

真偽値・文字列・繰り返し

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
```

(やってみよう)
1.even? とやると何が起こる？

is_even.rb

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

```
def c(n)
```

```
  if is_even(n)
```

```
    n/2
```

```
  else
```

```
    3*n + 1
```

```
  end
```

```
end
```

c.rb

文字列

```
irb(main):003:0> s = "abra"
```

```
=> "abra"
```

```
irb(main):004:0> t = "cadabra"
```

```
=> "cadabra"
```

```
irb(main):006:0> u = s + t
```

```
=> "abracadabra"
```

```
irb(main):007:0> "123" + "456"
```

```
=> "123456"
```

```
irb(main):009:0> s.length()
```

```
=> 4
```

```
irb(main):010:0> (s + t).length()
```

```
=> 11
```

```
irb(main):012:0> s[0..0]
```

```
=> "a"
```

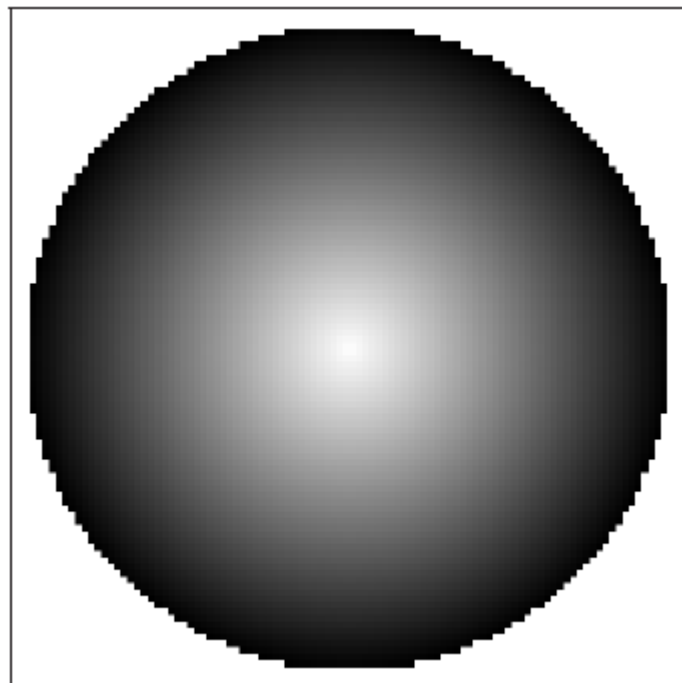
```
irb(main):013:0> s[1..2]
```

```
=> "br"
```

```
irb(main):014:0> t[1..(t.length()-1)]
```

```
=> "adabra"
```

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

- 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。

2次元配列を作る

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

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

```
>> 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]]
```

練習

- h 行 w 列の配列を作る $\text{make2d}(h,w)$ を定義せよ。ただし、作られる配列の中身は全て 0 にせよ。
- 式3.2 の計算をする関数 $b(r,x,y)$ を定義せよ (x, y) だけでなく r も引数となっていることに注意せよ)。

原点と (x,y) との間の距離

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

2重の繰り返し

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

sphere.rb

練習

- `show(sphere(20))` を実行して表示される画像を確認よ。

進捗状況の確認

1. `show(sphere(20))` がうまく行った時点で投票してください。
2. `sphere` を定義したが、`show(sphere(20))` がうまくいかない
3. `b(x,y,r)` までできた
4. `make2d` までできた
5. `make1d` までできた
6. まだ

3.4 定義のまとめ

真偽値: `true` と `false` はそれぞれ、真と偽を表わす値である。

文字列を作る: 2つの " 記号の間にある文字や記号は文字列を作る。

文字列の結合: `[式1] + [式2]` という式は、`[式1]`、`[式2]` の値が文字列のとき、それらの文字列をつなげた文字列を作る。

文字列の長さ: `[式1].length()` という式は、`[式1]` が表わす文字列の長さを求める。

部分文字列: `[式1][式2..式3]` という式は、`[式1]` が表わす文字列の `[式2]` 番目から `[式3]` 番目までを取り出した文字列を作る。

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
```

