

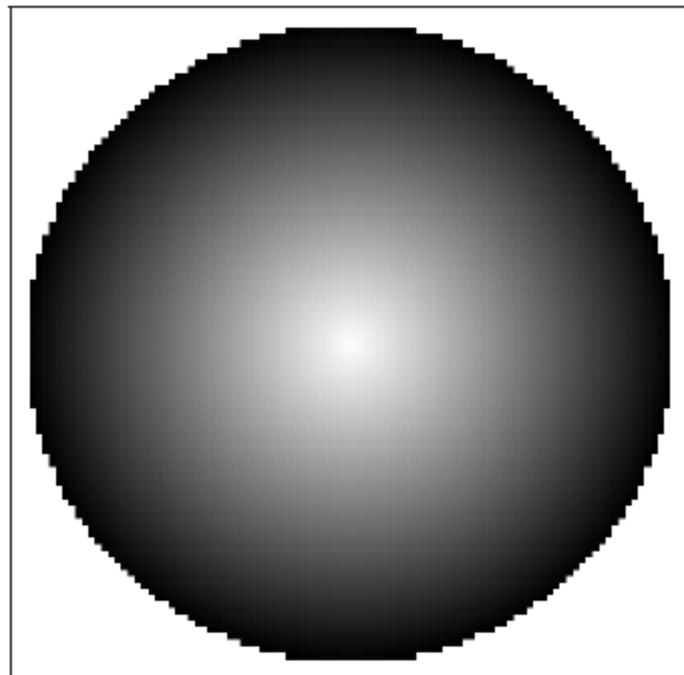
配列・繰り返し・文字列

教科書 P40～

自習状況の確認

1. 教科書の練習2.1(渦巻き), 2.4(国旗)など, すべて試した
2. 教科書の練習2.1(渦巻き)は試した
3. 簡単な画像(図2.2)は試した
4. 画像を描くプログラムは試していない

繰り返しによる画像の作成



与えられた大きさの 1次元配列を作る

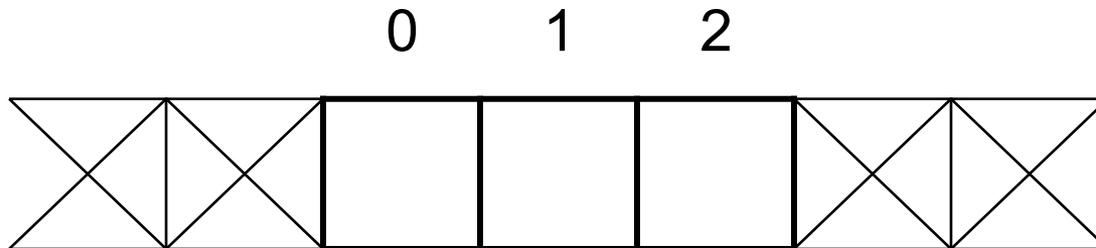
```
isrb(main):001:0> image = Array.new(6)
```

```
[nil , nil , nil , nil , nil , nil]
```

```
isrb(main):002:0> a = Array.new(3)
```

```
[nil , nil , nil]
```

空の容器(変数)は nil となる



繰り返しによって、 配列の全要素をそれぞれ変更する

```
isrb(main):003:0> for i in 0..2
```

```
isrb(main):004:1>   a[i] = 0
```

```
isrb(main):005:1> end
```

```
0..2
```

```
isrb(main):006:0> a
```

```
[0, 0, 0]
```

配列の作成: `Array.new(式)` という式は、大きさが `式` の値であるような配列を作る。作られた配列の中身は全て `nil` である。

繰り返し: `変数` の値を `式1` の値から `式2` の値まで 1 ずつ順に変化させながら、`命令1` から `命令n` を毎回実行する繰り返しは次のように書く。

```
for 変数 in 式1 .. 式2
  命令1
  ⋮
  命令n
end
```

0ばかりの1次元配列を作る

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

make1d.rb

わからない？

1. def
2. Array.new(n)
3. for i in 0..(n-1)
4. そのほか 具体的に書く

make1d.rb を授業のページからダウンロードする

2次元配列を作る

```
isrb(main):007:0> image = Array.new(6)
```

```
[nil , nil , nil , nil , nil , nil]
```

```
isrb(main):008:0> for i in 0..5
```

```
isrb(main):009:1>   image[i] = make1d(3)
```

```
isrb(main):010:1> end
```

```
0..5
```

```
isrb(main):011:0> image
```

```
=> [[0 , 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0,  
0, 0], [0, 0, 0]]
```

