

# CS第1 テーマ1

## テーマ1の目標

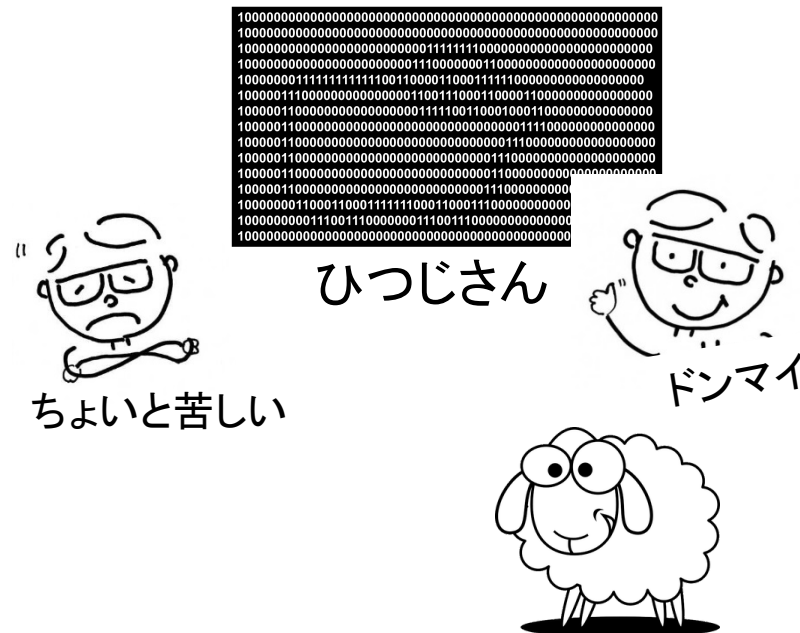
計算の基本要素を知る

## 本日の内容

1. データ = 数
2. コンピュータの中では

## 演習課題

四則演算でアニメーション



## クラスが 4b から 4a に変更になった学生へ

- 履修登録は, **4a** の方に行ってください
- 既に本登録を済ませた学生は, 教務課に行けば仮登録に戻せます. **今日(10/9)まで.**

## 1. はじめに

CSの**ところ**

# すべては計算である

- $(1 + 4) \times 5 =$
- 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
- 木の成長
- 脳の形成
- 銀行のATMの制御

ただし、コンピュータに  
載せるには ...



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

**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      21, -5, 3.25, 1/3
- 文字
- 画像
- 音
- 映像
- ....

10進数:  $21 = 2^4 + 2^2 + 1$

II

2進数: 10101

- 味, におい??

## 2. データは数である

**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  $21, -5, 3.25, 1/3$

- 文字  $a \leftarrow 01100001$  (=97),  $b \leftarrow 01100010$  (=98)

- ## ■ 画像

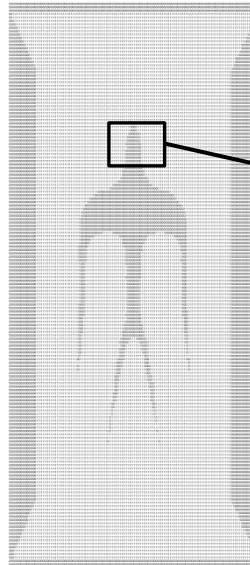
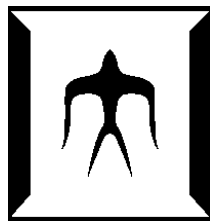
- 音

- ## ■ 映像

■ ■ ■ ■

- ・ 味, におい??

## ASCII という符号法



```

11111111111111111111
11111111111111111111
11111111111111111111
11111111111111111111
11111000111111111111
11111000011111111111
11111000011111111111
11110000000111111111
11100000000011111111