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？

```
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]]`

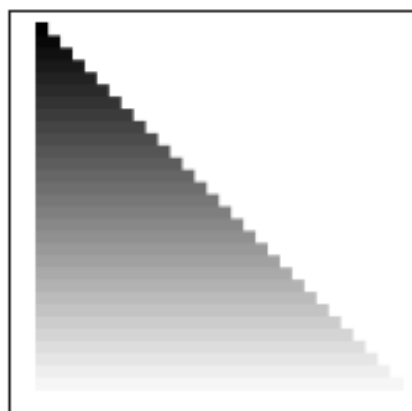
4. `[[1,1],[1,1]]`

5. `[0,0]`

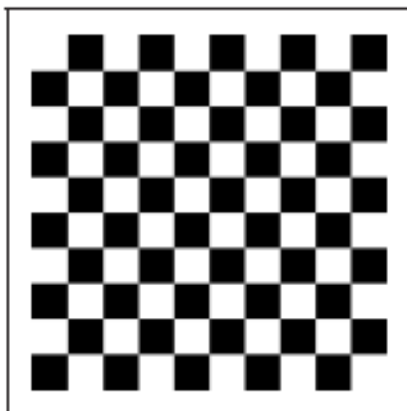
6. `[0,1]`

7. `[1,1]`

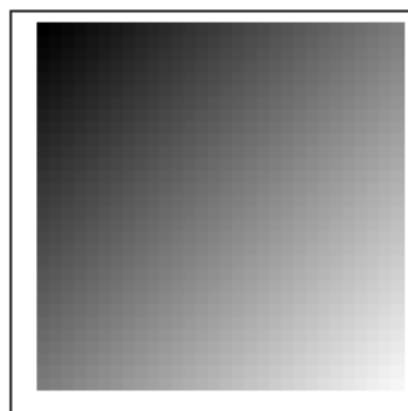
練習 26: (色々な図形*) 次のような画像を作成する関数をそれぞれ定義せよ。



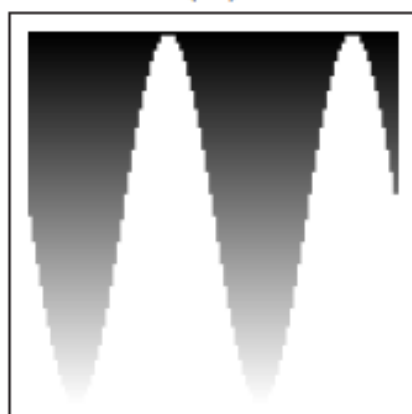
(a)



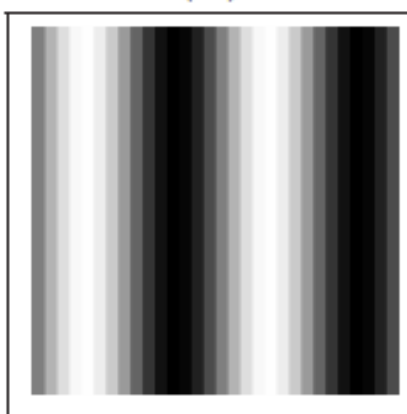
(b)



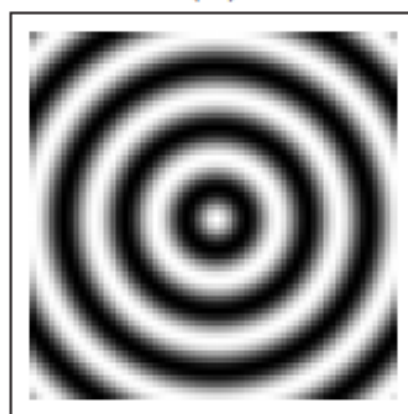
(c)



(d)



(e)



(f)