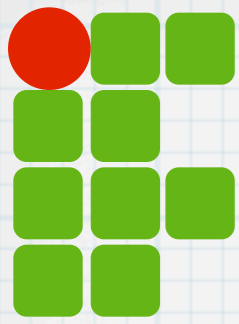


Entrada de dados - exemplo

tamanho_msg.rb

```
msg = gets
print msg
puts msg.size
```

```
tmp — bash — 51x7
Jorgiano:tmp jorgiano$ ruby2 tamanho_msg.rb
teste
teste
6
Jorgiano:tmp jorgiano$
```



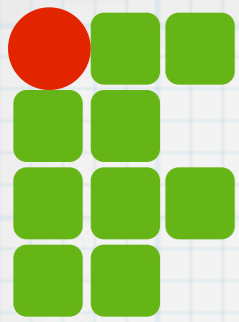
Entrada de dados - exemplo

tamanho_msg.rb

```
msg = gets  
print msg  
puts msg.size
```

Não
deveria
ser 5?

```
tmp — bash — 51x7  
Jorgiano:tmp jorgiano$ ruby2 tamanho_msg.rb  
teste  
teste  
6  
Jorgiano:tmp jorgiano$
```



Entrada de dados - exemplo

```
tamanho_msg.rb
```

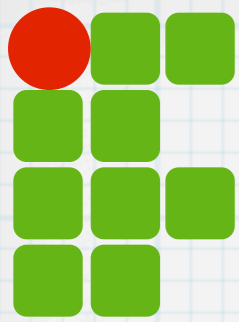
```
msg = gets  
print msg  
puts msg.size
```

O `gets` lê a linha inteira, incluindo a caractere de quebra de linha (CR/LF) ao final da linha.

Como eliminar esse caracter?

Não deveria ser 5?

```
tmp — bash — 51x7  
Jorgiano:tmp jorgiano$ ruby2 tamanho_msg.rb  
teste  
teste  
6  
Jorgiano:tmp jorgiano$
```



Entrada de dados

* Método chomp

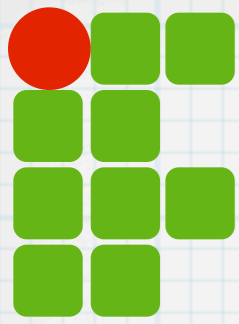
- * Retorna uma string sem o caractere de quebra de linha, caso ele exista.

tamanho_msg.rb

```
msg = gets.chomp  
print msg  
puts msg.size
```

Sem a
quebra de
linha

```
tmp — bash — 51x5  
Jorgiano:tmp jorgiano$ ruby2 tamanho_msg.rb  
teste  
teste5  
Jorgiano:tmp jorgiano$  
Jorgiano:tmp jorgiano$ _
```



Entrada de dados

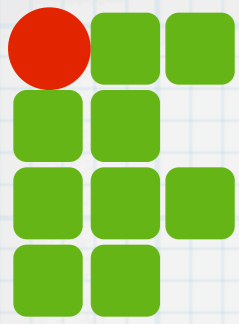
* Como ler inteiro e/ou real?

- * ~~Deve-se~~ converter com os métodos `to_i` OU `to_f`

media.rb

```
valor1 = gets.to_f
valor2 = gets.to_f
media = (valor1+valor2)/2
puts media
```

```
tmp — bash — 51x5
Jorgiano:tmp jorgiano$ ruby media.rb
8
7.2
7.6
Jorgiano:tmp jorgiano$
```



Entrada de dados

* Como ler inteiro e/ou real?

