プログラミング演習 I
– 第 7 回 –
メソッド その 1

1　はじめに

第 7 回の演習の目標は,

- メソッドとは何かを知る.
- メソッドの使い方を知る.
- メソッドを自分で作る（定義する）.

です.

2　簡単なメソッドを作る

教科書

p.52(p.29)- メソッドの作成

演習 7-1
次を入力し，出力を確かめよう.

```
------------------------------------------------
# 挨拶を表示するメソッド
def hello
  print("やあ，こんにちは．\n")
end

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

演習 7-2
次を入力し，出力を確かめよう．

```
-----------------------------------------------
# 消費税を計算するメソッド
#    引数 nedan: 商品などの値段
def zeikin(nedan)
  print(nedan * 0.05, "円\n")
end

zeikin(300)
zeikin(2600)
-----------------------------------------------
```

3　プログラミングの前に...

教科書

p.107(p.107)- メソッド

演習 7-3
一般にメソッド呼び出しは以下の構文にしたがいます．

オブジェクト . メソッド (引数,...)

いままで出てきたメソッドを列挙してみよう．

演習 7-4
次の言葉の意味を調べよう．

インスタンスメソッド

クラスメソッド

関数的メソッド

なお，本演習では関数的メソッドのみを扱う．

4　基本問題

演習 7-5
以下のプログラムを作ってみよう.
その 1

```
-----------------------------------------------
print("うにさん，こんにちは．\n")
print("かにさん，こんにちは．\n")
print("おにさん，こんにちは．\n")
-----------------------------------------------
```

その 2

```
-----------------------------------------------
# 挨拶を表示するメソッド
#   引数なし
def hello
  print("こんにちは．\n")
end

print("うにさん，")
hello
print("かにさん，")
hello
print("おにさん，")
hello
-----------------------------------------------
```

その 3

```
-----------------------------------------------
# 挨拶を表示するメソッド
#   引数 name: 挨拶を言う相手の名前（文字列）
def hello(name)
  print(name, "さん，こんにちは．\n")
end

hello("うに")
hello("かに")
hello("おに")
-----------------------------------------------
```

演習 7-6

次の用語を調べよう.

引数

演習 7-7

演習 7-5 その 3 のプログラムで，while を使って名前を 3 回キーボードから入力できるようにして，名前を入力するたびに挨拶をするプログラムにしよう.

演習 7-8

演習 7-5 その 3 は 3 人の名前を配列で渡すようにすることもできる．次の例で確かめよう.

```
------------------------------------------------
# 挨拶を表示するメソッド
#   引数 name: 挨拶を言う相手 3 人分の名前
#              各要素が文字列の要素数 3 の配列
def hello(name)
  i = 0
  while i < 3
    print(name[i], "さん，こんにちは．\n")
    i = i + 1
  end
end

name = ["うに", "かに", "おに"]
hello(name)
------------------------------------------------
```

演習 7-9

3 回リンリンと鳴く（表示する）メソッド suzumushi を while 文を使って定義し使ってみよう.

演習 7-10

上のメソッドを引数として指定した回数だけ鳴くメソッドに変更して使ってみよう.

演習 7-11

言葉の意味を調べよう.

返り値

演習 7-12
次のプログラムを実行しよう．どんなメソッドだろうか？コメントを埋めよう．

```
-----------------------------------------------
# ××をするメソッド
#   引数:     doushi ○○
#   返り値:   ＊＊＊
def san_tan_gen(doushi)
  return doushi + "s"
end

print("動詞を入力してください> ")
doushi = gets.chomp
print("3 人称単数現在の場合は, ", san_tan_gen(doushi), "としてください\n")
-----------------------------------------------
```

演習 7-13
もう少し使えるメソッドにしてみよう．have の 3 人称単数現在の形は has だ．

```
-----------------------------------------------
# ××をするメソッド
#   引数:     doushi ○○
#   返り値:   ＊＊＊
def san_tan_gen(doushi)
  if doushi == "have"
    return "has"
  else
    return doushi + "s"
  end
end

print("動詞を入力してください> ")
doushi = gets.chomp
print("3 人称単数現在の場合は, ", san_tan_gen(doushi), "としてください\n")
-----------------------------------------------
```

演習 7-14
次の 2 つのプログラムを入力し，出力を確かめよう．2 つのメソッドの違いに注意．それぞれ，どんな時に便利か？

☆「表示する」メソッド

```
-----------------------------------------------
# 税込み価格を表示するメソッド
#   引数:   nedan 計算する商品の値段
#   返り値: なし
def zeikomi1(nedan)
  print(nedan * 1.08, "円\n")
end

zeikomi1(300)
zeikomi1(2600)
-----------------------------------------------
```

☆「返す」メソッド

```
-----------------------------------------------
# 税込み価格を返すメソッド
#   引数:   nedan 計算する商品の値段
#   返り値: nedan の税込み価格
def zeikomi2(nedan)
  return nedan * 1.08
end

print(zeikomi2(300), "円\n")
print(zeikomi2(2600), "円\n")
-----------------------------------------------
```

演習 7-15
次のメソッドは直方体の体積を計算して「**返す**」メソッドである．

```
-----------------------------------------------
# 直方体の体積を返すメソッド
#   引数:   x, y, z それぞれ直方体の 1 辺の長さ
#   返り値: 直方体の体積
def volume(x, y, z)
  return x * y * z
end

print(volume(2, 3, 4), "\n")
print(volume(10, 20, 30), "\n")
```

```
x = 2
y = 3
z = 4
print(volume(x, y, z), "\n")
-----------------------------------------------
```

演習 7-16
このメソッド area は何を計算しているか？

