プログラミング言語　第六回

担当：篠沢 佳久
櫻井 彰人

平成29年 5月15日

本日の内容

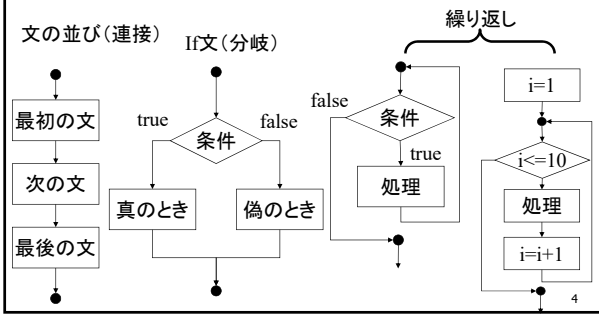
- 繰り返し（2）
- times, each
- 繰り返しを用いた標準入力

- 繰り返しについての練習問題
- 自習問題もしておいて下さい

前回の復習

制御構造
繰り返し（1）

制御構造（復習）



無限の繰り返し

```
loop{
        式

}
```

式が永久に実行される

停止するために break を用いる

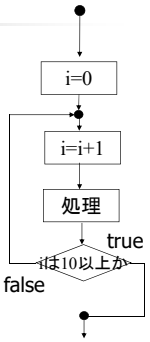
```
loop{
        式
        break 条件式
}
次の式
```

条件式を満たした場合のみ停止する（loopブロックの次の式を実行する）

無限の繰り返し①

10回「こんにちは」を表示するプログラム

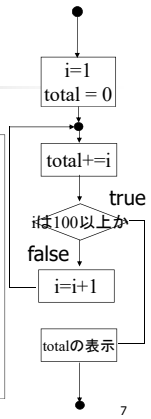
```
# coding: Windows-31J
i = 0
loop{
  i = i+1
  print( i , "回目のこんにちは¥n" )
  break  if i >= 10
}
```



無限の繰り返し②

100以下の整数の和を求めるプログラム

```
# coding: Windows-31J
i = 1
total = 0
loop{
   total += i
   break  if i >= 100
   i = i+1
}
print( "合計は" , total )
```



7

回数の決まった繰り返し

times

each

8

times①

- 同じ処理をn回繰り返したい
- n.times

```
n.times {
    式
}
```

式をn回繰り返す

9

回数がわかっている繰り返し①

- 10.times

```
# coding: Windows-31J
10.times {
  print( "やっほ～ " )
  puts( "     Yee-ha! " )
}
```

停止条件は書かなくてもよい
n.times で指定されたn回，式を繰り返す

10

loop{}で書く場合には

```
# coding: Windows-31J
10.times {
  print( "こんにちは¥n" )
}
```

```
# coding: Windows-31J
i = 0
loop{
   i = i+1
   print( "こんにちは¥n" )
   break  if i >= 10
}
```

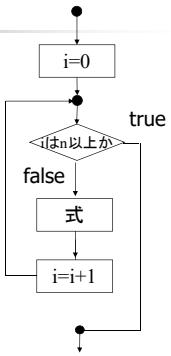
```
Z:¥Ruby>ruby sample.rb
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
こんにちは
```

11

times②

```
n.times{ |i|
   式
}
```

n回式を繰り返す
iには自動的に0からn-1が代入される
(i=0,1,2,…,n-1)



12

回数がわかっている繰り返し②

```
# coding: Windows-31J
10.times { |i|
  print( i , "回目のこんにちは¥n" )
}
```

```
Z:¥ruby>ruby sample.rb
0回目のこんにちは
1回目のこんにちは
2回目のこんにちは
3回目のこんにちは
4回目のこんにちは
5回目のこんにちは
6回目のこんにちは
7回目のこんにちは
8回目のこんにちは
9回目のこんにちは
```

