

1 もくじ

- [Pythonにおける構文](#)
- [for文](#)
 - [概要](#)
 - [forを使う](#)
 - [配列とループ](#)
 - [zip関数](#)
 - [enumerate関数](#)
 - [zip と enumerate](#)
 - [演習1 \(九九表\)](#)
 - [演習2 \(排他的論理和のピラミッド\)](#)
 - [演習3 \(三角形の描画\)](#)
- [while文](#)
 - [概要](#)
 - [whileを使う](#)
- [break/continue](#)
- [if文](#)
 - [概要](#)
 - [if-else文を使う](#)
 - [elif文](#)
- [総括演習](#)
 - [フィボナッチ数列](#)
 - [数字当てゲーム](#)

2 Pythonにおける構文

さきにPythonに存在する構文をいくつかあげておきます。

- if
- for in
- while
- try-except

C言語には `switch` がありますが, Pythonにはありません。

また, 例外に対する構文である `try-except` ですが, 今回は説明しないので注意してください。

3 for文

概要

Pythonの `for` はシーケンス型を利用したものになっていて、C言語のそれとは見た目がかなり異なります。シーケンスから要素を1つずつとってループを回す感じになります。

forを使う

作るもの

- 1~9までの数字をそれぞれ2乗して表示する

C言語

特に変哲もなく。

```
for (int i = 1; i < 10; i++) {  
    printf("%d\n", i*i);  
}
```

Python

```
1 | for val in range(1, 10):  
2 |     print(val**2)
```

```
1  
4  
9  
16  
25  
36  
49
```

64

81

PythonではC言語の標準的な `for` 文を書く時, `range` というものを使う.

```
1 | a = range(0, 3)
2 | print(a.__class__.__name__, a)
3 | print(list(a))
```

```
range range(0, 3)
```

```
[0, 1, 2]
```

`range(start, end+1, step)` のように利用する. (スライスと似た感じ)
`range` で生成されるのは リスト でも タプル なく, `range` である.
紹介はしなかったが, この `range` もシーケンス型の1つである.

この `range` から要素が1つずつとりだされ, `in` の左にある変数へ一個ずつ代入され, ループが回っていく.

配列とループ

● 作るもの

- 適当な文字列を配列に収納し, それらを3つ表示する.

● C言語では

```
char *a[] = {"hoge", "fuga", "piyo", "fizz", "buzz"};

for ( int i = 0; i < sizeof(a) / sizeof(char *); i++) {
    printf("%s\n", a[i]);
}
```

少しややこしいですね.

● pythonでは

```
str_list = ["hoge", "fuga", "piyo", "fizz", "buzz"]
```

```
for str in str_list:  
    print(str)
```

hoge

fuga

piyo

fizz

buzz

楽ですね.

先ほど言ったように `for a in シーケンス` のように書き, シーケンスから1つずつ値が取り出されていくので, それを表示するだけです.

zip関数

● 作るもの

- 適当な文字列の配列を作る
- 適当な数値の配列を作る
- インデックス `n` の数値文だけ文字列を表示する.

● C言語では

```
int a[4] = {3, 6, 2, 5};  
char *str[] = {"hoge", "fuga", "piyo", "fizz"};  
  
for ( int i = 0; i < sizeof(a) / sizeof(int); i++) {  
    for ( int try = 0; try < a[i]; try++ ) printf("%s", str[i]);  
    printf("\n");  
}
```

```
hogehogehoge
fugafugafugafugafugafuga
piyopiyo
fizzfizzfizzfizzfizz
```

Pythonでは

```
num_list = [3, 6, 2, 5]
str_list = ["hoge", "fuga", "piyo", "fizz"]

for num, str in zip(num_list, str_list):
    print(str * num)
```

`zip` でいくつかのシーケンス型をまとめて、それらの1要素ずつ取り出すことができる。
ちなみに `zip` で作成されるのは `zip` クラスであり、それをリストへ変換すると以下ようになる。

```
num_list = [3, 6, 2, 5]
str_list = ["hoge", "fuga", "piyo", "fizz"]

print(zip(num_list, str_list).__class__.__name__)
print(list(zip(num_list, str_list)))
```

このように要素がタプルになっており、それぞれが各変数へアンパックされ、代入される。

enumerate関数

`enumerate` は添字を得ることができます。
大したことはないのですが、テキストに説明します。

```
1 | str_list = ["hoge", "fuga", "piyo", "fizz"]
2 |
3 | for i, str in enumerate(str_list):
4 |     print(i, ":", str)
```

```
0 : hoge
1 : fuga
2 : piyo
3 : fizz
```

```
1 str_list = ["hoge", "fuga", "piyo", "fizz"]
2 print(enumerate(str_list).__class__.__name__)
3 print(list(enumerate(str_list)))
```

enumerate

```
[(0, 'hoge'), (1, 'fuga'), (2, 'piyo'), (3, 'fizz')]
```

zip と enumerate

もし `zip` と `enumerate` を同時に使いたい場合、以下のようにすると良いと思います。

```
1 str_list = ["hoge", "fuga", "piyo", "fizz"]
2 num_list = [5, 1, 3, 6]
3
4 for i, (num, str) in enumerate(zip(str_list, num_list)):
5     print(i, ":", num, "/", str)
```

```
0 : hoge / 5
1 : fuga / 1
2 : piyo / 3
3 : fizz / 6
```

演習1 (九九表)

● 内容

- 九九表を描画してください。

- もしできるなら, 乗数と被乗数も外側に描画してください.

