Information Science
4: Arrays:
Understanding with Images

Naonori Kakimura
垣村尚徳
kakimura@global.c.u-tokyo.ac.jp

2016/10/24

PEAK
Programs in English at Komaba

Remarks on Previous Exercises

- Before submission,
  - check whether your program works correctly
    - By executing the program with specified parameters
      - Need careful writing
    - Read error messages to resolve the error
  - Common mistakes
    - Ex. 1/2, that would return 0
    - Ex. 2x, that would return error
  - Consider why it works or why it does not
    - You can discuss with your friends

Quiz 2-2: Computing the logarithms

- Transforming the base of log

```
def log_3(x)
        log10(x)/log10(3)
# can be replaced with
#       log(x)/log(3)
end

def log_b(x,b)
        log10(x)/log10(b)
# can be replaced with
#       log(x)/log(b)
end
```

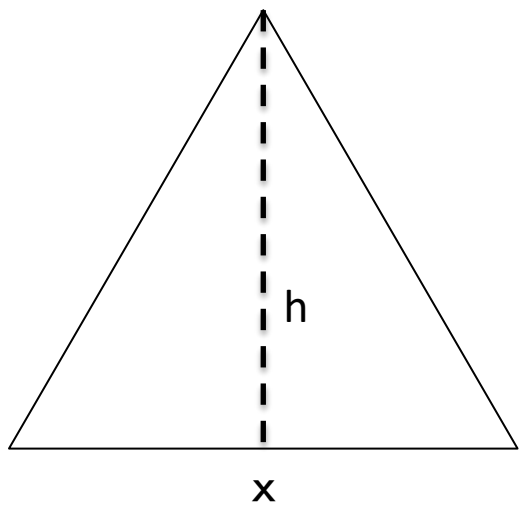
$$\log_a b = \frac{\log_x b}{\log_x a}$$

Consider why

Write carefully:
No log_10(x)
No log 10(x)

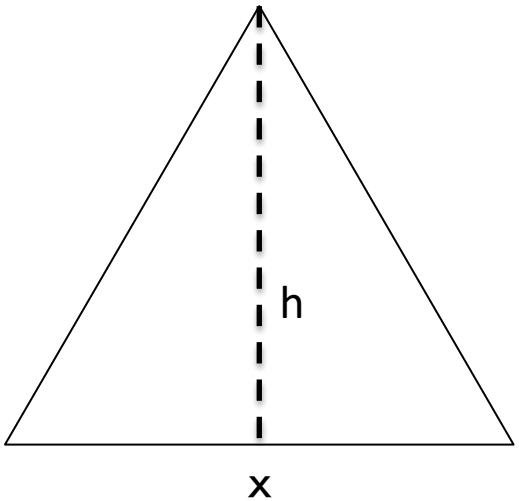
2-4: Computing the Triangle Area

- Area = 0.5*(base length x)*(height h)



2-4: Computing the Triangle Area

- Area = 0.5*(base length x)*(height h)



$$h = \sqrt{x^2 - \frac{x^2}{4}} = \frac{\sqrt{3}}{2}x^2$$

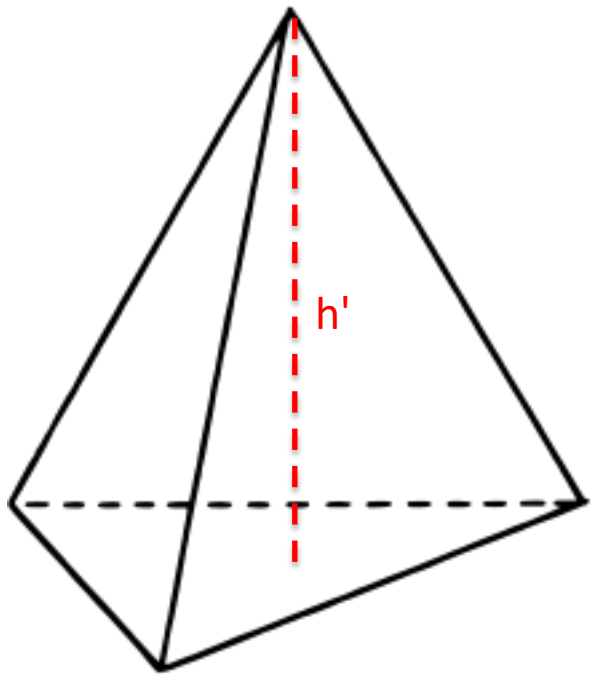
```
def triangle(x)
    sqrt(3.0)/4*x**2
end
```

Note:
1/2 returns 0 in Ruby
To obtain 0.5 type 1.0/2

Write carefully:
Don't forget *

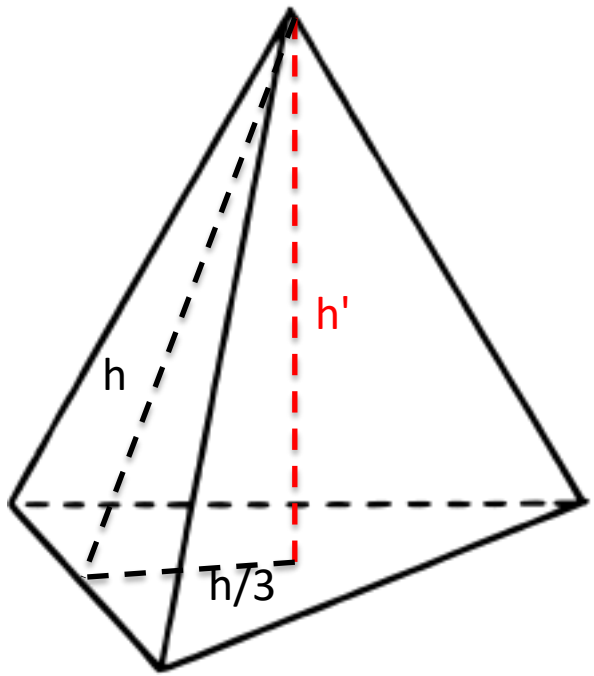
2-4:Computing the Volume of Tetrahedron

- Volume = 1/3*(base area)*(height h′)



2-4:Computing the Volume of Tetrahedron

- Volume = 1/3*(base area)*(height h′)



(base area) = triangle(x)

$$h' = \sqrt{\frac{2}{3}}x$$

$$\left(h = \sqrt{h^2 - \frac{h^2}{9}} = \frac{2\sqrt{2}}{3}h = \frac{2\sqrt{2}}{3}\frac{\sqrt{3}}{2}x^2 \right)$$

```
def tetrahedron(x)
    1.0/3*triangle(x)*sqrt(2.0/3)*x
end
```

Many other expressions

Review1: Conditional Processing

- Branching procedure according to a condition

```
if “condition”
   ”commands when the cond is satisfied"
else
   “commands when the cond is not satisfied"
end
```

```
def max(x,y)
    if y < x
        x
    else
        y
    end
end
```

max.rb

Review2: Repetitive Processing: WHILE

- "While ～, repeat the following"
  - While the condition "～" holds, do the commands between "while" and "end"

```
while CONDITION
    "COMMANDS"
end
```

Easy to understand if we put an indent(space) at the head of line

```
x = 5
while x >= 1
   print x
   print " hello!¥n"
   x = x-1
end
```

OUTPUT:

```
5 hello!
4 hello!
3 hello!
2 hello!
1 hello!
```

Review3: Repetitive Processing: FOR

- Repeat 100 times

```
for i in 1..100
    "COMMANDS"
end
```

i is a variable
(OK to have another name)
i changes from 1 to 100
during the repetition

```
for j in 1..5
  print j
  print " Hello!¥n"
end
```

OUTPUT:
1 Hello!
2 Hello!
3 Hello!
4 Hello!
5 Hello!