変数iには0から代入されていく

変数iには9まで代入されていく

13

回数がわかっている繰り返し②

変数

```
# coding: Windows-31J
n=10
n.times { |i|
  print( i , "回目のこんにちは¥n" )
}
```

```
Z:¥ruby>ruby sample.rb
0回目のこんにちは
1回目のこんにちは
2回目のこんにちは
3回目のこんにちは
4回目のこんにちは
5回目のこんにちは
6回目のこんにちは
7回目のこんにちは
8回目のこんにちは
9回目のこんにちは
```

変数iには0から代入されていく

変数iには9まで代入されていく

14

回数がわかっている繰り返し②

「1～10」回目のこんにちは」と表示させるには？

```
# coding: Windows-31J
10.times { |i|
  print( i+1 , "回目のこんにちは¥n" )
}
```

```
Z:¥ruby>ruby sample.rb
1回目のこんにちは
2回目のこんにちは
3回目のこんにちは
4回目のこんにちは
5回目のこんにちは
6回目のこんにちは
7回目のこんにちは
8回目のこんにちは
9回目のこんにちは
10回目のこんにちは
```

変数iには0から代入されていく
i+1 →「1回目のこんにちは」

変数iには9まで代入されていく
i+1 →「10回目のこんにちは」

15

回数がわかっている繰り返し②'

0から9までの合計を求めるプログラム

```
# coding: Windows-31J
total = 0
10.times { |i|
  total += i
}
print( "合計は" , total )
```

変数iは0から9まで1ずつ加算

```
Z:¥Ruby>ruby sample.rb
合計は45
```

16

回数がわかっている繰り返し②''

1から10までの合計を求めるには？

```
# coding: Windows-31J
total = 0
10.times { |i|
  total += i+1
}
print( "合計は" , total )
```

変数iは0から9まで
i+1とすると1から10まで合計される

```
Z:¥Ruby>ruby sample.rb
合計は55
```

17

回数がわかっている繰り返し③

10のべき乗を表示するプログラム

```
10.times{ |i|
  print( 10 ** i , "¥n" )
}
```

i は0から9まで1ずつ

```
Z:¥Ruby>ruby sample.rb
1
10
100
1000
10000
100000
1000000
10000000
100000000
1000000000
```

10^0

10^1

10^9

回数がわかっている繰り返し③

10のべき乗を表示するプログラム

```
10.times{ |i|
  print( 10 ** (i+1) , "¥n" )
}
```

i は0から9まで1ずつ
i+1 → 1から10まで

```
Z:¥Ruby>ruby sample.rb
10          10^1
100         10^2
1000
10000
100000
1000000
10000000
100000000
1000000000
10000000000   10^10
```

回数がわかっている繰り返し③'

10のべき乗を表示するプログラム

```
10.times{ |i|
  print( 10 ** i , "¥n" )
  break if i >= 5
}
```

iが5以上となった場合，break文によりループを停止

loopと同様に，break文にてループの停止も可能

```
Z:¥Ruby>ruby sample.rb
1           10^0
10          10^1
100
1000
10000
100000      10^5
```

20

times と loop の関係①

10のべき乗（10^0から10^9まで）を表示するプログラム

times を用いた場合

```
10.times{ |i|
  print( 10 ** i , "¥n" )
}
```

loop を用いた場合

```
i = 0
loop{
  print( 10 ** i , "¥n" )
  i = i+1
  break  if i >= 10
}
```

21

times と loop の関係②

10のべき乗（10^1から10^{10}まで）を表示するプログラム

times を用いた場合

```
10.times{ |i|
  print( 10 ** (i+1) , "¥n" )
}
```

loop を用いた場合

```
i = 1
loop{
  print( 10 ** i , "¥n" )
  break  if i >= 10
  i = i+1
}
```

