

プログラミング言語 第2回 4月17日

担当: 篠沢 佳久
櫻井 彰人

平成29年度: 春学期

1

講義のホームページ

- 講義に関する情報
 - <http://www.sakurai.comp.ae.keio.ac.jp/class.html>

2

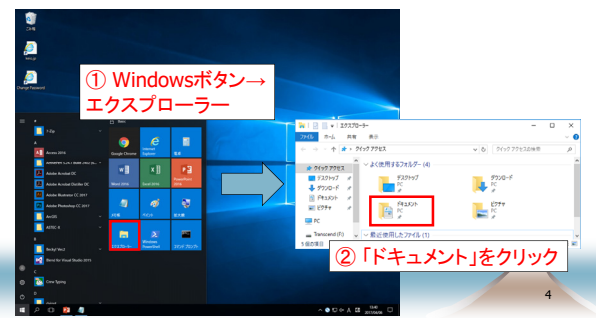
まずは注意点

- ドキュメントに「Ruby」というフォルダを作って、作成したプログラムはそこに保存するようにすること
- (第一回の講義資料を参照)

※日吉ITCのPCでは、ドキュメントの中の Ruby フォルダは、"Z:¥Documents¥Ruby" を指す

3

日吉ITCの場合 (OSはWindows10)



4

本日の内容

- interactive Ruby の使い方
- データの型
- 演算子
- 練習問題

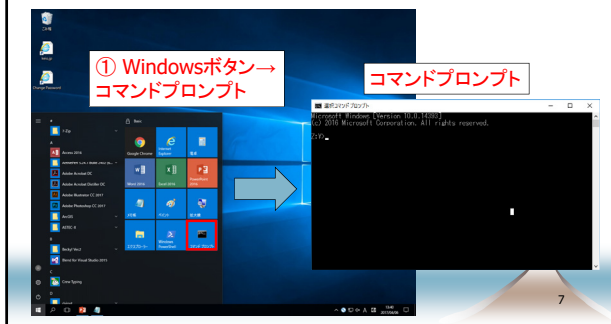
5

interactive Ruby

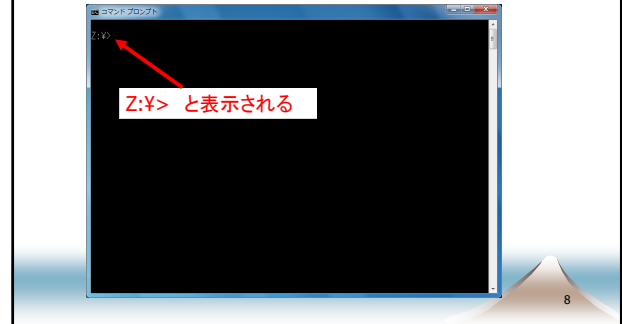
irbの起動と終了
irbでのRubyプログラムの実行方法

6

コマンドプロンプトの起動





コマンドプロンプトの画面

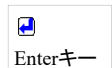


まずは: irb

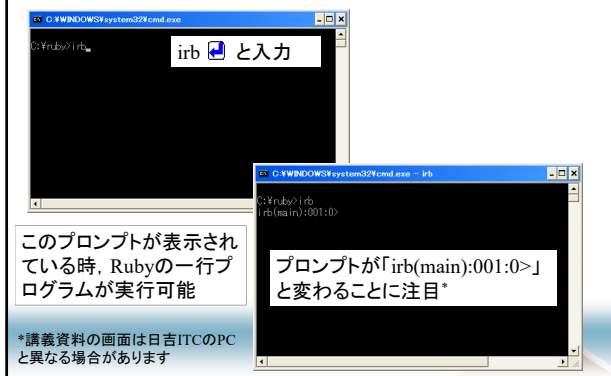
- プログラムは、多数の行で構成されている
- しかし、中には、実行したいことが、一行で書いてしまうこともある
- Ruby には、この「一行プログラム」の実行ができるツールが提供されている
 - 実は、複数行に渡っても実行できる、優れたもの
- それが、irb (interactive Ruby)

irb の起動①

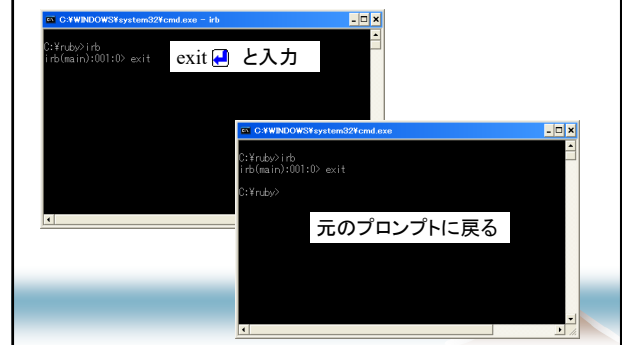
- コマンドプロンプト上で
irb 
と入力
- あとは interactive (会話的に)
 - 「コマンド」の入力
 - 実行結果の出力が無限に行われます
- 終了したいときには
exit 
と入力