Today's Contents

- Array
  - a sequence of variables
  - Preliminaries: use irsb instead of irb
  - Using Array in Ruby
  - 2-dimensional array
  - Color image
- Draw an image using repetition and condition
- Exercises
- Appendix: see when you have time

Use “isrb” only to Make an Image

- isrb: irb with visualization function “show”
  - used only if we make an image
    - today and next week

- (If necessary) See the PDF file on ITC-LMS on how to install isrb to your PC/Mac

Similar Appearance to irb

Prompt of isrb

```
cm12345$ isrb
>> a = 3
=> 3
```

Today's Contents

- Array
  - a sequence of variables
  - Preliminaries: use irsb instead of irb
  - Using Array in Ruby
  - 2-dimensional array
  - Color image
- Draw an image using repetition and condition
- Exercise
- Appendix: see when you have time

Array - 1

a sequence of variables

Define an array

```
irb(main):001:0> a = [1, 7, 3, 2]
=> [1, 7, 3, 2]
irb(main):002:0> a
=> [1, 7, 3, 2]
irb(main):003:0> a[0]
=> 1
irb(main):004:0> a[1]
=> 7
irb(main):005:0> a[3]
=> 2
```

0 1 2 3 ← index

Display the array a

Display
the 0th entry of array a

The index starts from 0

Array - 2


```
irb(main):006:0> a.length()
=> 4
irb(main):007:0> a[4]
=> nil
irb(main):008:0> a[-1]
=> 2
irb(main):009:0> a[-4]
=> 1
irb(main):010:0> a[-5]
=> nil
```

Return the length of the array a

There is no a[4]

= [a.length()-1]=a[3]

Special in Ruby

= [a.length()-4]=a[0]

index from the back

What is the Result?


```
irb(main):001:0> a = [3,1,4,1,5,9]
=> [3, 1, 4, 1, 5, 9]
irb(main):002:0> a.length()
```

1. nil
2. 1
3. 6
4. 9
5. [3, 1, 4, 1, 5, 9]

What is the Result?


```
irb(main):001:0> a = [3,1,4,1,5,9]
=> [3, 1, 4, 1, 5, 9]
irb(main):002:0> a.length()
```

1. nil
2. 1
3. 6 ← Answer
4. 9
5. [3, 1, 4, 1, 5, 9]

What is the Result?


```
iirb(main):001:0> a = [3,1,4,1,5,9]
rb(main):003:0> a[0] = a[4]
=> 5
irb(main):004:0> a[0]+a[2]
```

1. nil
2. 1
3. 6
4. 8
5. 7
6. [3, 1, 4, 1, 5, 9]

What is the Result?


```
iirb(main):001:0> a = [3,1,4,1,5,9]
rb(main):003:0> a[0] = a[4]
=> 5
irb(main):004:0> a[0]+a[2]
```

1. nil
2. 1
3. 6
4. 8
5. 9 ← Answer
6. [3, 1, 4, 1, 5, 9]

Make an Array with a Given Size


```
irb(main):001:0> image = Array.new(6)
=> [nil , nil , nil , nil , nil , nil]
```

Create a new array with size 6

```
irb(main):001:0> a = Array.new(3)
=> [nil , nil , nil]
```

All the entries are “nil”(nothing)

Change Each Entry of the array

Use repetition to change all the entries in an array

```
irb(main):001:0> for i in 0..2
irb(main):001:1>     a[i] = 0
irb(main):001:2> end
=> 0..2
irb(main):001:0> a
=> [0, 0, 0]
```

Examples of Repetitions and Arrays

- Define a function make1d(n) that makes a 1-dimensional array with size n, each of whose entry is 0.

```
def make1d(n)
  a = Array.new(n)
  for i in 0..(n-1)
    a[i] = 0
  end
  a
end
```

1. Make an empty array
2. For each index i (from 0 to n-1)
   assign 0 to a[i]
3. Return the array a

You can download make1d.rb from ITC-LMS

Today's Contents

- Array
  - a sequence of variables
  - Preliminaries: use irsb instead of irb
  - Using Array in Ruby
  - 2-dimensional array and image
  - Color image
- Draw an image using repetition and condition
- Exercises
- Appendix: see when you have time

Make a 2-dimensional array

An array of an array

```
>> a = [[1,2,3,4], [10,20,30,40]]
>> a[0][0]
=> 1
```

Display the (0,0) entry

```
>> a[1][2]
=> 30
```

Display the (1,2) entry
(Row is first, Column is second)

```
>> a[2][1]
=> (ERROR)
```

No (2,1) entry
(Row is first, Column is second)

Like a matrix

2nd indices

0 1 2 3

0	1	2	3	4
1	10	20	30	40

1st indices

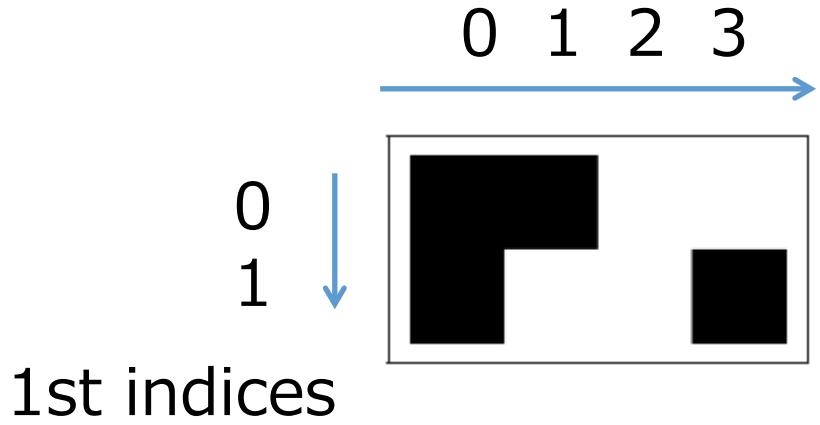
ISRB: Representing Data with Images


```
irb(main):002:0> ctrl D
cm12345$ isrb
>> a = [[0,0,1,1],
?>        [0,1,1,0]]
=> [[0 , 0, 1, 1], [0, 1, 1, 0]]

>> show(a)
=> nil
```

← Keyboard typing

Entries have to be between 0 and 1

Available only in isrb



0: Black

1: White

Modifying Images

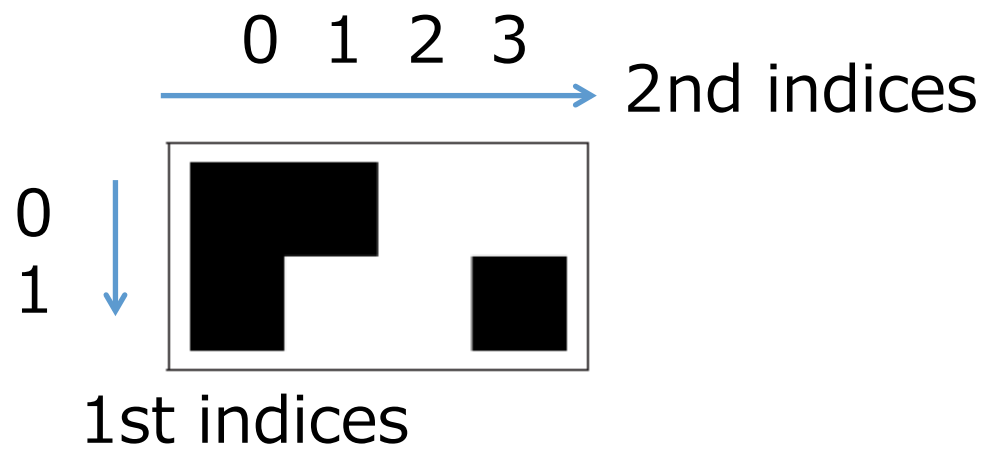

```
>> a[0][0]
=> 0
>> a[0][2]
=> 1

>> a[1][2]=0.5
=> 0.5
>> show(a)
=> nil
```

Display the brightness at the position (0,0)

Display the brightness at the position (0,2)

Change the brightness at the position (2,1)