練習

- m 行 n 列の配列を作る `make2d(m,n)` を定義せよ。ただし、作られる配列の中身は全て 0 とし、ファイル名は `make2d.rb` とする。

関数を定義したら投票

1. 定義した

0ばかりの2次元配列を作る

```
load("./make1d.rb")  
def make2d(m,n)  
  a = make1d(m)  
  for i in 0..(m-1)  
    a[i] = make1d(n)  
  end  
  a  
end
```

make2d.rb

練習(方針)

- `show(sphere(20))` を実行して表示される画像を確認する。ただし、その前に以下を実行する。
 - `sphere.rb` は授業のページからダウンロードする。
(次に見せるように教科書とは違うので注意！)
 - 後に書くように `make2d.rb` と `b.rb` を作成する。

(詳細は、このあとのスライドで)

sphere.rb (2重の繰返し)

```
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
```

sphere.rb

練習

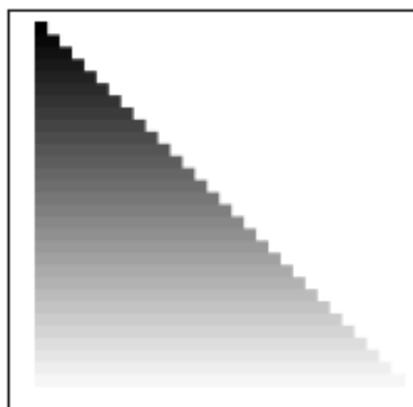
- m 行 n 列の配列を作る `make2d(m,n)` を定義する。配列の中身は全て 0。(make2d.rb)
- 式3.2 を計算する関数 `b(r,x,y)` を定義する。(x, y) だけでなく r も引数なので注意。(b.rb)

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

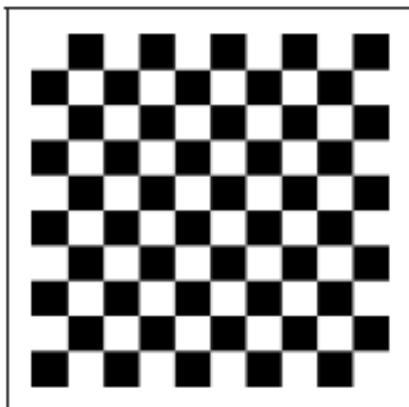
(r,r) と (x,y) との間の距離

- `show(sphere(20))` を実行して表示される画像を確かめたら、「1」を投票する。(練習 3.10へ)

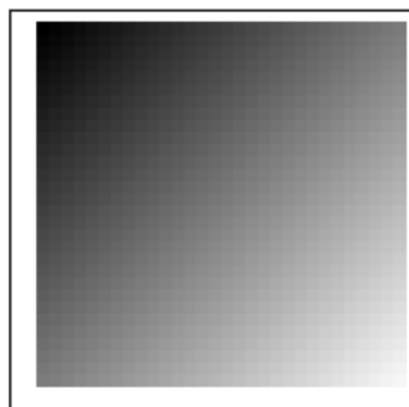
練習 3.10 (色々な図形*) 次のような画像を作成する関数をそれぞれ定義せよ。



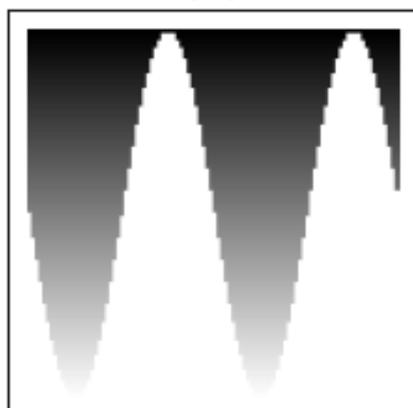
(a)



