関数をつくってみる

- 複雑なプログラムを作る場合、
「ヤヤコシイ」部分を関数として切り分けると、
意外と簡単になることも

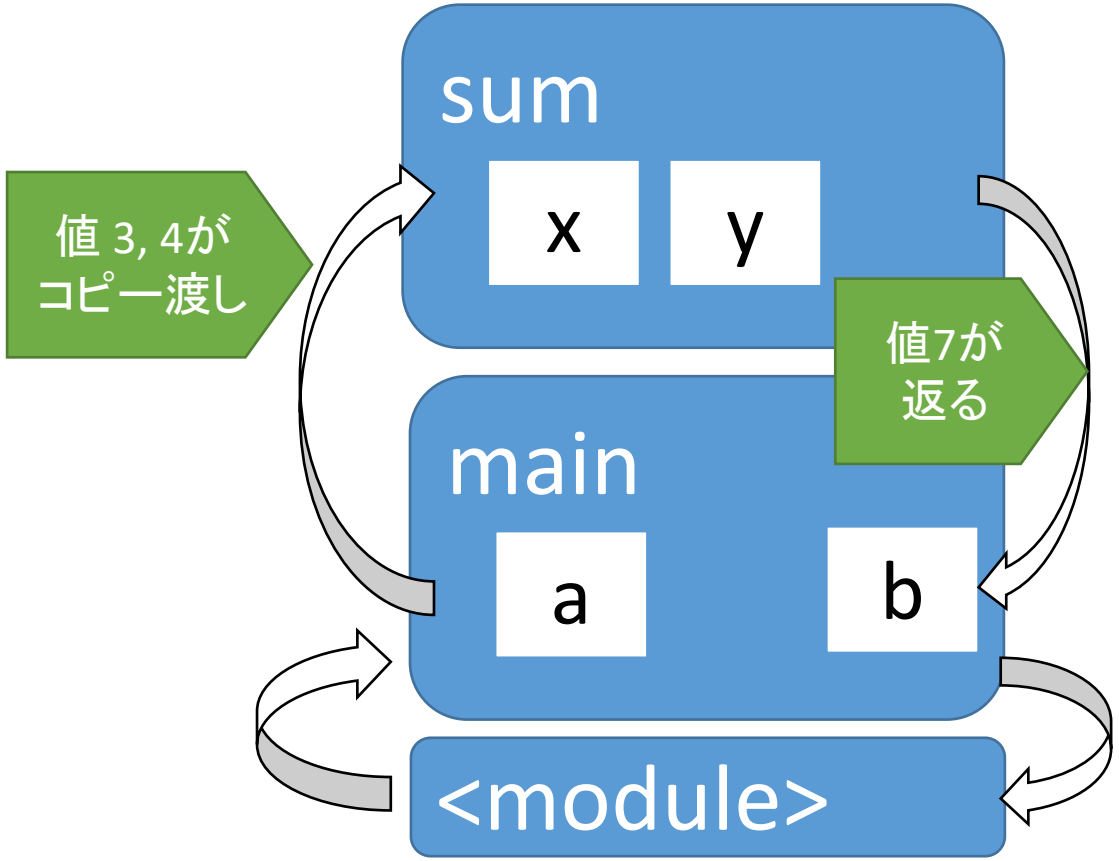
練習

- まずは、関数の実行イメージを身につける
- で、関数つかって面倒そうな問題を解いてみる
- 少しヤヤコシイ「引数の渡し方」の解説

関数呼び出し

引数の扱いについては、
後で詳しく

- main 関数が sum 関数を呼び出す例（ x + y の結果を返す）




```
def sum_inc(x, y):
    result = x + y
    x += 1
    print("in sum_inc(), x=" + str(x))
    return result


def main():
    a = 3
    b = sum_inc(a, 4)
    print("in main(): a=" + str(a)
                  + ", b=" + str(b))


main()
```

実行結果
in sum_inc(), x=4
in main(), a=3, b=7

ちょっとヤヤコシイ問題

2013年パソコン甲子園予選3
（本家 pdf）（レギオ用AtCoder）

- 野菜の苗に、雑草が混ざってしまった。
  - 野菜の長さは等差数列、雑草は違う
- 入力例
  - 1行目：野菜の本数、2行目：野菜＋雑草の長さ

```
5
1 2 3 6 4 5
```

```
4
5 7 9 11 12
```

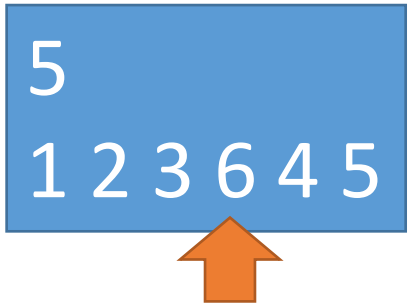
- 出力：雑草の長さ

解けそうな気はする？

- とりあえず、入力を取り込むのは簡単
  - List に
    いれておこう！

```
n = int(input())
line = input()
lst = list(map(int, line.split()))
```