Re-display



Exercise

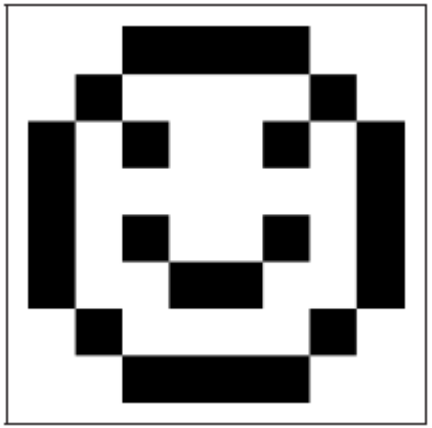
- Make the following data, and display it as an image

```
w = [[0 ,1 ,1 ,1 ,1 ,1] ,
     [0 ,1 ,0 ,0 ,0 ,1] ,
     [0 ,1 ,0 ,1 ,0 ,1] ,
     [0 ,1 ,1 ,1 ,0 ,1] ,
     [0 ,0 ,0 ,0 ,0 ,1]]
```

Cf) Exercise 1 (If you have time)

- Make the following data, and display it as an image

```
w = [[1 ,1 ,0 ,0 ,0 ,0 ,1 ,1] ,
     [1 ,0 ,1 ,1 ,1 ,1 ,0 ,1] ,
     [0 ,1 ,0 ,1 ,1 ,0 ,1 ,0] ,
     [0 ,1 ,1 ,1 ,1 ,1 ,1 ,0] ,
     [0 ,1 ,0 ,1 ,1 ,0 ,1 ,0] ,
     [0 ,1 ,1 ,0 ,0 ,1 ,1 ,0] ,
     [1 ,0 ,1 ,1 ,1 ,1 ,0 ,1] ,
     [1 ,1 ,0 ,0 ,0 ,0 ,1 ,1] ]
```



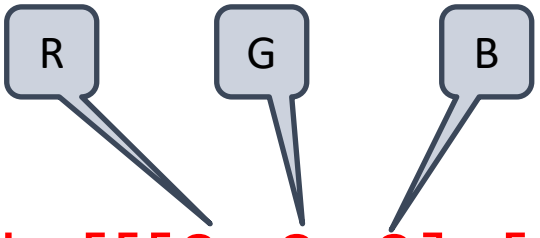
Today's Contents

- Array
  - a sequence of variables
  - Preliminaries: use irsb instead of irb
  - Using Array in Ruby
  - 2-dimensional array and images
  - Color image
- Draw an image using repetition and condition
- Exercises
- Appendix: see when you have time

Representing a Color Image – 3-dim array

An array of an array of an array

2-dim array, each of whose entry has 3 values(an array)




```
>> d=[[[0 ,0 ,0] ,[0 ,1 ,0] ,[0 ,0 ,1]] ,
?>      [[1 ,0 ,0] ,[1 ,1 ,0] ,[1 ,0 ,1]]]
=> [[[0 , 0, 0], [0, 1, 0], [0, 0, 1]] , [[1 , 0, 0],
[1, 1, 0], [1, 0, 1]]]
>> show (d)
=> nil
```

Example

- Easier to understand if we use a local variable representing colors

```
def flag()
   r=[1,0,0]
   w=[1,1,1]
   [[r,r,r,r,r],
    [r,r,w,r,r],
    [r,w,w,w,r],
    [r,r,w,r,r],
    [r,r,r,r,r]]
end
a = flag()
show(a)
```

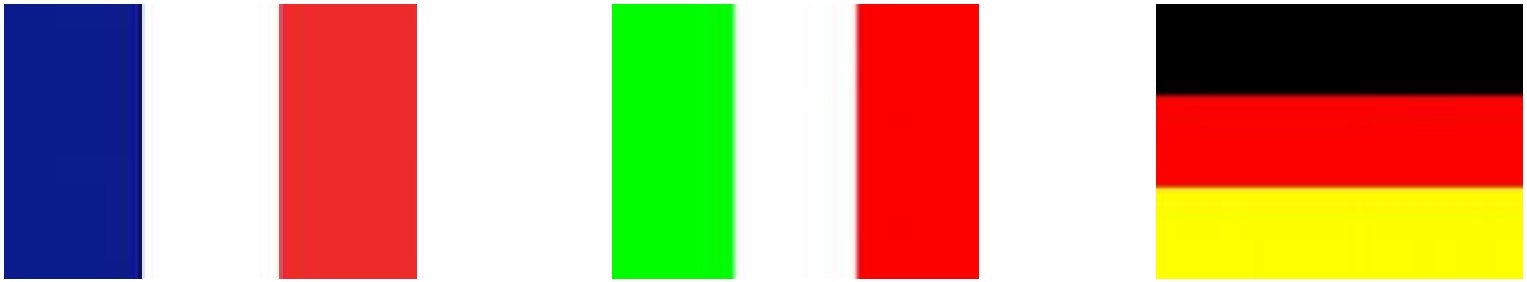
Define color red

Define color white

Return a 3-dim array

Exercise 1 (we will have some time later)

- Draw national flags using color image
  - Ex. France, Italy, Germany…
  - Refer to color information in the table



Color	Green	White	Red	Orange	Blue
Red	0	1	1	1	0
Green	0.6	1	0	0.4	0.2
Blue	0	1	0	0	0.6

Today's Contents

- Array
  - a sequence of variables
  - Preliminaries: use irsb instead of irb
  - Using Array in Ruby
  - 2-dimensional array and images
  - Color image
- Draw an image using repetition and condition
- Exercises
- Appendix: see when you have time

Program: Diagonal Line

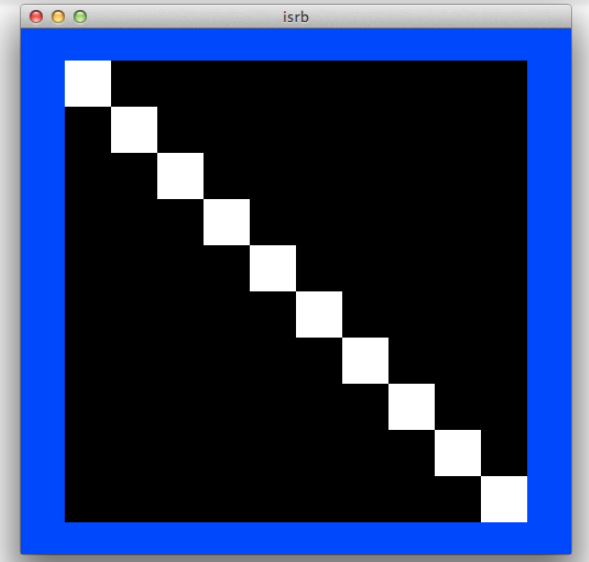

```
def diagonal(s)
    image = make2d(s, s)
    for x in 0..(s-1)
        image[x][x] = 1
    end
    image
end
```

Make a 2-dim array
all of whose entries are 0

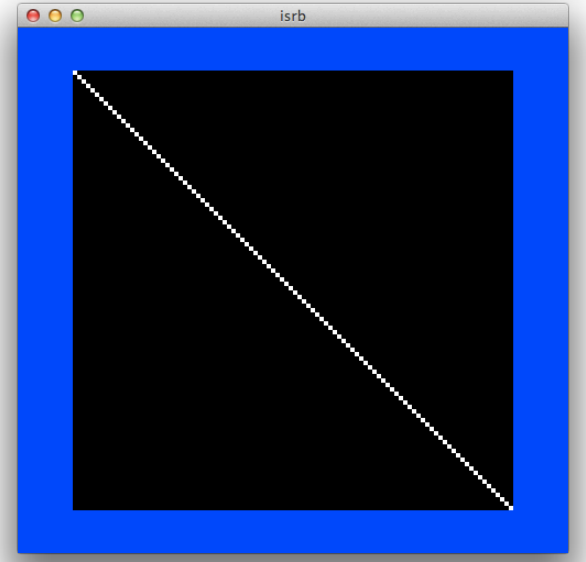
The "for"-part can be expanded to
- image[0][0] = 1
- image[1][1] = 1
- image[2][2] = 1
- . . . . .
- image[s-1][s-1] = 1

Examples

- show(diagonal(10))



- show(diagonal(100))



