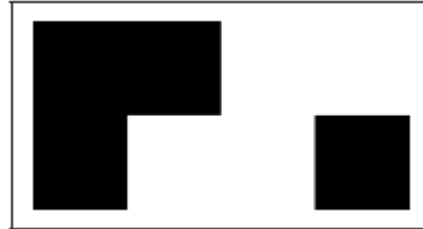


配列による画像の表示

画像の表現



isrb の
プロンプト

```
irb(main):002:0> コントロールD
```

```
cm12345$ isrb
```

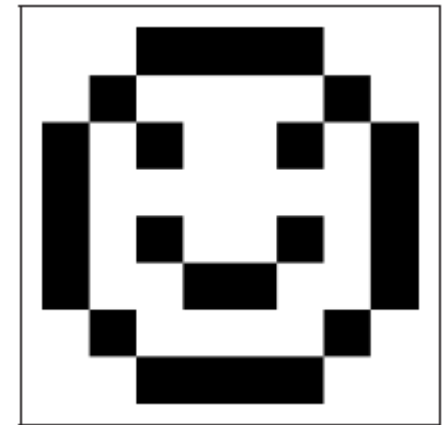
```
>> a = [[0,0,1,1],
```

```
?>       [0,1,1,0]]
```

```
=> [[0 , 0, 1, 1], [0, 1, 1, 0]]
```

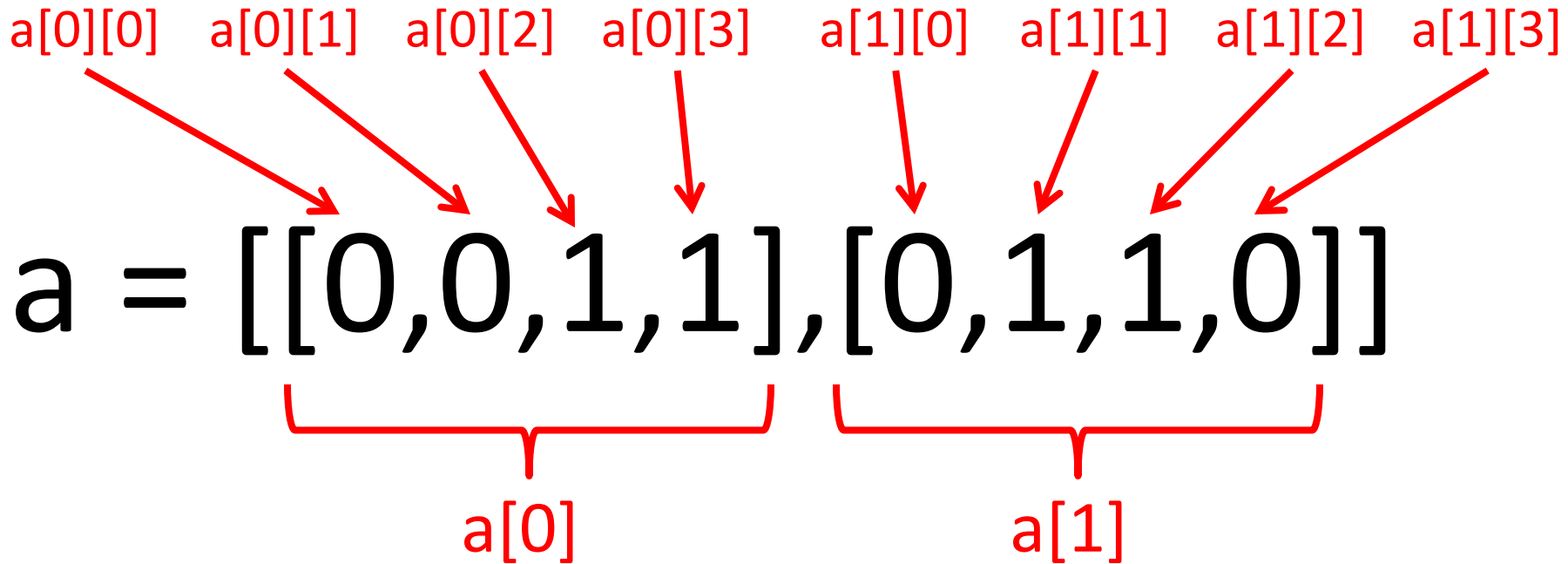
```
>> show(a)
```

```
=> nil
```



単なる参考

配列の要素



画像の操作

>> a[0][0]