22

回数がわかっている繰り返し④

```
10.times {|i| print i; puts "番目"}
10.times {|i| puts( "#{i} 番目" ) }
10.times {|j| puts "*"*j}
```

```
0 番目
1 番目
2 番目
3 番目
4 番目
5 番目
6 番目
7 番目
8 番目
9 番目
```

iやj=0,1,2,3,4,5,6,7,8,9 の順で，繰り返し，式の計算をする

j=0
j=1
j=9

```
*
**
***
****
*****
******
*******
********
*********
```

23

回数が分っている繰り返し⑤

- なぜ，このような出力になるでしょうか

```
n=5
n.times {|k| print( " "*(10-k), "*"*k, "¥n" ) }
```

出力結果

```
   *
  **
 ***
****
```

k	10-k	空白の数	*の数
0	10	10	0
1	9	9	1
2	8	8	2
3	7	7	3
4	6	6	4

24

回数が分っている繰り返し⑥

- どのような出力になるでしょうか

```
n=10
n.times {|k| print( " "*(10-k), "*", "¥n" ) }
```

```
n=10
n.times {|k|
  print( " "*(10-k), "*", " "*k*2 , "*" , "¥n" )
}
```

25

each①

```
n.times{ |i|
  式
}
```

n回式を繰り返す
iには自動的に0からn-1が代入される

```
(n..m).each{ |i|
  式
}
```

(m-n+1)回式を繰り返す（m>n）
iには自動的にnからmが代入される

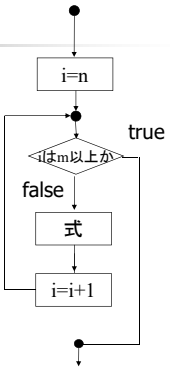
26

each②

```
(n..m).each{ |i|
  式
}
```

i には自動的にnからmが代入される
その結果、式は（m-n+1）回繰り返される

（m≥n の場合）



27

each③

```
(3..5).each{ |i|
  print( i , "¥n" )
}
```

```
Z:¥Ruby>ruby sample.rb
3
4
5
```
i=3,4,5

```
(1..4).each{ |i|
  print( i*2+1 , "¥n" )
}
```

```
Z:¥Ruby>ruby sample.rb
3
5
7
9
```
i=1,2,3,4

28

each③'

```
a=3
b=5
(a..b).each{ |i|
    print( i , "¥n" )
}
```
変数

```
Z:¥Ruby>ruby sample.rb
3
4
5
```
i=3,4,5

```
a=1
b=4
(a..b).each{ |i|
    print( i*2+1 , "¥n" )
}
```
変数

```
Z:¥Ruby>ruby sample.rb
3
5
7
9
```
i=1,2,3,4

29

each④

```
(-3..3).each{ |i|
  print( i , "¥n" )
}
```

```
Z:¥Ruby>ruby sample.rb
-3
-2
-1
0
1
2
3
```
i=-3,-2,-1,0,1,2,3

```
(3..-3).each{ |i|
      print( i , "¥n" )
}
```

```
Z:¥Ruby>ruby sample.rb
```
実行されない

30

変数範囲が決まっている繰り返し①

10のべき乗を表示するプログラム

each を用いた場合

```
(0..9).each{ |i|
  print( 10 ** i , "¥n" )
}
```

times を用いた場合

```
10.times{ |i|
  print( 10 ** i , "¥n" )
}
```

```
Z:¥Ruby>ruby sample.rb
1          10^0  i=0
10         10^1  i=1
100
1000
10000
100000
1000000
10000000   10^8  i=8
100000000
1000000000 10^9  i=9
```

31

変数範囲が決まっている繰り返し②

```
# coding: Windows-31J
(5..10).each {|i|
  print( i )
  if i%2==0 then
    print( " は偶数" )
  else
    print( " は奇数" )
  end
  print( "¥n" )
}
```

```
# coding: Windows-31J
s = 5
e = 10
(s..e).each {|i|
  print( i )
  if i%2==0 then
    print( " は偶数" )
  else
    print( " は奇数" )
  end
  print( "¥n" )
}
```

```
5 は奇数
6 は偶数
7 は奇数
8 は偶数
9 は奇数
10 は偶数
```

