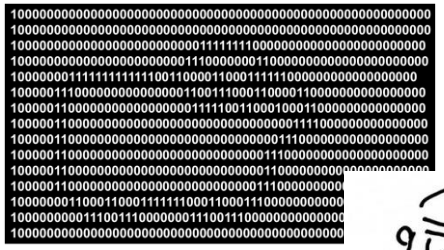
CS第1　テーマ1

テーマ1の目標

計算の基本要素を知る



ひつじさん

ちょいと苦しい

ドンマイ

演習課題

四則演算でアニメーション

内　容

1. はじめに
2. データ = 数　教科書 1.1
3. コンピュータの中では　教科書 2.1
4. 計算 = ±1 と繰り返し　教科書 1.2
5. Ruby での書き方　宿題
   - プログラムの基本
   - 分岐，繰り返し

1. はじめに

CSのこころ

すべては計算である

- （1＋4）×5 ＝
- 12 と 16 の最大公約数
- $x^2 + 2xy + y^2$ の因数分解
- 原子炉の設計図を作成する
- 遺伝子を解析する
- 銀行のATMの制御
- 木の成長
- 脳の形成

ただし，コンピュータに載せるには …

⇨ 必須

対象をデータとして表すこと
処理を基本演算の組合せで表すこと

2. データは数である

教科書 1.1

データ ＝ 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが二進列

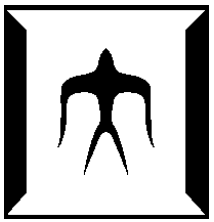
0 と 1 の列

確かめてみよう（例で考える）

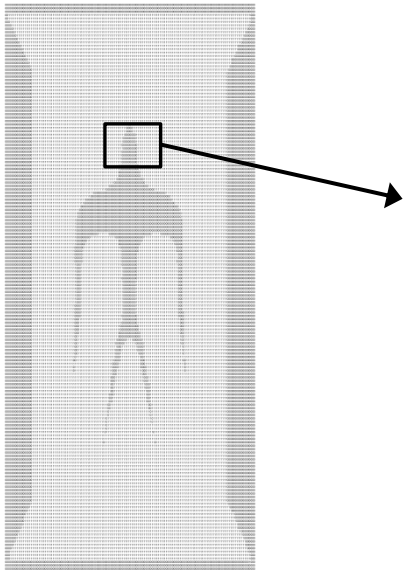
練習

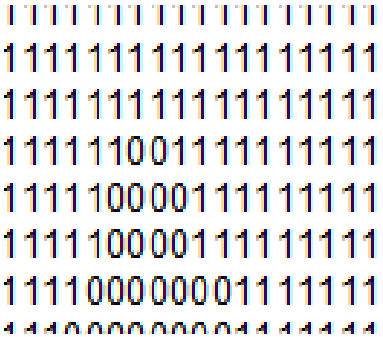
- 数　　18, –5, 3.25, 1/3
- 文字　　a ← 01100001 (=97), b ← 01100010 (=98)
- 画像
- 音
- 映像

ASCII という符号法



→





….