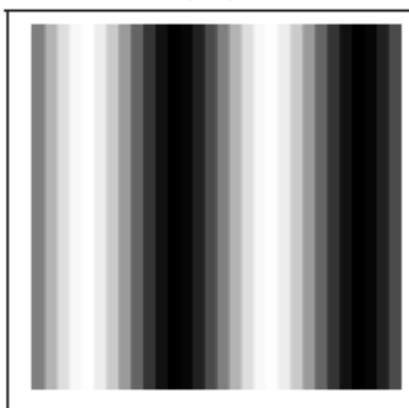
(b)



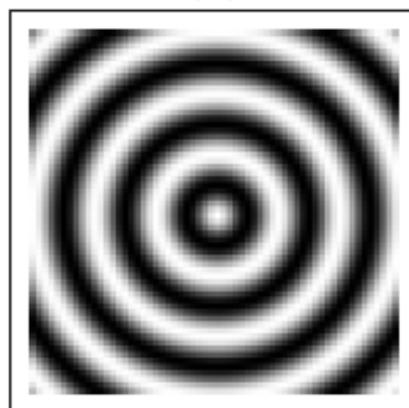
(c)



(d)



(e)



(f)

進捗状況の確認

1. `show(sphere(20))` がうまく行った時点で投票した
2. 実行したけれども、`show(sphere(20))` がうまくいかない
3. 関数 `b` ができていない
4. 関数 `make2d` ができていない
5. それ以外, 具体的に書く

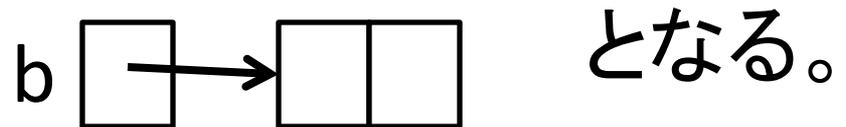
関数 $b(r,x,y)$

```
include(Math)
def b(r,x,y)
  d = sqrt((x-r)**2 + (y-r)**2)
  if d <= r
    (r - d) / r
  else
    1
  end
end
end
```

b.rb

値としての配列

- 配列の実体(実際のデータ)は、コンピュータのメモリのどこかに割りつけられている。
- 配列を変数に代入したり、配列を関数の引数として渡したり、関数の値として返したりするときは、実体の「場所」がやりとりされる。
- たとえば、変数 b に配列の実体の場所が値として入っているとき、

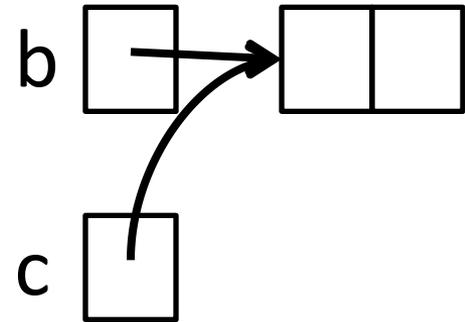


配列の代入

- たとえば、

$$c = b$$

とすると、同じ配列を二つの
変数が指す。



配列の代入

- たとえば、

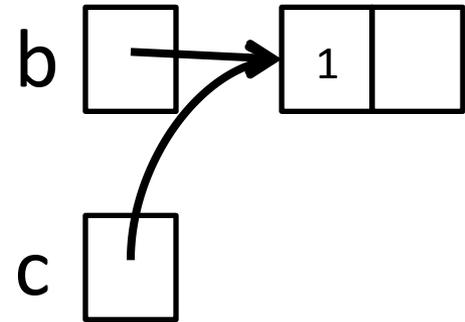
$$c = b$$

とすると、同じ配列を二つの
変数が指す。

- このとき、

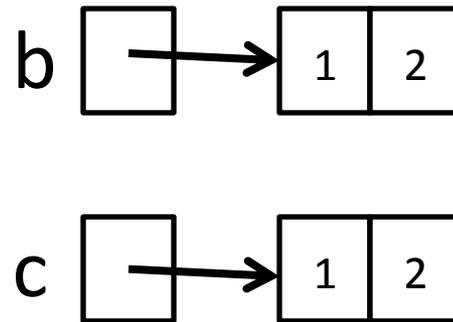
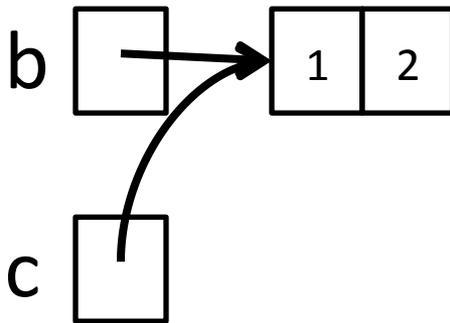
$$b[0] = 1$$

