

CS第1 テーマ1

テーマ1の目標

計算の基本要素を知る

演習課題

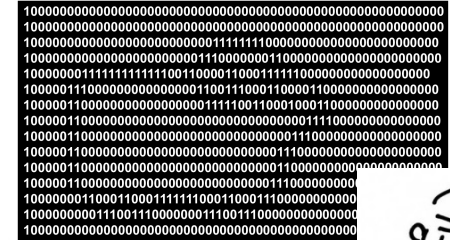
四則演算でアニメーション

本日の内容

1. Python を試してみる
 - ・ 四則演算, 変数
2. Python でプログラミング
 - ・ プログラムの基本
 - ・ 分岐, 繰り返し



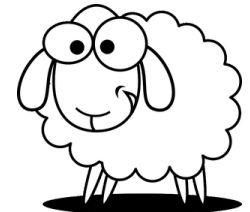
ちよいと苦しい



ひつじさん



ドンマイ



課題

Zip ファイル

データ圧縮およびアーカイブのためのファイルフォーマット

- 拡張子: .zip
- アーカイブ (複数のファイルを一つのファイルにまとめる)
- パスワードをつけることもできる

プログラミング演習のファイルを zip 形式で配布

演習ガイド: Mac

1. 講義のウェブページから day1.zip をダウンロードする.
 - Downloads(ダウンロード)フォルダに day1.zip が展開される
2. Terminal を実行
3. Terminal でフォルダ day1 をCS1 に移動
4. day1 フォルダに移動
5. 内容を確認

```
$ cd Documents/CS1
$ mv ~/Downloads/day1 ./
$ cd day1
$ ls
add.py      if2.py      med3.py      nested-if.py
div.py      max.py      mult.py      nested-if2.py
if1.py      max3.py     myname.txt   sum.py
```

演習ガイド: Windows

1. 講義のウェブページから day1.zip をダウンロードする.
 - Downloads(ダウンロード)フォルダに day1.zip を置かれる
 - ファイルを開く → すべてを展開
 - 展開先として, Documents¥CS1 を指定して, 展開
2. コマンドプロンプトを実行
3. day1 フォルダに移動
4. 内容を確認

```
C:¥Users¥minamide> cd Documents¥CS1
C:¥Users¥minamide¥Documents¥CS1> cd day1
C:¥Users¥minamide¥Documents¥CS1¥day1> dir

...
2020/10/08 09:14          224 add.py
...
2020/10/08 09:14          233 myname.txt
...
```

Terminal からテキストファイルの内容を確認

Windows の場合

```
> chcp 65001 (文字コードをutf-8に変更)
Active code page: 65001
> type myname.txt
# 下の名前を自分の名前に置き換えてください。
# 日本語の苗字と名前の間に全角(日本語の)空白を入力してください。

My name is Yasuhiko Minamide.
私の名前は南出靖彦です。
```

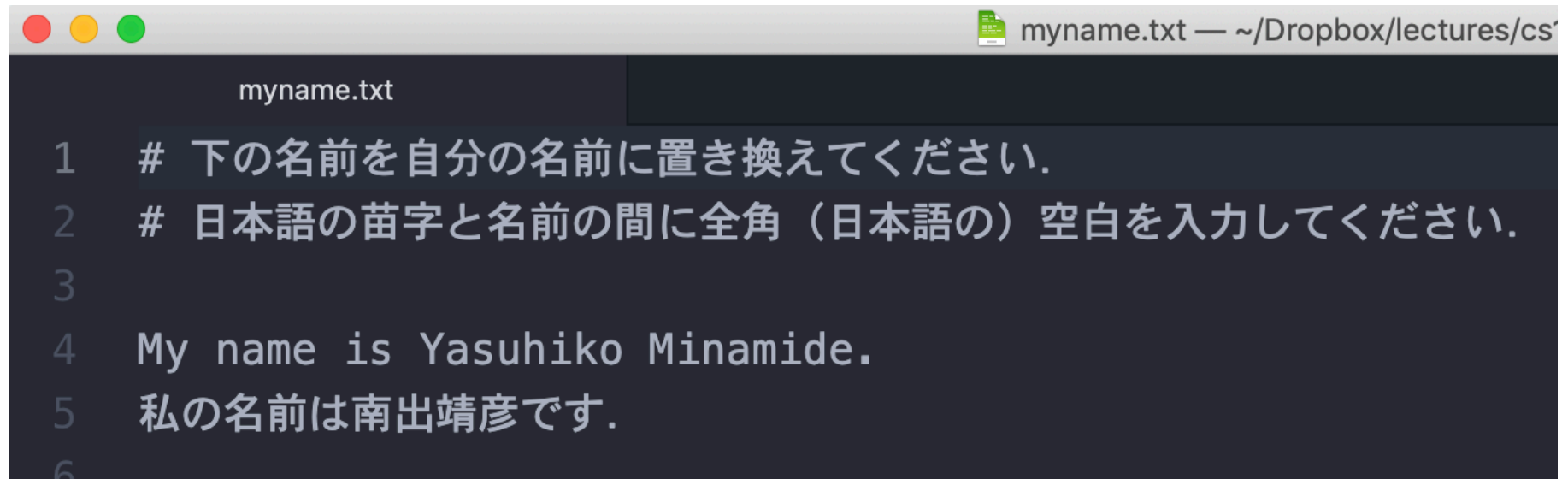
Mac の場合

```
$ cat myname.txt
# 下の名前を自分の名前に置き換えてください。
# 日本語の苗字と名前の間に全角(日本語の)空白を入力してください。

My name is Yasuhiko Minamide.
私の名前は南出靖彦です。
```