- 味，におい？？

2. データは数である

教科書 1.1

データ ＝ 計算の対象

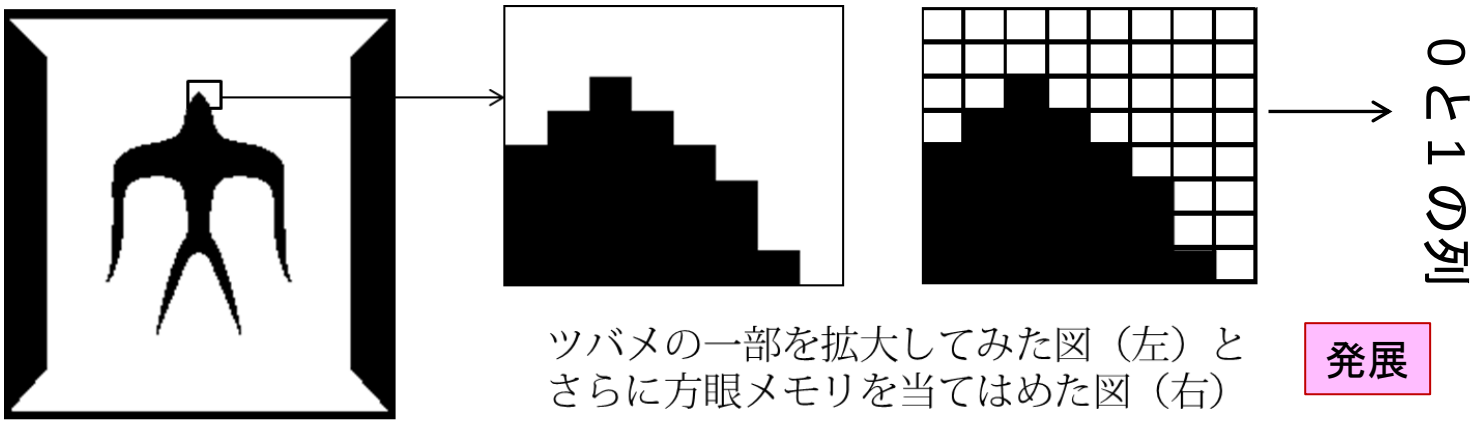
どこかで聞いたね

コンピュータの中ではすべてが**二進列**

0と1の列

確かめてみよう（例で考える）

- 数
- 文字
- 画像



ツバメの一部を拡大してみた図（左）と
さらに方眼メモリを当てはめた図（右）

発展

デジタル化とは
方眼紙をあてることなり

2. データは数である

教科書 1.1

データ ＝ 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが二進列

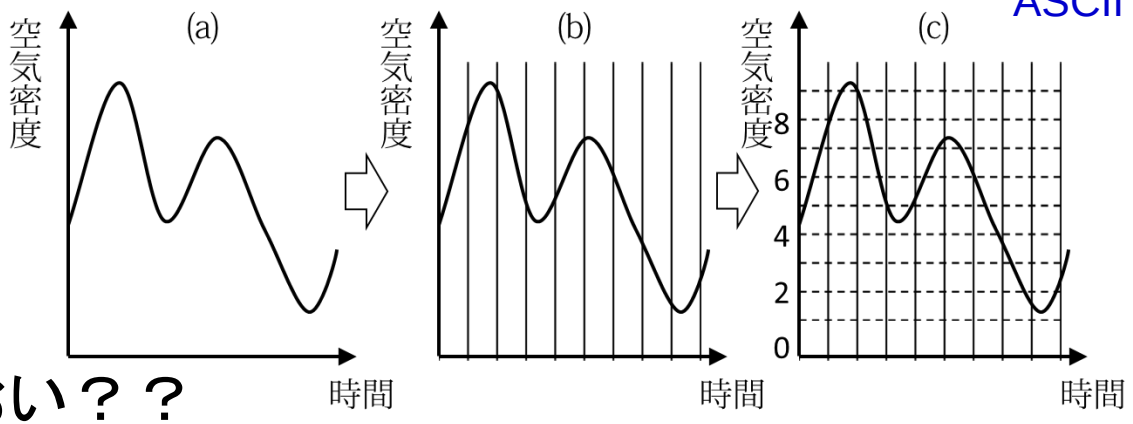
0 と 1 の列

確かめてみよう（例で考える）

- 数　　18, −5, 3.25, 1/3
- 文字　　a ← 01100001 (=97), b ← 01100010 (=98)
- 画像
- 音
- 映像

….