32

変数範囲が決まっている繰り返し③

10から100までの合計値を求めるプログラム

```
total=0
(10..100).each { |i|
   total += i
}
print( total )
```

nからmまでの合計値を求めるプログラム

```
# coding: Windows-31J
n=gets.chomp.to_i
m=gets.chomp.to_i
total=0
(n..m).each { |i|
   total += i
}
print( n , "から" , m , "までの合計は
   " , total )
```

標準入力

33

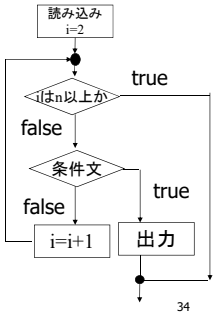
変数範囲が決まっている繰り返し④

何を調べているプログラムでしょう

```
# coding: Windows-31J
n=gets.chomp.to_i
(2..n-1).each { |i|
   if n % i == 0 then
      print( "〇〇ではありません" )
      break
   end
}
```

i は2～n-1

break でループを抜け出ることも可能



34

変数範囲が決まっている繰り返し⑤

負の場合

```
(-10..10).each{ |i|
  print( i , " " , i**2 , "¥n" )
}
```

i には-10から10まで代入される

```
Z:¥Ruby>ruby sample.rb
-10 100
-9 81
-8 64
-7 49
-6 36
-5 25
-4 16
-3 9
-2 4
-1 1
0 0
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
```

変数範囲が決まっている繰り返し⑤

(n..m).each

n>m の場合

```
(10..-10).each{ |i|
  print( i , " " , i**2 , "¥n" )
}
```

```
Z:¥Ruby>ruby sample.rb
```

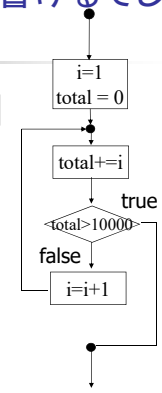
何も実行されない
→ n < m でなければならない

36

times と each では書けるでしょうか？

1からnまでの合計が10,000以上で終了

```
i = 1
total = 0
loop{
   total += i
   break  if total > 10000
   i = i+1
}
print( i )
```



37

繰り返しの場合分け

- 同じ処理をある回数だけ行ないたい
  - times, each
- ある条件が成立するまで，同じ処理を繰り返したい
  - loop, while（次週）

38

刻み幅に小数値を使用したい場合①

0から1まで0.1刻みで二乗の計算を行なう

i は0から10まで1刻みの整数値

```
11.times{ |i|
   print( i / 10.0 , " " , (i/10.0)**2.0 , "¥n" )
}
```

10.0で割る

39

刻み幅に小数値を使用したい場合①'

0から1まで0.1刻みで二乗の計算を行なう

```
11.times{ |i|
   print( i / 10.0 , " " , (i/10.0)**2.0 , "¥n" )
}
```

0.1刻みにしたい場合
i=0,1,2,･･･,10 として，この値を10で割る

```
Z:¥Ruby>ruby sample.rb
0.0 0.0
0.1 0.010000000000000002
0.2 0.04000000000000001
0.3 0.09
0.4 0.16000000000000003
0.5 0.25
0.6 0.36
0.7 0.48999999999999994
0.8 0.6400000000000001
0.9 0.81
1.0 1.0
```

loopで書こうとすると…

loop文の場合

```
i=0.0
loop{
   print( i , " " , i**2 , "¥n")
   break if i == 1.0
   i += 0.1
}
```

0.1ずつ加算

実行結果
→ 終了しない
（Ctrl+Cで強制終了）

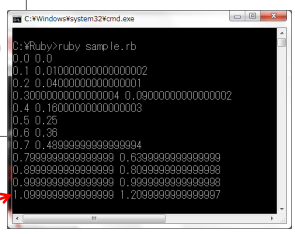
loopで書こうとすると…

loop文の場合

```
i=0.0
loop{
   print( i , " " , i**2 , "¥n")
   break if i >= 1.0
   i += 0.1
}
```