座標(0,0)の明度を参照

=> 0

>> a[0][2]

座標(2,0)の明度を参照

=> 1

>> a[1][2]=0.5

座標(2,1)の明度を変更

=> 0.5

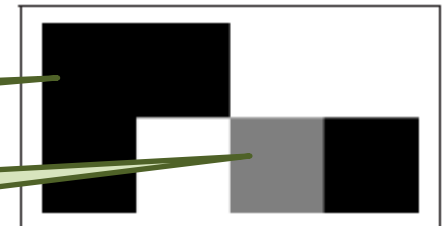
再表示

>> show(a)

=> nil

座標(0,0)

座標(2,1)



練習2.1

- 次のようなデータを作成し、画像として表示させよ。

$$w = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

条件分岐と繰り返し

条件分岐 --- 場合分けを使った計算

```
irb(main):003:0> load("./ max.rb")
```

```
=> true
```

```
irb(main):004:0> max(123, 456)
```

```
=> 456
```

```
irb(main):005:0> max(max(12, 34), max(56, 78))
```

```
=> 78
```

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

3 通りの場合分け

```
def sign(x)
  if x < 0
    -1
  else
    if 0 < x
      1      # not(x<0) and 0<x
    else
      0      # not(x<0) and not(0<x)
    end
  end
end
end
```

sign.rb

複雑な条件

- 色々な比較

書き方	数学	意味
<code>x > y</code>	$>$	xがyより大きい
<code>x >= y</code>	\geq	xがy以上
<code>x == y</code>	$=$	xとyが等しい (x=y でないことに注意)
<code>x < y</code>	$<$	xがyより小さい
<code>x <= y</code>	\leq	xがy以下
<code>x != y</code>	\neq	xとyが異なる

- 条件式の組合せ

書き方	意味
<code>x > y x == 0</code>	x > y <u>または</u> x == 0
<code>x < y && y < z</code>	x < y <u>かつ</u> y < z
<code>!(x < y && y < z)</code>	(x < y <u>かつ</u> y < z) <u>でない</u>

複雑な条件

= でないことに注意
= は代入

- 色々な比較

書き方	数学	意味
$x > y$	$>$	x が y より大きい
$x \geq y$	\geq	x が y 以上
$x == y$	$=$	x と y が等しい (x=y でないことに注意)
$x < y$	$<$	x が y より小さい
$x \leq y$	\leq	x が y 以下
$x != y$	\neq	x と y が異なる

- 条件式の組合せ

書き方	意味
$x > y \ \ x == 0$	x > y <u>または</u> x == 0
$x < y \ \&\& \ y < z$	x < y <u>かつ</u> y < z
$!(x < y \ \&\& \ y < z)$	(x < y <u>かつ</u> y < z) <u>でない</u>

どれが正しいか？

x の値が 7、y の値が 5、z の値が 3 であるとして

1. $x < y$
2. $x \leq y$
3. $y \neq z$
4. $z > x$
5. $z == x$

どれが正しいか？

x の値が 7、y の値が 5、z の値が 3 であるとして

1. $x < y$
2. $x \leq y$
3. $x < y \ \&\& \ y \neq z$
4. $x \leq y \ || \ y == z$
5. $!(x < y \ \&\& \ y == z)$