```

## 2. データは数である

**データ** = 計算の対象

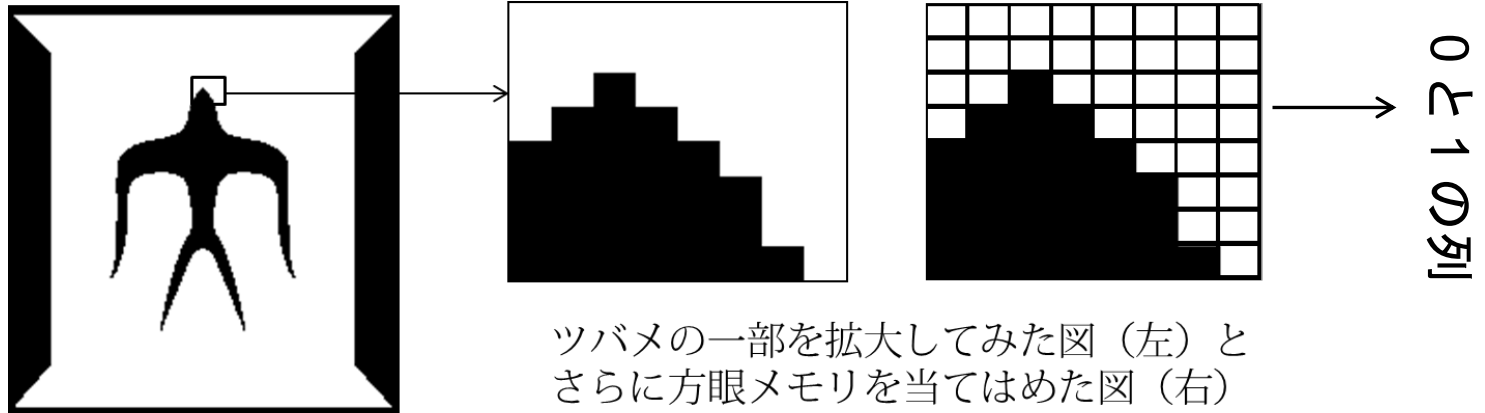
どこかで聞いたね

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



**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

**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

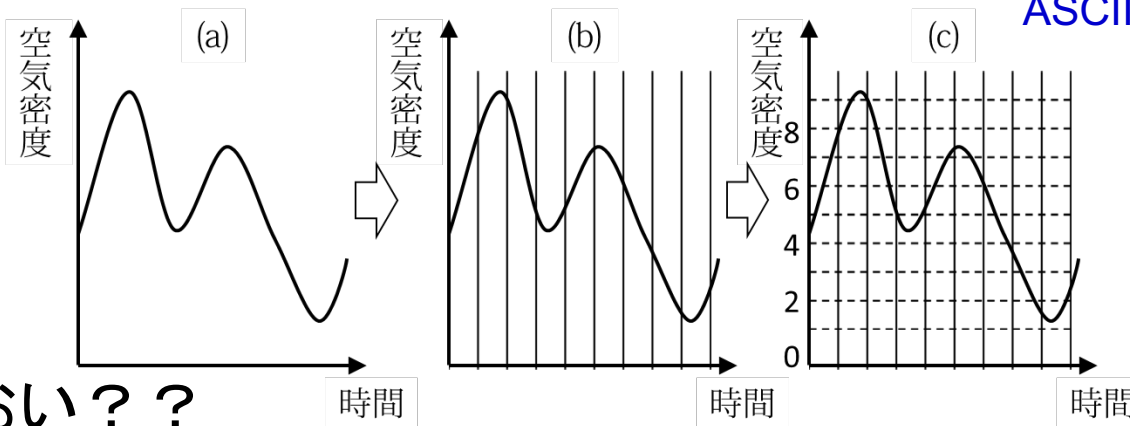
- 文字  $a \leftarrow 01100001$  (=97),  $b \leftarrow 01100010$  (=98)

- 画像       ASCII という符号法

- 音
- 映像

■ ■ ■ ■

- ・ 味, におい??



## 2. データは数である

**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~~

確かめてみよう（例で考える）

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

....

