プログラミング言語
第2回　4月21日

担当：篠沢 佳久
櫻井 彰人

平成20年度：春学期

まずは注意点

- マイドキュメントに「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

※日吉では、マイドキュメント と H ドライブは同一。
マイドキュメントの中の Ruby フォルダと、
“H:¥Ruby” は同じフォルダ（ディレクトリ）を指す。

本日の内容

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

interactive Ruby

irbの起動と終了
irbでのRubyプログラムの実行方法

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、一行で書け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

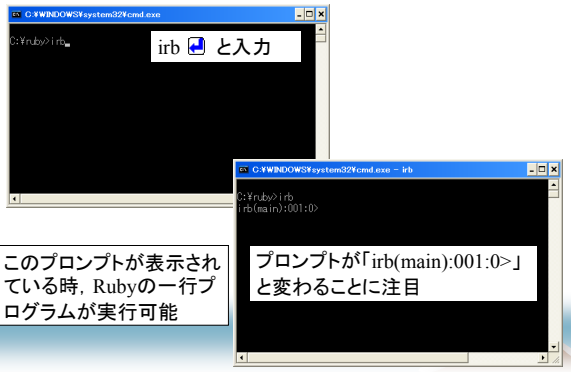
↵
Enterキー

- 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

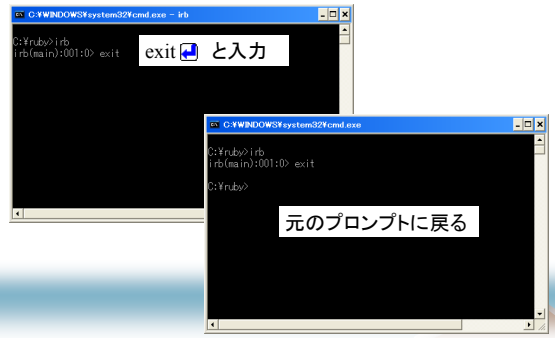
- 終了したいときには
  exit ↵
  と入力

irbの起動②

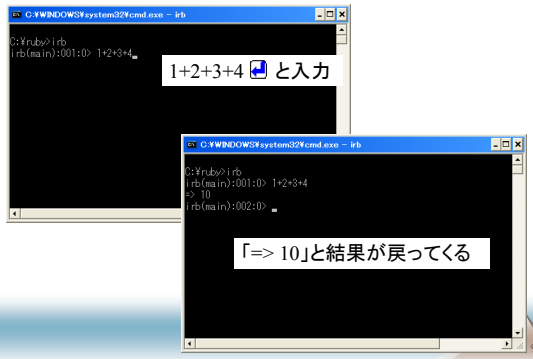


このプロンプトが表示されている時，Rubyの一行プログラムが実行可能

irbの終了



irb での入力方法



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
=> 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>
```

代入式

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

http://www.ruby-lang.org/ja/man/?cmd=view;name=Math

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

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

強い ↑

**
/ * %
+ −

演算の優先順位②

- 5 + － 3
  => 2
- －(3 － 10)
  => 7
- 3 ** －2
  => 0.11111111111111
- －3 ** 2
  => －9
- －3 ** －2
  => －0.111111111111
- (－3) ** －2
  => 0.111111111111

強い

括弧 （）
**
符号 ＋ －
/ * %
＋ －

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)

π
Math::PI

ラジアン

数学用関数②

- 自然対数
  - Math.log(Math::E)
- 常用対数
  - Math.log10(100)
- 指数
  - Math.exp(2)

e
Math::E

その他の数学関数の一例
http://www.ruby-lang.org/ja/man/?cmd=view;name=Math

練習①（計算してみて下さい）

① Math.sin(Math::PI/4)**2 + Math.cos(Math::PI/4) **2

② Math.sin(Math::PI/4)**2

③ (1-Math.cos(Math::PI/2))/2

④ Math.log(Math.exp(2))

代入式①

- x = 2
- y = 10
- (x + y) / 2
  => 6
- x * y
  => 20
- Math.sqrt(y / x)
  => 2.23606797749979

x , y を変数
x = 2 を代入式と呼ぶ

代入式②