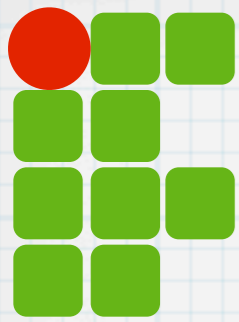
* Deve-se converter com os métodos `to_i` OU `to_f`

media.rb

```
valor1 = gets.to_f  
valor2 = gets.to_f  
media = (valor1+valor2)/2  
puts media
```

Não precisa
do `chomp`

```
tmp — bash — 51x5  
Jorgiano:tmp jorgiano$ ruby media.rb  
8  
7.2  
7.6  
Jorgiano:tmp jorgiano$
```



Mais sobre strings

* Uso de expressões em strings

* `{exp}`

* Apenas se delimitadas por aspas duplas ("")

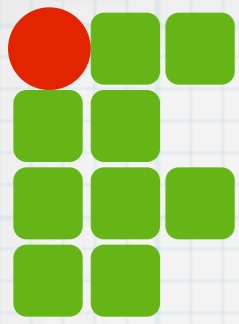
`oi.rb`

```
nome = gets.chomp  
puts "Oi #{nome}"
```

A expressão é avaliada e transformada em string.

`media.rb`

```
valor1 = gets.to_f  
valor2 = gets.to_f  
media = (valor1+valor2)/2  
puts "A sua media foi de #{media} esse ano!"
```

Mais sobre strings

* Concatenação

* com o +: cria uma nova string

```
saudacao = "Oi "  
nome = gets.chomp  
msg = saudacao + nome  
puts msg
```

saudacao

Oi

nome

Alfredo

msg

Oi Alfredo

* com o <<: Adiciona a segunda string ao final da primeira

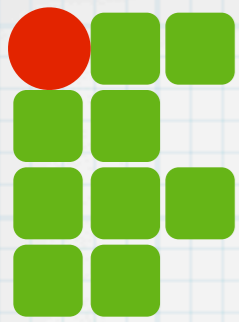
```
msg="Oi "  
nome = gets.chomp  
msg << nome  
puts msg
```

msg

Oi

nome

Alfredo



Mais sobre strings

* Concatenação

* com o +: cria uma nova string

```
saudacao = "Oi "  
nome = gets.chomp  
msg = saudacao + nome  
puts msg
```

saudacao

Oi

nome

Alfredo

msg

Oi Alfredo

* com o <<: Adiciona a segunda string ao final da primeira

```
msg="Oi "  
nome = gets.chomp  
msg << nome  
puts msg
```

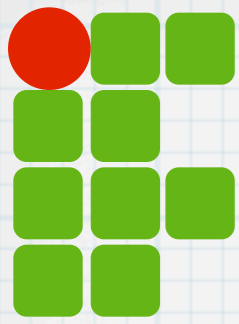
msg

Oi Alfredo

nome

Alfredo





Formatar números

* str % valor

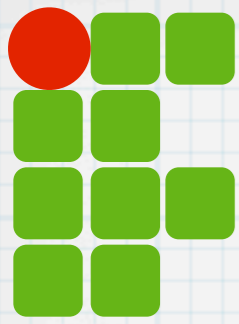
* Cria uma string convertendo “valor” no formato especificado por str

* Exemplo: `media_str = "%.2f" % media`

* Cria uma string que é a conversão do valor (real) da variável `media`, com duas casas decimais

* Exemplo: `seq_str = "%05d" % num`

* Cria uma string que é a conversão do valor (inteiro) da variável `num`, com pelo menos cinco casas, preenchendo com zeros as casas não usadas, no limite de 5

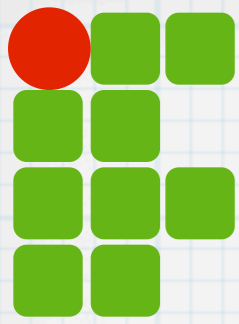


Formatar números

* Ejemplos

```
jorgiano — Desinstalador — ruby — 55x8
irb(main):007:0> media=7.213
=> 7.213
irb(main):008:0> media_str="%.2f" % media
=> "7.21"
irb(main):009:0> puts media_str
7.21
=> nil
irb(main):010:0>
```

```
jorgiano — Desinstalador — ruby — 55x8
irb(main):004:0> num=1
=> 1
irb(main):005:0> seq_str="%05d" % num
=> "00001"
irb(main):006:0> puts seq_str
00001
=> nil
irb(main):007:0>
```