irbの起動②



irbの終了



irb での入力方法①

1+2+3+4 と入力

「=> 10」と結果が戻ってくる

irb での入力方法②

1+2+3+ と入力

「irb(main):002:0*」と戻って来る
→ 式の入力途中を意味する

irb での入力方法③

4 と入力

「=> 10」と結果が戻ってくる

irb での入力方法④

Math.sqr(2) と入力

間違った入力のため、エラーメッセージが戻って来る

irb での入力方法⑤

puts "こんにちは" と入力

コマンドプロンプト上での日本語入力
「半角/全角」でローマ字変換

irb の実行例

- 電卓がわりにも使える
- 日本語以外は半角文字で入力する
- 入力した後は、Enterキーを入力すると結果が戻ってくる

```

irb(main):001:0> 1+2+3+4
=> 10
irb(main):002:0> 2*3*4*5*6*7*8*9*10
=> 3628800
irb(main):003:0> x=2.0 代入式
=> 2.0
irb(main):004:0> Math.sqrt(x)
=> 1.4142135623731
irb(main):005:0> Math::PI
=> 3.14159265358979
irb(main):006:0> Math.sin(Math::PI/4)
=> 0.707106781186547
irb(main):007:0> Math.sqrt(2)/2
=> 0.707106781186548
irb(main):008:0>
    
```

数学用関数	
平方根	Math.sqrt()
π	Math::PI
sin	Math.sin()

<http://docs.ruby-lang.org/ja/2.4.0/class/Math.html>

算術演算子

演算子	用途	例	演算結果
+	加算	3+2	5
-	減算	4-2	2
*	乗算	2*2	4
/	除算	4/2	2
%	剰余	5%2	1
**	べき	5**3	125

19

演算の優先順位①

- $5 + 10 / 5$
=> 7
- $(5 + 10) / 5$
=> 3
- $10 / 2 + 3$
=> 8
- $10 / (2 + 3)$
=> 2
- $3 * 4 ** 2$
=> 48
- $(3 * 4) ** 2$
=> 144

優先される

**
/ * %
+ -

4**2を行ない、その結果を3倍する

(3*4)を行ない、その結果を2乗する

20

括弧の書き方

```

irb(main):001:0> (3+6)
=> 9
irb(main):002:0> ((3+6))
=> 9
irb(main):003:0> (((3+6)))
=> 9
irb(main):004:0> ((3+6)
irb(main):005:1> )
=> 9
irb(main):006:0> (3+6)
SyntaxError: compile error
(irb):6: syntax error, unexpected ')', expecting Send
    
```

複数の()を用いてもよい

必ず閉じる

左右の個数が合っていない場合はエラーが表示される

21

演算の優先順位②

- $5 + - 3$
=> 2
- $-(3 - 10)$
=> 7
- $3 ** - 2$
=> (1/9)
- $- 3 ** 2$
=> -9
- $- 3 ** - 2$
=> -(1/9)
- $(- 3) ** - 2$
=> (1/9)

優先される

括弧 ()
**
符号 + -
/ * %
+ -

3 ** -2を行ない、その結果をマイナスとする

22

数学用関数①

- 平方根
 - `Math.sqrt(2)`

- 三角関数

- `Math.sin(Math::PI)`
- `Math.cos(Math::PI)`
- `Math.tan(Math::PI)`

sin π

cos π

tan π

π

Math::PI

単位はラジアン

23

数学用関数②

- 自然対数
 - `Math.log(Math::E)`

e
Math::E

- 常用対数
 - `Math.log10(100)`

- 指数
 - `Math.exp(2)`

その他の数学関数の一例
<http://doc.ruby-lang.org/ja/2.3.0/class/Math.html>

24

代入式①

- $x = 2$
- $y = 10$
- $(x + y) / 2$
=> 6
- $x * y$
=> 20
- $\text{Math.sqrt}(y / x)$
=> 2.23606797749979

x, y を変数
 $x = 2$ を代入式と呼ぶ

25

代入式②

- $x = 2$
- $y = 10$
- $a = (x + y) / 2$
- $b = x * y$
- $c = y / x$
- $d = \text{Math.sqrt}(c)$