0 と 1 の列

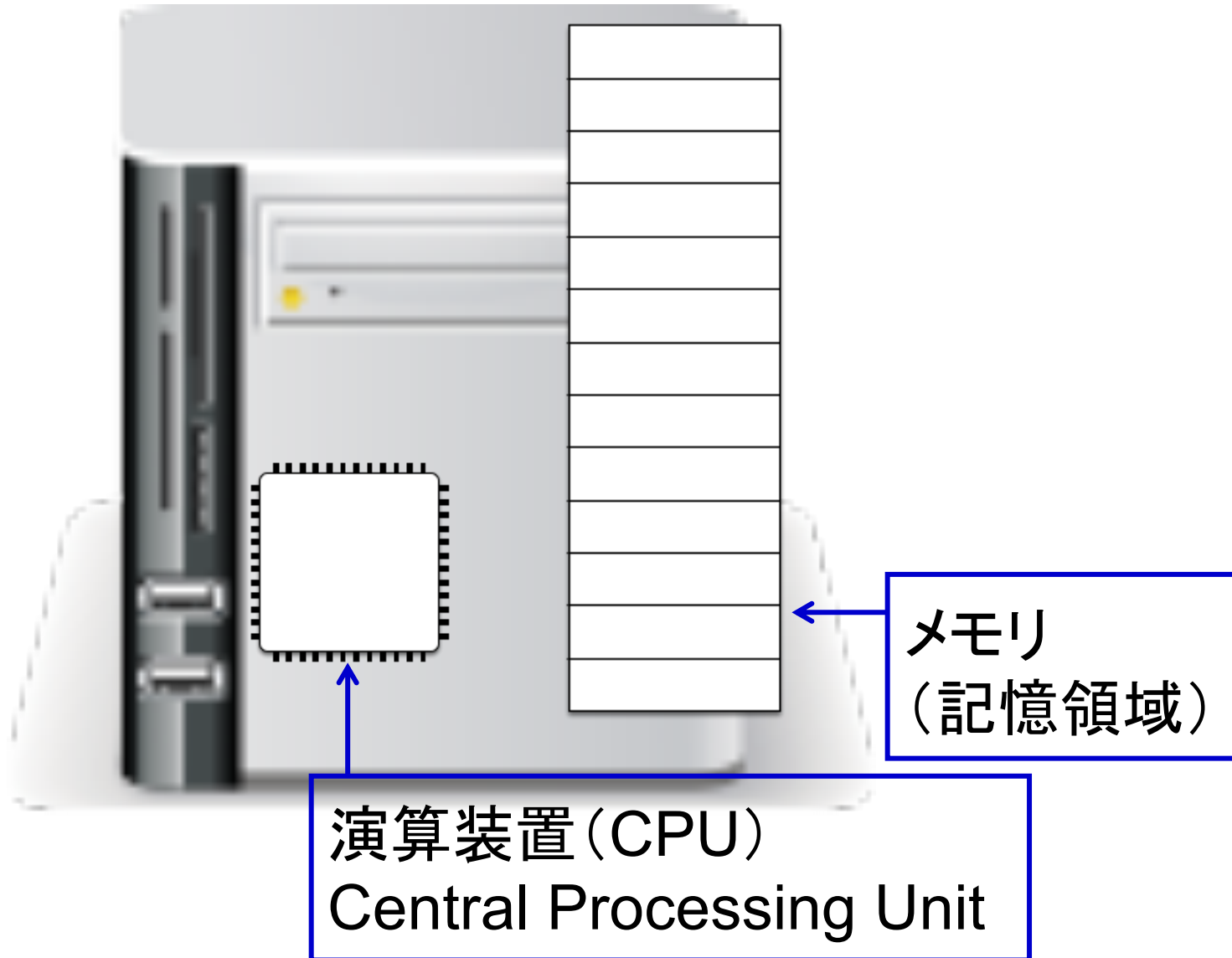
話を簡単にするため → 自然数

0, 1, 2, 3, ...



### 3. コンピュータの中では

コンピュータの基本 = 演算装置とメモリ



### 3. コンピュータの中では

メモリ = データ(数)を  
格納する箱



データ

番地

1234	[0]
78	[1]
0	[2]
123	[3]
9542	[4]
⋮	
0	[9998]
0	[9999]

メモリ

演算装置

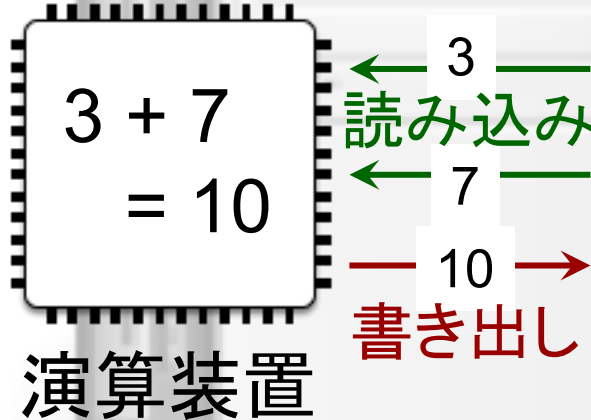
### 3. コンピュータの中では

演算装置 = 基本演算を行う装置

プログラムも  
メモリに格納  
される

プログラム

```
lw $x, M(100)
lw $y, M(101)
add $z, $x, $y
sw $z, M(102)
```



lw \$x, M(100) lw \$y, M(101) add \$z, \$x, \$y sw \$z, M(102)	
3	[20]
7	[21]
10	[22]
⋮	
0	[9998]
0	[9999]

