

# 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 &gt; y</code>	$>$	xがyより大きい
<code>x &gt;= 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 &lt; y</code>	$<$	xがyより小さい
<code>x &lt;= 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 &gt; y    x == 0</code>	x > y <u>または</u> x == 0
<code>x &lt; y &amp;&amp; y &lt; z</code>	x < y <u>かつ</u> y < z
<code>!(x &lt; y &amp;&amp; y &lt; 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

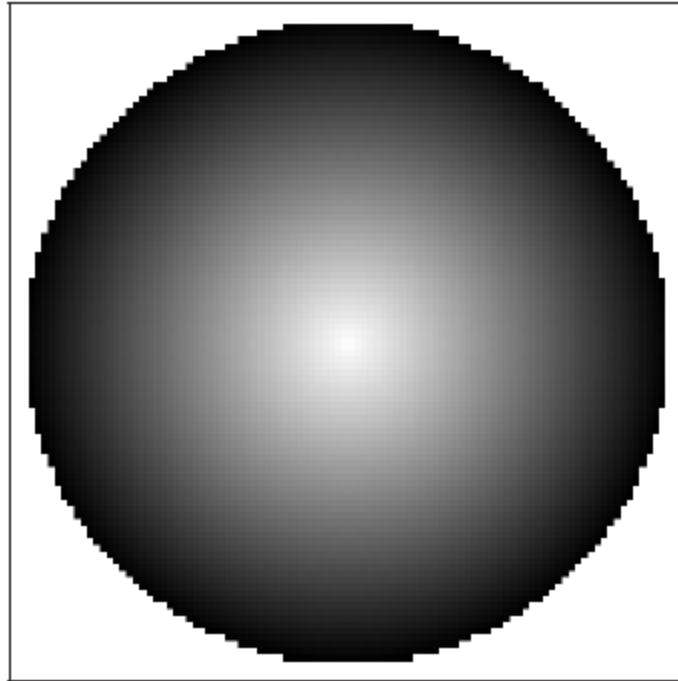
- 色々な比較

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>

# 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]
```

# make1d & make2d

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

```
=> true
```

```
>> make2d(2,3)
```

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

```
>> image = make2d(3,3)
```

```
=> [[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

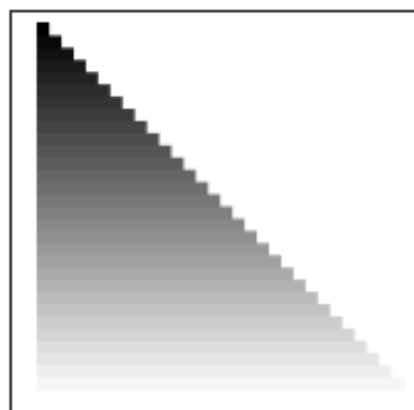
# 画面のファイルへの保存

- アプリケーションのユーティリティにある「グラブ」を起動する。
- メニューバーの「取り込み」メニューの「スクリーン」を選択し、指示に従って操作する。
- メニューバーの「ファイル」メニューの「保存」を選択する。ファイル名を聞かれるのでファイル名を入力する。特に指定しないと、「書類」のフォルダに保存されるので注意。
- レポートのメールにファイルを添付する。
  - 「Windows 対応の添付ファイルを送信」にチェックが入っていることを確認せよ。

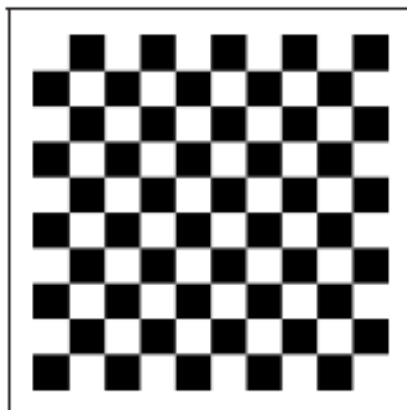
# クラブできたか？

1. できた
2. まだ

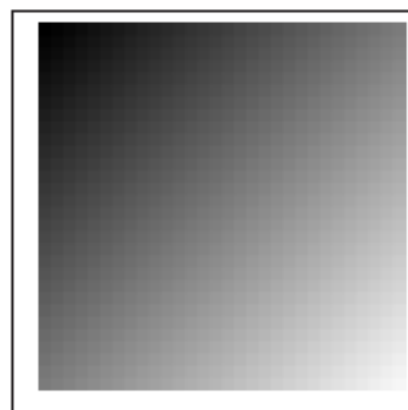
練習 26: (色々な図形\*) 次のような画像を作成する関数をそれぞれ定義せよ。



