コンピュータサイエンス第２

南出 靖彦

第１回

講義スケジュールと評価

スケジュール

- 12 月 9 日
- 12 月 16 日
- 12 月 23 日
- 1 月 6 日
- 1 月 13
- 1 月 20 日
- 1 月 27 日

評価

- 課題：3〜4 回, 30〜50 %
- レポート課題：2〜3 回, 50〜70 %

今日の内容

- 関数 (サブルーチン)
- 再帰関数

準備: Windows

- コマンドプロンプトを実行，Documents フォルダに CS2 フォルダを作る
  - `cd Documents`
  - `mkdir CS2`
  - `cd CS2`
- 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¥CS2 を指定して，展開
- day1 フォルダに移動
  - `cd day1`

準備: Mac

- Documents フォルダに CS2 フォルダを作る
  - `cd Documents`
  - `mkdir CS2`
  - `cd CS2`
- 講義のウェブページから `day1.zip` をダウンロードする.
- Terminal で day1 を CS2 に移動
  - `mv ~/Downloads/day1 ./`
  - `cd day1`

Python による文字列等の出力

print 関数

```
$ python                          (Mac の場合は python3)
...
>>> print("abc", 10)
abc 10
```

改行したくない時

```
>>> print("abc", 10, end="")
bc 10>>>
>>> exit()
```

復習：関数

```
# 関数 kaijou(n) (n の階乗)

def kaijou(n):
    ans = 1
    for i in range(1,n+1):
        ans = ans * i
    return(ans)

# kaijou の定義ここまで

# ここからプログラム本体
x = int(input())
print(kaijou(x))
```

値を返さない関数 (サブルーチン)

```
# 関数 writeX(a)
# 働き：文字 X を a 個横に並べて書いて改行する

def writeX(a):
    for i in range(a):
        print("X", end="")
    print()

# writeX の定義ここまで

# ここからプログラム本体
n = int(input())
writeX(n)
```

- return を持たない

writeX(式) という文を実行すると，

1. 式 の値が計算される．
2. その値が関数 writeX の変数 a に代入される．
3. writeX の定義部分が実行開始される．
   （関数が呼び出される，と言う）
4. 実行が終了すると本体に戻ってくる

用語：式 や a のことを引数と呼ぶ

再帰呼び出し

関数の定義の中で，今定義している自分自身を呼び出す．

```
# 階乗の再帰的な定義
def kaijou(n):
    if n <= 1:
        return 1
    else:
        return n*kaijou(n-1)

# ここからプログラム本体
x = int(input())
print(kaijou(x))
```

kaijou(3) を実行すると

```
   kaijou(3)
=> 3 * kaijou(2)
=> 3 * 2 * kaijou(1)
=> 3 * 2 * 1
   ...
=> 6
```

練習: フィボナッチ数

F_n: n 番目のフィボナッチ数

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_{n+2} &= F_n + F_{n+1} \quad (n \geq 0) \end{aligned}$$

F_0	F_1	F_2	F_3	F_4	F_5	$\cdots$	F_{10}	$\cdots$
0	1	1	2	3	5	$\cdots$	55	$\cdots$

練習: ファイル `fib.py` の中の関数 `fib` を完成せよ.

```
$ python fib.py
5
5
$ python fib.py
10
55
```

練習: 指数関数

次の考え方で，n^k を計算する関数 exp(n,k) を書け

$$\begin{aligned} n^{2k} &= (n^2)^k \\ n^{2k+1} &= n \times (n^2)^k \end{aligned}$$

練習: ファイル exp.py の中の関数 exp を完成せよ.

```
$ python exp.py
2
10
1024
1024                    (Python の組み込みの指数関数での計算結果)
$ python exp.py
3
100
515377520732011331036461129765621272702107522001
515377520732011331036461129765621272702107522001
```

再帰関数はどう実現されているか

メモリの一部をスタック として使う.

スタック：データを積み上げたもの．伸びたり縮んだりする.