C言語のように書式変換指定子を使いたい場合, 以下のようにできます.

```
1 | val1 = 3
2 | val2 = 8
3 | val3 = 2341
4 | print("①->{0:3d}, ②->{1:03d}, ③->{2}".format(val1, val2, val3))
5 | print("①->{1:3d}, ②->{2:03d}, ③->{0}".format(val1, val2, val3))
6 | print("①->{:3d}, ②->{:03d}, ③->{}".format(val1, val2, val3))
```

①-> 3, ②->008, ③->2341

①-> 8, ②->2341, ③->3

①-> 3, ②->008, ③->2341

文字列中に {何番目の引数を表示するか:変換指定子} を挿入し, 文字列.format() で引数を指定していく.
何番目の引数を表示するかは省略でき, そのときは順に読み込まれていくことになる.

また, print の末尾改行をなくすには

```
print("~~~~", end="")
```

のようになる.

わからない場合は, 以下にポイントを列挙していくので, 参考にしてください.
また, 最後にプログラム例と出力結果を示します.

● ポイント1

- 範囲が[1, 9]のループをネストする

● プログラム例

周りなし

```
1 | for i in range(1, 10):
2 |     for j in range(1, 10):
```

```

3         print( "{:3d}".format(i * j), end="")
4     print()

```

```

1  2  3  4  5  6  7  8  9
2  4  6  8 10 12 14 16 18
3  6  9 12 15 18 21 24 27
4  8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81

```

罫線等あり

```

1     print(" |", *["{:2d}".format(i) for i in range(1, 10)])
2     print("--" + "|" + "-"*30)
3     for i in range(1, 10):
4         print("{:2d}|".format(i), end="")
5         for j in range(1, 10):
6             print( "{:3d}".format(i * j), end="")
7         print()

```

```

|  1  2  3  4  5  6  7  8  9
--|-----
1|  1  2  3  4  5  6  7  8  9
2|  2  4  6  8 10 12 14 16 18
3|  3  6  9 12 15 18 21 24 27
4|  4  8 12 16 20 24 28 32 36
5|  5 10 15 20 25 30 35 40 45
6|  6 12 18 24 30 36 42 48 54
7|  7 14 21 28 35 42 49 56 63

```



```
8| 8 16 24 32 40 48 56 64 72
9| 9 18 27 36 45 54 63 72 81
```

リスト内包表記

```
1 list9x9 = [[ "{:2d}".format(i*j) for i in range(1, 10)] for j in range(1, 10)]
2
3 print(" ", "|", *["{:2d}".format(i) for i in range(1, 10)])
4 print("--" + "|" + "-"*28)
5 for i, hline in enumerate(list9x9):
6     print(i+1, "|", *hline)
```

```
| 1 2 3 4 5 6 7 8 9
--|-----
1 | 1 2 3 4 5 6 7 8 9
2 | 2 4 6 8 10 12 14 16 18
3 | 3 6 9 12 15 18 21 24 27
4 | 4 8 12 16 20 24 28 32 36
5 | 5 10 15 20 25 30 35 40 45
6 | 6 12 18 24 30 36 42 48 54
7 | 7 14 21 28 35 42 49 56 63
8 | 8 16 24 32 40 48 56 64 72
9 | 9 18 27 36 45 54 63 72 81
```

演習2 (排他的論理和のピラミッド)

内容

- 大きさ10程度のリストへ `0` か `1` をランダムに入れていく.
- それを左から順に `XOR` (排他的論理和)をしていく.
- また, 排他的論理和が済んだ列に対し, 同様の操作を行っていく.

出力例

```
1 1 0 0 1 1 0 1 1 1
0 1 0 1 0 1 1 0 0
1 1 1 1 1 0 1 0
0 0 0 0 1 1 1
0 0 0 1 0 0
0 0 1 1 0
0 1 0 1
1 1 1
0 0
0
```

排他的論理和は $A \oplus B$ のように計算でき, $0 \oplus 0 = 1 \oplus 1 = 0$, $1 \oplus 0 = 0 \oplus 1 = 1$ である.

