レギオ講習会　神戸＠甲南大

- 初日：プログラム実行を理解する
  - 変数やデータ構造のイメージ
  - 関数の利用イメージ
    - デバッガで動作確認
  - 計算量と実行時間
- ２日目：アルゴリズムの理解と応用
  - 計算量と実行時間
  - 動的計画法
  - ライブラリの利用
  - 探索問題

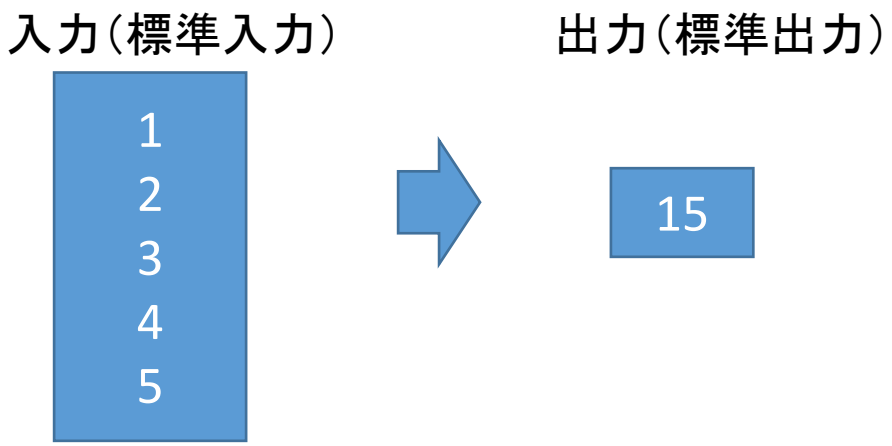
自分の書いたプログラムの
実行イメージを
しっかりつかむ

動作を確認する

実行時間のイメージを持つ

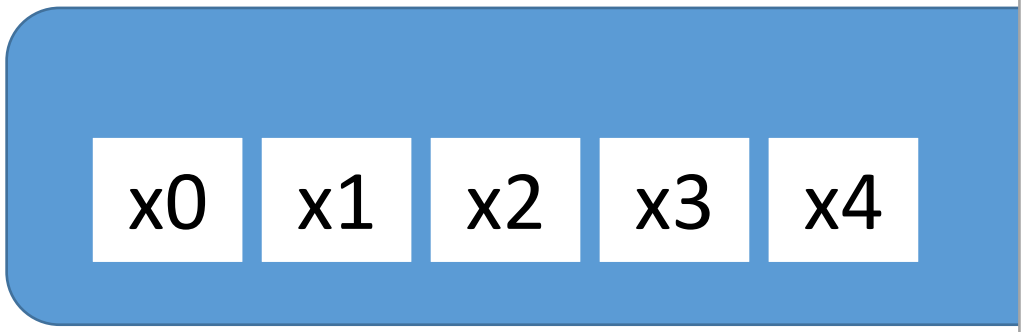
初歩の初歩

- 5個の整数が与えれられたとき、その和を計算して出力するプログラムを作れ



初歩の初歩

- 「繰り返し」なしで解いてみる
- 変数 x0, x1, …, x4



input(): 1行読み込み
int(str): 文字str をint に

pro1a.py

```
x0 = int(input())
x1 = int(input())
x2 = int(input())
x3 = int(input())
x4 = int(input())

print(x0+x1+x2+x3+x4)
```

プログラムは、以下に配置：
https://github.com/tomiokamada/regio/tree/master/samples

Python (.py), C (.c), C++ (.cpp)
を並べておいてあるので、
比較したい人はどうぞ。

デモンストレーション

- デバッガで、プログラム動作の様子を眺めてみましょう。
  - プログラムの動作イメージが視覚的に描けることは重要
  - 思った通りにプログラムが動作しているか、確かめることもできる

初歩の初歩

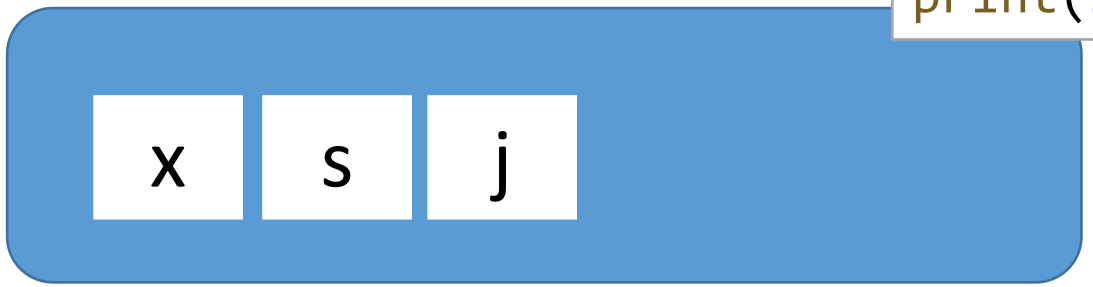
- 「繰り返し」なしで解いてみる
- 変数
  - x 入力用
  - s 計算結果用


```
s = 0
x = int(input())
s += x
x = int(input())
s += x
x = int(input())
s += x
x = int(input())
s += x
x = int(input())
s += x

print(s)
```