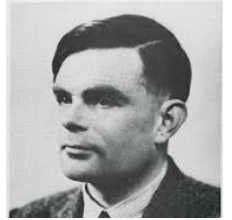
メモリ

誰が指示するの？

### 3. コンピュータの中では

演算装置 = 基本演算を行う装置  
とは？

計算の原子に  
相当する物は何？



チューリング

繰り返し

+1

-1

+

-

×

÷

√

sin

⋮

経路探索

囲碁, 将棋

四則演算

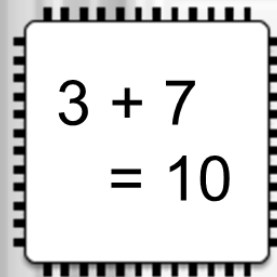
### 3. コンピュータの中では

入出力 = 外とのやり取り

プログラム

入力 ⇒

⇒ 出力



演算装置

コンピュータ内部  
でのやり取り

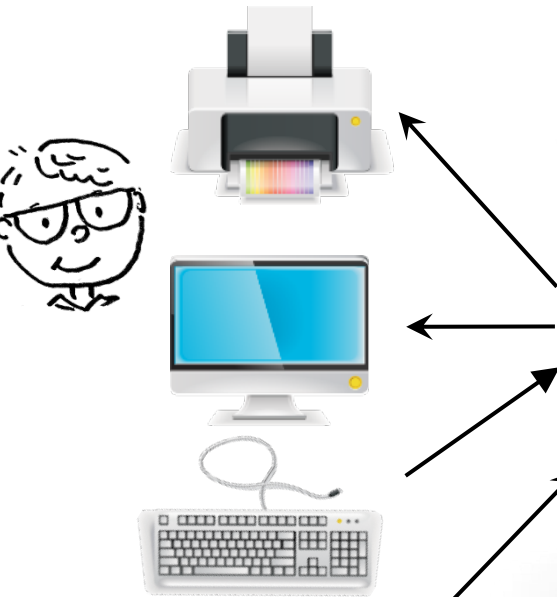
読み込み  
書き出し

lw \$x, M(100)	
lw \$y, M(101)	
add \$z, \$x, \$y	
sw \$z, M(102)	
3	[20]
7	[21]
10	[22]
⋮	
0	[9998]
0	[9999]

メモリ

### 3. コンピュータの中では

入出力 = 外とのやり取り



インターネット



3 + 7  
= 10

演算装置

コンピュータ内部  
でのやり取り

3  
読み込み  
7  
10  
書き出し

プログラム

```
lw $x, M(100)
lw $y, M(101)
add $z, $x, $y
sw $z, M(102)
```

3

[20]

7

[21]

10

[22]

⋮

0

[9998]

0

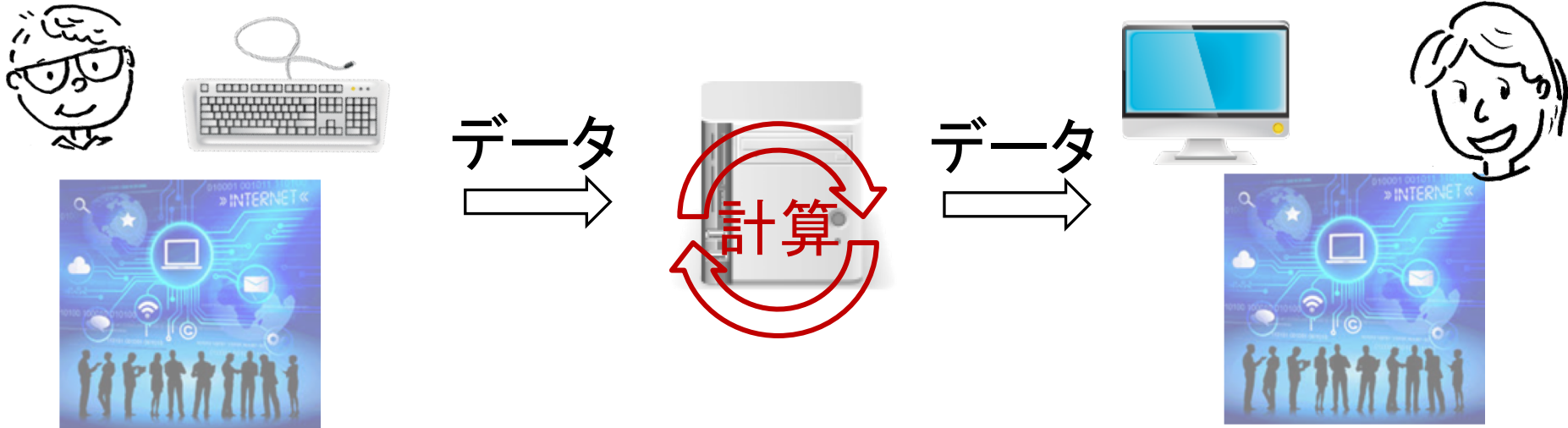
[9999]