● ポイント1

- 各「段数」と「左側にある空白の文字数」の関係を考える

● ポイント2

- データを保持する必要があるのは, 以前のブール配列とそれから計算されるブール配列の二つ. (一応うまくやれば配列は1つでもできる.)
- このデータ配列を2重ループで回す

● プログラム例

```
1 import random
2
3 def main():
4     bit_list = random.choices([0, 1], k=10 + random.choice(range(-2, 3)))
5     print(*bit_list)
6
7     for try_num in range(0, len(bit_list) - 1):
8         next_bit_list = [0] * (len(bit_list) - 1)
9         print(" " * (try_num+1), end="")
10        for i in range(0, len(next_bit_list)):
```

```

11         next_bit_list[i] = bit_list[i] ^ bit_list[i+1]
12
13     bit_list = next_bit_list
14     print(*bit_list)
15
16
17 if __name__ == '__main__':
18     main()

```

```


0 1 1 0 1 1 0 0 1
1 0 1 1 0 1 0 1
1 1 0 1 1 1 1
0 1 1 0 0 0
1 0 1 0 0
1 1 1 0
0 0 1
0 1
1


```

演習3 (三角形の描画)

内容

以下の条件を満たすプログラムを作成してください。

- 6~20の整数値' n 'を標準入力から受け取る. (もしくはコマンドラインから受け取る)
- 1, 1, 2, 3, 5, ...となるようなフィボナッチ数列を20番目まで出力する.
(コマンドライン引数として受け取る.)
- ' n ' 個の  を底辺とする三角形を出力する.

例えば  のとき

```

**
*****

```

```
*****
```

となり, `n=11` のとき

```
  *
 ***
*****
*****
*****
*****
*****
```

のようになればよい.

ちなみにコマンドライン引数は以下のように受け取れる.

```
import sys
n = int(sys.argv[1])
```

また, 標準入力から受け取るには

```
n = int(input("> 三角形の底辺の*の数nを入力してください : "))
```

のように受け取れる.

わからない場合は, 以下にポイントを列挙していくので, 参考にしてください.
また, 最後にプログラム例と出力結果を示します.

● ポイント1

- `n` と, 三角形の段数の関係に注目する.

● ポイント2

- 各段差に番号を振り, その番号と空白/星の数の関係を見出す

● ポイント3

- 各関係を「底辺が奇数のとき」、「底辺が偶数のとき」に分けて考える。

● ポイント4

- 段差を上から $n, n - 1, \dots, 2, 1$ とし, 描画段階における段差を N とおく。
- そのとき, 各段差における左側の空白の数は $\text{stage}-1$
- 奇数の時, 中心を1列真ん中においた時, 左右にある星の数はそれぞれ $n - N$ 個. 中心の1つも合わせれば全部で $2(n-N) + 1$ 個あることになる。
- 偶数の時, 中心を2列真ん中においたと考えると, その左右にある星の数はそれぞれ $n - N$ 個. 中心に2つあるので, 全部で $2(n-N) + 2$ 個あることになる。

● プログラム例

```
1  import sys
2
3  def main():
4      n = int(sys.argv[1])
5      stage_num = (n + 1) // 2
6
7      for stage in range(stage_num, 0, -1):
8          print(" " * (stage-1) + "*" * (2*(stage_num - stage) + 2 - n%2) )
9
10 if __name__ == '__main__':
11     main()
```

```
  **
 ****
*****
*****
*****
*****
```

4 while文

概要

使い方はC言語とほぼ同じです。

whileの後に来る条件文が `True` のうちはwhileブロックのプログラムが行われます。

whileを使う

作るもの

- 文字列"end"が入力されたらプログラムが終了するプログラム

python3では `input()` で入力が戻り値で得られる。

区切りは改行など。

また, `input("入力前に表示する文字列")` のように, 入力前に文字列を表示することもできる。

C言語なら

```
char input[20];  
do {  
    scanf("%s", input);  
} while (strcmp(input, "end") != 0);
```

Pythonなら

```
while input("> 文字列を入力してください : ") != "end":  
    pass
```

Pythonでは `while` の処理ブロックを何も書かないということができない。

そのため, 明示的にここではなにもしませんよということを示すために `pass` と書かなければならない。

5 break/continue

すこし横道にそれますが, `break` と `continue` について触れておきます。

for/whileのループでは途中に `break` を挟むと, そこでループを抜け出すことができます。

また, `continue` を挟むと以降の処理をスルーして, 次のループへ移ることができます。

```
1  a = 0
2
3  while True:
4      a += 1
5
6      if a in [2, 4, 7]:
7          print()
8          continue
9
10     if a >= 10:
11         break
12
13     print(a)
```

```
1
2
3
4
5
6
7
8
9
```

6 if文

概要

Pythonの `if` は直後にくる値がブール値の `True` か `False` かで後のコードブロックの実行の有無を判断します。

if-else文を使う

● 作るもの

- 配列を作成し, そこへ整数を10個ほど入れる.
- 配列の各要素に対し, 「偶数ならそれを表示, 奇数なら二乗して表示」を行う.