ASCII という符号法



- 味，におい？？

練習

2. コンピュータの中では

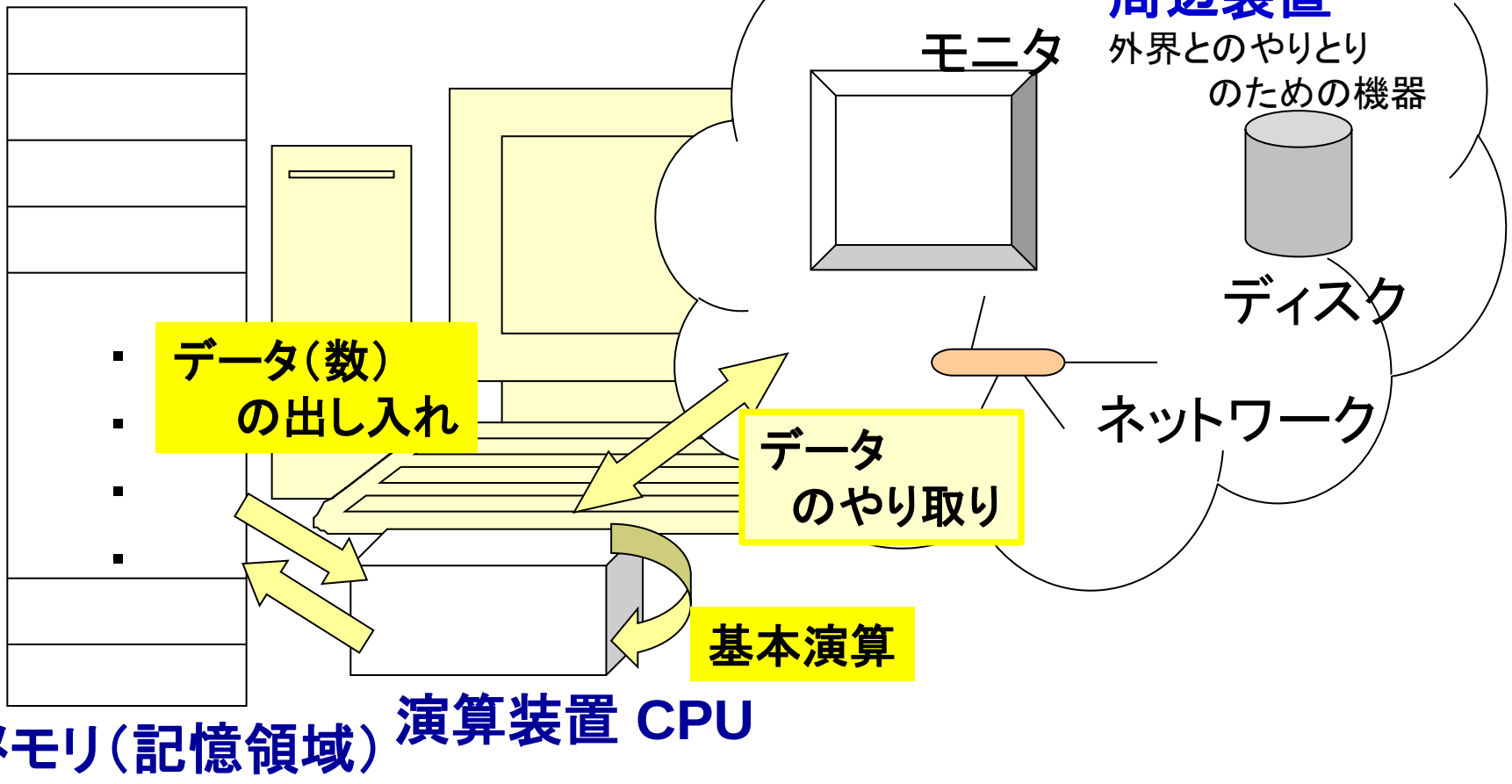
教科書 2.1

データ ＝ 計算の対象 ＝ ~~二進列~~ 自然数 ↔ 0, 1, 2, 3, ...

