

ACM ICPC Python3 CheetSheet

Aya Tokikaze

2017 年 7 月 13 日

目次

1	入出力	2
1.1	文字列	2
1.2	数値 (int or float)	2
1.3	配列	2
1.4	数値配列	3
1.5	一度に複数の変数へ代入	3
2	制御構造	4
2.1	三項演算子	4
2.2	スワップ	4
3	スライス	5
3.1	N 番目の要素を取り出す	5
3.2	末尾からの取り出し	5
3.3	N 番目から M 番目の要素を取り出す	6
3.4	ステップ数を指定して取り出し	6
4	数値関係	8
4.1	数値の桁数	8
5	文字列とか	8
5.1	0 で桁あわせ	8

1 入出力

1.1 文字列

ある文字列 S を読み込む際は `input()` を用いる。

```
S = input()
```

1.2 数値 (int or float)

整数及び浮動小数点を読み込みたい場合は `input()` で文字列として読み込んだ後、`int` でキャストしてやればいい。

```
N = int(input())
```

1.3 配列

もし、 a_0, a_1, \dots, a_n のような入力を配列として読み込みたい場合は `split()` で切り分けて変数に入れることでおk。

```
a = input().split()
```

1.4 数値配列

`split()` した `list` をすべて数値として入力させたいときは、`map(int, list)` とすればすべての要素を `int` 型に変換できる。

```
a = list(map(int, input().split()))
```

`float` など他の型にもできる。

```
a = list(map(float, input().split()))
```

1.5 一度に複数の変数へ代入

例えば以下のような入力があったとする。

```
10 5
```

それぞれの数値を m 、 r という変数に代入したい場合、アンパック代入を用いて要素ごとに左辺の変数に代入できる。

```
m, r = list(map(int, input().split()))
```

2 制御構造

2.1 三項演算子

python での三項演算子は

```
<条件式がTrueのときの値> if <条件式> else <Falseのときの値>
```

でできる。

```
>>> a = True
>>> print('wei' if a else 'soiya')
wei
```

2.2 スワップ

python でのスワップは c みたいに値を tmp に突っ込まなくても、以下のようにすればできる。

```
a, b = b, a
```

3 スライス

スライスは配列や文字列の任意の部分もしくは範囲を取り出す方法。けっこう便利なので知っておくべき。

以下のような配列 a を定義しておく。

```
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

3.1 N 番目の要素を取り出す

基本中の基本、説明することはない。

```
a[N]  
  
>>> a[1]  
1
```

3.2 末尾からの取り出し

負の数を指定すると配列の末尾からの指定になる

```
# -1 で 一 番 最 後  
>>> a[-1]  
9  
  
>>> a[-8:-5]  
[2, 3, 4]  
  
# 始 点 省 略  
>>> a[:-6]  
[1, 2, 3]  
  
# 終 点 省 略  
>>> a[-2:]  
[8, 9]
```

3.3 N 番目から M 番目の要素を取り出す

コロンで区切ると範囲指定ができる。

```
# n 番目から m-1 番目の要素を取り出す
a[n:m]

# 1 番目から 3 番目の要素を取り出す
>>> a[1:4]
[1, 2, 3]

# 始点を省略すると 0 番目から
>>> a[:3]
[0, 1, 2, 3]

# 終点を省略すると最後まで
>>> a[7:]
[7, 8, 9]
```

3.4 ステップ数を指定して取り出し

奇数項だけ取り出したいときとかにつかう

```
# コロンを 2 個続けるとステップして抜き出します
>>> a[::3]
[1, 4, 7]

# もちろん切り出しつつのステップも可能
>>> a[2:7:3]
[3, 6]

# 負の数を使って後ろから
>>> a[::-3]
[9, 6, 3]

# ややこしい・・・orz
>>> a[-2:-8:-3]
```

```
[8, 5]
```

```
# コロンが2個あって且つ後ろからステップする場合は  
# 逆順にしてから抜き出してるっぽい  
# よって以下は何も取り出せない  
>>> a[-8:-2:-3]  
[]
```

3.4.1 スライスを用いたリバース

`reversed()` よりもこっち使ったほうが簡潔だしいと思います。

```
# 配列の末尾から取り出し  
>>> a[::-1]  
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

4 数値関係

4.1 数値の桁数

```
>>> a = 1234567
>>> int(math.log10(a) + 1)
7
```

5 文字列とか

5.1 0 で桁あわせ

0 埋めの桁合わせをしたいときは一旦 str に変換してから zfill メソッドで指定した桁数分 0 埋めできる。

```
# 元のすうち
>>> a = 12345
# 合わせたい桁
>>> l = 7
>>> str(a).zfill(l)
0012345
```