前ページと同じ計算
求めた値を別の変数(a, b, c, d)に代入することも可能

26

代入式③

```
コマンドプロンプト - rb
C:\Users\shino>irb
irb(main):001:0> x=2
=> 2
irb(main):002:0> y=10
=> 10
irb(main):003:0> a=(x+y+z)/2
NameError: undefined local variable or method `z' for main:Object
from (irb):3:
from C:/Program Files/Ruby-2.3-x64/bin/irb.cmd:19:in `main'
irb(main):004:0>
```

エラーメッセージ
→ 変数zは未定義

$x=2$
 $y=10$
 $a=(x+y+z)/2$
→ 変数zは未定義

27

代入式③'

```
C:\WINDOWS\system32\cmd.exe - irb
C:\Ruby>irb
irb(main):001:0> x=2
=> 2
irb(main):002:0> y=10
=> 10
irb(main):003:0> z=0
=> 0
irb(main):004:0> a=(x+y+z)/2
=> 6
irb(main):005:0>
```

$x=2$
 $y=10$
 $z=0$
 $a=(x+y+z)/2$

代入式の右側に現れる変数はそれ以前に定義しなければならない

28

エラー出力①

- 式をRubyの文法通りに記述しなかった場合、エラーが出力されます

```
コマンドプロンプト - rb
C:\Users\shino>irb
irb(main):001:0> z
NameError: undefined local variable or method `z' for main:Object
from (irb):1:
from C:/Program Files/Ruby-2.3-x64/bin/irb.cmd:19:in `main'
irb(main):002:0> Math::Pi
NameError: uninitialized constant Math::Pi
Did you mean?  Math::PI
from (irb):2:
from C:/Program Files/Ruby-2.3-x64/bin/irb.cmd:19:in `main'
irb(main):003:0>
```

変数zは未定義

定数Math::Piは初期化されていない

29

エラー出力②

- 式をRubyの文法通りに記述しなかった場合、エラーが出力されます
 - エラーの原因が書かれています
 - まずは読んで意味を考えて下さい
 - なぜ間違えたのか、どこに誤りがあるのかを考えて下さい

30

irbでの編集

irb で入力した文字を修正する方法

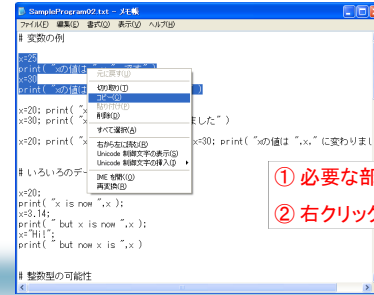
- backspace: カーソル直左の一文字消去
- ←→: カーソルの移動
 - Ctrl+b, Ctrl+f と同じ
- ↑↓: 過去入力した行への切替
 - Ctrl+n, Ctrl+p と同じ
- delete: カーソル位置の一文字を消去
- Home, End: 入力行の先頭、末尾へ
 - Ctrl+a, Ctrl+e と同じ
- Ctrl+k でカーソルから右側全部削除
- 困った時はCtrl+c