- さて、どうやって解く？
  - 意外に悩ましい。。。。
- 単純な案
  - i 番目の要素を無視して、
    等差数列になっているかチェックしよう！



プログラムの概略は？

- 雑草 (skip) を仮決めしては、
  等差数列か
  確認すればいいよね。

```
for skip in range(n + 1):
    skip が雑草かどうか、等差数列チェック
    雑草だったら、 lst[skip] を出力して終了
```

- とはいえ、雑草を考慮した等差数列チェックって？
  - ごちゃごちゃして分からん。。。

そういうときは、仕事を切り分けると、
頭の整理がつくかも。

等差数列チェックは作れそう？

- 与えられた vector が、等差数列かチェックする
  - `d = lst[1] - lst[0]`が
  - `di = lst[i] - lst[i - 1]` と
    `i: 2..n` で一致すればOK

- ただ、skip の扱いは面倒そう。。。
  - 案1: skip を除いた List をその都度作る
  - 案2:「skip を考慮した i番目」にアクセスする
    - 今回はこちらでいってみましょう

ややこしいものは関数に切り分け

- 関数の作ることにしよう！
- `def check(skip, n, vec)`
  - skip 番目が雑草として、vec が等差数列か判定。

- 呼び出し側はスッキリ
  - あとは、関数つくるだけ

```
for skip in range(n + 1):
    if check(skip, n, lst):
        print(lst[skip])
        return
```

check 関数作るぞ

- 与えられた vector が、等差数列かチェックする （再掲）
  - `d = lst[1] - lst[0]`が
  - `di = lst[i] - lst[i - 1]` と
    `i: 2..n` で一致すればOK

- skip 考慮
  した i 番目も
  ヤヤコシイ

```
def check(skip, n, lst):
    d = lst[1番目] - lst[0番目]
    for i in range(2, n):
        di = lst[i番目] - lst[i-1番目]
        if di != d:
            return False
    return True
```

再度、ヤヤコシイ物は切り分け

- skip 考慮した i 番目って、何番目？
  - def nth(skip, i)
- 呼ぶ側は、簡単に。

```
def check(skip, n, lst):
    d = lst[nth(skip, 1)] - lst[nth(skip, 0)]
    for i in range(2, n):
        di = lst[nth(skip, i)] - lst[nth(skip, i - 1)]
        if di != d:
            return False
    return True
```