Make2d(h,w): Making a 2-dim array

- Make a 2-dimensional array with h rows and w columns, each of whose entry is 0.
  - You can download it, and see how it works

```
>> make2d(2,3)
=> [[0, 0, 0], [0, 0, 0]]
>> make2d(4,4)
=> [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

Program: Gradation

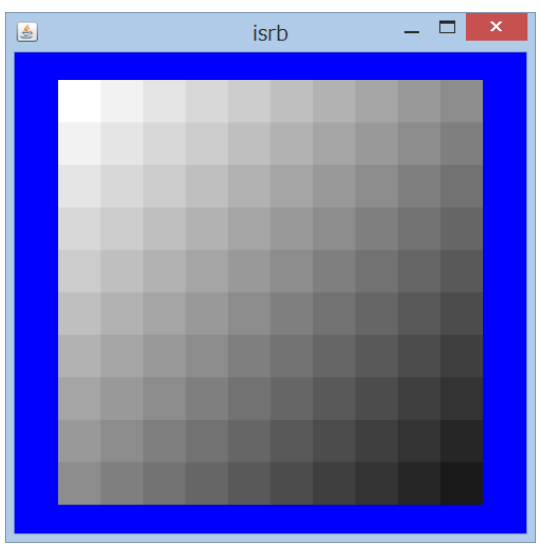

```
def gradation(s)
    image = make2d(s, s)
    for y in 0..(s-1)
        for x in 0..(s-1)
            image[y][x] = (2.0*s – x - y)/(2*s)
        end
    end
    image
end
```

Make a 2-dim array
all of whose entries are 0

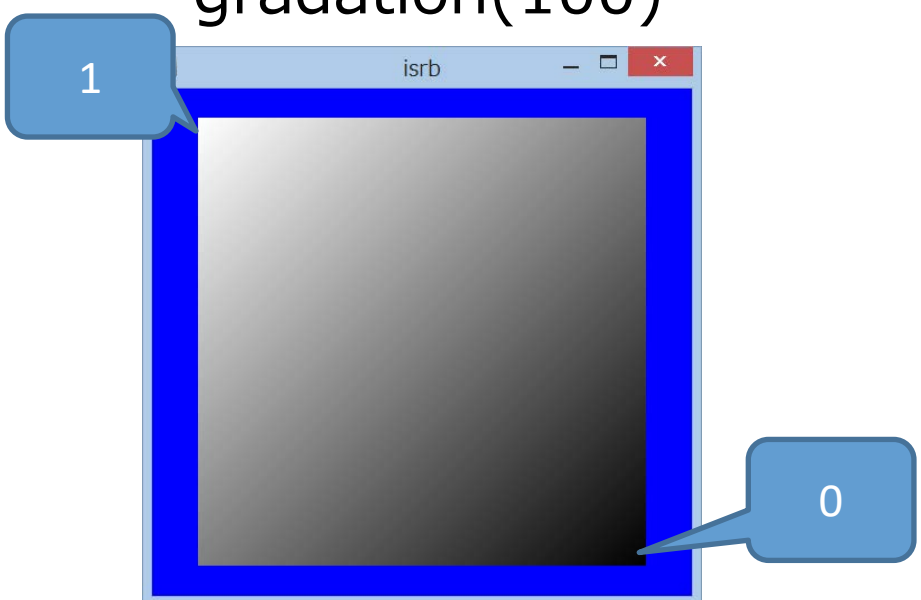
For each entry,
determine the brightness

```
def gradation(s)
    image = make2d(s, s)
    for y in 0..(s-1)
        for x in 0..(s-1)
            image[y][x] = (2.0*s – x - y)/(2*s)
        end
    end
    image
end
```

gradation(10)



gradation(100)



```
def gradation(s)
    image = make2d(s, s)
    for y in 0..(s-1)
        for x in 0..(s-1)
            image[y][x] = (2.0*s – x - y)/(2*s)
        end
    end
    image
end
```

```
image[0][0]=1.0
image[0][1]=0.8333333333333334
image[0][2]=0.6666666666666666
image[1][0]=0.8333333333333334
image[1][1]=0.6666666666666666
image[1][2]=0.5
image[2][0]=0.6666666666666666
image[2][1]=0.5
image[2][2]=0.3333333333333333
```

when y=0

Change x from 0 to 2

when y=1

Change x from 0 to 2

when y=2

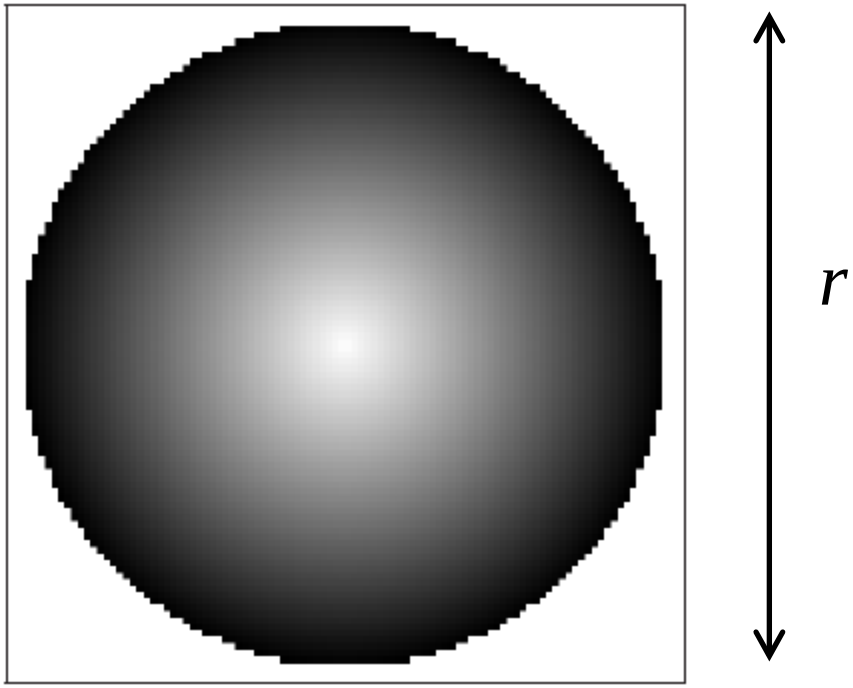
Change x from 0 to 2

…

Today's Exercise during the session

Making the image below by using "Repetition"

Size is r



A point outside the circle(distance > r) is white

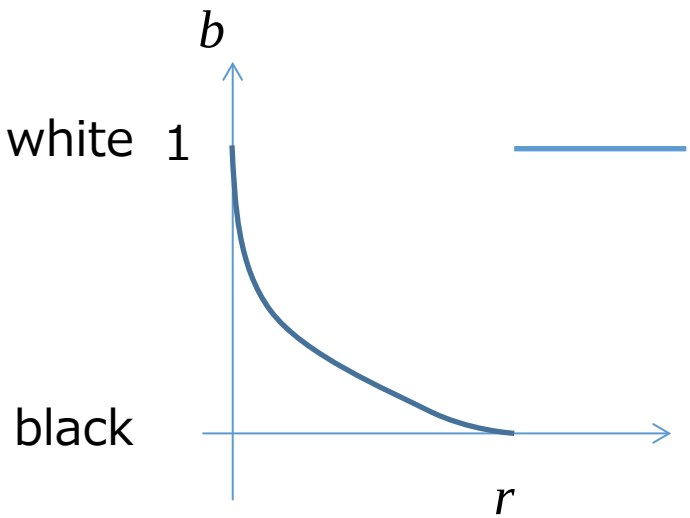
A point inside the circle becomes darker
if the distance is larger

Exercises to make the image

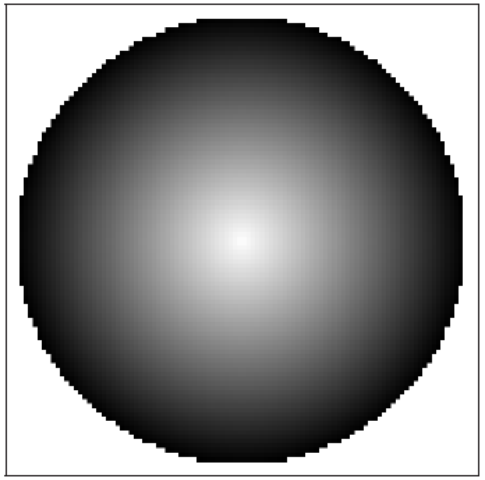
- We define the brightness of each point as follows

$$b(x,y) = \begin{cases} \frac{r-d(x,y)}{r} & (d(x,y) \leq r) \\ 1 & (d(x,y) > r) \end{cases}$$

Distance of (x,y) from the center



Distance from the center



Exercise

- We define the brightness of each point as follows

$$b(x,y) = \begin{cases} \frac{r-d(x,y)}{r} & (d(x,y) \leq r) \\ 1 & (d(x,y) > r) \end{cases}$$

Distance of (x,y) from the origin

- Define the function $b\,(r, x, y)$ that computes $b(r, x, y)$ when r is specified.
  - we here assume that r is also a parameter.

Rem.

We have already made a function $d(x, y)$ in prev. exercise

Drawing a Sphere: Partial Program is Downloadable

Using $b(r, x, y)$, we can define a function to draw a sphere