話を簡単にするため

とする

コンピュータ内の処理



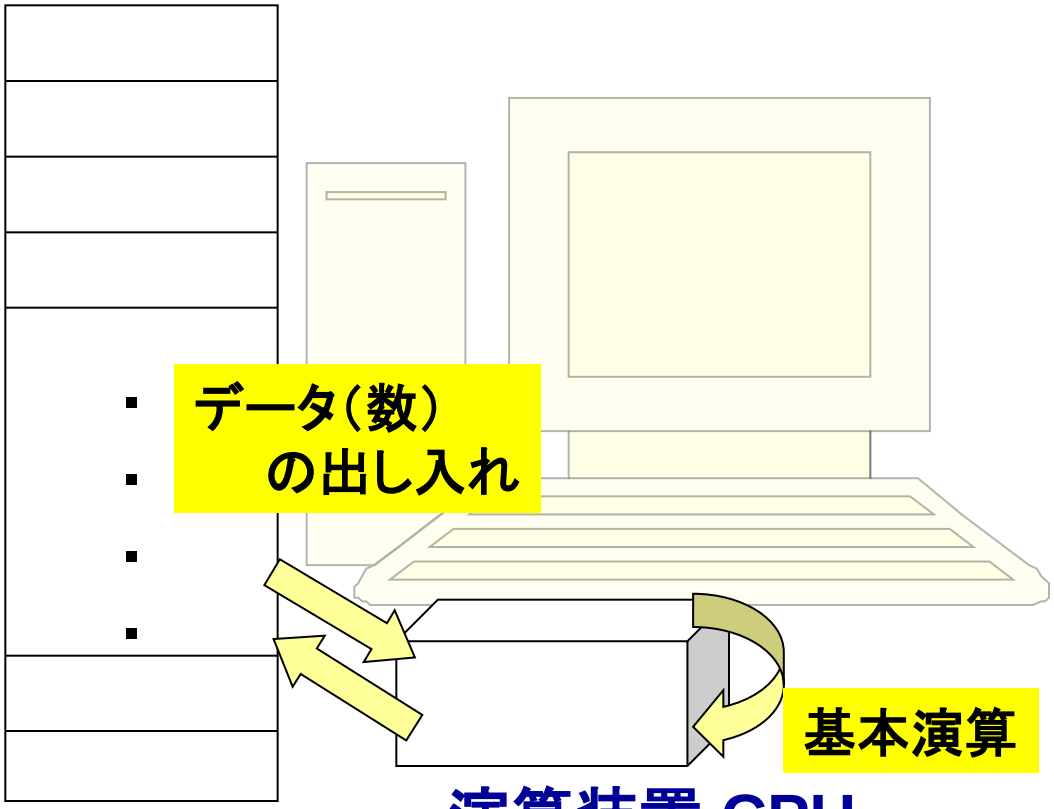
3.計算 = ±1 と繰り返し

教科書 1.2

データ = 自然数

計算 = 超基本要素は
±1 と繰り返し

コンピュータ内の処理



物質で言えば原子
のような基本要素

究極的に絞り込むと
(この話は少し後で)

四則演算
繰り返し
条件分岐

3.計算 = ±1 と繰り返し

教科書 1.2

データ = 自然数

計算 = 超基本要素は
±1 と**繰り返し**
まあ，**条件分岐**も！

これだけで ??

計算の設計図
プログラム ⟷ 指示書，命令書

プログラムの例：平方根を求めるプログラム

```
sq.rb
n = gets().to_i
a = 1;  a2 = a * a
while a2 <= n
   a = a + 1
   a2 = a * a
end
puts(a - 1)
```

sq.rb — プログラムの名前
n = gets().to_i — データの入力
a = 1; a2 = a * a / while a2 <= n / a = a + 1 / a2 = a * a / end — 実際の計算の部分
puts(a - 1) — データの出力

使われ方
実行例

```
$ ruby sq.rb
26
5
$
```

4. Ruby での書き方（その1）

Ruby ↔ プログラミング言語（プログラムを書く言葉）の1つ．まつもと氏が開発した言語

プログラム（指示書）一般に共通するルール

- プログラムはプログラム名のファイルに格納．
- プログラムは命令の列．原則として上から下に順に実行．（繰り返し，条件分岐以外は）
- **変数** ↔ データの格納場所
- 基本命令は
  - (1) 変数への**代入文**
    （右辺の計算結果を代入）
  - (2) **繰り返し文**
  - (3) **条件分岐文**

sq.rb （例）

```
n = gets()
a = 1;  a2 = a * a
while a2 <= n
   a = a + 1
   a2 = a * a
end
puts(a - 1)
```