とすれば、 $c[0]$ も同じく 1 になる。



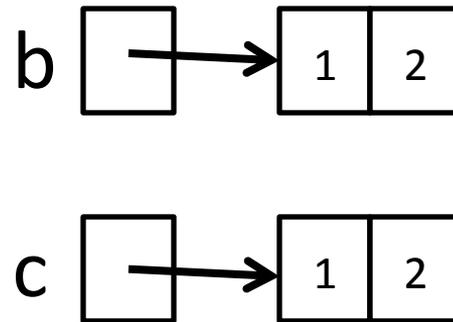
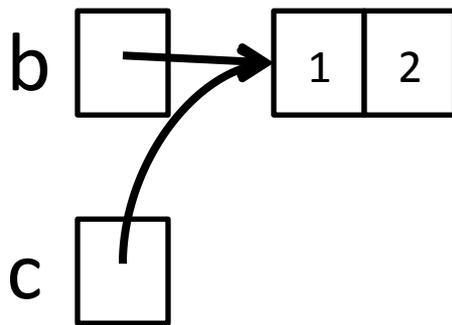
配列の表示

- 配列を表示させると、その要素が順に並べて表示される。場所の情報は明示されない。



配列の表示

- 配列を表示させると、その要素が順に並べて表示される。場所の情報は明示されない。



- 表示させても、両者は区別できない。

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

基本型の変数

irb(main):001:0> x = 1

=> 1

x

1

 y

--

irb(main):002:0> y = x

=> 1

x

1

 y

1

irb(main):003:0> y

=> 1

irb(main):004:0> y = 2

=> 2

x

1

 y

2

irb(main):005:0> y

=> 2

irb(main):006:0> x

=> 1

配列の変数

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

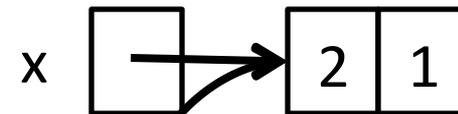
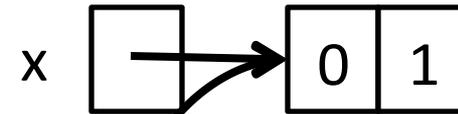
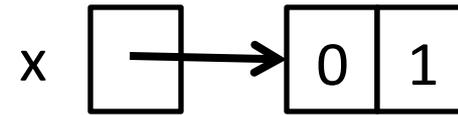
=> 2

irb(main):005:0> y[0]

=> 2

irb(main):006:0> x[0]

=> 2



配列の複製(コピー)

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]

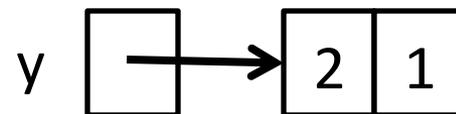
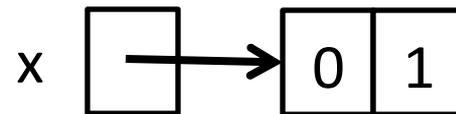
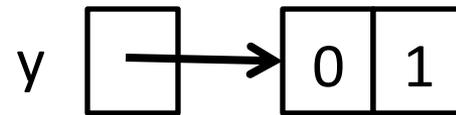
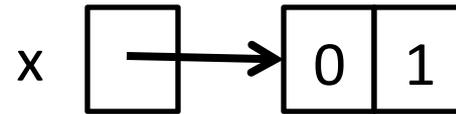
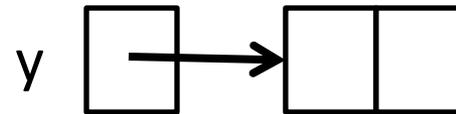
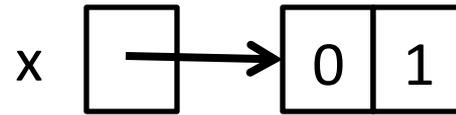
=> 1

irb(main):005:0> y[0] = 2

=> 2

irb(main):006:0> x

=> [0, 1]



ここまで

1. わかった
2. それ以外一わからないところを書く

結局: 2次元配列って...

