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 4 日
- 12 月 11 日
- 12 月 18 日
- 12 月 25 日
- 1 月 8 日
- 1 月 15 日
- 1 月 22 日
- 1 月 29 日： 期末試験

評価

- 課題：3〜4 回, 40〜50 %
- 期末試験：50〜60 %

今日の内容

- 関数 (サブルーチン)
- ローカル変数
- グローバル変数
- 再帰呼び出し

準備

- 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

```
>>> print("abc", 10)
abc 10
```

改行したくない時

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

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

ローカル変数とグローバル変数

- ローカル変数：関数の仮引数，関数内で定義される変数
  - 関数（又はプログラム本体）の中でのみ有効
  - プログラム本体や他の関数内に同じ名前の変数があっても実体は別

- グローバル変数：関数の外で定義される変数
  - プログラム本体からも各関数内部からも触ることができる
  - プログラム実行中ずっと存在し続ける
  - 関数内でグローバル変数に代入する場合は，global 宣言を用いる

ローカル変数

- プログラム本体や他の関数内に同じ名前の変数があっても実体は別なので混同は起こらない。
- 同じ関数が何度も呼び出される場合にもその関数定義内のローカル変数は毎回名前は同じだが実体は別になる。

```
def test(a):
    a = 100
    b = 200
    print(a, b)

a = 1
b = 2
test(a)
print(a, b)
```

実行結果

```
100 200
1 2
```

- 関数 test 中の変数 a,b とプログラム本体中の変数 a,b は，別の変数として扱われる

グローバル変数

グローバル変数：関数外の変数，global 宣言された変数

- プログラム本体からも各関数内部からも触ることができる
- プログラム実行中ずっと存在し続ける

```
def test(a):
    global b
    a = 100
    b = 200
    print(a, b)
```

実行結果

```
100 200
1 200
```

```
a = 1
b = 2
test(a)
print(a, b)
```

- 関数 test の中のグローバル変数 b への代入が，プログラム本体へも影響する.

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.

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.

グローバル変数の応用

関数が何回呼び出されるか？

```
def gcd(m,n):
    global calls
    calls = calls + 1
    if n == 0:
        return m
    else:
        return gcd(n, m % n)

m = int(input())
n = int(input())
calls = 0
print(gcd(m,n))
print(calls)
```

gcd(100,60) を実行すると

```
   gcd(100,60)
=> gcd(60,40)
=> gcd(40,20)
=> gcd(20,0)
=> 20
```

再帰呼び出し 6.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。

```
X
XX
XXX
XXXX
XXXXX
```

等価なプログラム 7.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】8.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 (kadai.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 11 日 午前 10:40
  - 12 月 11 日　午後 9:00 までの提出は採点しますが，大きく減点します.
- 提出するファイル: kadai.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。