0.1ずつ加算

1を超えて出力



42

刻み幅に小数を使う場合の注意

```
i=0
loop{
  print( i/10.0 , " " , (i/10.0)**2 , "¥n")
  break if i >= 10
  i += 1
}
```

```
Z:¥Ruby>ruby sample.rb
0.0 0.0
0.1 0.010000000000000002
0.2 0.04000000000000001
0.3 0.09
0.4 0.16000000000000003
0.5 0.25
0.6 0.36
0.7 0.48999999999999994
0.8 0.6400000000000001
0.9 0.81
1.0 1.0
```

1.0で停止

43

刻み幅に小数を使う場合の注意

```
i=0.0
loop{
  print( i , " " , i**2 , "¥n")
  break if i >= 1.0
  i += 0.1
}
```

0.1を10回加算しても1とはならない

```
i=0
loop{
  print( i/10.0 , " " , (i/10.0)**2 , "¥n")
  break if i >= 10
  i += 1
}
```

刻み幅は必ず整数とする

44

刻み幅に小数値を使用したい場合②

```
(0..10).each{ |i|
  print( i / 10.0 , " " , (i/10.0)**2.0 , "¥n" )
}
```

0から1まで，刻み幅0.1ごと

```
(0..100).each{ |i|
  x = i.to_f / 100
  print( x, " " ,  x**2.0 , "¥n" )
}
```

0から1まで，刻み幅0.01ごと

45

刻み幅に小数値を使用したい場合③

0から10まで，刻み幅0.5ごと

```
(0..20).each{ |i|
  print( i / 2.0 , " " , (i/2.0)**2.0 , "¥n" )
}
```

0から10まで，刻み幅0.25ごと

```
(0..40).each{ |i|
  print( i / 4.0 , " " , (i/4.0)**2.0 , "¥n" )
}
```

46

標準入力と繰り返し

47

キーボードからの入力（復習）

- line = gets.chop
- line = gets.chomp

- gets
  - キーボードから文字列を読み込む
  - この場合，改行文字が文字列の最後に含む
- chop（chomp）
  - 最後の一文字を削除する（最後の一文字が改行ならば削除する）
- line には読み込まれた文字列が代入される
- 文字列のため，数字に「to_i」「to_f」を用いて数値に変換する

48

標準入力①（復習）

- getsによる標準入力
- 文字列型で入力される
- 末尾に"¥n"（もしくは"¥r¥n"）が挿入される

- 整数値（小数値）として利用したい場合
- 末尾の改行を削除
- 文字列型から整数（小数）へ変換する必要がある

49

標準入力②（復習）

入力

```
irb(main):001:0> x=gets
3.1415
=> "3.1415¥r¥n"
irb(main):002:0> x.chomp
=> "3.1415"
irb(main):003:0> x.chomp.to_f
=> 3.1415
irb(main):004:0> x.chomp.to_i
=> 3
```

chop で最後の一文字（改行）を削除

文字列型を小数に変換

文字列型を整数に変換

50

標準入力③（復習）

入力

```
irb(main):001:0> x=gets
abcd
=> "abcd¥r¥n"
irb(main):002:0> x.chomp
=> "abcd"
```

chop で最後の一文字（改行）を削除

51

標準入力（復習）

「5↵」と入力

```
# coding: Windows-31J
n = gets.chomp.to_i
i = 0
loop{
  i = i+1
  print( i , "回目のこんにちは¥n" )
  break  if i >= n
}
```

```
Z:¥Ruby>ruby sample.rb
5
1回目のこんにちは
2回目のこんにちは
3回目のこんにちは
4回目のこんにちは
5回目のこんにちは
```

gets により "5¥n"
chomp により "5"
to_i により整数5に変換される

52

簡単に書くには…①

```
# coding: Windows-31J
a = gets.chomp.to_i
b = gets.chomp.to_i
c = gets.chomp.to_i
d = gets.chomp.to_i
e = gets.chomp.to_i
sum = a+b+c+d+e
print( "合計は" , sum , "です¥n" )
```