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

```
=> [[0, 1], [3], [5, 6]]
```

```
irb(main):002:0> a.length()
```

```
=> ?
```

```
irb(main):003:0> a[0].length()
```

```
=> ?
```

```
irb(main):004:0> a[1].length()
```

```
=> ?
```

```
irb(main):005:0> a[2].length()
```

```
=> ?
```

結局: 2次元配列って...

```
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()
```

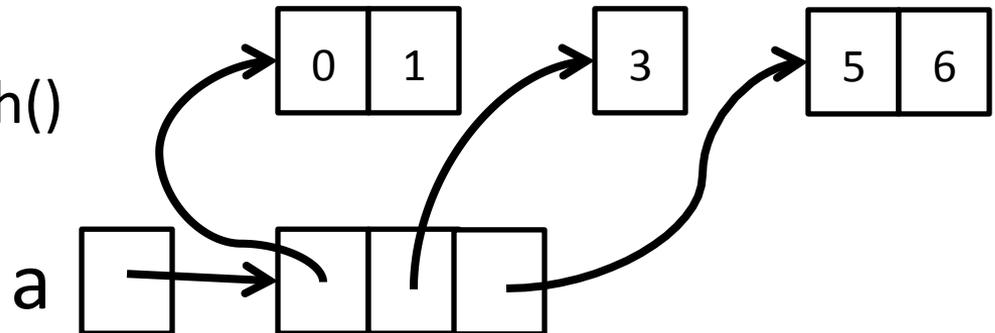
```
=> 2
```

```
irb(main):004:0> a[1].length()
```

```
=> 1
```

```
irb(main):005:0> a[2].length()
```

```
=> 2
```



ここまで

1. わかった
2. それ以外一わからないところを書く

次は何を返す？

```
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]`

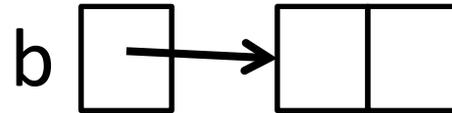
7. `[1,1]`

次は何を返す？

```
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
```

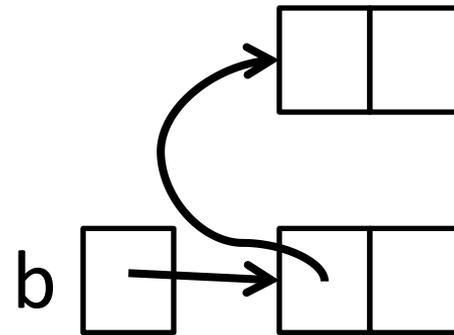
次は何を返す？

```
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
```



次は何を返す？

```
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
```

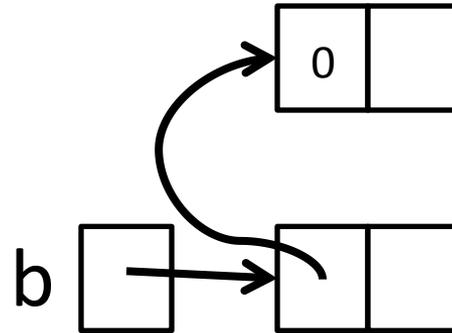


`i = 0`

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

次は何を返す？

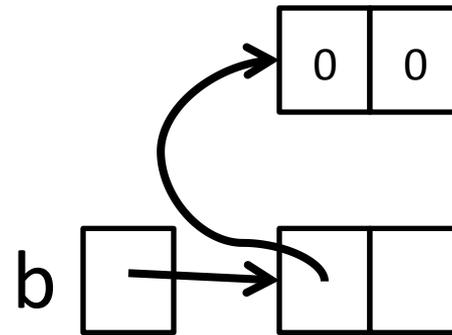
```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
b
```



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

次は何を返す？

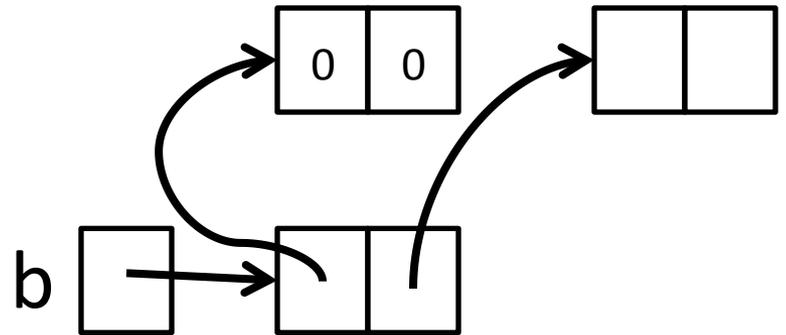
```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
b
```



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

次は何を返す？

