アルゴリズム入門
第２回

千葉　滋

東京大学
情報理工学系研究科
創造情報学専攻

1

感想より

- この講義を落としたら、留年なので、本当によろしくお願いします。
- PC弱者なので厳しい授業になると思うががんばってついていきたい。
- スライドが遠くから見にくい

2

感想より

- pythonに於けるreturn的なものはrubyに存在しないのですか？
  - ありますが、最後の文なら省略可能です
- 少しPythonやるかと期待してました
  - そんなに違いありません
  - def bmi(h, w):
    return w / (h / 100.0) ** 2

```
def bmi(h, w)
  w / (h / 100.0) ** 2
end
```

print

- x = 3.0　などと表示したい
- a = 3
  print “x=“, a, “\n”

\は option と ¥ の同時押し。 \n は改行の意。

- 「プログラミングは普通、表示の仕方から教えるじゃないですか…」

今週の課題の準備

- show 関数を定義する
  - 教科書では isrb で show 関数を使うが、我々は iruby を使うので…
  - Web の講義資料の show 関数の定義を iruby に copy & paste して、shift + return
    - 最初に一回 shift + return で実行すればよい
    - 定義されたかテスト

括弧 () とカギ括弧 [] を区別

```
In [4]: show([[1,0], [0, 1]])
```

絵がでればOK

```
Out[4]: true
```

中身

- 40行もある

jupyter show.rb ✔ 3 hours ago

File Edit View Language

```
require "zlib"

def show(image)
  def chunk(type, data)
    [data.bytesize, type, data, Zlib.crc32(type + data)].pack("NA4A*N")
  end

  def make_png(raw_data)
    height = raw_data.size
    width = raw_data[0].size

    s = "\x89PNG\r\n\x1a\n".force_encoding("ascii-8bit")
    s1 = chunk("IHDR", [width, height, 8, 2, 0, 0, 0].pack("NNCCCCC"))
    s2 = chunk("IDAT", Zlib::Deflate.deflate(raw_data.map {|line|
                         ([0] + line.flatten).pack("C*") }.join))
    s3 = chunk("IEND", "")
    s + s1 + s2 + s3
  end

  def to_8bit(c)
    if c.is_a?(Array)
      if c[0].is_a?(Numeric)
        c.map{|e| (e * 255).to_i }
      else
        raise "show(): bad array element " + c.to_s
      end
    else
      [(c * 255).to_i] * 3
    end
  end

  if !image.is_a?(Array) || !image[0].is_a?(Array)
    puts "show(): not a 2d array" ; false
  else
    h = image.size
    s = h < 10 ? 20 : h < 60 ? 10 : h < 400 ? 400 / h : 1
    data = image.flat_map{|r| [r.flat_map{|c| [to_8bit(c)] * s }] * s}
    IRuby.display make_png(data), mime: "image/png" ; true
  end
end
```

毎週やるのは面倒なので

- プログラムをダウンロードして保存
  - show.rb という名で保存
    - 保存したものをファイルと呼ぶ

Web ページ上の「ダウンロード」をクリックするとダウンロードされて、Downloads フォルダの中に保存されるはず

- load で読み込む

保存されたプログラムが実行される

  - ノートブック .ipynb と同じフォルダ

```
In [5]: load "show.rb"
Out[5]: true
```

  - Downloads フォルダに show.rb がある

```
In [6]: load "~/Downloads/show.rb"
Out[6]: true
```


ファイル（というメタファー）

- コンピュータ内に保存されたデータ（文書や画像、プログラムなど）

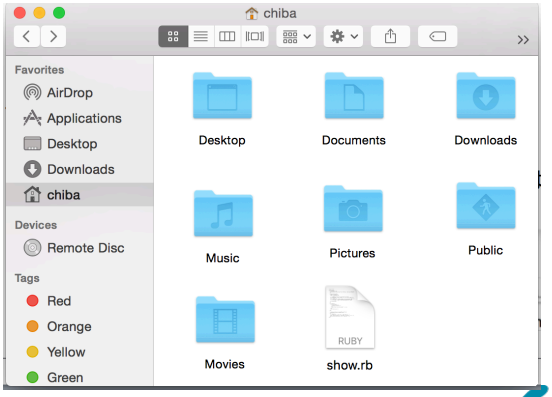
show.rb

- テキスト・ファイル
  - テキスト形式で保存された文書ファイル
  - 文字コードをただ並べて保存しただけ。装飾等はなし。
  - 色々なエディタで編集可能

Emacs.app

フォルダ（ディレクトリ）

- ファイルの入れ物？　しまう場所？
  - 入れ子にできる

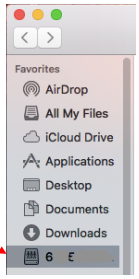


Path 名

- ファイルの指定方法
  - どのフォルダ（ディレクトリ）にあるファイルか？
- 起点となるフォルダ
  - ~　　home

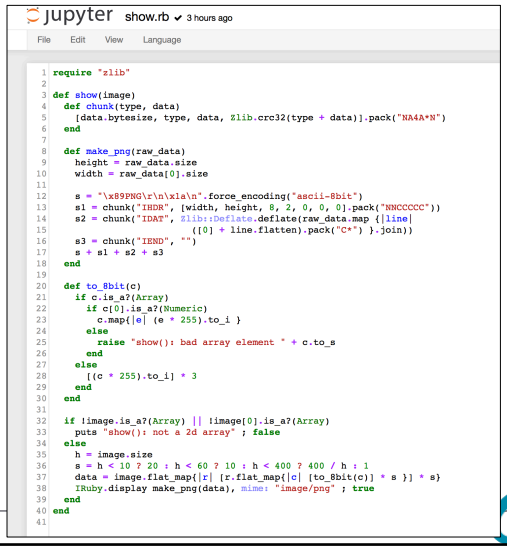
- フォルダの入れ子
  - ~/Downloads
    - home フォルダ内の Downloads フォルダ
  - ~/Downloads/show.rb
    - home フォルダ内の Downloads フォルダ内の show.rb
  - show.rb　(./show.rb の省略形)
    - 今、着目している（具体的にどこかは色々）フォルダ内の show.rb