Ctrl+a
Ctrlキーを押しながらa

31

irbへの貼り付け(paste)①

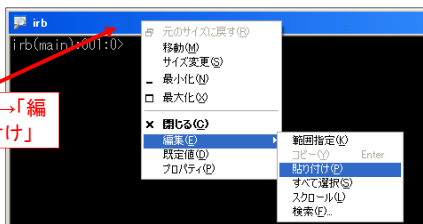
メモ帳で書いたプログラムをirbで動かしたい



- ① 必要な部分をドラッグ
- ② 右クリック→「コピー」

32

irbへの貼り付け(paste)②



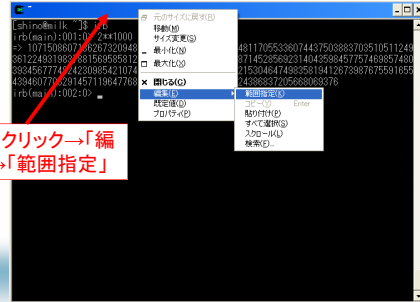
- ③ 右クリック→「編集」→「貼り付け」

複数行のコピー&ペーストもできますが、一行ごとに実行して下さい

33

irbからのコピー①

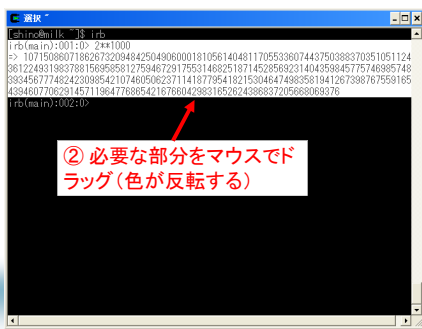
irb上のプログラム、結果をメモ帳などにコピーしたい場合



- ① 右クリック→「編集」→「範囲指定」

34

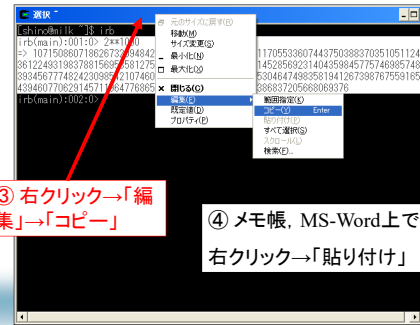
irbからのコピー②



- ② 必要な部分をマウスでドラッグ(色が反転する)

35

irbからのコピー③



- ③ 右クリック→「編集」→「コピー」

- ④ メモ帳、MS-Word上で
右クリック→「貼り付け」

36

irbの画面のコピー①

irbの画面をMS-Word等に貼り付けたい場合

```
irb(main):001:0> 2 * 10000
=> 20000
irb(main):002:0>
```

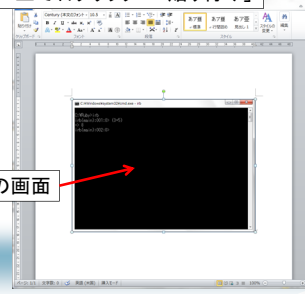
- ① irbのウィンドウをクリック
- ② Alt キーを押しながら PrintScrn

37

irbの画面のコピー②

③ MS-Word上で右クリック→「貼り付け」

実行結果の画面



38

データの型

- 整数型
- 浮動小数点型
- 文字列型

39

数値には型がある①

さて、次のようになる理由を考えてみよう

```
irb(main):005:0> 3
=> 3
irb(main):006:0> 3.0
=> 3.0
irb(main):007:0> 3/2
=> 1
irb(main):008:0> 3.0/2
=> 1.5
irb(main):009:0> 3.0/2.0
=> 1.5
irb(main):010:0>
```

ヒント:
小数点がある数と小数点がない数に違いがある

40

数値には型がある②

- 「3」は「整数」
- 「3.0」は「小数」
- 「3/2」は「整数を整数で割り算」
→ 結果は「整数」
- 「3.0/2」は「小数を整数で割り算」
→ 結果は「小数」
- 「3.0/2.0」は「小数を小数で割り算」
→ 結果は「小数」

41

データの型①

- データ
 - コンピュータの演算・操作の対象
 - 文字列、小数点のある数、小数点のない数
- データの型
 - そのデータに適用が許される演算・操作の集合
- 小数点のない数: 小数点のない数による加減乗除
- 小数点のある数: 小数点のある数による加減乗除
 - 小数点のない数に小数点のある数を足そうとすると(それはできない)、前者を小数点のある数に変換して、足し算をする
 - この変換を「型変換」という

42

型変換

- 「3.0/2」
 - 「小数を整数で割り算」することはできない
 - 整数「2」を少数「2.0」に**型変換**
 - 「3.0/2.0」として「小数を小数で割り算」
 - 結果は小数「1.5」

43

データの型②

- Rubyにある主なデータ型:
 - 小数点のない数: 整数、固定小数点数
 - integer or fixed-point number
 - 小数点のある数: 浮動小数点数
 - floating-point number
 - 文字列

44

データには型がある(例)

```
irb(main):001:0> 2+3
=> 5
irb(main):002:0> 2+3.0
=> 5.0
irb(main):003:0> 2.0+3
=> 5.0
irb(main):004:0>

irb(main):010:0> 2/3
=> 0
irb(main):011:0> 2.0/3
=> 0.6666666666666666
irb(main):012:0> 2/3.0
=> 0.6666666666666666
irb(main):013:0>
```

整数と小数を計算すると整数は小数に自動的に変換され結果は小数となる

45

データには型がある(例)

```
irb(main):002:0> 10e5
=> 1000000.0
irb(main):003:0> 10e-5
=> 0.0001
irb(main):004:0> 10**5
=> 100000
irb(main):005:0> 10**-5
=> (1/100000)
irb(main):006:0> 10.0**-5
=> 1.0e-05
```

10e5
→ 10 × 10.0⁵

表示の違いに注目

46

データには型がある(例)

```
irb(main):002:0> 10e5
=> 1000000.0
irb(main):003:0> 10e-5
=> 0.0001
irb(main):004:0> 10**5
=> 100000
irb(main):005:0> 10**-5
=> (1/100000)
irb(main):006:0> 10.0**-5
=> 1.0e-05
```