- x = 2
- y = 10
- a = (x + y) / 2
- b = x * y
- c = y / x
- d = Math.sqrt(c)

前ページと同じ計算
求めた値を変数に代入することも可能

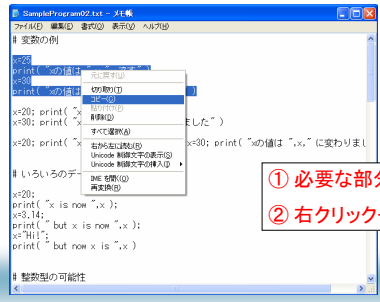
irbでの編集

- 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

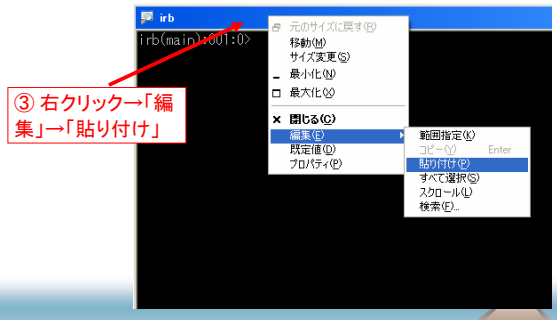
Ctrl+a
Ctrlキーを押しながらa

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

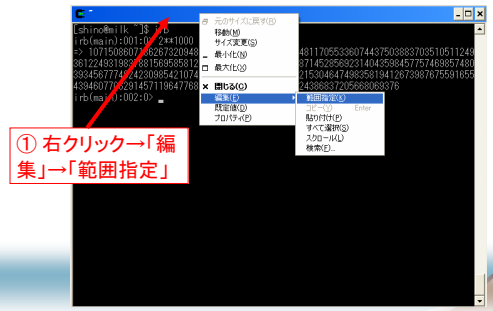


irbへの貼り付け(paste)②

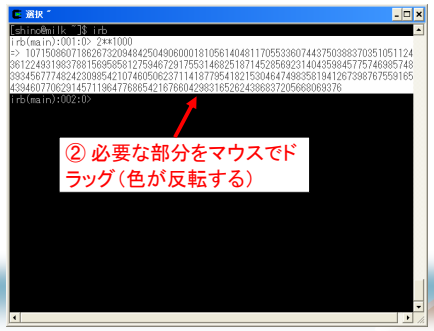


irbからのコピー①

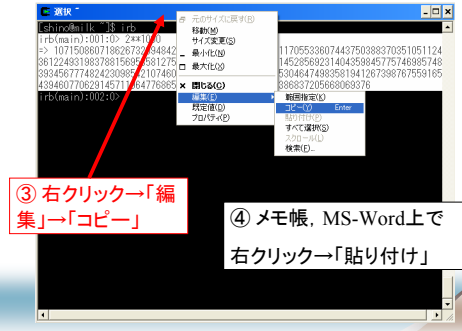
irb上のプログラム，結果をメモ帳などにコピーしたい場合



irbからのコピー②

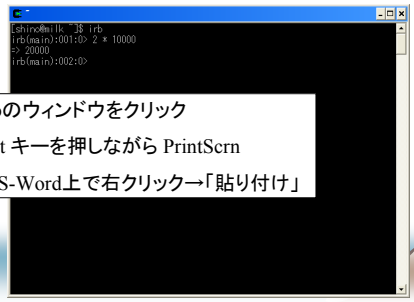


irbからのコピー③



irbの画面のコピー

irbの画面をMS-Word等に貼り付けたい場合



① irbのウィンドウをクリック
② Alt キーを押しながら PrintScrn
③ MS-Word上で右クリック→「貼り付け」

データの型

整数型
浮動小数点型
文字列型

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

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」

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

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

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
=> 0.666666666666667
irb(main):012:0> 2/3.0
=> 0.666666666666667
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

練習②（データには型がある）

① 1+2+3+4+5
- 結果は？

② 1+2+3+4+5.0
- 結果は？

③ （1+2+3+4+5）/ 2
- 結果は？

④ (1+2+3+4+5) / 2.0
- 結果は？

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>
```

"
2 ふ

式も" "で囲むと文字列

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

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
        from (irb):20
        from :0
```

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

文字列にも演算が可能②

```
irb(main):00:0> "x+y" + "z"
=> "x+yz"
irb(main):01:0> ( "abcd" + "efg" ) * 2
=> "abcdefgabcdefg"
irb(main):02:0> "abcd" * 2 + "efg"
=> “abcdabcdefg”
```

x+y は式ではなく文字列

括弧も可能

「＊」の方が優先順位は強い

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

文字列の操作（例）