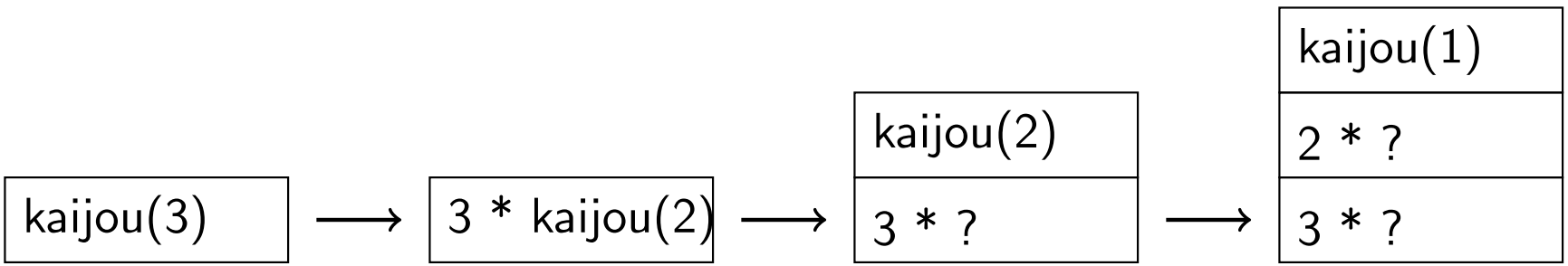
- 伸びたり縮んだりできる配列だと思っても良い

階乗の再帰的な定義

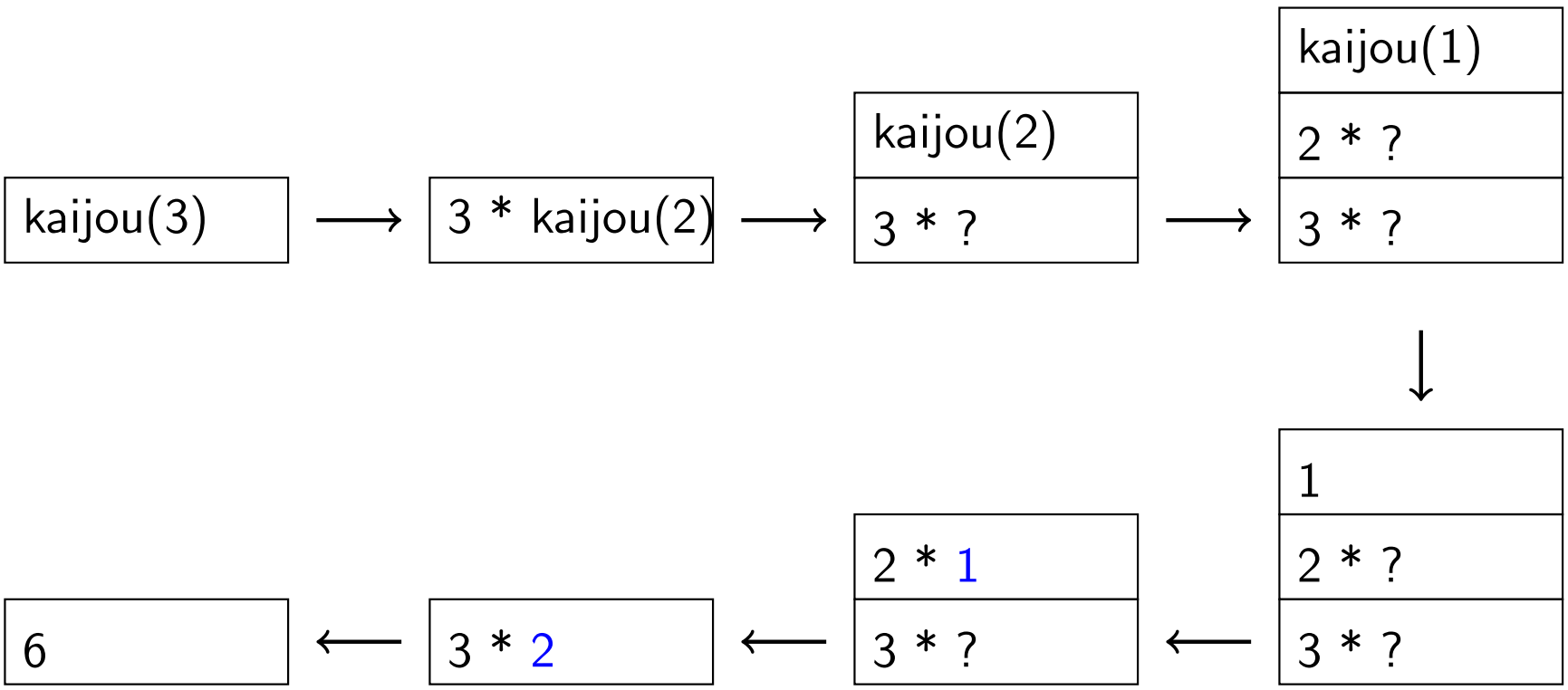
```
def kaijou(n):
    if n <= 1:
        return 1
    else:
        return n*kaijou(n-1)
```