Formatar números

* Sintaxe:

* `%[opções][tamanho][.precisão]tipo`

* Onde,

* **tipo é obrigatório e pode ser:**

* **inteiro:** b, B, d, i, o, u, x, X

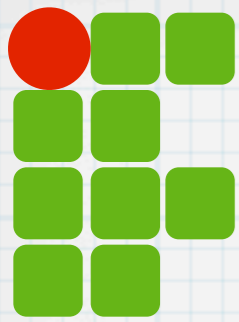
* **real (float):** e, E, f, g, G, a, A

* **outros:** c, p, s, %

* **opções, tamanho e precisão são opcionais e dependem do tipo**

* Informações

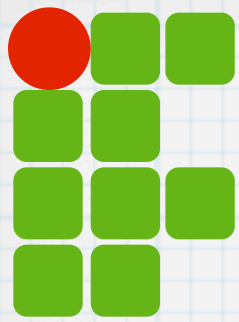
* <http://www.ruby-doc.org/core-1.9.3/Kernel.html#method-i-sprintf>



Formatar números

* Inteiro (d ou i)

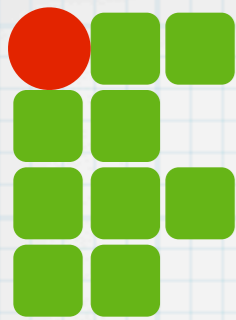
Expressão	resultado
“%d” % 123	“123”
“%5d” % 123	“ 123”
“%05d” % 123	“00123”
“%+d” % 123	“+123”
“%10.5d” % 123	“ 00123”
“%-10d” % 123	“123 ”
“%-10.5d” % 123	“00123 ”



Formatar números

* Real (f)

Expressão	resultado
“%f” % 123	“123.000000”
“%12f” % 123	“ 123.000000”
“%5.2f” % 123	“123.00”
“%5.2f” % (10.0/3)	“ 3.33”
“%.2f” % 324.3281	“324.33”
“%+10.2f” % (10.0/3)	“ +3.33”
“%+010.2f” % (10.0/3)	“+000003.33”



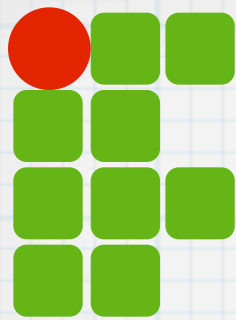
Exemplo

media.rb

```
nota1 = gets.to_f  
nota2 = gets.to_f  
media = (nota1*2+nota2*3)/5  
media_str = "%.1f" % media  
puts media_str
```

OBS:
8 e 9.1 são
valores digitados
pelo usuário

```
jorgiano — Desinstalador — bash — 44x7  
Jorgiano:~ jorgiano$ ruby media.rb  
8  
9.1  
8.7  
Jorgiano:~ jorgiano$ _
```