```
def sphere(r)
    image = make2d(2*r+1, 2*r+1)
    for y in 0..(2*r)
        for x in 0..(2*r)
            image[y][x] = b(r, x, y)
        end
    end
    image
end
```

Make a 2-dim array
all of whose entries are 0

For each entry,
determine the brightness
by the function b

sphere.rb

Today's Contents

- Array
  - a sequence of variables
  - Preliminaries: use irsb instead of irb
  - Using Array in Ruby
  - 2-dimensional array and images
  - Color image
- Draw an image using repetition and condition
- Exercises
- Appendix: see when you have time

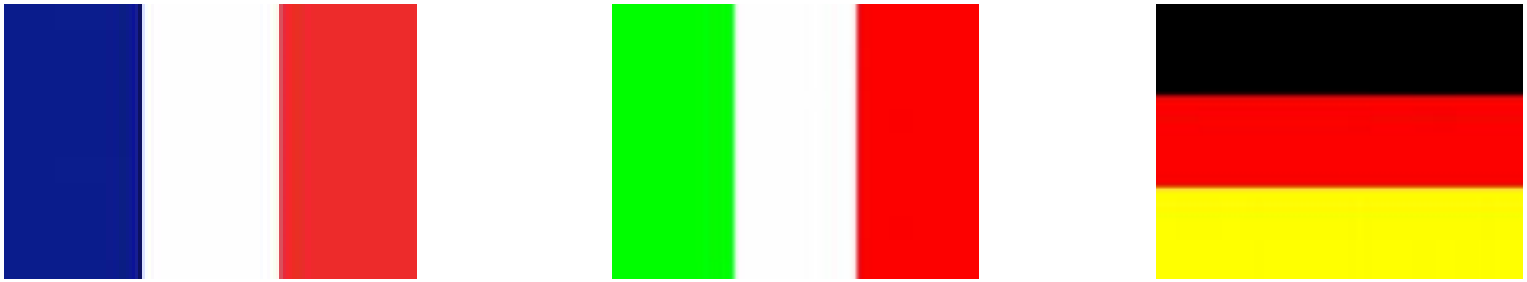
Today's Exercises

- 2 quizzes
  - About arrays
  - About making a sphere

- **(If you have time) Exercise 4-6 and/or 4-7**

(Recap.) Exercise 1 (4-3)

- Draw national flags using color image
  - Ex. France, Italy, Germany…
  - Refer to color information in the table



Color	Green	White	Red	Orange	Blue
Red	0	1	1	1	0
Green	0.6	1	0	0.4	0.2
Blue	0	1	0	0	0.6

Exercise 2: Make missing components

- Download
  - make1d.rb
  - make2d.rb
  - sphere.rb

- Ex2: Define the function $b(r, x, y)$ that computes $b(r, x, y)$ when r is specified.
  - We assume that r is also a parameter.
  - Function d should be defined if necessary

- Execute “show(sphere(100))” on isrb, and confirm that we can obtain a desired image.
  - You can save the image as an image file

Skelton of Function Sphere is Provided

- Fill in the part of function $b(r, x, y)$

```
def b(r, x, y)
  # WRITE THE DEFINITION OF B DOWN HERE
end

def sphere(r)
    image = make2d(2*r+1, 2*r+1)
    for y in 0..(2*r)
        for x in 0..(2*r)
            image[y][x] = b(r, x, y)
        end
    end
    image
end
```

How to Submit Exercises

- Through ITC-LMS as in the last week:
  - Create a text file ([StudentID].txt) containing all the answers
    - program to make the flag image
    - function b(r,x,y)
  - Upload it through “assignments” in ITC-LMS
  - Upload an image of your national flag
- (If you have time) Exercise 4-6 and/or 4-7
- Deadline: Nov. 2(Wed), 23:59

Next Week (Oct. 31)

- More arrays and more images

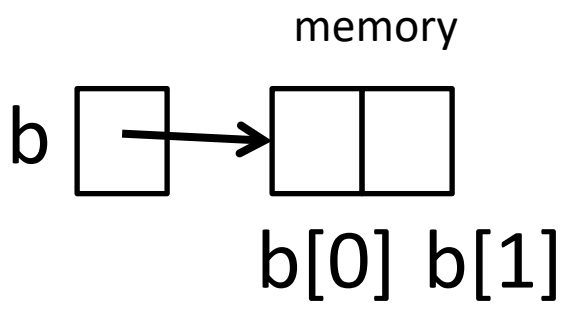
Appendix: Note on Using Arrays

- Differences between variables and arrays
- Intuition of 2-dimensional arrays
- Behaviors of arrays
  - Corresponding to Exercises 4-11 and 4-12

Cf. Array as Values

- Real data in an array is assigned somewhere else in computer memory

- Ex. array b points to memory having the value



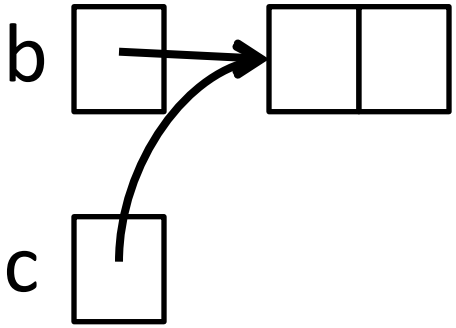
Assignment to an array

- For example,

  c = b

  means that

  the two variables point the same array.



Copy of the values are not made
Both are referring to the same memory

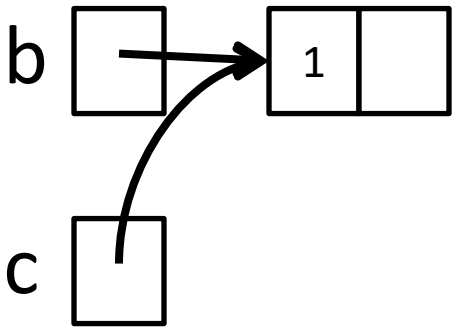
Assignment to an array

- For example,

  c = b

  means that

  the two variables point the same array.



- If we write

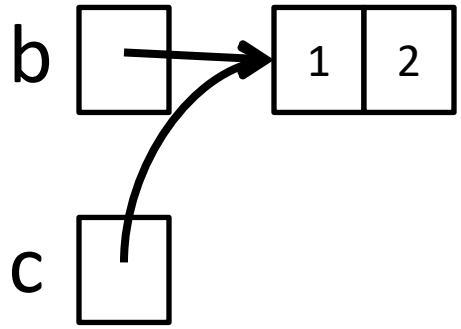
  b[0] = 1 changing value in the pointed memory

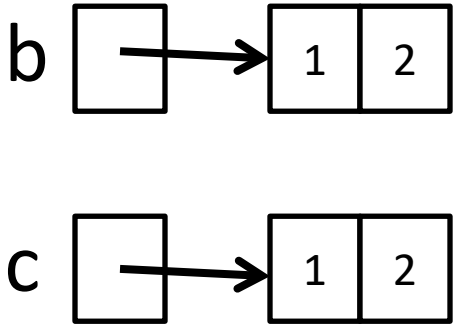
  then c[0] also becomes 1.

Display an Array

- We cannot distinguish the above two cases

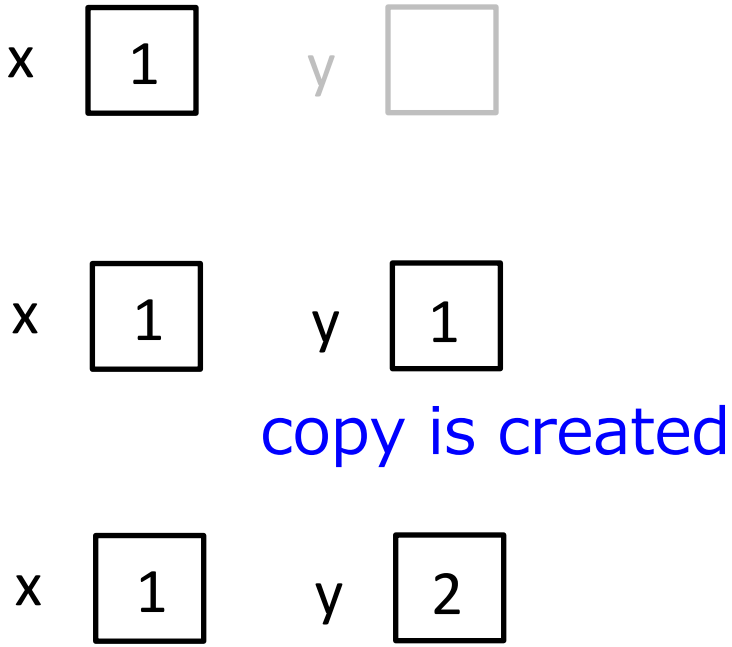
b => [1, 2] c => [1, 2]





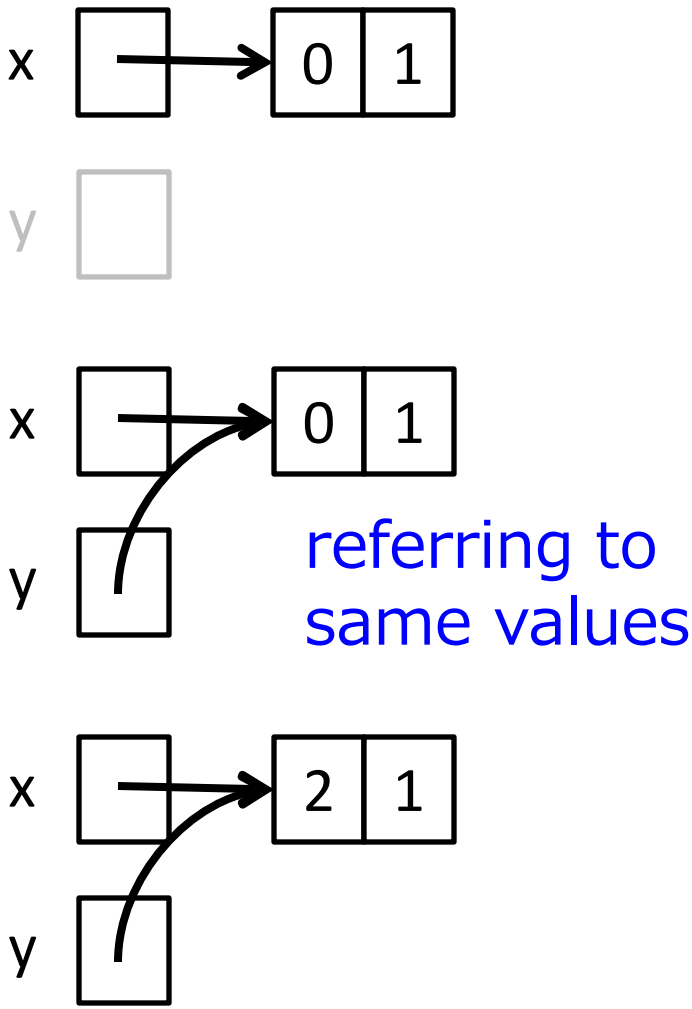
Behavior of Variables


