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"
```

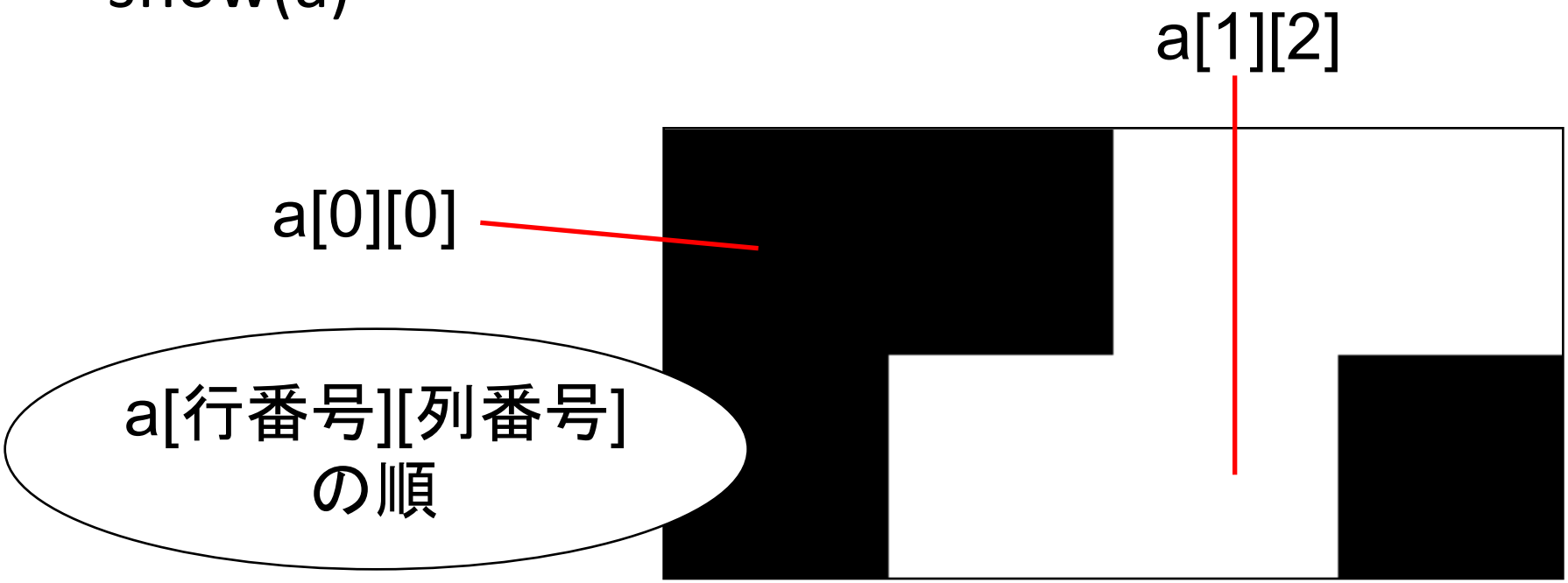
s = "abra"
t = "cadabra"

```
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"
```

復習: 図形の表示

- irbを一度終了してisrbを起動する
- a=[[0,0,1,1], [0,1,1,0]]
- show(a)



練習問題1.11 解説

- 局所変数を使わずに、計算回数を増やさずに同じ計算ができるか?

```
def heron(a,b,c)
    s = 0.5*(a+b+c)
    sqrt(s * (s-a) * (s-b) * (s-c))
end
```

NG

0.5*(a+b+c)が4回になる

```
def heron(a,b,c)
  sqrt(0.5*(a+b+c) * (0.5*(a+b+c) -a)
          * (0.5*(a+b+c) -b) * (0.5*(a+b+c) -c))
end
```

練習問題1.11 解説

```
heron(3,4,5)
=>heron_aux(0.5*(3+4+5),3,4,5)
=>heron_aux(6,3,4,5)
=>sqrt(6*(6-3)*(6-4)*(6-5))
=>sqrt(36)
=>6
```

計算回数を増やさず

```
end
```

答:

```
def heron_aux(s,a,b,c)
  sqrt(s * (s-a) * (s-b) * (s-c))
end

def heron(a,b,c)
 heron_aux(0.5*(a+b+c),a,b,c)
end
```

※関数の引数は
展開する前に
計算される

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

```
>> a = Array.new(3)
=> [nil , nil , nil]
```

```
>> for i in 0..2
>>     a[i] = 0
>> end
=> 0..2
>> a
=> [0, 0, 0]
```

練習

完成したら
進捗アンケート

- 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(3)
[0, 0, 0]
```

ファイルmake1d.rbに

```
a = Array.new(3)

for i in 0..2
    a[i] = 0
end

a
```

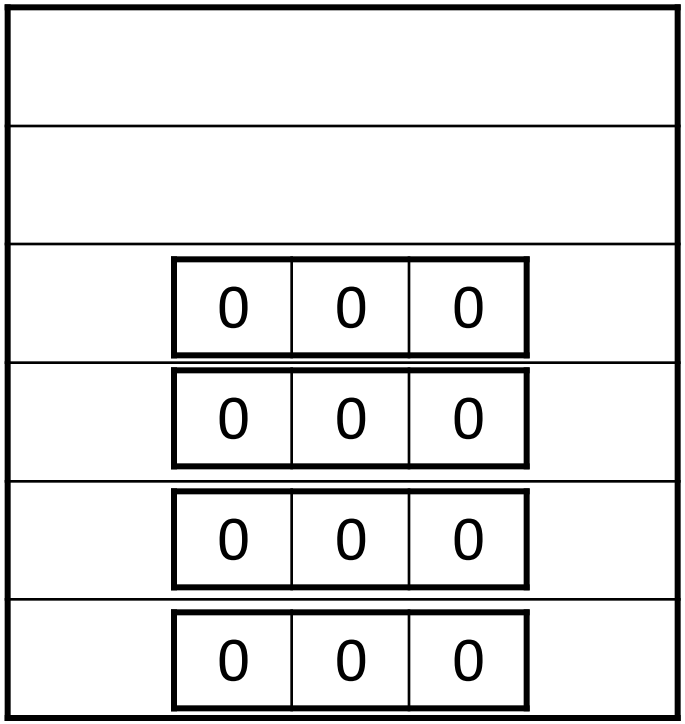
2 次元配列を作る

```
>> image = Array.new(6)
=> [nil , 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]]
```

0	0	0

image:



復習

$$\mathrm{sign}(x) = \begin{cases} -1 & (x < 0) \\ 1 & (0 < x) \\ 0 & (\text{それ以外}) \end{cases}$$

sign(x)として正しい関数は?

投票: 1

```
def sign(x)
  if x < 0
    1
  end
  if 0 < x
    1
  end
  if x == 0
    0
  end
end
```

2

```
def sign(x)
  if x < 0
    -1
  else
    if 0 < x
      1
    else
      0
end
```

3

```
def sign(x)
  if x < 0
    -1
  else
    if 0 < x
      1
    else
      0
    end
  end
end
```

4

```
def sign(x)
  if x < 0
    -1
  if 0 < x
    1
  else
    0
  end
end
```