4. Ruby での書き方（その2）

Ruby での書き方のルール（例を見ながら）

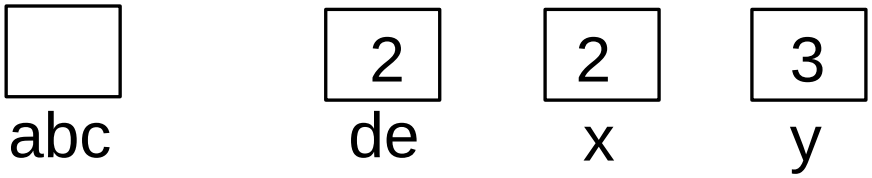
8 + 3 を求める計算

代入文 ↔ 変数に値を格納せよ，という命令

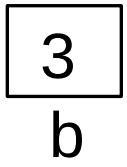
【文の構造】

変数名 = 計算式

例） abc = 3 * (4 - de) + 6 / x * y



例） b = b - 1



add_8_3.rb

```
a = 8
b = 3
wa = a
while b > 0
    wa = wa + 1
    b = b - 1
end
puts(wa)
```

4. Ruby での書き方　補足

【演算子】

演算	使用例	意味
+	x + y	x と y の足し算
-	x - y	x から y の引き算
*	x * y	x と y の掛け算
/	x / y	x を y で割った商
%	x % y	x を y で割った余り
**	x ** y	x の y 乗

・この講義では**整数変数**（整数格納用の変数）のみとする.
（最初に整数を入れると整数変数と認識される. ※1）
・整数変数が入った割り算の答えは整数（切り捨て値）となる.

※1　Ruby 特有の方便. 本格プログラムではお勧めできない.
※2　Ruby 以外の言語では ** は使えない場合も多い.

4. Ruby での書き方（その3）

Ruby での書き方のルール（例を見ながら）

8 + 3 を求める計算

繰り返し文　while 文

↔ 指定の条件が成立する間繰り返せという命令

【文の構造】

```
while  条件
    繰り返される命令列
    （複数の文もOK）
end
```

例）

add_8_3.rb

```
a = 8
b = 3
wa = a
while b > 0
    wa = wa + 1
    b = b - 1
end
puts(wa)
```

4. Ruby での書き方　例題

掛算を±1 と繰り返しのみで実現する

mult.rb

```
x = 入力データ
y = 入力データ
seki = 0
while y > 0
    seki = seki + x
    y = y - 1
end
puts(seki)
```

add.rb　（参考）

```
a = 入力データ
b = 入力データ
wa = a
while b > 0
    wa = wa + 1
    b = b - 1
end
puts(wa)
```

±1 以外も使ってる

4. Ruby での書き方　例題

掛算を±1 と繰り返しのみで実現する

mult.rb

```
x = 入力データ
y = 入力データ
seki = 0
while y > 0
  seki = seki + x
  y = y - 1
end
puts(seki)
```

```
a = seki
b = x
wa = a
while b > 0
  wa = wa + 1
  b = b - 1
end
seki = wa
```

```
x = 入力データ
y = 入力データ
seki = 0
while y > 0
  wa = seki; b =x;
  while b > 0
    wa = wa + 1
    b = b - 1
  end
  seki = wa
  y = y - 1
end
puts(seki)
```

宿題：次回の授業開始まで
次のプログラムを紙に書いてくること
(1) 引き算を±1 と繰り返しのみで計算する
(2) 割り算を加減算と繰り返しのみで計算する

4. Ruby での書き方　補足

条件式 ↔ 条件を指定する式．while 文，if 文などで使う

例）while　a >= x ** 2
　　if　b % 2 != 0　（if 文については次で）

【関係演算子】

関係	使用例	意味
>=	x >= y	x は y より大きいかまたは等しい
>	x > y	x は y より大きい
==	x == y	x は y と等しい
!=	x != y	x は y は等しくない
<	x < y	x は y より小さい
<=	x <= y	x は y より小さいかまたは等しい