```
irb(main):001:0> x = 1
=> 1
irb(main):002:0> y = x
=> 1
irb(main):003:0> y
=> 1
irb(main):004:0> y = 2
=> 2
irb(main):005:0> y
=> 2
irb(main):006:0> x
=> 1
```



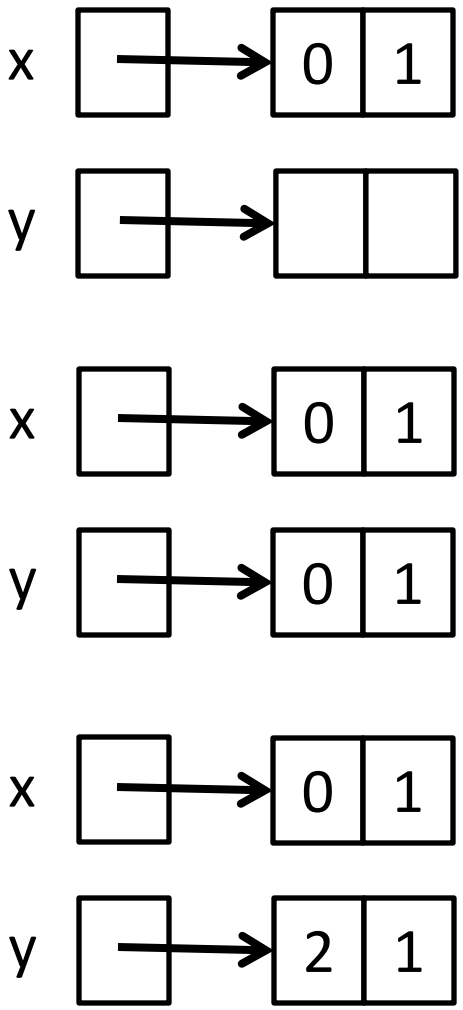
Behavior of Arrays


```
irb(main):001:0> x = [0,1]
=> [0, 1]
irb(main):002:0> y = x
=> [0, 1]
irb(main):003:0> y[0]
=> 0
irb(main):004:0> y[0] = 2
=> 2
irb(main):005:0> y[0]
=> 2
irb(main):006:0> x[0]
=> 2
```



Copying an Array


```
irb(main):001:0> x = [0,1]
=> [0, 1]
irb(main):002:0> y = Array.new(2)
=> [nil, nil]
irb(main):003:0> y[0] = x[0]
=> 0
irb(main):004:0> y[1] = x[1]
=> 1
irb(main):005:0> y[0] = 2
=> 2
irb(main):006:0> x
=> [0, 1]
```



Create an empty array

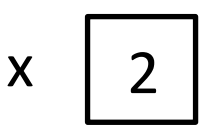
Copy values

Difference between Variable and Array


```
def plus(a)
  a=a+1
end
```

```
irb(main):004:0> x=2
=> 2
irb(main):005:0> plus(x)
=> 3
irb(main):006:0> x
=> 2
```

x is not changed

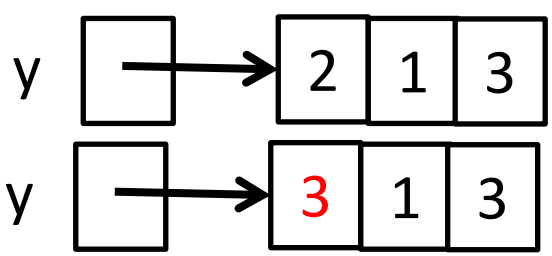


copy is created in *plus*