Atom を使ってみる

Atom で myname.txt を開いてください。
指示に従い編集して、保村して保存。



```
myname.txt
1  # 下の名前を自分の名前に置き換えてください。
2  # 日本語の苗字と名前の間に全角（日本語の）空白を入力してください。
3
4  My name is Yasuhiko Minamide.
5  私の名前は南出靖彦です。
6
```

⇒ Terminal から変更されているか確認してください

Python: バージョン2と3 (Mac の場合)

バージョン2: Macに標準でインストールされている

```
$ python
Python 2.7.10 (default, Feb 22 2019, 21:55:15)
[GCC 4.2.1 Compatible Apple LLVM 10.0.1 ...
Type "help", "copyright", "credits" or "license"
for more information.
>>>
```

バージョン3: 追加でインストールが必要 (授業はこっち)

```
$ python3
Python 3.8.2 (default, Aug 25 2020, 09:23:57)
[Clang 12.0.0 (clang-1200.0.32.2)] on darwin
Type "help", "copyright", "credits" or "license"
for ...
>>>
```

Pythonを使ってみる

対話的な環境で整数の四則演算をやってみる

加算, 減算, 乗算

```
$ python (Mac の場合は python3)
Python 3.7.3 (default, Mar 27 2019, 16:54:48)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: ...
Type "help", "copyright", ... for more information.
>>> 3+5
8
>>> 3*5
15
>>> 1 + 2 * 3
7
>>> (1 + 2) * 3
9
```

四則演算の結合の強さは普通の数学と同じ

Pythonを使ってみる: 続き

除算(割り算): 注意が必要

```
>>> 10/3  
3.3333333333333335
```

計算結果が実数(浮動小数点数): コンピュータサイエンス第一では使わない (バージョン2だと動作が異なる)

整数の除算(割り算): 式 **//** 式

```
>>> 8//3  
2
```

余りは切り捨て

Python(インタプリタ)の終わり方

```
$ python
```

```
Python 3.7.3 (default, Mar 27 2019, 16:54:48)
```

```
...
```

```
>>> 1 + 2
```

```
3
```

```
>>> exit()
```

```
$
```

以下でも終了できます

- Mac: control + d
- Windows: control + z

変数：計算結果に名前をつける

3 + 5 の計算結果を変数 x に代入する

- 3 + 5 の計算結果に x という名前を付ける

```
>>> x = 3 + 5
```

```
>>> x
```

```
8
```

```
>>> x * x
```

```
64
```

代入文 変数 = 式

- 変数名に使える文字
 - 小文字の英字, 大文字の英字
 - 数字, アンダースコア(_)
- 数字は最初の文字に使えない

変数：計算結果に名前をつける

代入文 変数 = 式

- 変数名に使える文字
 - 小文字の英字, 大文字の英字
 - 数字, アンダースコア()
- 数字は最初の文字に使えない

```
>>> wa_3_5 = 3 + 5
```

```
>>> wa_3_5 * wa_3_5
```

```
64
```

```
>>> 3_5_wa = 3 + 5
```

```
File "<stdin>", line 1
```

```
  3_5_wa = 3 + 5
    ^
```

```
SyntaxError: invalid token
```

Pythonを使ってみる: 真偽値とブール式

真偽値: True と False

```
>>> True
True
>>> False
False
```

比較: `==` (等しいか), `<`, `<=` (以下), `>`, `>=`, `!=` (等しくない)

```
>>> 10 == 3 + 7
True
>>> 10 == 11
False
>>> 10 < 11
True
>>> 10 < 10
False
>>> 10 <= 10
True
```

最初の Python プログラム: add.py

- 二つの整数をターミナルから入力する.
- 二つの整数の和を表示する.

```
# add.py
# 入力: 整数 a, b
# 出力: a + b

a = int(input())    # 入力された整数を a に代入
b = int(input())    # 入力された整数を b に代入
wa = a + b
print(wa)           # wa の値を出力
```

Atom から add.py を開いてみよう

最初の Python プログラム: add.py

- 二つの整数をターミナルから入力する.
- 二つの整数の和を表示する.

```
# add.py
# 入力: 整数 a, b
# 出力: a + b

a = int(input())    # 入力された整数を a に代入
b = int(input())    # 入力された整数を b に代入
wa = a + b
print(wa)           # wa の値を出力
```

- プログラムは, 最初の行から一行ずつ実行される
- **コメント**: # から 行末まで
プログラムの実行では無視される

プログラムを実行してみる

- 青の部分がタイプする部分

```
$ python add.py  
3  
8  
11  
$ python add.py  
11  
13  
24
```