小数型

整数型

結果を分数(有理数)で表示

47

分数の扱い①

```
irb(main):001:0> 1/3
=> 0
irb(main):002:0> Rational(1,3)
=> (1/3)
irb(main):003:0> 1/3+2/3
=> 0
irb(main):004:0> Rational(1,3)+Rational(2,3)
=> (1/1)
irb(main):005:0> Rational(5,7)*Rational(2,3)
=> (10/21)
```

分数
Rational(分子, 分母)

分数の演算

48

分数の扱い②

```
irb(main):016:0> 1+Rational(1,3)
=> (4/3)
irb(main):021:0> 1.0+Rational(1,3)
=> 1.3333333333333333
```

整数と分数の演算結果 → 分数表示
小数と分数の演算結果 → 小数表示

49

文字列型

- 文字列を使用する場合は" "(ダブルクォート)で囲む

```
irb(main):001:0> "abcd"
=> "abcd"
irb(main):002:0> "xyz"
=> "xyz"
irb(main):003:0> "3+3"
=> "3+3"
irb(main):004:0>
```

式も" "で囲むと文字列

2 ふ

50

文字列型

- ' '(シングルクォート)で囲んでもよいのですが、講義では使いません

```
irb(main):001:0> 'abcd'
=> "abcd"
irb(main):002:0> 'xyz'
=> "xyz"
irb(main):003:0> '3+3'
=> "3+3"
```

7 や

51

文字列型の演算子

- +
 - 文字列 + 文字列
 - 二つの文字列の連結
- *
 - 文字列 * 整数
 - 文字列の繰り返し

52

文字列にも演算が可能①

```
irb(main):017:0> "abcd"*2
=> "abcdabcd"
irb(main):018:0> "abcd"+"efgh"
=> "abcdefgh"
irb(main):019:0> "Abc"*3
=> "AbcAbcAbc"
```

「+」「*」は可能

```
irb(main):020:0> "abcd"-"ab"
NoMethodError: undefined method '-' for "abcd":String
```

引き算は不可能

実行できないプログラムを入力した場合エラーメッセージが返ってくる

53

文字列にも演算が可能②

```
irb(main):00:0> "x+y" + "z"
=> "x+yz" x+y は式ではなく文字列
irb(main):01:0> ("abcd" + "efg") * 2
=> "abcdefghabcdefgh" 括弧も可能
irb(main):02:0> "abcd" * 2 + "efg"
=> "abcdabcdefg" 「*」の方が優先順位は強い
```

54

文字列の操作①

- 文字列.length
- 文字列.size
 - 文字列の長さを求める(値は整数)
- 文字列.reverse
 - 文字列を反転する
- 文字列[n..m]
 - 文字列のn番目からm番目を取り出す
 - ただし先頭の文字を0番目とする

55

文字列の操作②

- 文字列1.index(文字列2)
 - 文字列1の中から文字列2の位置を調べる
 - ただし先頭の文字を0番目とする
- 文字列[a , length]
 - 文字列のa番目から長さlengthの文字列を取り出す
 - ただし先頭の文字を0番目とする

56

文字列の操作③

- 文字列.upcase
 - 文字列中の小文字を大文字に変換
- 文字列.downcase
 - 文字列中の大文字を小文字に変換

57

文字列の操作(例)①

```
irb(main):00:0> "abcd".size
=> 4
irb(main):01:0> "abcd".length
=> 4
irb(main):02:0> "abcd".reverse
=> "dcba"
irb(main):03:0> "abcd"[0..1]
=> "ab"
irb(main):04:0> "abcd"[0..2]
=> "abc"
irb(main):05:0> "ruby programming".index("pro")
=> 5
irb(main):06:0> "ruby programming"[5, 10]
=> "programmin"
```

文字列の先頭の位置

"ruby programming"

0	1	2	3	4	5	6	7	8	9	10	11
r	u	b	y		p	r	o	g	r	a	m

12	13	14	15
m	i	n	g

"ruby programming"[5, 10]
=> "programmin"

先頭の r は文字列中で0番目の文字として扱われる

59

文字列の操作(例)②

```
irb(main):001:0> "abcd".upcase
=> "ABCD"
irb(main):002:0> "ABCD".downcase
=> "abcd"
irb(main):003:0> "abCD".upcase
=> "ABCD"
irb(main):004:0> "abCD".downcase
=> "abcd"
```