```
def plus_array(a)
  a[0]=a[0]+1
end
```

```
irb(main):004:0> y=[2,1,3]
=> [2,1,3]
irb(main):005:0> plus_array(y)
=> 3
irb(main):006:0> y
=> [3,1,3]
```

y has been changed



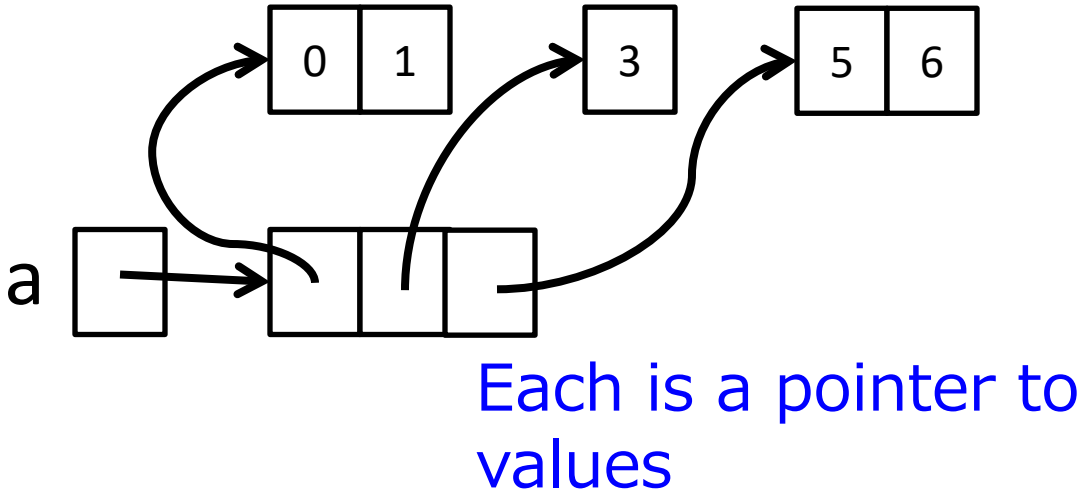
Appendix: Note on Using Arrays

- Differences between variables and arrays
- Intuition of 2-dimensional arrays
- Behaviors of arrays
  - Corresponding to Exercises 4-11 and 4-12

Two-Dimensional Array

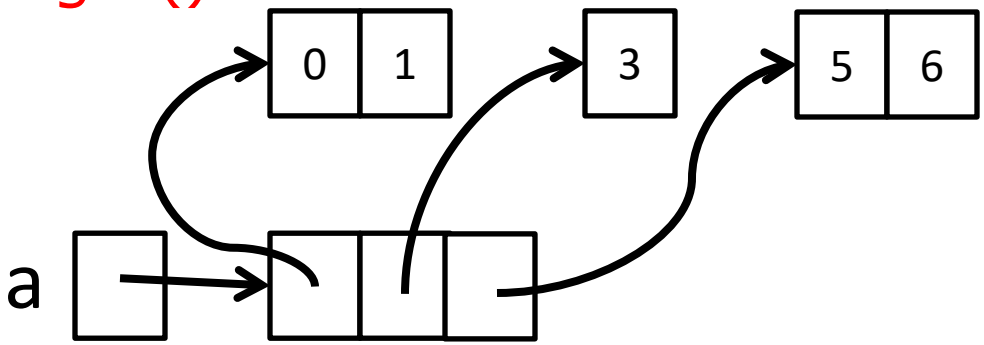
- Array of "Arrays"
  - Array = a pointer to values

```
irb(main):001:0> a = [[0,1],[3],[5,6]]
=> [[0, 1], [3], [5, 6]]
```



Exercise: Behavior of 2-dimensional Array


```
irb(main):001:0> a = [[0,1],[3],[5,6]]
=> [[0, 1], [3], [5, 6]]
irb(main):002:0> a.length()
=> 3
irb(main):003:0> a[0].length()
=> 2
irb(main):004:0> a[1].length()
=> 1
irb(main):005:0> a[2].length()
=> 2
```



Appendix: Note on Using Arrays

- Differences between variables and arrays
- Intuition of 2-dimensional arrays
- Behaviors of arrays
  - Corresponding to Exercises 4-11 and 4-12

What does this program return?


```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
    b[i] = a
    for j in 0..1
        b[i][j] = i
    end
end
b
```

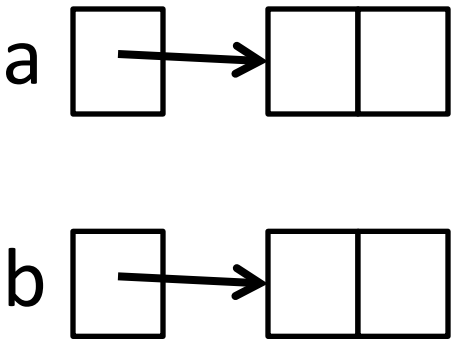
1. [[0,0],[0,0]]
2. [[0,0],[1,1]]
3. [[0,1],[0,1]]
4. [[1,1],[1,1]]
5. [0,0]
6. [0,1]
7. [1,1]

What does this program return?


```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
    b[i] = a
    for j in 0..1
        b[i][j] = i
    end
end
b
```

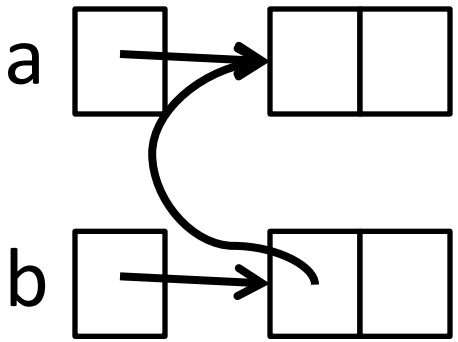
What does this program return?


```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
    b[i] = a
    for j in 0..1
        b[i][j] = i
    end
end
b
```



What does this program return?


```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
    b[i] = a
    for j in 0..1
        b[i][j] = i
    end
end
b
```

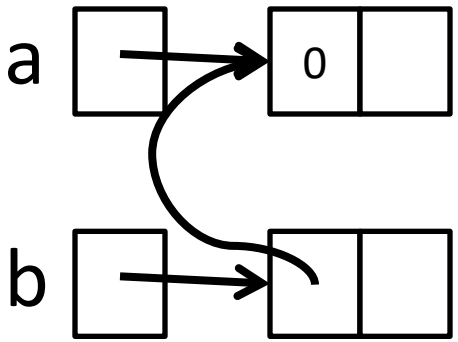


i = 0

b[i] = a

What does this program return?

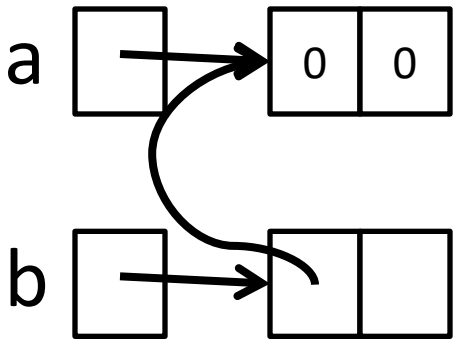

```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
    b[i] = a
    for j in 0..1
        b[i][j] = i
    end
end
b
```




```
i = 0
b[i] = a
  j = 0
  b[i][j] = i
```

What does this program return?

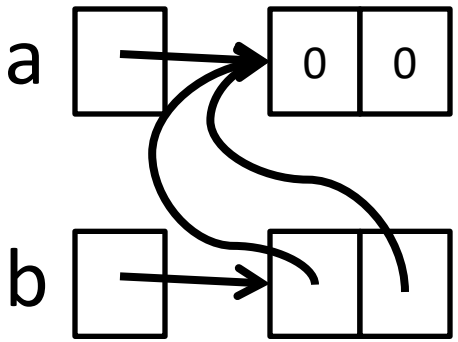

```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
    b[i] = a
    for j in 0..1
        b[i][j] = i
    end
end
b
```




```
i = 0
b[i] = a
  j = 0
  b[i][j] = i
  j = 1
  b[i][j] = i
```

What does this program return?


```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
    b[i] = a
    for j in 0..1
        b[i][j] = i
    end
end
b
```

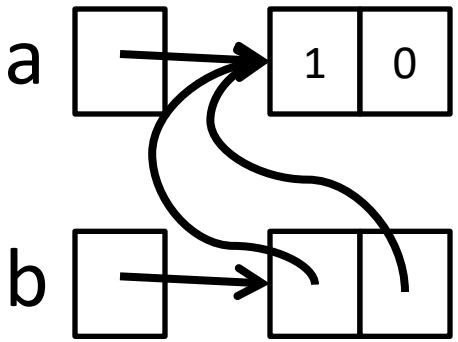


i = 1

b[i] = a

What does this program return?

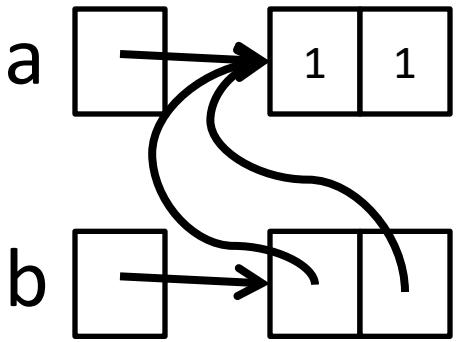

```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
    b[i] = a
    for j in 0..1
        b[i][j] = i
    end
end
b
```




```
i = 1
b[i] = a
  j = 0
  b[i][j] = i
```

What does this program return?