Exemplo

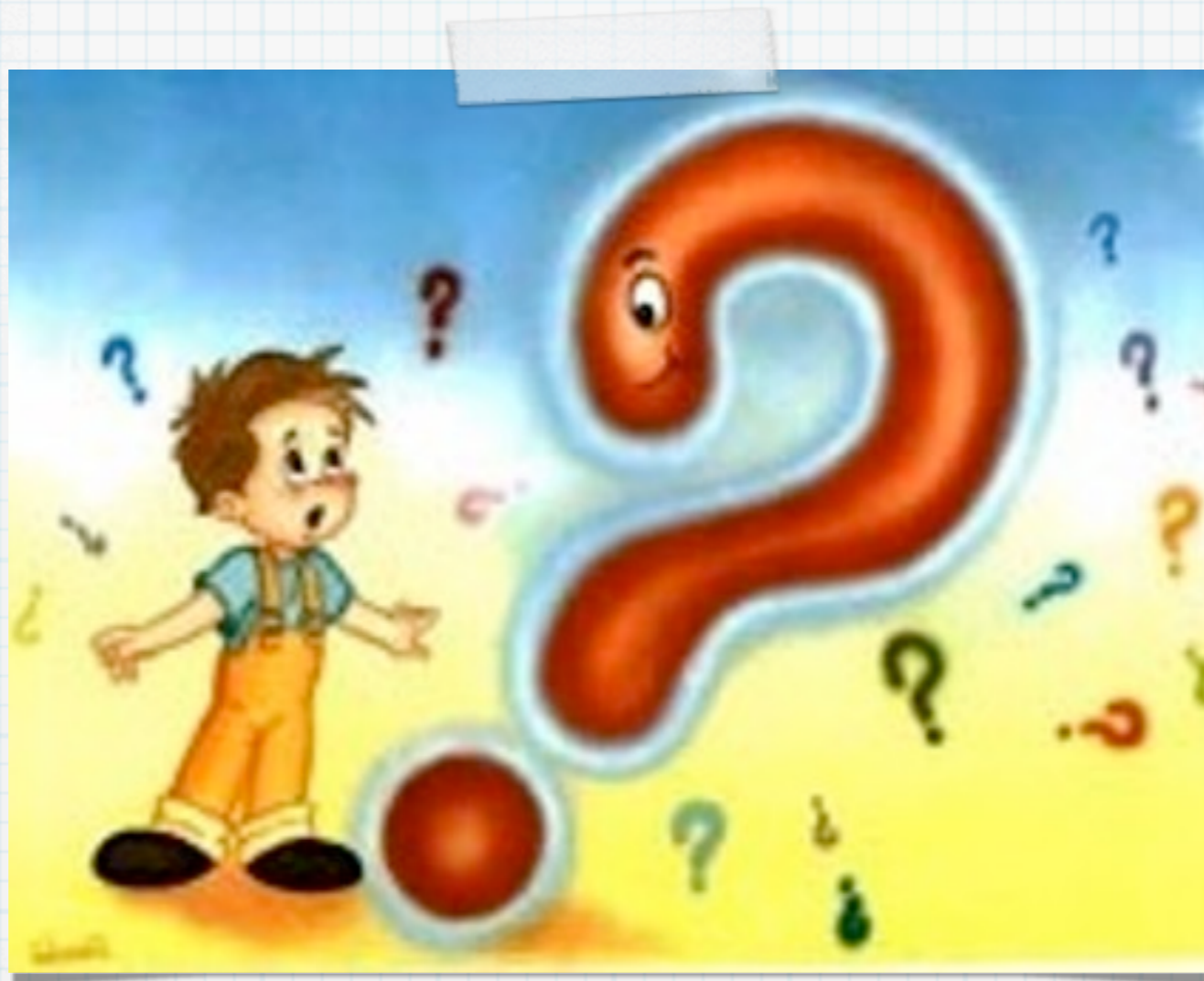
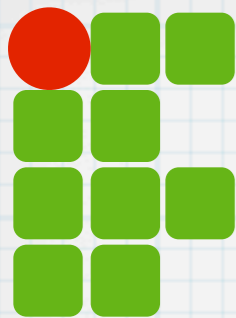
media.rb

```
nota1 = gets.to_f  
nota2 = gets.to_f  
media = (nota1*2+nota2*3)/5  
media_str = "%.1f" % media  
puts media_str
```

OBS:
8 e 9.1 são
valores digitados
pelo usuário

A média sem a
formatação é
8.66

```
jorgiano — Desinstalador — bash — 44x7  
Jorgiano:~ jorgiano$ ruby media.rb  
8  
9.1  
8.7  
Jorgiano:~ jorgiano$ _
```



Dúvidas?