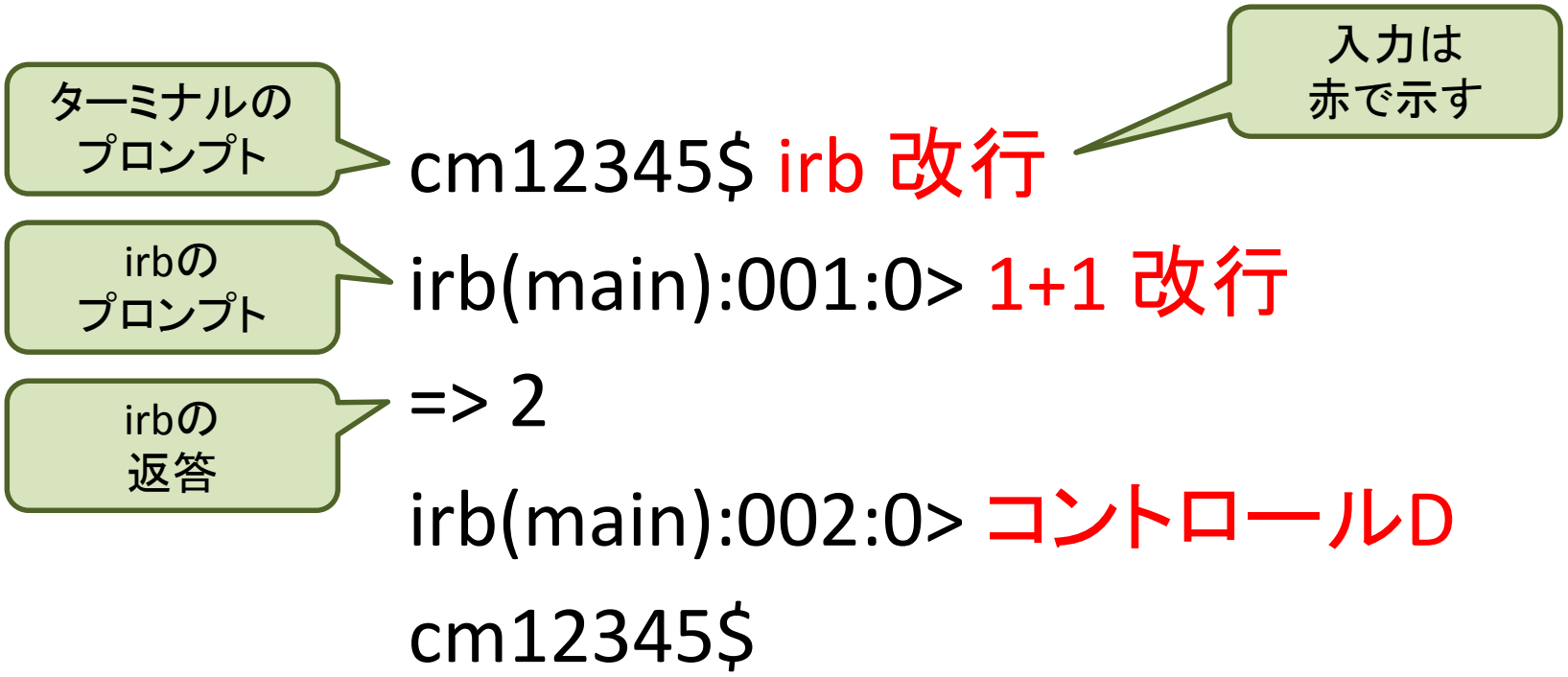
数の計算と関数

コンピュータとの対話

- ターミナルの起動 ⇒ irbの起動 ⇒ 数式の入力



- 今回は、指示されるまで、irbを使いながら…

数式の計算 --- 電卓がわり

以下、
改行は省略

7を2で割った
余り

```
irb(main):003:0> 7 – 2
=> 5
rb(main):004:0> 7 * 2
=> 14
irb(main):005:0> 7 / 2
=> 3
```

```
irb(main):006:0> 7 % 2
=> 1
irb(main):007:0> 7 ** 2
=> 49
```