- 負の整数も入力してみよう

条件分岐: if 文

if 条件式:

 ブロック1 # 条件が**真**の時に実行される

else:

 ブロック2 # 条件が**偽**の時に実行される

ブロックとは

- **インデント**(字下げ)が同じ1行以上の文
- 新しいブロックを始める時: 通常**4文字**インデント
- インデントは, **半角スペース**を使う
全角スペースを使うとエラーになる.
タブは非推奨

条件分岐: 最大値

ファイル max.py

```
# 入力: 整数 a, b
# 出力: a と b の最大値

a = int(input())    # 入力された整数を a に代入
b = int(input())    # 入力された整数を b に代入
if a > b:
    max = a
else:
    max = b
print(max)
```

if1.py

```
a = int(input())
b = int(input())
if a == b:
    print("A")
else:
    print("B")
    print("C")
```

} ここがブロック

if2.py

```
a = int(input())
b = int(input())
if a == b:
    print("A")
else:
    print("B")
print("C")
```

```
$ python if1.py
0
0
A
$ python if1.py
0
1
B
C
```

```
$ python if2.py
0
0
A
C
$ python if2.py
0
1
B
C
```

条件文の入れ子(ネスト): nested-if.py

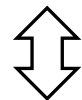
```
a = int(input())    # 入力された整数を a に代入
if a < 0:
    print("N")      # Negative
else:
    if a == 0:
        print("Z")  # Zero
    else:
        print("P")  # Positive
```

実行例

```
$ python nested-if.py
-1
N
$ python nested-if.py
0
Z
$ python nested-if.py
1
P
```

else if の構文

```
a = int(input())    # 入力された整数を a に代入
if a < 0:
    print("N")      # Negative
else:
    if a == 0:
        print("Z")  # Zero
    else:
        print("P")  # Positive
```



等価：同じ計算を行う

```
a = int(input())    # 入力された整数を a に代入
if a < 0:
    print("N")      # Negative
elif a == 0:
    print("Z")      # Zero
else:
    print("P")      # Positive
```

if 文: else がない場合

if 条件式:

ブロック1

条件が**真**の時に実行される

max3.py: 3つの整数の最大値を出力

```
a = int(input())    # 入力された整数を a に代入
b = int(input())    # 入力された整数を b に代入
c = int(input())    # 入力された整数を c に代入
max = a             # a の値を最大値の候補とする
if b > max:
    max = b
if c > max:
    max = c
print(max)          # max の値を出力
```

練習: 3つの数の中央値(medium) med3.py

??? に適切な条件式を書け

```
if a < b:
    min = a          # 最小値の候補
    max = b          # 最大値の候補
else:
    min = b
    max = a
# この時点で, min <= max
if ???:
    print(min)
elif ??? :
    print(c)
else:
    print(max)
```

実行例

```
$ python med3.py
1
2
3
2
$ python med3.py
2
1
3
2
$ python med3.py
7
9
9
9
```

少し本格的なプログラムを書く: while ループ

- 非負整数 n をターミナルから入力する.
- $1 + 2 + \dots + n$ を表示する.

ファイル: sum.py

```
n = int(input())    # 入力された整数を n に代入
i = 1
sum = 0
while i <= n:
    sum = sum + i
    i = i + 1
print(sum)
```


while 文の実行

while 条件式:
ブロック

1. 条件式を評価(計算する)
 - True の時:
 - ① ブロック部分を実行する
 - ② 1に戻る(while文の先頭に戻る)
 - Falseの時:
while文の実行を終わる


n = 3 の場合

```
i = 1
sum = 0
while i <= n:
    sum = sum + i
    i = i + 1
print(sum)
```

	変数 i	変数 sum	
i = 1	1	?	
sum = 0	1	0	
while i <= n	1	0	True
sum = sum + i	1	1	
i = i + 1	2	1	
while i <= n	2	1	True
sum = sum + i	2	3	
i = i + 1	3	3	
while i <= n	3	3	True
sum = sum + i	3	6	
i = i + 1	4	6	
while i <= n	4	6	False
print(sum)	4	6	6 = 1 + 2 + 3

プログラム例：乗算を加算と減算で実現

- 非負整数 x と y をターミナルから入力する.
- x を y 回加算: $x + x + \dots + x$



ファイル: mult.py

```
x = int(input())    # 入力された整数を x に代入
y = int(input())    # 入力された整数を y に代入
seki = 0
while y > 0:        # y が 0 より大きい間は ブロックの実行を繰り返す
    seki = seki + x
    y = y - 1
print(seki)
```

課題1 : Python でプログラミング

割り算を加減算と繰り返しのみで計算する

```
# div.py
# 入力: 自然数 x, y
# 出力:  $x \div y$  の商と余り

x = int(input())    # 入力された自然数を x に代入
y = int(input())    # 入力された自然数を y に代入
shou = ...
amari = ...
while ???:
    shou = ...
    amari = amari - y
print(shou)         # shou の値を出力
print(amari)        # amari の値を出力
```

提出方法: OCW-I, 締め切り: 10/19(月) 22:00

- div.py を提出