5個の整数を読み込み，合計を出力

```
Z:¥Ruby>ruby sample.rb
43
34
2
1
1
合計は81です
```

53

簡単に書くには…②

```
# coding: Windows-31J
sum = 0
i = 0
loop{
        x = gets.chomp.to_i
        sum += x
        i += 1
        break if i >= 5
}
print( "合計は" , sum , "です¥n" )
```

変数xに整数を入力

5回入力したら停止

```
Z:¥Ruby>ruby sample.rb
4
1
4
6
7
合計は22です
```

54

入力の繰り返し①

```
# coding: Windows-31J
max = 0
i = 0
loop{
        x = gets.chomp.to_i
        if x > max then
           max = x
        end
        i += 1
        break if i >= 5
}
print( "最大は" , max , "です¥n" )
```

変数xに整数を入力

5回入力したら停止

```
Z:¥Ruby>ruby sample.rb
23
12
68
45
23
最大は68です
```

55

入力の繰り返し②

```
# coding: Windows-31J
sum = 0
loop{
        x = gets.chomp.to_i
        sum += x
        break if sum > 100
}
print( "合計は" , sum , "です¥n" )
```

変数xに整数を入力

sumの値が100を越えたら停止

```
Z:¥Ruby>ruby sample.rb
34
23
17
56
合計は130です
```

56

入力回数が分からない場合

- 前々頁のプログラムは入力が5回
- 前頁のプログラムは条件式によって停止
- 入力回数が分からない場合はどうすればよいか
  - 特定の文字(Enterなど)を入力した場合のみ入力を終了させるようにする

57

繰り返し入力できるようにするためには①

```
# coding: Windows-31J
loop{
  print( "何か文字を入れて下さい(Enterで終了します)¥n" )
  line = gets.chomp
  break if line == ""
  print( line , "¥n" )
}
```

```
Z:¥Ruby>ruby sample.rb
何か文字を入れて下さい(Enterで終了します)
24
24
何か文字を入れて下さい(Enterで終了します)
abcd
abcd
何か文字を入れて下さい(Enterで終了します)
sdds
sdds
何か文字を入れて下さい(Enterで終了します)
```

Enterで終了

キーボードからの入力

キーボードでEnterキーを入力した場合

```
# coding: Windows-31J
loop{
   line = gets.chomp
   break if line == ""
   print( line , "¥n" )
}
```

chomp で改行文字が削除されるため，lineには空白文字が入る

lineには空白文字が入っているため break が実行され停止する

59

改行の処理(復習)

```
irb(main):013:0> gets

=> "¥n"
irb(main):014:0> x = gets
=> "¥n"
irb(main):015:0> print( x )

=> nil
irb(main):016:0> x.chomp
=> ""
```

Enterキーのみを入力

chomp により改行を削除されるため空白文字となる

60

繰り返し入力できるようにするためには②

```
# coding: Windows-31J
loop{
  print( "何か文字を入れて下さい(stopで終了します)¥n" )
  line = gets.chomp
  break if line == "stop"
  print( line , "¥n" )
}
```

```
Z:¥Ruby>ruby sample.rb
何か文字を入れて下さい(stopで終了します)
32
32
何か文字を入れて下さい(stopで終了します)
stop
```

stopで終了

61

繰り返し入力できるようにするためには③

```
# coding: Windows-31J
sum = 0
loop{
  print( "整数を入れて下さい(Enterで終了します)¥n" )
  line = gets.chomp
  break if line == ""
  print( line , "¥n" )
  sum += line.to_i
}
print( "合計値は" , sum )
```

末尾の改行を削除

空白かどうか

文字型の変数line
整数に型変換

62

```
# coding: Windows-31J
sum = 0
loop{
  print( "整数を入れて下さい(Enterで終了します)¥n" )
  line = gets.chomp
  break if line == ""
  print( line , "¥n" )
  sum += line.to_i
}
print( "合計値は" , sum )
```

```
G:¥Ruby>ruby sample.rb
整数を入れて下さい(Enterで終了します)
32
32
整数を入れて下さい(Enterで終了します)
23
23
整数を入れて下さい(Enterで終了します)
11
11
整数を入れて下さい(Enterで終了します)

合計値は66
```