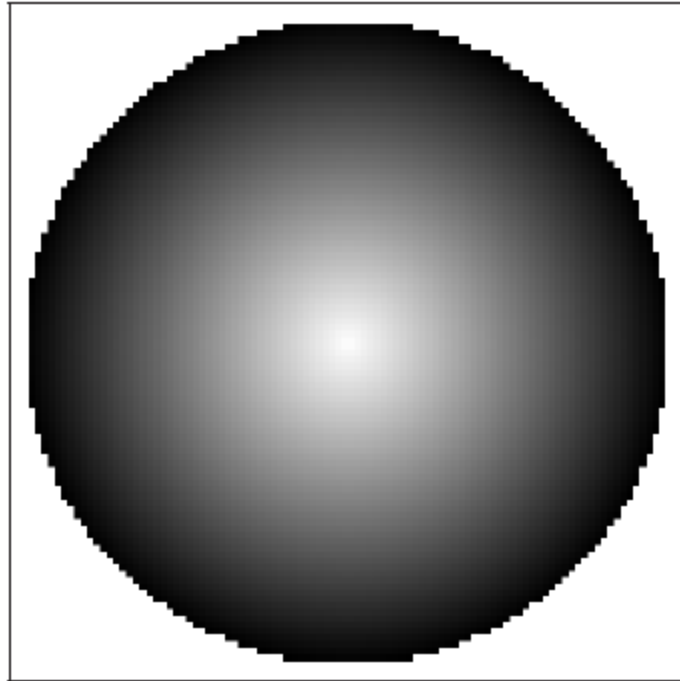
練習3.3c)

c) 3つの異なる値 x, y, z が与えられたときの中央値を求める `median(x,y,z)`. 中央値とは大きさ順に並べたときに真ん中に来る値のことである。(median.rb というファイルを作れ。)

進捗状況の確認

1. できた(できた時点で投票してください)
2. まだ

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

```
>> image = Array.new(6)
```

```
=> [nil , nil , nil , nil , nil , nil]
```

```
>> a = Array.new(3)
```

```
=> [nil , nil , nil]
```

繰り返しによって、 配列の全要素をそれぞれ変更する

```
>> for i in 0..2
```

```
>>   a[i] = 0
```

```
>> end
```

```
=> 0..2
```

```
>> a
```

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

練習 (make1d)

- 大きさ n で中身が全て 0 であるような 1 次元配列を作る関数 `make1d(n)` を定義せよ。
(`make1d.rb` というファイルに格納する。)

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

変数 `a` に入っている配列が
関数の値として返される

時間がないので...

- 共通資料の配布プログラムから、make1d.rbをダウンロード

```
>> load("./make1d.rb")
```

```
=> true
```

```
>> make1d(3)
```

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

2次元配列を作る

```
>> for i in 0..5
```


```
>>   image[i] = make1d(3)
```

```
>> end
```

```
=> 0..5
```

```
>> image
```

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



6行3列の配列

練習 (make2d)

- h 行 w 列の配列を作る `make2d(h,w)` を定義せよ。ただし、作られる配列の中身は全て 0 にせよ。(もちろん、`make1d` を使う。`make2d.rb` というファイルに格納する。)
 - `Array.new` によって配列を作るのを忘れずに。
 - 最後に配列を関数の値として返すのを忘れずに。

時間がないので...

- 共通資料の配布プログラムから、make2d.rbをダウンロード

```
>> load("./make2d.rb")
```

```
=> true
```

```
>> make2d(6, 3)
```

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

2重の繰り返し

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

変数 image に入っている配列が
関数の値として返される

練習3.2

- (sphere.rb をダウンロード。)
- 式(3.2)の計算をする関数 $b(s,x,y)$ を定義せよ。
(sphere.rb の中で定義すればよい。)

$$b(s, x, y) = \begin{cases} \frac{r - d(x, y, r, r)}{r} & (d(x, y, r, r) \leq r) \\ 1 & (d(x, y, r, r) > r) \end{cases} \quad (3.2)$$

(ただし $r = s/2.0$)

(r,r) と (x,y) の間の距離

- `show(sphere(100))` を実行して表示される画像を確かめよ。

進捗状況の確認

1. `show(sphere(100))` がうまく行った時点で投票してください。
2. `show(sphere(100))` がうまく行かない。
3. `b` が定義できた
4. まだ