演習棟の場合



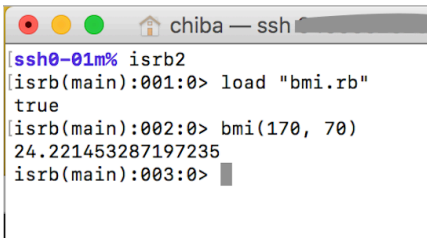
ファイルの中身を編集・保存する

- エディタで編集
- それを
  load で読み込んで実行する
  流儀もある
  - こちらがむしろ普通



教科書 isrb2

- プログラム全体をエディタで編集
- ターミナルから isrb2 を実行
- load “ex2.rb” のようにプログラムを読み込んで実行

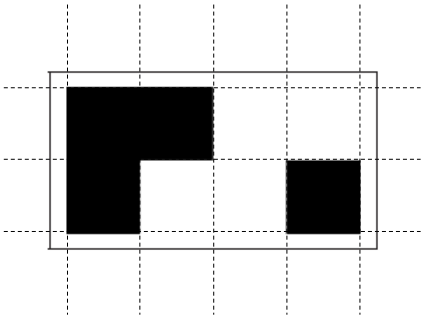


データの表現　画像の表現 (p.26)

- 配列を使って絵を描く
  - 0 は黒、1 は白

```
In [4]: a = [[0,0,1,1],
             [0,1,1,0]]
        show(a)
```

```
Out[4]: true
```



配列 (array)

- 行列？　ベクトル？

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

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

- `a[i][j]`　a_{ij}

  a_{00} が左上すみ

あるいは

$$\begin{pmatrix} v_0 \\ v_1 \end{pmatrix}$$

ただし

$v_0 = \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix}$

$v_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}$

画像の操作 (p.27)

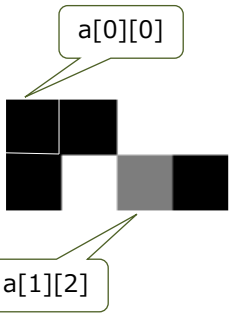
- 続き

```
In [5]: a[0][0]
Out[5]: 0

In [6]: a[0][2]
Out[6]: 1

In [7]: a[1][2] = 0.5
Out[7]: 0.5

In [8]: show(a)
```



代入？？

- 既に値が決まっている変数の値を異なる値に変えられる。
  - それ以降の行では、新しい値を使って計算する。

```
a = [[0,0,1,1],
     [0,1,1,0]]
show(a)
a[1][2]=0.5
show(a)
```

変数の値を調べるタイミングによって
値が異なる可能性がある

カラー画像の表現 (p.28)

- 三原色の明るさを指定

Red Green Blue

```
In [3]: # 3x2 のカラー画像
        d = [[[0 ,0 ,0], [0 ,1 ,0], [0 ,0 ,1]],
             [[1 ,0 ,0], [1 ,1 ,0], [1 ,0 ,1]]]
        show (d)
```

```
Out[3]: true
```


コメント

- プログラム中で
  # から行末までは何を書いても無視される
  - 後でプログラムを読むときの助けになるように、プログラムの説明をメモ的に書くのに使われる
  - たくさん書く方がよいという流儀も、
    そうでない流儀もある
    - そうでない流儀では、変数名や関数名を意味ある名前にする

3次元配列 (array)

- ベクトルのベクトルのベクトル？
- `a = [[[0,1,1],[0,1,0],[1,0,1]],`
  `[[1,0,0],[0,0,0],[1,1,1]]]`
- `a[i][j][k]` a_{ijk}

$$\begin{pmatrix} v_0 \\ v_1 \end{pmatrix}$$

ただし

$$v_0 = \begin{pmatrix} w_0 & w_1 & w_2 \end{pmatrix}$$
$$v_1 = \begin{pmatrix} u_0 & u_1 & u_2 \end{pmatrix}$$

$$w_0 = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}$$
$$w_1 = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$$
$$\vdots$$

a_{000} が左上すみ

本日の課題

- 練習2.4 (国旗, p.31)

flag()という関数を定義してみる

```
def flag()
  r=[1,0,0]
  w=[1,1,1]
  [[r,r,r,r,r],
   [r,r,w,r,r],
   [r,w,w,w,r],
   [r,r,w,r,r],
   [r,r,r,r,r]]
end
a = flag()
show(a)
```




関数の定義

- 式をいくつも書ける
  - 最後の式が本体
  - flag の後ろの () は、引数なし、の意味。
    - 省略してもよい
- 名前を変えれば関数はいくつも定義できる
  - 関数名や変数名は小文字から始める

```
def flag()
  r = [1,0,0]
  w = [1,1,1]
  [[r,r,r,r,r],
   [r,r,w,r,r],
   [r,w,w,w,r],
   [r,r,w,r,r],
   [r,r,r,r,r]]
end
```