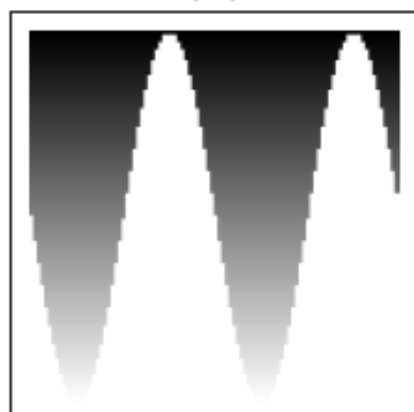
(a)



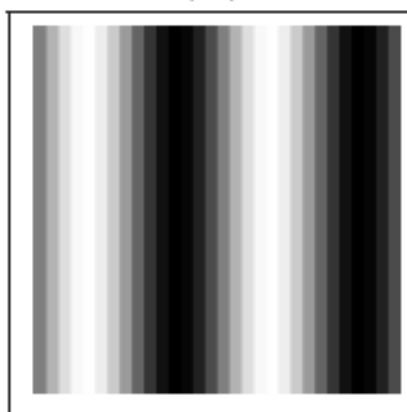
(b)



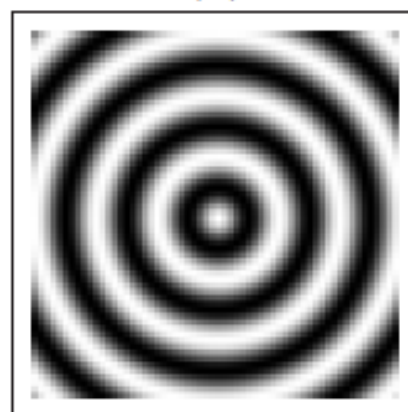
(c)



(d)



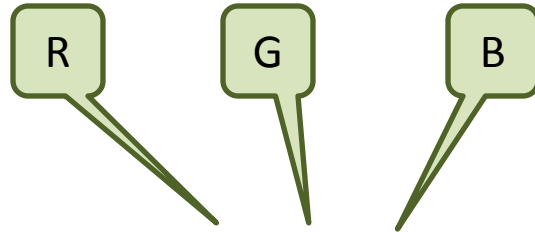
(e)



(f)



# カラー画像の表現



```
>> 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
```

## 3.5 定義のまとめ

条件分岐:  $\boxed{\text{式}_1}$  が成り立つときは  $\boxed{\text{式}_2}$  を、そうでないときは  $\boxed{\text{式}_3}$  を計算する条件分岐は次のように書く。このとき  $\boxed{\text{式}_1}$  を条件式という。

```
if  $\boxed{\text{式}_1}$   
   $\boxed{\text{式}_2}$   
else  
   $\boxed{\text{式}_3}$   
end
```

真偽値: true と false はそれぞれ、真と偽を表わす値である。

# 真偽値を与える論理演算

irb(main):003:0>  $x = 3$

=> 3

irb(main):004:0>  $1 < x$

=> true

irb(main):005:0>  $x == 2$

=> false

```
def is_even(x)
```

```
  x%2 == 0
```

```
end
```

is\_even.rb

```
load("./is_even.rb")
```

```
def tnpo(n)
```

```
  if is_even(n)
```

```
    n/2
```

```
  else
```

```
    3*n + 1
```

```
  end
```

```
end
```

tnpo.rb

空の値: nil は「無い」ことを表わす特別な値である。

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すべて nil である。

繰り返し: 変数 の値を 式<sub>1</sub> の値から 式<sub>2</sub> の値まで 1 ずつ順に変化させながら、命令<sub>1</sub> から 命令<sub>n</sub> 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
```

# 練習問題確認プログラム

- ex03.rb (練習3.1と3.3-3.9)の結果を以下のアドレスに送れ。(3.7と3.8はやらなくてよい。)

[is-komaba@lyon.is.s.u-tokyo.ac.jp](mailto:is-komaba@lyon.is.s.u-tokyo.ac.jp)

- Subject:欄は、

[is-komaba](#)

としてください。

- ✕切: **11月3日 (祝日なので気を付けろ)**
  - 練習問題確認プログラムの結果は出席点の一部とします。結果の内容は問いません。(習得状況を知りたい。)