プログラミング演習I
– 第6回 –
文字列，復習

1　はじめに

第 6 回の演習の目標は，

- 文字列についてのさまざまな操作を理解する．
- 配列，ハッシュ，条件判断，繰り返しについて理解を深める．

です．

2　文字列あれこれ

教科書

p.275(p.273)- 文字列 (String) クラス

演習 6-1
次の 2 行の出力の違いを確認しよう．2 行目と同じ出力になるように 1 行目を変更するにはどうすればよいだろう．ただし，文字列は"(ダブルクォート) の中身だけを変更することとする．

```
------------------------------------------------
print("hoge\n")
print('hoge\n')
------------------------------------------------
```

演習 6-2
キーボードから入力した回数（整数）だけ,「カッコー」と鳴く（表示する）プログラムを作ろう．

演習 6-3
キーボードからいろいろな文字列を入力して，出力を確かめよう．特に，日本語の場合について考えてみよう．

```
------------------------------------------------
line = gets.chomp
print(line.length, "\n")
print(line.size, "\n")
print(line.bytesize, "\n")
------------------------------------------------
```

演習 6-4
文字列がつながることを確かめよう.

```
-----------------------------------------------
who = "私は"
what = "宙返りを"
where = "実習室で"
why = "寝坊したので"
how = "華麗に"

phrase_1 = who + what + "した"
print(phrase_1, ". \n")

phrase_2 = who + why + where + how + what + "した"
print(phrase_2, ". \n")
-----------------------------------------------
```

演習 6-5
文字列と配列は同じ？以下の出力を確認してみよう.

```
-----------------------------------------------
string = "abcdef"
array = ["a", "b", "c", "d", "e", "f"]
print(string[0], "\n")
print(string[1], "\n")
print(string[0].chr, "\n")
print(string[1].chr, "\n")
print(string[1,2], "\n")
print(string[1,3], "\n")
print(array[0], array[1], array[2], "\n")
-----------------------------------------------
```

演習 6-6
以下の出力を確認しよう.

```
-----------------------------------------------
print("aaa" == "baa")
print("aaa" == "aaa")
print("aaa" != "bbb")
print("aaa" < "bbb")
-----------------------------------------------
```

演習 6-7
キーボードから入力した自分の姓名が正しいかを判定するプログラムを作ろう.
補足：コマンドプロンプトの文字コードが Windows-31J であるため，gets で日本語を扱う場合には，1 行目を以下に書き換える必要がある.

```
#encoding: Windows-31J
```

演習 6-8
出力を確かめてメソッドの意味を確認しよう.

```
-----------------------------------------------
string = "知識五郎丸, ちしきごろうまる,185,99,B"
column = string.split(/,/)
i = 0
while i < column.size
  print(column[i], "\n")
  i = i + 1
end
-----------------------------------------------
```

演習 6-9
chomp と chop の違いを確かめてみよう.

```
-----------------------------------------------
line = gets.chomp
print(line, "\n")
print(line.chomp, "\n")
print(line.chop, "\n")
-----------------------------------------------
```

演習 6-10
出力を確かめてメソッドの意味を確認しよう.

```
-----------------------------------------------
string = "012345:-)9012345678901:-)56789"
print(string.index(":-)"), "\n")
print(string.rindex(":-)"), "\n")
print(string.include?(":-<"), "\n")
-----------------------------------------------
```

演習 6-11
次のプログラムの出力を確認しよう.

```
-----------------------------------------------
string = "abcdefg"
print(string, "\n")
print(string[2..4], "\n")
print(string[2, 4], "\n")
string[2] = "C"
print(string, "\n")
-----------------------------------------------
```

演習 6-12
次のプログラムの出力を確認しよう.

```
-----------------------------------------------
string = "abcdefg"
print(string, "\n")
print(string.slice!(2..3), "\n")
print(string, "\n")
-----------------------------------------------
```