```
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
```

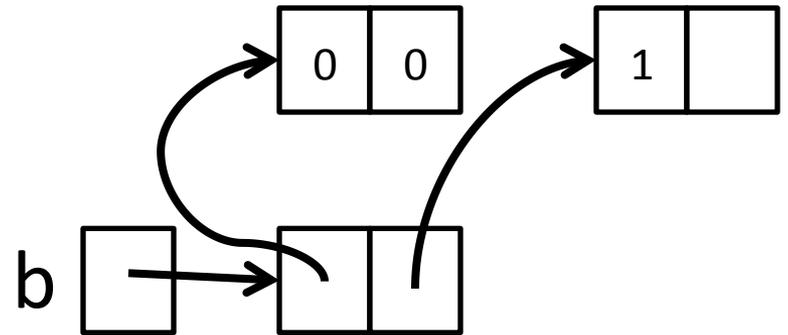


`i = 1`

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

次は何を返す？

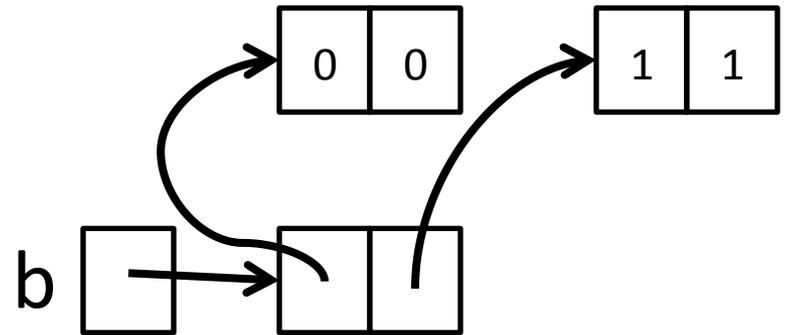
```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
b
```



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

次は何を返す？

```
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
```

次は何を返す？

```
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]`

次は何を返す？

```
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
```

次は何を返す？

```
a = Array.new(2)
```

```
b = Array.new(2)
```

```
for i in 0..1
```

```
  b[i] = a
```

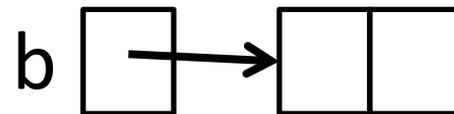
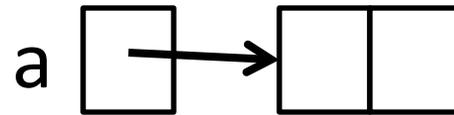
```
  for j in 0..1
```

```
    b[i][j] = i
```

```
  end
```

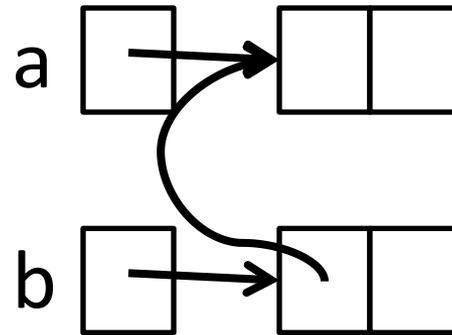
```
end
```

```
b
```



次は何を返す？

```
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
```

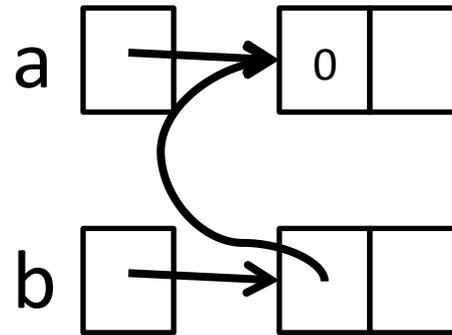


$i = 0$

$b[i] = a$

次は何を返す？

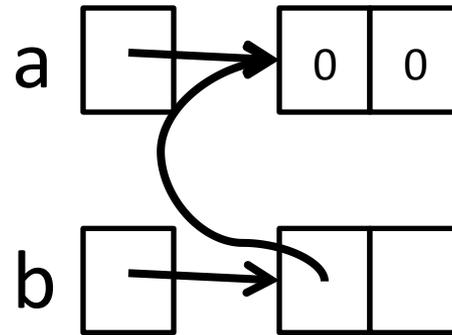
```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
  b[i] = a
  for j in 0..1
    b[i][j] = i
  end
end
b
```