```
-----------------------------------------------
# ××を計算するメソッド
#   引数:  x ○○
#          y △△
#          z □□
#   返り値: ＊＊＊
def area(x, y, z)
  return (x * y + y * z + z * x) * 2
end

print(area(2, 3, 4), "\n")
print(area(10, 20, 30), "\n")

a = 2
b = 3
c = 4
print(area(a, b, c), "\n")
-----------------------------------------------
```

[**注**] メソッドの引数に使用している変数名と呼び出すときの変数名を同じにする必要はない．演習 7-15 と 7-16 を比べてみよう．

演習 7-17
次のメソッド max は，2 つの数のうち大きい方を返すメソッドである．これをもとに小さい方を返すメソッド min を作って，使ってみよう．

```
-----------------------------------------------
# 2 つの数のうち大きい方を返すメソッド
#     引数: a, b （大きさを比較する 2 つの数）
#     返り値: a と b のうち大きい方の数
def max(a, b)
  if a > b
    return a
  else
```

```
    return b
  end
end

print(max(2, 3), "\n")
-----------------------------------------------
```

演習 7-18
配列で渡す場合は次のようになる.

```
-----------------------------------------------
# 2 つの数のうち大きい方を返すメソッド
#     引数: a （大きさを比較する 2 つの数を要素とする配列）
#     返り値: a の 2 つの要素のうち大きい方の数
def max(a)
  if a[0] > a[1]
    return a[0]
  else
    return a[1]
  end
end

b = [2, 3]
print(max(b), "\n")
-----------------------------------------------
```

演習 7-19
2 つの数の平均を求めるメソッド

```
-----------------------------------------------
# 2 つの数の平均を求めるメソッド
#   引数:   x, y 平均を求める 2 つの整数
#   返り値: 2 つの整数の平均
def heikin(x, y)
  return (x + y) / 2
end

print(heikin(3, 7), "\n")
print(heikin(3, 4), "\n")
-----------------------------------------------
```

演習 7-20
演習 7-19 を変更して，割り切れない場合も正しい平均を返すメソッドにするにはどうしたらよいか?

変数のスコープ（ローカル変数）

教科書

p.61(p.63)- ローカル変数のところ

演習 7-21
次の出力を確認して，2 つの変数 uni の関係を考えよう．

```
-----------------------------------------------
# 値を表示するメソッド
#   変数のスコープを確認するためのサンプル
#   引数:   uni 表示する変数
def uni_nedan(uni)
  print("たしかに，うには", uni, "円\n")
  uni = 300
  print("本当は，うには", uni, "円\n")
end

uni = 100
print("うには", uni, "円\n")
uni_nedan(uni)
print("でも，やっぱり，うには", uni, "円\n")
-----------------------------------------------
```

5　練習，練習

演習 7-22
演習 7-8 のメソッド hello を，引数を変えずに配列のサイズが 3 以外でも使えるように変更しよう．

演習 7-23
演習 7-13 をもうちょっと使えるものにしたい．まずは，ルールは以下の通りに変更しよう．

1. 普通は s をつける
2. 末尾が y なら y を ies に変える
3. have なら has にする

さて，他にどんなルール，あるいはルールの改良が必要か？

演習 7-24
配列を引数とし，含まれる値を全て「**表示する**」メソッドを作成しよう．また，表示を後ろからにするメソッドを作成しよう．

演習 7-25
2 つの文字列を引数とし，連結した文字列を「**返す**」メソッドと「**表示する**」メソッドを考え，2 つのメソッドの定義と呼び出しを含んだプログラムを作ってみよう．ただし，以下の通りプログラムの構成をコメントで書いてから，これにあわせてプログラムを書くこと．

```
# 連結した文字列を返すメソッド（メソッド 1）
#   引数：文字列 2 つ
#   返り値：連結した文字列

# 連結した文字列を表示するメソッド（メソッド 2）
#   引数：文字列 2 つ
#   返り値：なし

# 2 つの文字列をキーボードから入力

# メソッド 1 を呼ぶプログラム

# メソッド 2 を呼ぶプログラム
```

演習 7-26
年齢によって飲酒可能かどうかを判定するメソッドを次の 2 通り作ってみよう．引数はどちらも年齢である．

1. 飲酒可能かどうかを「**表示する**」メソッド．
2. 飲酒可能かどうかを「**返す**」メソッド．（可能なら true，だめなら false を返す）

演習 7-27
球の体積，表面積を「**求める**」メソッドをそれぞれ定義し，使ってみよう．なお，半径と高さをキーボードから入力できるようにする．また，π は，`Math::PI` を利用する．
(チェックのための例：半径が 5 のとき，体積は 523.598775598299，表面積は 314.159265358979)
[Q] 「求める」メソッドと言われたら,「表示する」べきか，それとも「返す」べきか？

演習 7-28
演習 7-18 のメソッドを，配列を引数とし，配列に含まれる数の平均を返すように変更してみよう．

演習 7-29
与えられた数までの九九表を「**表示する**」メソッドを作成しよう．

演習 7-30
演習 7-23 を参考にして，動詞の現在形を過去形へ変換するメソッドを作ってみよう．

1. 普通は ed をつける
2. ....