● C言語では

あくまで一例ですが, 以下のようにと思います.
(forの `()` 内で変数を宣言する文法はかなり古いgccではできないので注意.)

```
#include<stdio.h>

#define N 10

void main(void) {
    int a[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    for (int i = 0; i < N; i++) {
        printf("%2d -> ", a[i]);
        if (a[i] % 2 == 0) {
            printf("%d\n", a[i]);
        }else {
            printf("%d\n", a[i] * a[i]);
        }
    }
}
```

● Pythonでは

```
1 import random
2
3 a = random.sample(range(1, 30), k=10)
4
5 for val in a:
6     if val % 2 == 0:
```



```
7         print(val, "->", val)
8     else:
9         print(val, "->", val**2)
```

```
22 -> 22
11 -> 121
2 -> 2
19 -> 361
21 -> 441
16 -> 16
14 -> 14
15 -> 225
3 -> 9
24 -> 24
```

とかで書けます。

Pythonの `if` では `()` が必要なく、後ろに `:` をつける必要があります。
また、実行する

elif文

● やること

- 数字を適当に10個程度配列へ用意する。
- 各要素 n に対して以下の処理を行う
 - $n > 30$ なら -1 をかけて表示
 - $30 \geq n > 20$ なら 2 をかけて表示
 - $20 \geq n > 10$ なら 2 乗して表示
 - $10 \geq n$ なら そのまま表示

● C言語では

```
int a[] = {1, 11, 5, 27, 33, 2, 38, 17, 22};

for (int i = 0; i < (sizeof(a)/sizeof(int)); i++) {
```

```
int tmp = a[i];
if (tmp > 30) {
    tmp *= -1;
}else if (tmp > 20) {
    tmp *= 2;
}else if (tmp > 10) {
    tmp *= tmp;
}
printf("%d\n", tmp);
}
```

pythonでは

```
1 import random
2
3 val_list = random.sample(range(1, 40), k=10)
4 print(val_list)
5
6 for val in val_list:
7     if val > 30:
8         print(-val)
9     elif val > 20:
10        print(val*2)
11    elif val > 10:
12        print(val**2)
13    else:
14        print(val)
```

[10, 30, 37, 39, 2, 28, 38, 17, 12, 32]

10

60

-37

-39

2
56
-38
289
144
-32

7 総括演習

フィボナッチ数列

● 作るもの

- 1, 1, 2, 3, 5, ...となるようなフィボナッチ数列を10番目まで出力する.

(ちなみにフィボナッチ数列は再帰関数で作成するのが一般的ですが, 今回はfor/ifで作成してください.)

以下にポイントを載せていくので, 作成に詰まったら1つずつ見てみてください.
また最後にプログラム例と出力例を載せておきます.

● ポイント1

● プログラム例

```
1 val1 = 0
2 val2 = 0
3 N = 10
4
5 print("try_num :", *["{:3d}", ".format(i+1) for i in range(N)], sep="")
6 print("-" * (N * 5 + 10))
7
8 print("fib_num :", end="")
9 for i in range(N):
```

```

10     if val1 + val2 == 0:
11         val2 = 1
12         print("{:3d}".format(val1 + val2), end=" ")
13     else:
14         print("{:3d}".format(val1 + val2), end=" ")
15         val1, val2 = val2, val1+val2

```

```
try_num : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
```

```
-----
fib_num : 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,
```

数字当てゲーム

作るもの

1. プログラム中で $1 \leq N \leq 100$ となるような N をランダムに得る
2. その後, 整数値の入力 `input` を受け付け, N と比較する.
 - `input = N` : プログラムを終了する.
 - `input ≠ N` : 入力値が N より大きい小さいかを出力する.
3. 2を繰り返す.

ランダムに値を得るには `random` を `import` として, `random.choice` を使うと良い.

```

import random

N = random.choice(range(1, 101))

```

また入力は `input("表示する文字列")` で得られるが, 得られる値は文字列であるため以下のように `int()` で整数値に変換する必要がある.

```

n = input(">整数値を入力してください")
n = int(n)

```

ちなみに `int()` では対象が整数値に変換できないような内容だとエラーが出るので注意(数値以外が入力されたときなど)

ポイント1

- breakでwhileから抜け出すようにする

プログラム例

```
1  import random
2
3  N = random.choice(range(1, 101))
4
5  while True:
6      n = input("> 1以上, 100以下の整数値を入力してください : ")
7      if int(n) == N:
8          print("> 正解です")
9          break
10     elif int(n) > N:
11         print("> 値が大きいです")
12     else:
13         print("> 値が小さいです")
14     print()
```