スタックを用いた計算

- `3 * ?`: 関数呼び出しの戻り値が `n` の時, `3 * n` を計算



```
def kaijou(n):
    if n <= 1:
        return 1
    else:
        return n*kaijou(n-1)
```



スタックオバーフロー

多くのプログラミング言語では，再帰呼び出しが深くなりすぎるとエラー
⇒ スタックが大きくなりすぎる
⇒ スタックオバーフロー

```
$ python kaijou2.py
100
933262154439441526816992388562667004907159682643 8...
$ python kaijou2.py
999
Traceback (most recent call last):
...
RecursionError: maximum recursion depth exceeded in comparison
```

- python の場合，深さ 1000 ぐらいが限界

再帰呼び出し figure1.py

```
# 関数 figure(n)
# 働き：大きさ n の三角形を文字 X で描く

def figure(n):
    if n <= 1:
        writeX(n)
    else:
        figure(n-1)
        writeX(n)
```

figure(5) を呼び出せば次が画面に出力される。

```
$ python figure1.py
5
X
XX
XXX
XXXX
XXXXX
```

等価なプログラム figure2.py

```
# 関数 figure(n)
# 働き：大きさ n の三角形を文字 X で描く

def figure(n):
    if n >= 2:
        figure(n-1)
    writeX(n)
```

等価なプログラム：なぜ？

```
def figure(n):
    if n >= 2:
        figure(n-1)
    writeX(n)
```

条件の真と偽を入れ替える

```
def figure(n):
    if n <= 1:
       # 何もしない.
    else:
        figure(n-1)
    writeX(n)
```

writeX(n) を if 文の中に入れる

```
def figure(n):
    if n <= 1:
        writeX(n)
    else:
        figure(n-1)
        writeX(n)
```

【練習問題】figure3.py

次のプログラムを実行するとどのような出力がされるか？

```
def test(n):
    if n == 1:
        writeX(1)
    elif n == 2:
        writeX(2)
    else:
        test(n - 1)
        test(n - 2)
        writeX(n)

test(5)
```

【課題 1 】: 再帰プログラム

課題用プログラム (kadai1.py) を, 以下の問題 (A)(B) の指示に従って改造せよ。そして改造したプログラムを OCWi の課題提出機能で提出せよ。

- 締め切り： 12 月 14 日 午前 22:00
- 提出するファイル: kadai1.py
  - 問題 A, 問題 B の両方の変更を行ったもの.

(問題 A)

関数 figure の定義の中身だけを変更して，次のような出力をするプログラムにせよ。figure の定義以外を変更してはいけない。

１を入力した場合

```
X
```

２を入力した場合

```
X
XX
X
```

３を入力した場合

```
X
XX
X
XXX
X
XX
X
```

４を入力した場合

```
X
XX
X
XXX
X
XX
X
XXXX
X
XX
X
XXX
X
XX
X
```

５以上の入力も同様（類推せよ）

(問題 B)

関数 `writeX` を，働きは変えずに再帰呼び出しを使った定義に書き換えよ
(ただし `writeX` に 0 や負数を与えたときの働きは変わってもよい)。
while や for などの繰り返し構文を使用してはいけない。