60

文字列の操作(例)③

```
irb(main):001:0> "abcd"*2.length
NoMethodError: undefined method `length' for 2:Fixnum
2.lengthは演算不可能

irb(main):002:0> ("abcd"*2).length
=> 8

irb(main):003:0> "abcd"*"2".length
=> "abcd"
"2".lengthは演算可能

irb(main):004:0> "abcd"[0..2].length
=> 3
"abcd"[0..2]実行した後、その結果"abc".lengthを実行
```

61

型により不可能な演算①

```
irb(main):020:0> "abcd"-ab
NoMethodError: undefined method `-` for "abcd":String
文字列同士の引き算は不可能

irb(main):021:0> 2**abcd
TypeError: String can't be coerced into Fixnum
整数に文字列は掛けることができない

irb(main):022:0>
```

62

型により不可能な演算②

```
irb(main):001:0> 1 + "abcd"
TypeError: String can't be coerced into Fixnum
整数と文字列の足し算は不可能

irb(main):002:0> 1 + 1.0
=> 2.0
整数と小数の足し算は可能

irb(main):003:0> 1 + "3"
TypeError: String can't be coerced into Fixnum
整数と文字列の足し算は不可能

irb(main):004:0> "1" + "3"
=> "13"
文字列と文字列の足し算は可能
```

63

型により不可能な演算③

```
irb(main):004:0> 3.length
NoMethodError: undefined method `length' for 3:Fixnum
整数にlengthは利用できない

irb(main):008:0> "abcd".reverse.length
=> 4

irb(main):009:0> "abcd".length.reverse
NoMethodError: undefined method `reverse' for 4:Fixnum
"abcd".lengthで整数(4)となり、
整数にreverseは利用できない
```

64

演算子

算術演算子
比較演算子

65

整数型の算術演算子

演算子	用途	例	演算結果
+	加算	3+2	5
-	減算	4-2	2
*	乗算	2*2	4
/	除算	4/2	2
%	剰余	5%2	1
**	べき	5**3	125

66

浮動小数点数型の算術演算子

演算子	用途	例	演算結果
+	加算	3.1+2.2	5.3
-	減算	4.2-2.1	2.1
*	乗算	2.1*2.1	4.41
/	除算	4.2/2.1	2.0
%	剰余	5.0%2.1	0.8
**	べき	2.1**0.5	1.44913

67

比較演算子

演算子	用途	例	演算結果
==	等	3==2	false
>	大	4 > 2	true
<	小	4 < 2	false
>=	大or等	4>=2	true
<=	小or等	4<=2	false
!=	非等	3 != 2	true

68

比較演算子①

□ 比較も演算です

```

irb(main):001:0> 2 == 2
=> true
irb(main):002:0> 2 < 3
=> true
irb(main):003:0> 2 > 3
=> false
irb(main):004:0> 2 != 2
=> false
irb(main):005:0>
    
```

これは等号
等しいか否かを
判定する演算子

演算式が正しい場合
→「true」を返す

演算式が正しくない
場合→「false」を返す

69

比較演算子②

```

irb(main):029:0> 2 == 2
=> true
irb(main):030:0> 2 < 3
=> true
irb(main):031:0> 2 >= 3
=> false
irb(main):032:0> 2 != 3
=> true
irb(main):033:0> "ab" == "ab"
=> true
irb(main):034:0> "ab" == "abc"
=> false
irb(main):035:0> "ab" < "abc"
=> true
irb(main):036:0>
    
```

整数と小数の比較は可能

```

irb(main):036:0> 2.1 < 2.2
=> true
irb(main):037:0> 2.1 > 2.2
=> false
irb(main):038:0> 2.0 == 2
=> true
irb(main):039:0> 2 < "2"
ArgumentError: comparison of Fixnum
with String failed
irb(main):040:0>
    
```

整数と文字列の比較
は不可能

文字列の比較も可能

70

比較演算子③

```

irb(main):001:0> (5+4) < 20
=> true
irb(main):002:0> (2+3) == 8
=> false
irb(main):003:0> (2+3) == (10-5)
=> true
irb(main):004:0> (2+3) != (10-5)
=> false
irb(main):001:0> "abcd".size < 4
=> false
irb(main):002:0> "abcd".upcase == "ABCD"
=> true
    
```