※ 究極的には > だけに限ってもよい（つまり，他を言い換えられる）．

発展

4. Ruby での書き方（その4，最後 ;-)

Ruby での書き方のルール（例を見ながら）

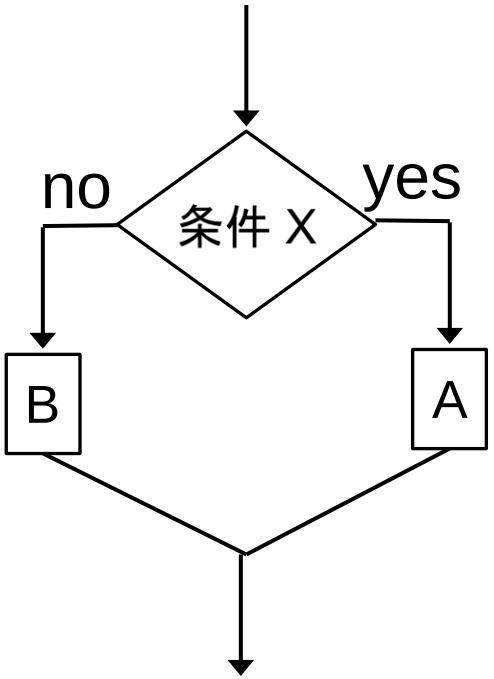
絶対値を求める計算

条件分岐文 if 文

↔ 条件に応じて分岐する（計算の流れを変える）命令

【文の構造】

```
if 条件 X
   命令列 A
else
   命令列 B
end
```



abs.rb

```
# abs.rb
x = 入力された数
if  x >= 0
    puts( x )
else
    puts( x*(-1) )
end
```

※ 究極的には if 文は不要（while 文で代用できるので）.

発展

4. Ruby での書き方（その4，最後 ;-)

Ruby での書き方のルール（例を見ながら）

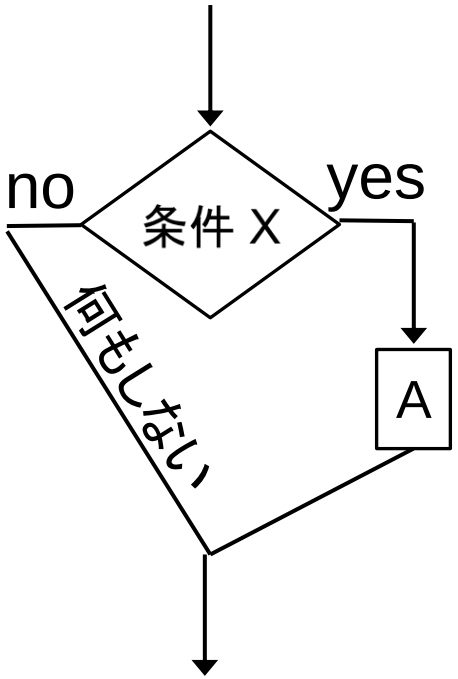
絶対値を求める計算

条件分岐文　if 文

↔ 条件に応じて分岐する
（計算の流れを変える）命令

【文の構造】（変形版）

```
if  条件 X
    命令列 A
end
```



abs.rb

```
# abs.rb
x = 入力された数
if  x < 0
    x = - x
end
puts( x )
```

4. Ruby での書き方　まとめ（1）

【演算子】

演算	使用例	意味
+	x + y	x と y の足し算
-	x - y	x から y の引き算
*	x * y	x と y の掛け算
/	x / y	x を y で割った商
%	x % y	x を y で割った余り
**	x ** y	x の y 乗

【関係演算子】

関係	使用例	意味
>=	x >= y	x は y より大きいかまたは等しい
>	x > y	x は y より大きい
==	x == y	x は y と等しい
!=	x != y	x は y は等しくない
<	x < y	x は y より小さい
<=	x <= y	x は y より小さいかまたは等しい

4. Ruby での書き方　まとめ（2）

【論理演算子】

論理記号	使用例	意味
&&	x && y	x と y の論理積 (両方が真のとき真)
||	x || y	x と y の論理和 (少なくとも一方が真のとき真)
!	!x	x の否定 (x が真のとき偽，x が偽のとき真)