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

次は何を返す？

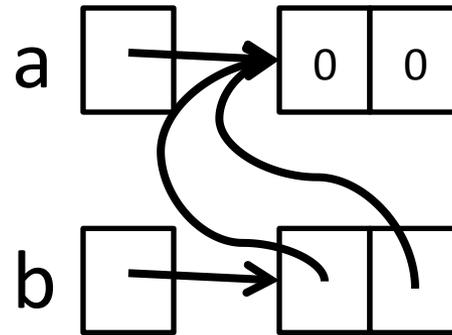
```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
  b[i] = a
  for j in 0..1
    b[i][j] = i
  end
end
b
```



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

次は何を返す？

```
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
```



$i = 1$
 $b[i] = a$

次は何を返す？

```
a = Array.new(2)
```

```
b = Array.new(2)
```

```
for i in 0..1
```

```
  b[i] = a
```

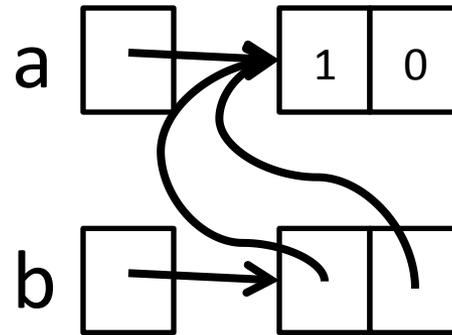
```
  for j in 0..1
```

```
    b[i][j] = i
```

```
  end
```

```
end
```

```
b
```



```
i = 1
```

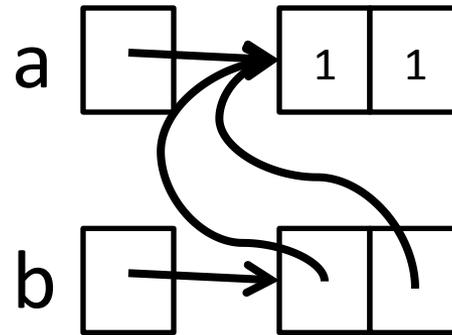
```
b[i] = a
```

```
j = 0
```

```
b[i][j] = i
```

次は何を返す？

```
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
```

ここまで

1. わかった
2. それ以外一わからないところを書く

配列の引数・返り値

```
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
```

配列の引数・返り値

irb(main):006:0> a = [1,2]

=> [1, 2]

irb(main):007:0> inc1(a)

=> [2, 3]

irb(main):008:0> a

=> [2, 3]

irb(main):009:0> plus1(a)

=> [3, 4]

irb(main):010:0> a

=> [2, 3]

練習

- a の x 番目の値と、その前後の値の平均値を求める `array_average(a,x)` を作れ。ただし前後の値とは、 a の $(x - 1)$ 番目と $(x + 1)$ 番目の値のうち、 a の範囲内のものだとする。例えば a が $[1,2,3]$ のとき `array_average(a,0)` は 1.5, `array_average(a,1)` は 2.0, `array_average(a,2)` は 2.5 になる。
- 終わったら、「1」を投票してください。
- 次週までの課題に挑戦する。

配列の前後の平均を求める

```
def array_average(a,x)
  n = 0
  s = 0.0
  for j in -1..1
    if i+j >= 0 && i+j < a.length()
      n = n+1
      s = s+a[i+j]
    end
  end
  if n == 0
    0
  else
    s / n
  end
end
```

array_average.rb

練習

- 画像データの座標 (x, y) の点と、その周囲の点の明度の平均値を計算する関数 `image_average(image,x,y)` を作れ。周囲の点とは指定された点の上下左右 8 点のうち、画像の範囲内の点だとする。例えば座標 $(0, 0)$ の周囲の点は、 $(0, 1)$, $(1, 1)$, $(1, 0)$ の 3 点になる。