式と式との比較も可能

"abcd".sizeにて整数4となる

71

少々不思議な結果①

```

irb(main):001:0> 0.9
=> 0.9
irb(main):002:0> 0.99
=> 0.99
irb(main):003:0> 0.999
=> 0.999
irb(main):004:0> 0.9999
=> 0.9999
irb(main):005:0> 0.99999
=> 0.99999
irb(main):006:0> 0.999999
=> 0.999999
irb(main):007:0> 0.9999999
=> 0.9999999
irb(main):008:0> 0.99999999
=> 0.99999999
irb(main):009:0> 0.999999999
=> 0.999999999
    
```

```

irb(main):010:0> 0.9999999999
=> 0.9999999999
irb(main):011:0> 0.99999999999
=> 0.99999999999
irb(main):012:0> 0.999999999999
=> 0.999999999999
irb(main):013:0> 0.9999999999999
=> 0.9999999999999
irb(main):014:0> 0.99999999999999
=> 0.99999999999999
irb(main):015:0> 0.999999999999999
=> 0.999999999999999
irb(main):016:0> 0.9999999999999999
=> 1.0
irb(main):017:0> 0.99999999999999999
=> 1.0
    
```

有限桁数かつ四捨五入の世界だからです

72

少々不思議な結果②

```
irb(main):040:0> 2.0/3  
=> 0.6666666666666666  
irb(main):041:0> 2.0/3 == 0.6666666666666666  
=> false  
irb(main):042:0> 2.0/3 == 0.6666666666666666666666666666  
=> true  
irb(main):043:0>
```

勿論、有限桁数かつ四捨五入の世界だからです

73

少々不思議な結果③

```
irb(main):026:0> x=0.99; for i in 3..18; x=0.9+x/10.0; print i, " ",x,"\n";end  
3 0.999  
4 0.9999  
5 0.99999  
6 0.9999990000000001  
7 0.9999999  
8 0.9999999000000001  
9 0.999999999  
10 0.9999999999  
11 0.99999999999  
12 0.999999999999  
13 0.999999999999001  
14 0.999999999999999  
15 0.9999999999999999  
16 0.99999999999999999  
17 1.0  
18 1.0  
=> 3..18=> 3..18  
irb(main):027:0>
```

これは、後ほど

xの初期値を0.99として、
 $x=0.9+x/10.0$
の計算を繰り返すプログラム

正しい値?

本当は0.9999999999999999のはず

本当は0.9999999999999999のはず

桁数が有限だから起こる不思議です

74

型変換

75

型変換①

- 整数と小数の演算
→整数は小数に自動的に変換され、結果は小数となる
- 整数(小数)と文字列の演算
 - 不可能
- 他のデータ型に変換することを型変換と呼ぶ

76

整数型への変換①

- 整数に変換

`3.1415.to_i`

整数へ変換

`"3".to_i`

値.to_i

`"3".to_i + 5`

77

整数型への変換②

```
irb(main):001:0> 3.1415.to_i  
=> 3  
irb(main):005:0> "3".to_i  
=> 3  
irb(main):009:0> "3" + 5  
TypeError: can't convert Fixnum into String  
from (irb):9:in `+'  
from (irb):9  
irb(main):006:0> "3".to_i + 5  
=> 8
```

文字列と整数の足し算は不可能

文字列を整数に変換することで可能

78

to_i を使う時の注意①

```
irb(main):001:0> "123".to_i + 10
=> 133
irb(main):002:0> "123".to_i+10
=> 133
irb(main):003:0> "123".to_i + 10
=> 123
irb(main):004:0> "123".to_i + 10
=> 133
```

なぜ答えが123?

to_i を使う時の注意②

- 「to_i X」でX進数に変換されます
↑
スペース
- Xを省略した場合は10進数に変換されます
↑
スペース
- 「"110".to_i + 10」は「文字列110を整数に変換した後、10を加算する」のではなく、「文字列110を+10進数に変換する」ということになります

to_i を使う時の注意③

```
irb(main):003:0> "123".to_i + 10
=> 123
```

空白(スペース)を空ける→
次の値(+10)を基数として整数変換

```
irb(main):004:0> "123".to_i + 10
=> 133
```

空白(スペース)を空ける→+の後も空白
→ 10を足すと解釈

to_i を使う時の注意④

```
irb(main):027:0> "110".to_i 2
=> 6
```

"110"を2進数で整数変換

```
irb(main):028:0> "110".to_i 10
=> 110
```

"110"を10進数で整数変換

```
irb(main):029:0> "110".to_i + 10
=> 110
```

```
irb(main):030:0> "110".to_i + 2
=> 6
```

"110"を-10進数?で整数変換
→不可能

```
irb(main):033:0> "110".to_i -10
ArgumentError: invalid radix -10
  from (irb):33:in `to_i'
  from (irb):33
  from C:/Program Files/Ruby-2.3.0/bin/irb.bat:19:in `<main>'
```

to_i を使う時の注意⑤

- 文字列123を整数に変換した後、10を加算したい場合
- (○) "123".to_i + 10
- (○) "123".to_i+10
- (○) "123".to_i + 10
- (×) "123".to_i + 10