63

繰り返し入力できるようにするためには④

```
# coding: Windows-31J
loop {
  print( "Enter your score: " )
  line = gets.chomp
    break if line==""
  score = line.to_f
  grade =
    if score >= 70 then
      if score >= 80 then "A" else "B" end
    else
      if score >= 60 then "C" else "D" end
    end
  print( "Your score #{score} corresponds to #{grade}¥n" )
}
```

改行キー(Enter)のみ入力された場合，停止

```
G:¥Ruby>ruby sample.rb
Enter your score: 90
Your score 90.0 corresponds to A
Enter your score: 40
Your score 40.0 corresponds to D
Enter your score:
```

64

どこが違うでしょうか

```
# coding: Windows-31J
loop{
  print( "何か文字を入れて下さい(0で終了します)¥n" )
  line = gets.chomp
  break if line == "0"
  print( line , "¥n" )
}
```

```
# coding: Windows-31J
loop{
  print( "何か文字を入れて下さい(0で終了します)¥n" )
  line = gets.chomp.to_i
  break if line == 0
  print( line , "¥n" )
}
```

実は英文字でも終了します
なぜでしょう

65

どこが違うでしょうか

英文字を整数型に型変換すると…

```
irb(main):001:0> "a".to_i
=> 0
irb(main):002:0> "x".to_i
=> 0
irb(main):003:0> "abc".to_i
=> 0
irb(main):004:0> "1".to_i
=> 1
irb(main):005:0> "10".to_i
=> 10
```

66

練習問題

練習①～④
（簡単な人は練習⑤も試してみて下さい）

練習①

- 1から10までの整数のうち奇数のみを印字するプログラムを times，each，loopを用いて書きなさい

```
Z:¥Ruby>ruby sample.rb
1
3
5
7
9
```


練習②

- 1から100までの整数の二乗和を印字するプログラムを times，eachを用いて書きなさい

二乗和 $\sum_{i=1}^{100} i^2$

```
Z:¥Ruby>ruby sample.rb
自乗和は338350
```


練習②をloopで書いた場合（先週の練習問題④）

loop を用いて書いた場合

```
# coding: Windows-31J
i = 1
total = 0
loop{
   total += i**2
   break  if i >= 100
   i = i+1
}
print( "二乗和は" , total )
```

1から100までの整数の二乗和を求める

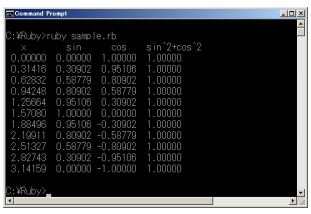
練習③

- 整数を5個キーボードから読み込みなさい．5個の整数の積を印字するプログラムを書きなさい

```
Z:¥Ruby>ruby sample.rb
2
4
5
7
3
積は840です
```


練習④

- 0からπまで，0.1π刻みでsin，cos，$\sin^2+\cos^2$を印字するプログラムを書きなさい



練習⑤

- 34ページのスライドのプログラムは素数でないことを判定するプログラムです

```
# coding: Windows-31J
n=gets.chomp.to_i
(2..n-1).each { |i|
    if n % i == 0 then
        print( "素数ではありません" )
        break
    end
}
```

73

練習⑤

- このプログラムを改良し，整数nが素数かどうかを判定するプログラムに書き直して下さい

```
Z:¥Ruby>ruby sample.rb
9999973
9999973は素数です
```

```
Z:¥Ruby>ruby sample.rb
123456789
123456789は素数ではありません
```

74

練習問題

- 練習問題①から④を行ないなさい
- （簡単な人は練習⑤も試してみて下さい）

- プログラムと実行結果をワープロに貼り付けて，keio.jp から提出して下さい

75