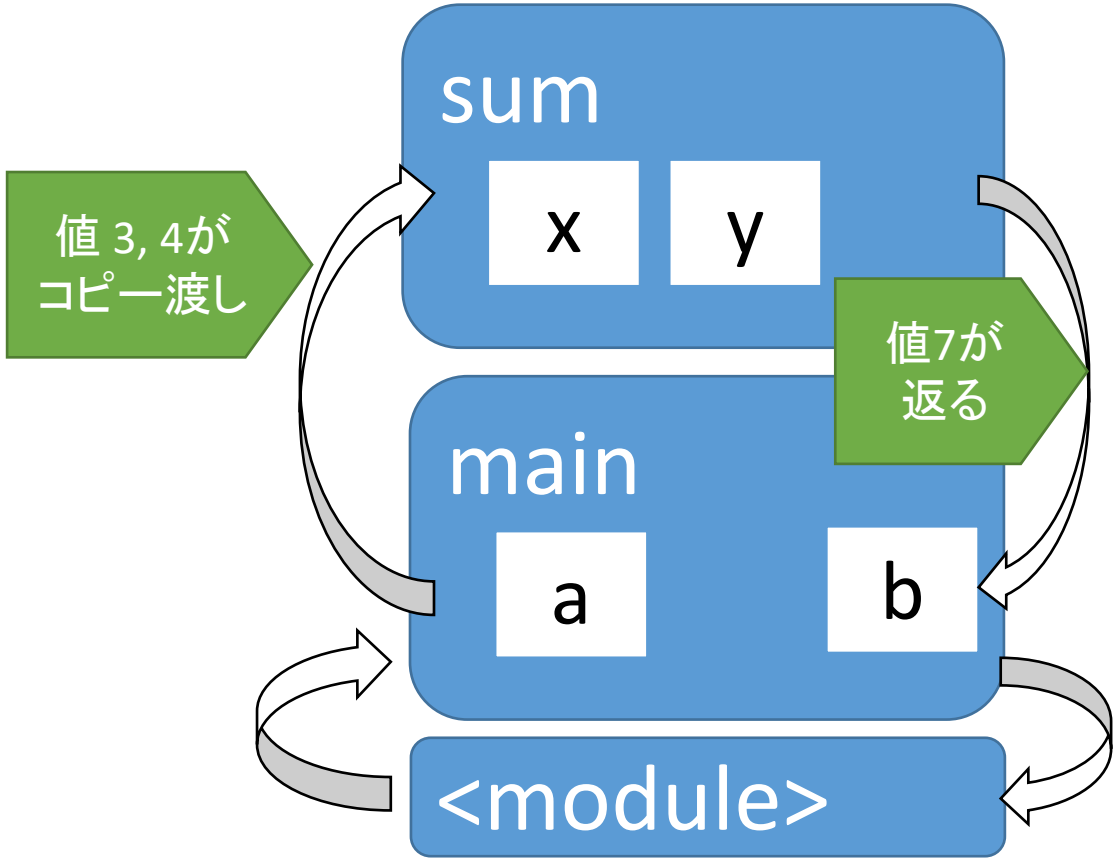
再度、ヤヤコシイ物は切り分け（続）

- skip 考慮した i 番目って、何番目？
  - `def nth(skip, i)`
- この関数はそれほど難しくない
  - i < skip なら、素直に i 番目
  - i>= skip なら、i+1番目

```
def nth(skip, i):
    if i < skip:
        return i
    else:
        return i + 1
```

関数呼び出し

- main 関数が sum 関数を呼び出す例（ x + y の結果を返す）



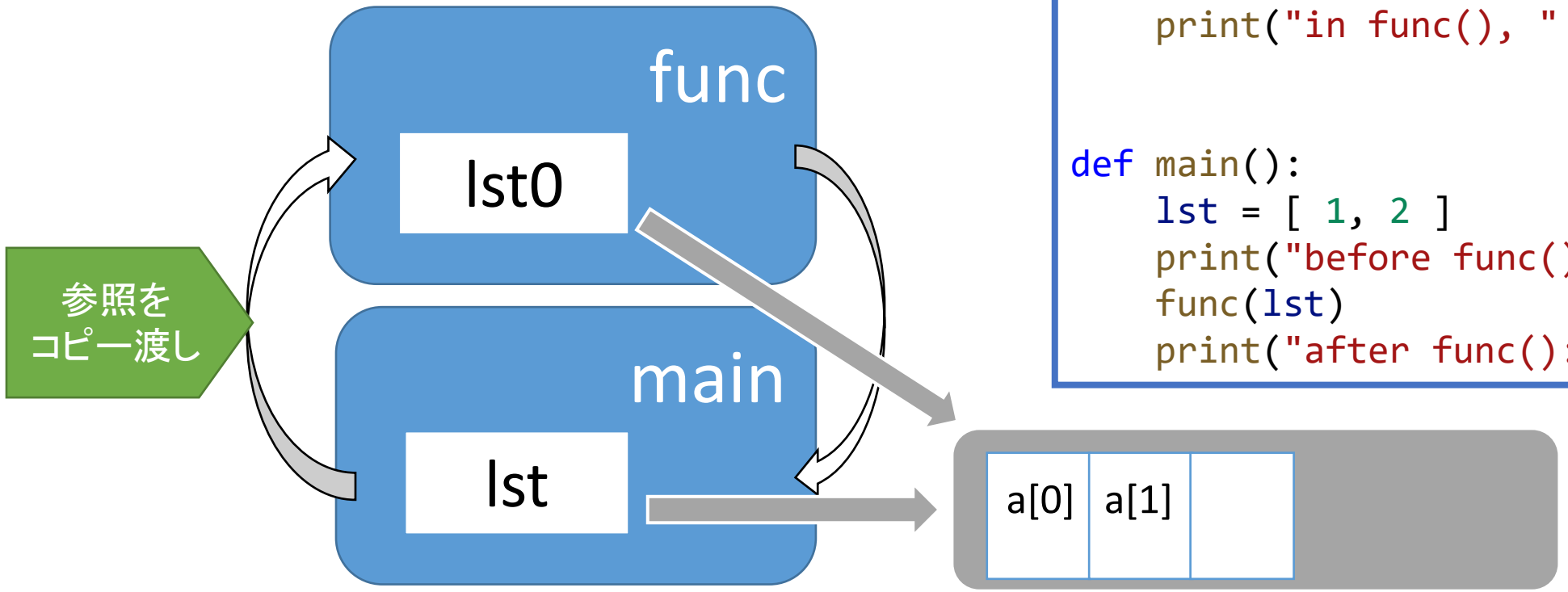

```
def sum_inc(x, y):
    result = x + y
    x += 1
    print("in sum_inc(), x=" + str(x))
    return result


def main():
    a = 3
    b = sum_inc(a, 4)
    print("in main(): a=" + str(a)
                  + ", b=" + str(b))


main()
```

実行結果
in sum_inc(), x=4
in main(), a=3, b=7

関数呼び出し(List の場合)

デバッガで確認

- func(lst) で List に要素追加




```
def func(lst0):
    lst0.append(3)
    print("in func(), " + str(lst0))


def main():
    lst = [ 1, 2 ]
    print("before func():" + str(lst))
    func(lst)
    print("after func():" + str(lst))
```

実行結果
before func():[1, 2]
in func(), [1, 2, 3]
after func():[1, 2, 3]