スペースを入れずに書くことをおすすめ

小数型への変換①

- 小数に変換

3.to_f

小数へ変換

"3.1415".to_f

値.to_f

"3".to_f

"3.1415".to_f * 2.5

小数型への変換②

```
irb(main):001:0> 3.to_f
=> 3.0
irb(main):002:0> "3.1415".to_f
=> 3.1415
irb(main):003:0> "3".to_f
=> 3.0
irb(main):004:0> "3.1415".to_f*2.5
=> 7.85375
```

85

文字列型への変換①

- 文字列に変換

`3.to_s`

`3.1415.to_s`

`3.to_s + "5"`

`3.1415.to_s * 2`

文字列へ変換
値.to_s

86

文字列型への変換②

```
irb(main):007:0> 3.to_s
=> "3"
irb(main):008:0> 3.1415.to_s
=> "3.1415"
irb(main):009:0> 3.to_s+"5"
=> "35"
irb(main):010:0> 3.1415.to_s*2
=> "3.14153.1415"
```

87

型変換のまとめ

```
irb(main):007:0> "3.1415".to_f
=> 3.1415
irb(main):008:0> "3.1415".to_f.to_i
=> 3
irb(main):009:0> "3.1415".to_f.to_i.to_s
=> "3"
```

小数に変換

小数→整数に変換

小数→整数→文字列に変換

88

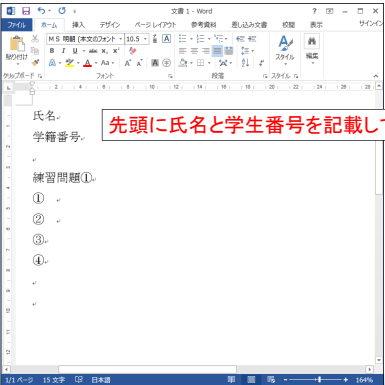
練習問題

89

練習問題

- 次頁以降の練習問題を行ないなさい。
- MS-Wordに回答を記述し、keio.jpに提出して下さい
 - 提出先
 - 第二回講義練習問題
 - 提出締め切り
 - 4/17(月)16時30分まで
 - 提出するMS-Wordファイルの先頭に学籍番号、氏名を書いて下さい

90



先頭に氏名と学生番号を記載して下さい

氏名
学籍番号

練習問題①

①
②
③
④

ファイル名は自由につけて下さい

91

練習問題

- 練習①～⑤の実行結果がどうなるか答えなさい。練習⑥について答えなさい。
(実行する前に、答えを考えて、その答えが当たっているかを確認するため、実行することが望ましい)
- 結果が true , false になる式を5個ずつ考えなさい

92

練習①(データには型がある) 結果がどうなるか考えて下さい

- $1+2+3+4+5$
• 結果は?
- $1+2+3+4+5.0$
• 結果は?
- $(1+2+3+4+5) / 2$
• 結果は?
- $(1+2+3+4+5) / 2.0$
• 結果は?

93

練習②(整数と小数) 結果がどうなるか考えて下さい

- $10^{**}2$
- $10.0^{**}2$
- $(1/3)*3$
- $(1/3)*3.0$
- $(1.0/3)*3$

94

練習③整数型への変換 結果がどうなるか考えて下さい

- $3.1415 * 2$
- $3.1415.to_i * 2$
- $3.1415.to_i*2.0$
- $(3.1415 * 2).to_i$
- $Rational(4,3).to_i$

95

練習④小数型への変換 結果がどうなるか考えて下さい

- $3 / 2$
- $3.to_f/2$
- $3.1415.to_i+2.to_f$
- $(3.1415.to_i + 2.to_f) / 3.1415.to_i$
- $Rational(2,7).to_f$

96

練習⑤文字列型への変換
結果がどうなるか考えて下さい

- ① `3.to_s*2`
- ② `3.1415.to_s.length`
- ③ `3.1415.to_i.to_s`
- ④ `("abc" > "abcd").to_s`

97

練習⑥(型変換)

- ① $1/2 + 3$ を行なうと、結果は 3 となります。結果を型変換 (`to_f`) によって 3.5 にするためには、式のどこを変えればよいか？
- ② $(1/3 + 2/3)/3$ を行なうと、結果は 0 となります。結果を型変換 (`to_f`) によって 0.3333333333333333 にするためには、式のどこを変えればよいか？

98