pro1b.py

初歩の初歩

- 「繰り返し」を使ってみる
- 変数
  - x 入力用
  - s 計算結果用
  - j 回数カウント用



j のように値を使わない変数は
_ と書けば省略可能

range(5) で
0..4 の繰り返し相当

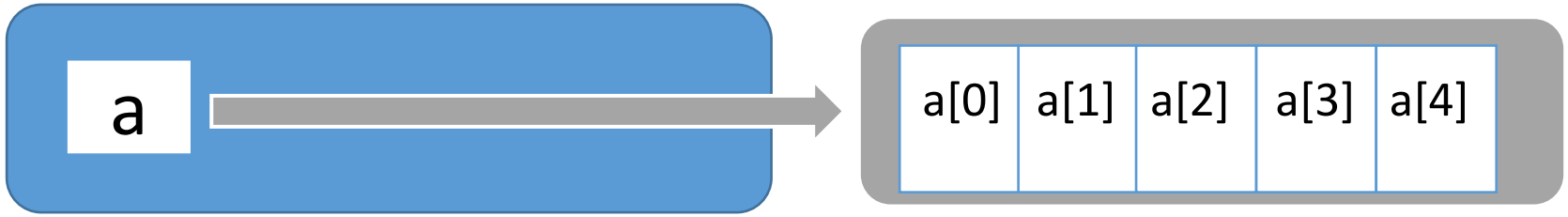
```
s = 0
for j in range(5):
  x = int(input())
  s += x

print(s)
```

pro1c.py

List の紹介

- この問題で list を使う必要はない
- でも、データ入手を先に済ませたほうが、わかりやすいことも多い

■ List: a = [1, 2, 3, 4, 5]



変数とは別に、
値格納用の領域（オブジェクト）
ができて、それを指す（参照）感じ

```
a = []
for i in range(5):
  a.append(int(input()))

s = 0
for i in range(5):
  s += a[i]

print(s)
```

pro1e.py

append で要素追加

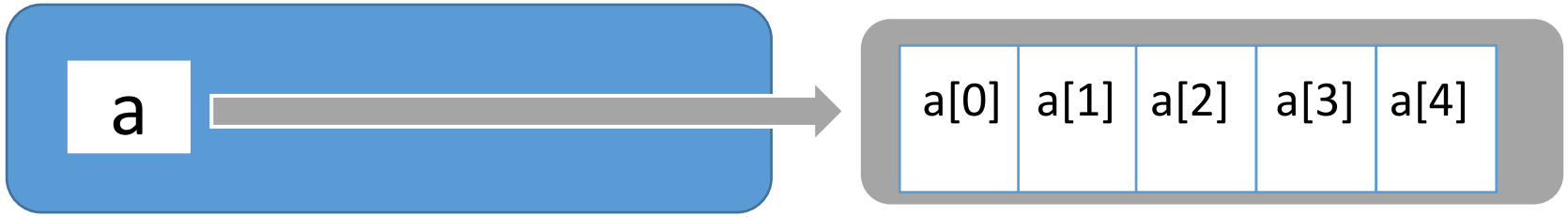
要素アクセス

詳しい人へ：
正確にいうと int の場合も、別領域にオブジェクトができている。ただ、計算の都度、新規オブジェクトができるので、前述のようなイメージで捉えた方が初心者には馴染むかと。

List の紹介

- この問題で list を使う必要はない
- でも、データ入手を先に済ませたほうが、わかりやすいことも多い

List: a = [1, 2, 3, 4, 5]




```
s = 0
for i in range(5):
  s += a[i]
```

添え字でアクセス

```
s = 0
for v in a:
  s += v
```

a の各要素を v に割り当てながら、繰り返し処理

pro1f.py

```
s = sum(a)
```

こういう関数も利用可能

List Comprehensions

List の初期化（内包表記）

数学

- 集合から別の集合を作るときの記法

$$\{ x^2 | \, x \in \mathbb{N} \}$$

Python

- List a: [1, 2, 3] から b: [1*1, 2*2, 3*3] を作成

```
a = [1, 2, 3]
b = [ i*i for i in a]
print(b)
```

List の初期化（入力処理）

- 各行に1つある整数を、5行分取り込む

```
lst = [ int(input()) for _ in range(5) ]
```

pro1f.py

- １行に列挙された複数の整数を取り込む

```
# 1行に数字が列挙されている場合は、行を split() してから
num_strs = input().split()
# 各要素に int を施したものを list として扱う
# (map の返り値は iterator)
lst = list(map(int, num_strs))
```

pro1line.py

プログラム置き場
https://github.com/tomiokamada/regio/tree/master/debug

デバッグ練習

- デバッガで実際にバグを見つけてみましょう
  - $0.1 \times 0.1 + 0.2 \times 0.2 + \cdots + 0.9 \times 0.9$ を求めるつもりのプログラム
  - 実行すると、結果は 3.84999
  - 正解は 2.85
    - 誤差はでるけど、1違うって？？？

debug_test2.py

```
now = 0.1
result = 0.0

while now < 1.0:
    result += now * now
    now += 0.1
print("result:", result)
```

参考：浮動小数点

- 浮動小数点で 0.1 は正確に表現できていない
  - 今回、10回足しても 1.0 よりわずかに小さい値になっていた
- 0.1 = 1/16 + 1/32 + 1/256 + 1/512 + ...
  = 0.0001100110011… (2進数)

10進数	2進数
0.5	0.1
0.25	0.01
0.125	0.001
0.0625	0.0001
0.03125	0.00001

教訓：
- 浮動小数点は誤差を含む
- 条件判定の際は誤差を注意する
- あるいは整数で簡単に処理できるならそうしよう

予選 A 問題を解いてみよう

第9回日本情報オリンピック 予選
A - レシート

問題

- 太郎君は 10冊の本を買ったが、レシートが一枚汚れて見えない。
  9冊のレシートと、総額から計算しよう。

入力

- 最初の行:総額
- 残り9行:9冊の値段

```
9850
1050
800
420
380
600
820
2400
1800
980
```

- 自分でプログラムを解いてみよう
- AtCoder のアカウントから問題を提出してみよう