演習 6-13
つぎのプログラムの出力を確認しよう.

```
-----------------------------------------------
string = "abc"
print(string, "\n")
string.concat("def")
print(string, "\n")
-----------------------------------------------
```

演習 6-14
つぎのプログラムの出力を確認しよう.

```
-----------------------------------------------
s = "abcabc"
print(s, "\n")
print(s.delete("b"), "\n")
print(s, "\n")
print(s.delete!("b"), "\n")
print(s, "\n")
-----------------------------------------------
```

演習 6-15
つぎのプログラムの出力を確認しよう.

```
-----------------------------------------------
string = "A man, a plan, a canal --Panama!"
print(string, "\n")
print(string.reverse, "\n")
-----------------------------------------------
```

演習 6-16
つぎのプログラムの出力を確認しよう.

```
-----------------------------------------------
print("hogehoge.hogehoho".count("h"), "\n")
print("abababcababababc".scan(/abc/).length, "\n")
-----------------------------------------------
```

演習 6-17
次のプログラムの出力を確認しよう.

```
-----------------------------------------------
string = "abcdefg"
print(string.tr("d", "D"), "\n")
print(string, "\n")
print(string.tr("a-z", "A-Z"), "\n")
print(string, "\n")
print(string.tr!("cde", "CDE"), "\n")
print(string, "\n")
-----------------------------------------------
```

演習 6-18
表 1 に文字列操作のメソッドをまとめよう.

表 1: 文字列操作のメソッド

メソッド名	説明,「たのしい Ruby」参照ページ	使い方の例
length		
size		
empty?		
split		
index		
rindex		
include?		
delete		
reverse		
strip		
upcase		
downcase		
swapcase		
capitalize		
tr		
concat		
count		
slice		

3　応用・復習問題

演習 6-19
キーボードから姓と名を入力すると，以下のことを実行するプログラムを作ろう．姓名の入力はアルファベットで行うとする．

1. 姓と名それぞれの長さを表示する
2. 姓と名の長さの和を表示する
3. 入力形式に関わらず，姓は全て大文字で表示し，名前は先頭だけ大文字で表示する．
4. 姓と名をこの順で空白 1 文字で区切って表示する．
5. 二人分の姓名を入力できるようにして，それぞれ上記と同様の処理をする．
6. 二人の姓名を姓のアルファベット順で表示する．

演習 6-20
要素数 10 の配列に整数を入れておく．キーボードから入力した整数と配列の各要素の大小を比較した結果を表示するプログラムを作成しよう．

演習 6-21
10 個の文字列の配列を用意し，それらのうち最大の長さの文字列を求めるプログラムを作成しよう．

演習 6-22
変数名を入力すると，本演習での ruby コーディング規約の命名規約に違反しているかどうかをチェックして，良い変数名を提案してくれるプログラムを作ろう．まずは，以下のことをやってくれるようにしてみよう．

1. 入力した変数名に記号類が含まれていた場合，‘_’ で置き換えたものを推薦してくれる．
   （例．`apple-juice` や `apple+juice` と入力したら，`apple_juice` と出力する）
2. 入力した変数名の先頭が大文字だった場合，小文字で置き換えたものを推薦してくれる．
   （例．`Apple` と入力したら，`apple` と出力する）
3. 入力した変数名が 2 つ目以降の単語を大文字で始める形式の場合，‘_’ で繋ぐ形式にしたものを推薦してくれる．
   （例．`appleJuice` と入力したら，`apple_juice` と出力する）

   その他に，どんなルールが必要か考えて言葉にしてみよう．その後で，それをプログラムとして書いてみよう．

4. （入力した変数名　　）場合，
   （　　）を推薦してくれる．
   （例．　　と入力したら，　　と出力する）
5. （入力した変数名　　）場合，
   （　　）を推薦してくれる．
   （例．　　と入力したら，　　と出力する）
6. （入力した変数名　　）場合，
   （　　）を推薦してくれる．
   （例．　　と入力したら，　　と出力する）