

プログラミング演習I

– 第8回 –

メソッド その2

1 はじめに

第8回の演習の目標は、

- メソッドからメソッドを呼ぶ方法を理解する.
- メソッドの扱いに慣れる.

です.

2 メソッドが別のメソッドを呼ぶ

次のようなプログラムはもう何度か出てきた.

```
-----  
# hello と表示するメソッド  
def hello  
  print("hello\n")  
end
```

```
hello  
-----
```

ここで、hello メソッドの中の print もメソッドであることに注意しよう。つまり、メソッドの中で別のメソッドを呼び出していることになる。メソッドの中でメソッドを呼ぶことは可能である。もちろん自分の定義したメソッドを呼ぶことも問題ない。

演習 8-1

hello メソッドを 3 回呼ぶメソッドを作って試してみよう。

```
-----  
# hello と 1 回表示するメソッド  
def hello  
  print("hello\n")  
end
```

```
# hello と 3 回表示するメソッド  
def hello_3  
  i = 0  
  while i < 3  
    hello  
    i = i + 1  
  end  
end
```

```
hello_3  
-----
```

演習 8-2

三角柱の体積を求めるメソッドを定義しよう。以下の手順で行う。

1. メソッドを定義せず、三角柱の体積を求めるプログラムを書く。
2. 三角柱の体積を求める部分をメソッドとして定義する。
3. 三角形の面積を求めるメソッドを定義し、三角柱の体積を求めるメソッドから呼び出す。

その 1

```
-----  
base = 3  
height_1 = 4  
height_2 = 5  
print(base * height_1 / 2.0 * height_2, "\n")  
-----
```

その 2

```
-----  
# 三角柱の体積を求めるメソッド  
# 引数 base: 底辺  
# height_1: 底面の三角形の高さ  
# height_2: 三角柱の高さ  
# 戻り値 三角柱の体積
```

```

#
def triangular_prism(base, height_1, height_2)
  return base * height_1 / 2.0 * height_2
end

base = 3
height_1 = 4
height_2 = 5
print(triangular_prism(base, height_1, height_2), "\n")
-----

```

その3

```

-----
# 三角形の面積を求めるメソッド
# 引数 base: 底辺
#       height: 高さ
# 返回值 三角形の面積
#
def triangle(base, height)
  return base * height / 2.0
end

# 三角柱の体積を求めるメソッド
# 引数 base: 底辺
#       height_1: 底面の三角形の高さ
#       height_2: 三角柱の高さ
# 返回值 三角柱の体積
#
def triangular_prism(base, height_1, height_2)
  return triangle(base, height_1) * height_2
end

base = 3
height_1 = 4
height+2 = 5
print(triangular_prism(base, height_1, height_2), "\n")
-----

```

(重要) メソッドに対するコメントの例を示した。ポイントは、以下を適切に書くことである。十分練習しよう。

- 何をするメソッドか？
- 引数は何か？変数名に対応させて意味を書く
- あるか？ないか？あるなら何を返すのか？

演習 8-3

三角錐の体積を求めるメソッドを定義しよう。三角形の面積を求めるメソッドを呼び出す場合と、三角柱の体積を呼び出す場合を考えてみよう。どちらが便利だろうか？

演習 8-4

三角形の面積を使うような操作を考えてメソッドとして定義し、そのメソッドから `triangle` を呼び出してみよう。

3 返り値を使う

演習 8-5

以下の成績評価プログラムを試してみよう。合否判定メソッドの返り値によって処理を分岐している例である。

```
-----  
# 合否判定メソッド  
# 引数 tensu: 点数  
# 返り値 合格か不合格 (合格: true, 不合格: false)  
#  
def goukaku?(tensu)  
  if tensu >= 60  
    return true  
  else  
    return false  
  end  
end  
  
print("あなたの点数を入れてください")  
tensu = gets.chomp.to_i  
  
if goukaku?(tensu)  
  print("おめでとうございます。合格です。 \n")  
else  
  print("不合格です。追試を受けてください。 \n")  
end  
-----
```

(補足) `true` か `false` を判定するようなメソッドの名前の末尾には、慣例として `?` が付く。「どう?○?×?」というニュアンスが伝わるので理解しやすいと思う。

演習 8-6

合否判定メソッドを改良して、成績判定メソッドを作ってみよう。メソッドは点数を引数とし、成績 (A か B か C か D) を返すように作成する。成績ごとにメッセージを考えて表示しよう。

```
4 i = 0
  while i < 10
    練習
  end
```

演習 8-7

さあ、練習練習。この節の見出しが意味する練習回数は何回？(_____) 回

演習 8-8

正の整数 n を引数とし、1 から n までの和を返り値とするメソッドを作って使ってみよう。

演習 8-9

文字列を要素とする配列と 1 つのキーとなる文字列を引数とし、配列にキーとなる文字列が含まれるかどうかを判定するメソッドを作って使ってみよう。

演習 8-10

2 つの数値と四則演算子の記号を引数とし、計算結果を返すメソッドを作って使ってみよう。

演習 8-11

お店をキーとし値段を値とするハッシュを受け取って、値段の平均と最安値のお店と値段を表示するメソッドを作って使ってみよう。

演習 8-12

顔文字を引数とし、別の顔文字に変換して返すメソッドを作成しよう。変換ルールは複数設定すること。例：変換ルール $\wedge \rightarrow @$ と $\rightarrow o$

入力: (\wedge . \wedge) (\wedge _ \wedge)

出力: (@o@) (@_@)

参考：顔文字図書館 (<http://www.kaomoji.com/kao/text/>)

演習 8-13

演習 7-18 を、引数の配列の要素数によらず最大値を返すように変更するにはどうしたよいか。

演習 8-14

文字列の配列を引数とし、それらのうち最大の長さの文字列を返すメソッドを作って使ってみよう。

演習 8-15

2 つの正の整数 x, y を引数とし、べき乗 x^y を返り値とするメソッドを定義しよう。結果は $x ** y$ と比較せよ。また、 x, y が正の整数でないときにはどうなるか確認してみよう？

演習 8-16

球の体積を求めるメソッドを作成しよう。ただし、べき乗の計算は演習 8-15 で作成したメソッドを使うこと。