7の2乗

電卓がわり

```
irb(main):009:0> 7 - 2 * 3
=> 1
irb(main):010:0> (7 - 2) * 3
=> 15

irb(main):012:0> 7.0 / 2
=> 3.5
irb(main):013:0> 7 / 2.0
=> 3.5
```

17 - 17/3*3 の値は

1. 0.0
2. 0
3. 2
4. 15.111111111111
5. 16

56 の16乗として間違っているのは

1. 56 ** 16
2. (7 * 8) ** 16
3. 7 * 8 ** 16
4. 56 ** 4 ** 2
5. 56 ** (4 ** 2)

さまざまなエラー

```
irb(main):001:0> 3/0
ZeroDivisionError: divided by 0
        from (irb):1:in `/'
        from (irb):1
irb(main):002:0> 7 - 2 3
SyntaxError: compile error
(irb):2: syntax error, unexpected tINTEGER, expecting $end
        from (irb):2
irb(main):003:0> (7 -
irb(main):004:1* 2) * 3)
SyntaxError: compile error
(irb):4: syntax error, unexpected ')', expecting $end
        from (irb):4
irb(main):005:0>
```

式の途中で改行すると
プロンプトが異なる

さまざまなエラー

```
irb(main):013:0> bm1(188.0 ,104.0)
NoMethodError: undefined method 'bm1' for main:
   Object
    from (irb):13
```

わけがわからなくなったら

- ともかくコントロールCを押す
  - irbはトップレベルの戻る

数学関数

```
irb(main):003:0> include(Math)
=> Object
irb(main):004:0> sqrt(2)
=> 1.4142135623731
irb( main ):005:0> cos(3.141592/3)
=> 0.50000018867511
```

数学関数を使う準備
irbを起動し直す
たびに必要

黄金比の値は

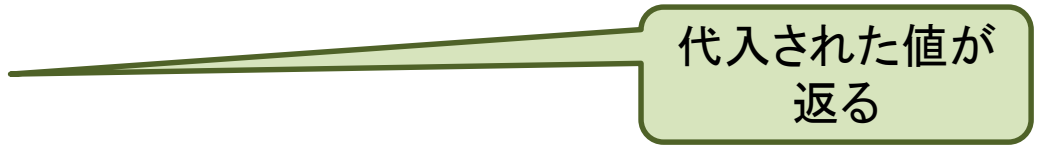
1. 1.61803398874989
2. 1.61803398874988
3. 1.61803398874987
4. 1.61803398874986
5. 1.61803398874985

$$\frac{1+\sqrt{5}}{2}$$

変数 --- 値に名前を付ける


```
irb(main):003:0> h=188.0
=> 188.0
irb(main):004:0> w=104.0
=> 104.0

irb(main):006:0> w / (h /100.0) ** 2
=> 29.4250792213671
```



変数を使うわけ

- 式の意味が理解しやすくなる

  w

  weight

  body_weight_in_pound

変数（局所変数）は、
小文字で始まる英数字列
アンダースコアは
小文字と考える

- 違う値で計算のやり直しができる

  irb(main):008:0> w=104.0-10

  => 94.0

  irb(main):009:0> w / (h/100.0) ** 2

  => 26.5957446808511

w=w-10
としてもよい

irbへの入力

- コントロールPもしくは上矢印を入力すると、直前の入力が復活する。
- コントロールB（もしくは左矢印）でカーソルは左に移動。
- コントロールF（もしくは右矢印）でカーソルは右に移動。
- 通常の文字はカーソル位置に挿入される。
- コントロールDでカーソル位置の文字が削除される。
- バックスペースでカーソルの直前の文字が削除される。