メモリ

## ここまでのまとめ

**データ** = 自然数

**計算** = 超基本要素は  $\pm 1$  と繰り返し



プログラム ↔ 計算の設計図  
指示書, 命令書

# 基本演算

掛算を $\pm 1$  と繰り返しのみで実現する

add.py

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

mult.py

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

$\pm 1$  以外も使ってるよ



# 基本演算

掛算を±1 と繰り返しのみで実現する

mult.py

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

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

```
x = 入力データ
y = 入力データ
seki = 0
while y > 0:
    a = seki
    b = x
    wa = a
    while b > 0:
        wa = wa + 1
        b = b - 1
    seki = wa
    y = y - 1
print(seki)
```

# CS第1 演習ガイド

1. ログインする.
2. Terminal を動かす
3. 講義のウェブページから プログラム をダウンロードする.
  - Downloads(ダウンロード)フォルダに day2.zip が展開される
4. Terminal で day2 をCS1 に移動
  - `cd Documents/CS1`
  - `mv ~/Downloads/day2 ./`
  - `cd day2`

# プログラムを走らせてみる

1. **ls** その部屋にあるファイルを表示させる.
2. そこにあるプログラムをいくつか実行してみる.
  - (1) **python** mult.py   たとえば mult.py を実行してみる.  
200                   乗算したい2数を入力  
1200
  - (2) **cat** mult.py       mult.py はどんなプログラムか見る.
  - (3) **python** mult2.py   mult2.py を同様に実行してみる.

# プログラムの実行が止まらないことがある

```
a = int(input())
b = int(input())
wa = a
while b > 0:
    wa = wa + 1
    b = b - 1
print(wa)
```

```
a = int(input())
b = int(input())
wa = a
while b != 0:
    wa = wa + 1
    b = b - 1
print(wa)
```

```
$ python add.py
3
5
8
$ python add.py
3
0
3
$ python add.py
3
-1
3
```

止まる  
正しい加算の答えではない

```
$ python add-alt.py
3
5
8
$ python add-alt.py
3
0
3
$ python add-alt.py
3
-1
```

止まらない  
control + c で止める

## 練習: 引き算 $a-b$ を $\pm 1$ と繰り返しのみで計算

できれば  $a < b$  のときは 0 を出力するようにする.  
(できない人は  $a \geq b$  と仮定)

ファイル: sub.py

```
a = int(input())
b = int(input())
sa = a
while ???:
    sa = ...
    b = ...
print(sa)
```

???, ... に適切な式を書く

cotEditor で開くには

```
$ open -a coteditor sub.py
```

実行例

```
$ python sub.py
15
7
8
$ python sub.py
15
21
0
```

# 論理演算子: Python

論理記号	使用例	意味
and	x and y	x と y の論理積 (両方が真のとき真)
or	x or y	x と y の論理和 (少なくとも一方が真のとき真)
not	not x	x の否定 (x が真のとき偽, x が偽のとき真)

```
>>> 0 == 0 and 0 < 1
```

```
True
```

```
>>> 0 == 0 and 0 > 1
```

```
False
```

```
>>> 0 == 1 or 0 == 0
```

```
True
```

```
>>> not True
```

```
False
```

```
>>> not 0 == 0
```

```
False
```

## 参考: Terminal コマンド

命令	使用例	意味
mkdir	mkdir day2	day2 というフォルダ(部屋)を作る
cd	cd day2	day2 というお部屋に入る
	cd ..	上の(大きな)部屋に戻る
	cd ../..	上の上の部屋に戻る
ls	ls	その部屋にあるファイルを表示する
cat	cat foo.py	foo.py の中身を表示する
rm	rm foo.py	foo.py を消す(戻らないので注意)