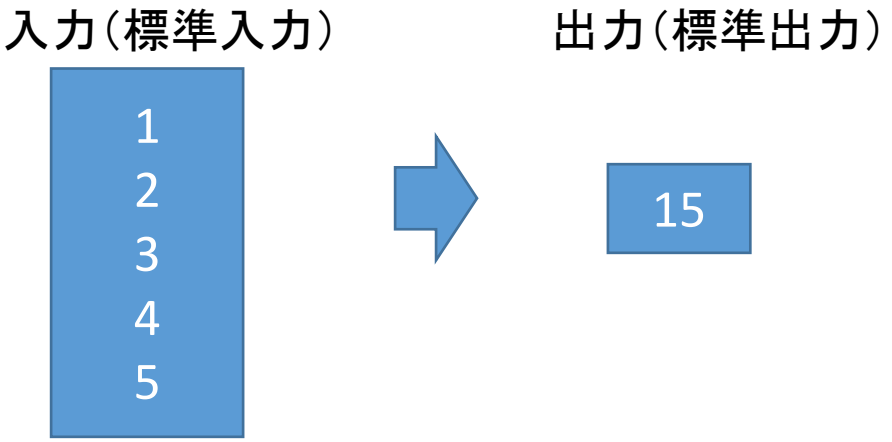
初歩の初歩

- 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



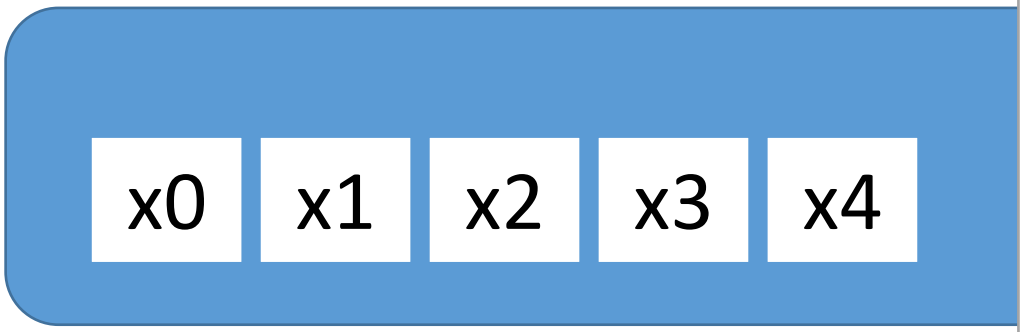
初歩の初歩

- 「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

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

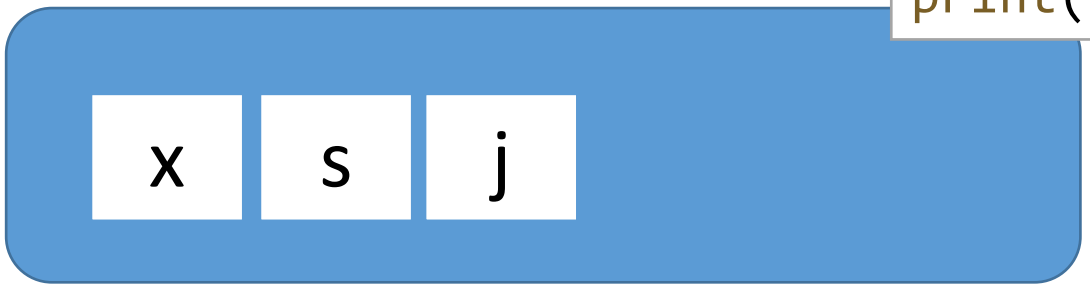
- 「繰り返し」なしで解いてみる
- 変数
  - x 入力用
  - s 計算結果用


```
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

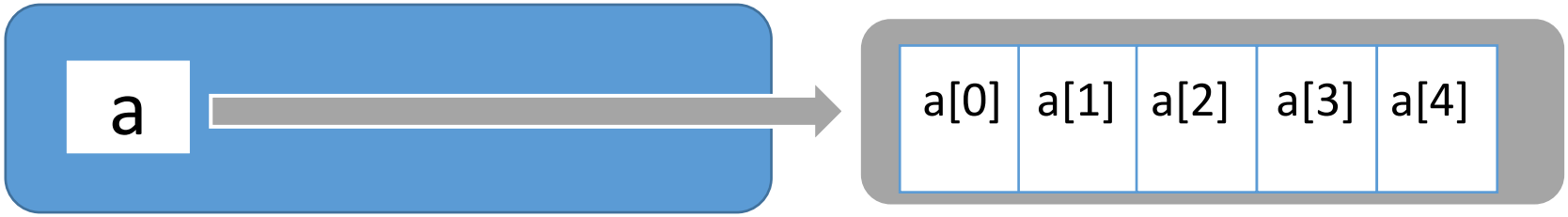
```
s = 0
for j in range(5):
  x = int(input())
  s += x

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

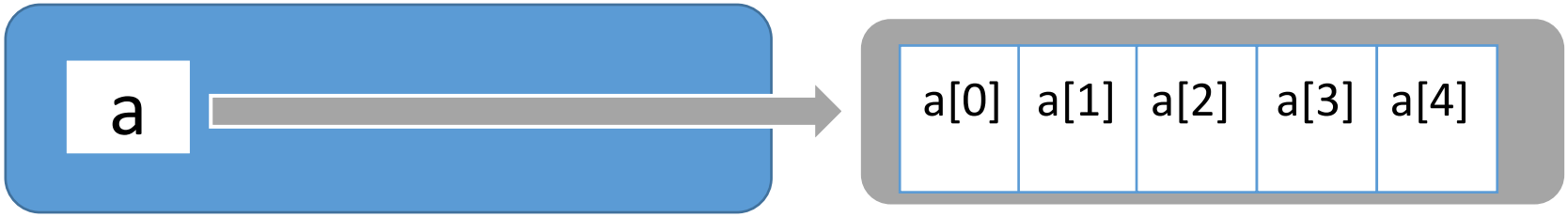
要素アクセス

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

a の各要素を
v に割り当てながら、
繰り返し処理

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

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
  - 1 + 1 / 2 + 1 / 3 + ... + 1 / 9 を求めるプログラム。
  - 実行すると、結果は 1

debug_test.py

```
result = 0.0
for i in range(1, 10):
    tmp = 1 // i
    result += tmp
print(result)
```

range(1, 10) で
1..9 の繰り返し相当