関数の定義 --- BMIを求める関数

```
irb(main):003:0> def bmi(height , weight )
irb(main):004:1>     weight / (height/100.0) ** 2
irb(main):005:1> end
=> nil

irb(main):007:0> bmi(188.0 ,104.0)
=> 29.4250792213671
irb(main):008:0> 1.1*bmi(174.0, 119.0 * 0.454)
=> 19.6289470207425
```

練習

1. 平面上の2 点(x, y) と(u, v) の距離を求める distance(x,y,u,v).
2. f フィートi インチをセンチメートルに変換する feet_to_cm(f,i). ただし、1 フィート= 12 インチ = 30.48 cm である。
3. p ポンドo オンスをキログラムに変換する pound_to_kg(p,o). 1 ポンド= 16 オンス= 0.4536 kg である。

関数を使う関数

関数も、
小文字で始まる英数字列
アンダースコアは
小文字と考える

```
irb( main ):010:0> def bmi_yp (f,i,p,o)
irb( main ):011:1>     bmi(feet_to_cm (f,i),
irb( main ):012:2*         pound_to_kg(p,o))
irb( main ):013:1> end
=> nil

irb( main ):015:0> bmi_yp(5,11,170,0)
=> 23.710342996960538
```

式の途中で改行すると
プロンプトが異なる

ファイルに保存した関数定義 ---ファイルからの読み込み

#以下
行末まで
コメント

```
# BMI of a person with height (cm) and weight (kg)
def bmi(height , weight )
   weight / ( height /100.0) ** 2
end
```

bmi.rb

```
irb(main):003:0> load("./ bmi.rb")
=> true

irb(main):005:0> bmi(188.0 , 104.0)
=> 29.4250792213671
```

ファイルを読み込むファイル

```
load ("./ bmi.rb")
load ("./ yardpound .rb")


def bmi_yp (f,i,p,o)
    bmi(feet_to_cm (f,i), pound_to_kg(p,o))
end
```

bmi_yp.rb

定数関数

```
# BMI of a person with height (cm) and weight (kg)
def bmi(height , weight )
   weight / ( height /100.0) ** 2
end
def k_height ()       #K選手の身長
   188.0
end
def k_weight ()       #K選手の体重
   104.0
end
```

bmi.rb

定数関数

```
irb( main ):004:0> load("./ bmi.rb")
=> true
irb( main ):005:0> k_weight()
=> 104.0
irb( main ):006:0> bmi(k_height(), k_weight())
=> 29.4250792213671
```

局所変数

```
def heron (a,b,c)
    s = 0.5*(a+b+c)
    sqrt(s * (s-a) * (s-b) * (s-c))
end
```

局所変数の定義

練習

- 二次方程式$ax^2 + bx + c = 0$ に関して

  (a) 判別式$b^2 – 4ac$ を求めるd(a,b,c).

  (b) 解の1 つを求めるsolution1(a,b,c). (d を使って定義せよ。)

  (c) もう1 つの解を求めるsolution2(a,b,c).

  (d) 二次関数$f(x) = ax^2+bx+c$ の値を求める quadratic(a,b,c,x).

1.6　定義のまとめ

この章で紹介した式や命令をまとておこう。

`include(Math)`： `cos` や `sqrt` などの数学関数を使う前に、実行しておかなければいけない命令。

変数名 = 式：右辺の式を計算した値を、左辺に書かれた名前の変数にしまう**代入命令**。

関数名(式$_1$,...,式$_n$)：　式$_1$,...,式$_n$を計算した値を、関数に渡す**関数呼び出し式**。関数に渡される式の値のことを**引数**(ひきすう)という。関数名は `sqrt` のような予め定義されている関数でも、`bmi` のような自分で定義した関数でもよい。

def 関数名(式$_1$,...,式$_n$) ↵ 式 ↵ end： **関数定義**。少し複雑なので、具体例で説明する。

def と end は、定義の範囲を示している。**関数名**(ここでは bmi) は、変数名と同様、好きな名前を付けることができる。次の (height,weight) は、関数が値を受け取るために使う変数名である。