

例題 1 プログラムの基本を復習

数字の1～9の「花文字」を使って、プログラムの基本を復習

※花文字の例

```
      A      BBBB   CCCC
     A A     B  B   C  C
    A  A     BBBB   C
   AAAAAA   B  B   C
  A        A  B  B   C  C
 A          A BBBB   CCCC
```

(1) ex01_number1.rb

まずは、0～9 のそれぞれの花文字を1つずつ上から順番に表示してください。
花文字1つのサイズは 5×5 または 6×6 文字くらいがちょうどいいです。

[ヒント] 

• puts の羅列(簡単すぎ?)

• 0～9 のフォント(文字)を使うのがつまらない場合は
rbpad だけで使うことができる「絵文字」を使ってみよう!

※右クリック → [絵文字を挿入...]で絵文字のリストが出てくる



例題 1 プログラムの基本を復習

(2) ex01_number2.rb

上記の(1)のままだと、プログラムの実行のたびに 0～9 までの文字が表示されてしまうので、好きな数字の花文字を1つずつ簡単に表示できるようにプログラムを変更してみましょう。

0～9 の花文字を1つずつ表示するための `put_0`, `put_1`, ... `put_9` という名前のメソッドを作り、それらのメソッドを使って「1」「3」「0」の花文字を表示してください。

```
put_0 ... 0の花文字を表示するメソッド
put_1 ... 1の花文字を表示するメソッド
put_2 ... 2の花文字を表示するメソッド
put_3 ... 3の花文字を表示するメソッド
put_4 ... 4の花文字を表示するメソッド
put_5 ... 5の花文字を表示するメソッド
put_6 ... 6の花文字を表示するメソッド
put_7 ... 7の花文字を表示するメソッド
put_8 ... 8の花文字を表示するメソッド
put_9 ... 9の花文字を表示するメソッド
```

[ヒント] 

- ・メソッドとは、呼び出すだけでひとまとまりの処理をしてくれるもの
- ・独自のメソッドは次のような書式で作ることができる

```
def メソッド名
  ~
  ~
end
```

- ・メソッドを呼び出すときは「メソッド名」を書く

```
put_1
put_3
put_0
```

例題 1 プログラムの基本を復習

(3) ex01_number3.rb

上記の(2)だと、put_0 から put_9 までの10個のメソッドを覚える必要があり、使い勝手があまりよくありません。

put_0, put_1, ..., puts_9 という各メソッドは残したままで、どの数字の花文字も表示できる put_n というメソッドを追加してください。

なお put_n は、引数として1桁の数字を受け取ることができ、受け取った値の花文字を表示するものとします。

put_n ができたら、それを使って「1」「3」「0」の花文字を表示してください。

[ヒント] 

- ・引数とは、メソッドに渡すデータのこと
- ・引数を受け取ったメソッドは、その引数の値に応じて処理の内容を変えることができるので1つのメソッドでいろいろなことができるようになる
- ・引数を受け取るメソッドは次のような書式で作ることができる

```
def メソッド名(引数名)
  ~
  ~
end
```

- ・引数の値に応じて処理を分けるには if 文を使う

```
if 引数が〇〇
  引数が〇〇の場合の処理
end
if 引数がXX
  引数がXXの場合の処理
end
```

- ・既存の put_0 から put_9 をうまく活用するのがポイント
- ・引数を渡してメソッドを使うときは「メソッド名(引数の値)」と書く

```
put_n(1)
put_n(3)
put_n(0)
```

例題 1 プログラムの基本を復習

(4) ex01_number4.rb

さらに `put_n` というメソッドの内容を拡張して、2桁以上の数字も処理できるようにしてください。

なお、`put_n` が2桁以上の数字を受け取った場合、
左側の桁から順に各桁の数字の花文字を表示するものとします。

それを使って、「`put_n(130)`」と呼び出すことで
「1」「3」「0」の花文字を表示してください。



・数字の各桁の値の調べ方(一例)

```
s = 130.to_s    # 数字の「130」を文字列に変換して s という変数に代入
puts s[0]       # 文字列のいちばん左側の文字を表示
puts s[1]       # 文字列の左側から2番目の文字を表示
puts s[2]       # 文字列の左側から3番目の文字を表示
```

・「`数値.to_s`」で「数値 → 文字列」に変換
「`文字列.to_i`」で「文字列 → 数値」に変換

・文字列を1文字ずつ処理するには、文字数分だけループさせるのが便利

```
s = "Ruby"
mojisu = s.length
mojisu.times do |i|
  puts s[i]
end
```

・文字列の文字数を調べるには
「`length`」または「`size`」というメソッドが便利

・ループ(繰り返し)の処理はいろいろな書き方があるけれど、
回数が決まっているのであれば「`times`」メソッドを使うのが簡単

```
回数.times do
  ~
end

回数.times do |i|  # ループのたびに i の値が 0, 1, 2 ...と増えていく
  ~
end
```