```
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> "ruby programming".index( "pro" )
=> 5
irb(main):04:0> "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

先頭の r は文字列中で0番目の文字として扱われる

練習③（文字列の操作）

① ("abcd" + "efg").length

② (("abcd" + "efg") * 2).length

③ "ruby programming".length + 5

④ "ruby programming".length + 5.0

型により不可能な演算①

```
irb(main):020:0> "abcd"-"ab"
NoMethodError: undefined method `-' for "abcd":String
        from (irb):20
        from :0
irb(main):021:0> 2*"abcd"
TypeError: String can't be coerced into Fixnum
        from (irb):21:in `*'
        from (irb):21
        from :0
irb(main):022:0>
```

文字列同士の引き算は不可能

整数に文字列は掛けることができない

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
        from (irb):1:in `+'
        from (irb):1
irb(main):002:0> 1 + 1.0
=> 2.0
```

整数と文字列の足し算は不可能

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
        from (irb):3:in `+'
        from (irb):3
irb(main):004:0> "1" + "3"
=> "13"
```

整数と文字列の足し算は不可能

文字列と文字列の足し算は可能

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

演算子

- 算術演算子
- 比較演算子

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

浮動少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

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

比較演算子①

- 比較も演算です。

これは等号。
等しいか否かを
判定する演算子

```
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>
```

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

比較演算子②

整数と小数の比較は可能

```
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>
```

文字列の比較も可能

```
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
        from (irb):39:in `<'
        from (irb):39
        from :0
irb(main):040:0>
```

整数と文字列の比較は不可能

少々不思議な結果②

```
irb(main):040:0> 2.0/3
=> 0.666666666666667
irb(main):041:0> 2.0/3 == 0.66666666666666
=> false
irb(main):042:0> 2.0/3 == 0.666666666666666666666
=> 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

少々不思議な結果③

これは、後ほど

```
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  0.999999
7  0.9999999
8  0.99999999
9  0.999999999
10  0.9999999999
11  0.99999999999
12  0.999999999999
13  0.9999999999999
14  0.99999999999999
15  0.999999999999999
16  1.0
17  1.0
18  1.0
=> 3..18
irb(main):027:0>
```

xの初期値を0.999として，

x=0.9+x/10.0

の計算を繰り返すプログラム

本当は0.9999999999999999のはず

本当は0.99999999999999999のはず

本当は0.999999999999999999のはず

桁数が有限だから起こる不思議です

型変換

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

整数型への変換①

- 整数に変換

3.1415.to_i

"3".to_i

"3".to_i + 5

整数へ変換
値.to_i

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
        from :0
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

練習④整数型への変換
結果がどうなるか考えて下さい

① 3.1415 * 2

② 3.1415.to_i * 2

③ 3.1415.to_i * 2.0

④ (3.1415 * 2).to_i

小数型への変換

- 小数に変換

3.to_f

"3.1415".to_f

"3".to_f

"3.1415".to_f * 2.5

小数へ変換
値.to_f

練習⑤小数型への変換
結果がどうなるか考えて下さい

① 3 / 2

② 3.to_f / 2

③ 3.1415.to_i + 2.to_f

④ (3.1415.to_i + 2.to_f) / 3.1415.to_i

文字列型への変換

- 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

練習問題

1．練習①～⑤についてirb上で実行しなさい

2．2/3 == 2.0/3
　2/3 < 2.0/3
を実行し，出力結果のようになる理由を考察しなさい

慶應義塾共通認証システム（keio.jp）

- WEBメール
- 図書館利用状況
- 教育支援システム
- お知らせ
- 各種サービス

keio.jp のアカウント

- keio.jp アカウントをまだ取得していない学生は，取得しておいて下さい
  - この講義ではレポートの提出に利用します
  - つけた名前，パスワードは絶対に忘れないで下さい

keio.jpアカウント

- メールアドレスの形式
  - 電子メールアドレスとしても利用
- 希望の文字列をつけることができる

自動的に決まる

keiotaro@z1.keio.jp

好きなようにつけて良い

http://keiojp.itc.keio.ac.jp/manual/activation/index.html