```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
    b[i] = a
    for j in 0..1
        b[i][j] = i
    end
end
b
```




```
i = 1
b[i] = a
  j = 0
  b[i][j] = i
  j = 1
  b[i][j] = i
```

What does this program return?


```
b = Array.new(2)
for i in 0..1
    b[i] = Array.new(2)
    for j in 0..1
        b[i][j] = i
    end
end
b
```

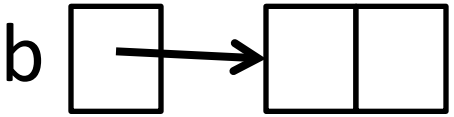
1. [[0,0],[0,0]]
2. [[0,0],[1,1]]
3. [[0,1],[0,1]]
4. [[1,1],[1,1]]
5. [0,0]
6. [0,1]
7. [1,1]

What does this program return?


```
b = Array.new(2)
for i in 0..1
    b[i] = Array.new(2)
    for j in 0..1
        b[i][j] = i
    end
end
b
```

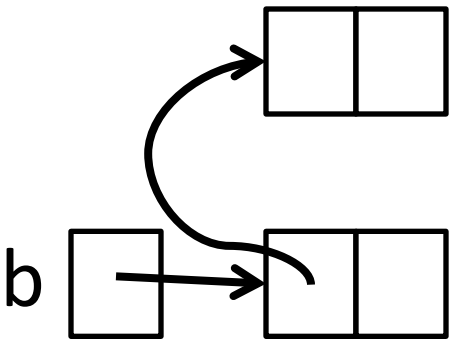
What does this program return?


```
b = Array.new(2)
for i in 0..1
   b[i] = Array.new(2)
  for j in 0..1
        b[i][j] = i
    end
end
b
```



What does this program return?


```
b = Array.new(2)
for i in 0..1
    b[i] = Array.new(2)
    for j in 0..1
        b[i][j] = i
    end
end
b
```

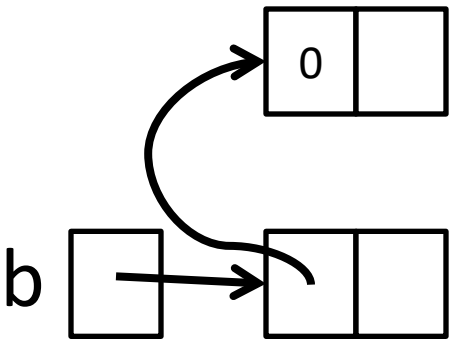


i = 0

b[i] = Array.new(2)

What does this program return?

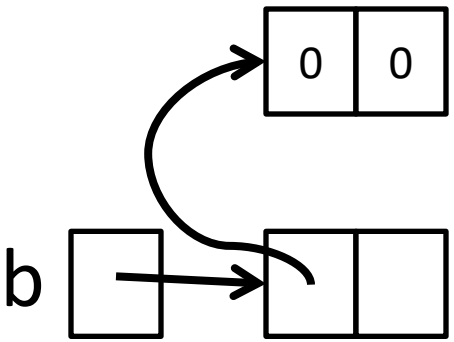

```
b = Array.new(2)
for i in 0..1
    b[i] = Array.new(2)
    for j in 0..1
        b[i][j] = i
    end
end
b
```



i = 0
b[i] = Array.new(2)
  j = 0
  b[i][j] = i

What does this program return?

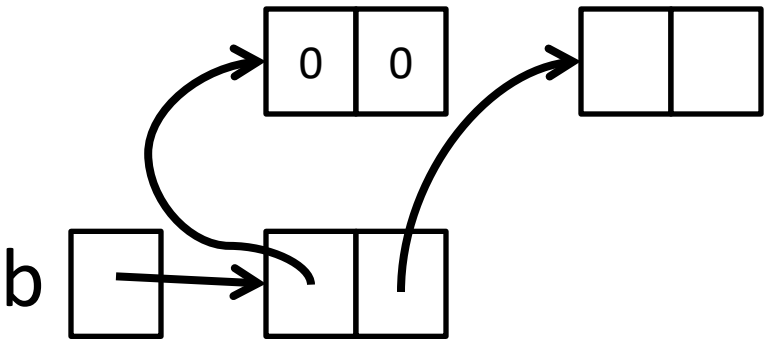

```
b = Array.new(2)
for i in 0..1
    b[i] = Array.new(2)
    for j in 0..1
        b[i][j] = i
    end
end
b
```




```
i = 0
b[i] = Array.new(2)
  j = 0
  b[i][j] = i
  j = 1
  b[i][j] = i
```

What does this program return?


```
b = Array.new(2)
for i in 0..1
    b[i] = Array.new(2)
    for j in 0..1
        b[i][j] = i
    end
end
b
```

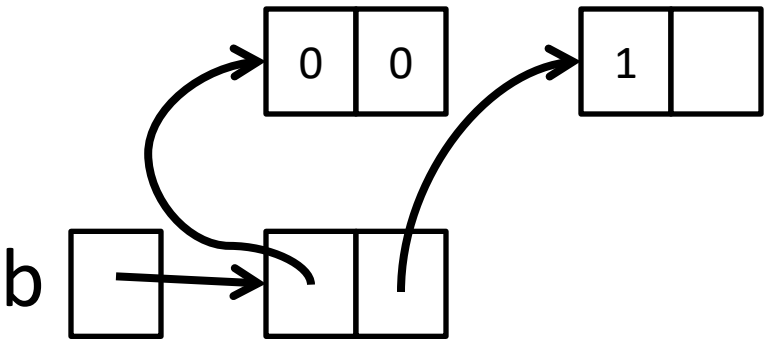


i = 1

b[i] = Array.new(2)

What does this program return?

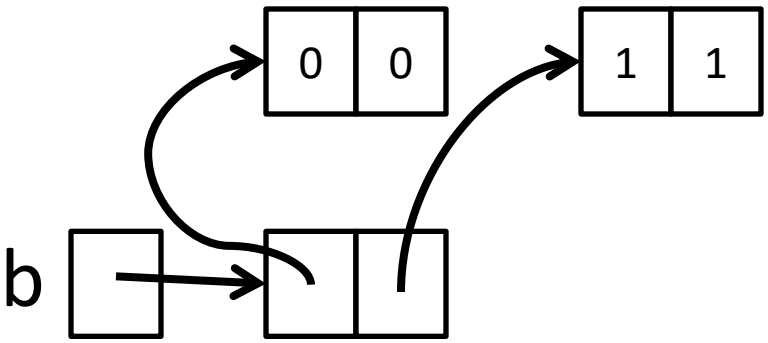

```
b = Array.new(2)
for i in 0..1
    b[i] = Array.new(2)
    for j in 0..1
        b[i][j] = i
    end
end
b
```



i = 1
b[i] = Array.new(2)
j = 0
b[i][j] = i

What does this program return?


```
b = Array.new(2)
for i in 0..1
    b[i] = Array.new(2)
    for j in 0..1
        b[i][j] = i
    end
end
b
```




```
i = 1
b[i] = Array.new(2)
  j = 0
  b[i][j] = i
  j = 1
  b[i][j] = i
```

Parameter and Returned Value of Array


```
def  inc1(b)
  n = b.length()

  for i in 0..n-1

      b[i] = b[i]+1
  end
  b
end
```

```
def plus1(b)
   n = b.length()
   c = Array.new(n)
   for i in 0..n-1
      c[i] = b[i]+1
   end
   c
end
```

Parameter and Returned Value of Array


```
>> a = [1,2]
>> plus1(a)
=> [1, 2]                    => [3, 4]
>>  inc1(a)
>> a
=> [2, 3]                    => [2, 3]
>> a
=> [2, 3]
```