



プログラミング言語 第十一回

担当: 篠沢 佳久
栗原 聡

2019年 7月1日



本日の内容

- 二次元配列
- 二次元配列と繰り返し
- ファイル操作

- 練習問題①～⑤



二次元配列

二次元配列の宣言
要素の参照, 代入

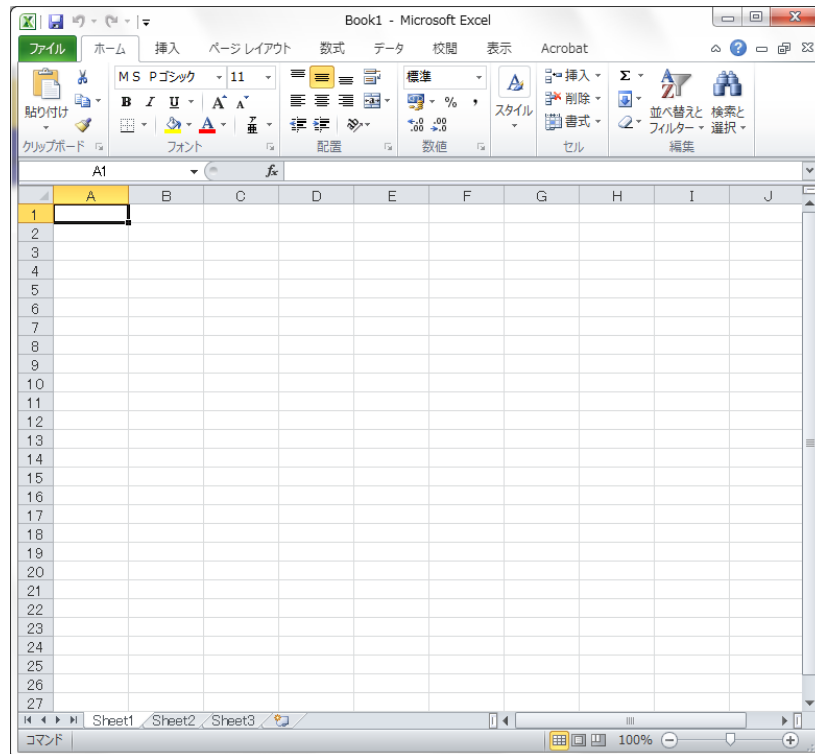


2次元配列

- 表が使えると、随分便利です
- スプレッドシートを思い起こしてください
 - スプレッドシートって何ですか？

二次元の配列＝二次元の表(行列)

- 表といえば，二次元かな
- 表計算ソフトも2次元だしな



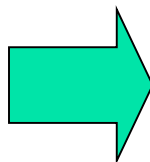
二次元配列の宣言①

3×3の行列 a

$$a = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

表の場合

1	2	3
4	5	6
7	8	9



Pythonでの宣言

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

二次元配列の宣言②

3×3の行列 a

1	2	3
4	5	6
7	8	9

Pythonでの宣言

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

さらに[]で囲む

「,」で区切る

二次元配列の宣言③

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

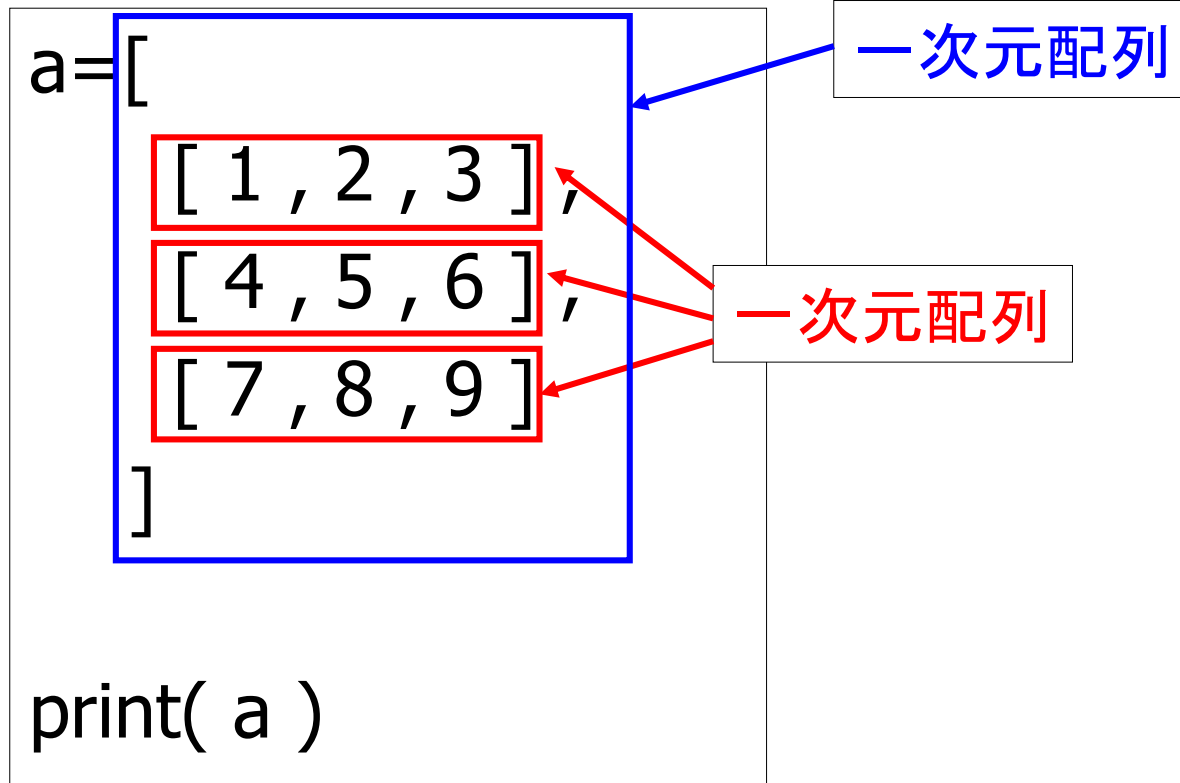
```
print( a )
```

```
>python sample.py  
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

一次元配列

二次元配列は一次元配列の要素を一次元配列として
いるとみなせる

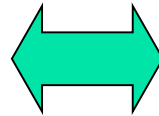
二次元配列の宣言④



二次元配列は一次元配列の要素を一次元配列として
いるとみなせる

二次元配列の要素の参照①

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



a[0][0]	a[0][1]	a[0][2]
a[1][0]	a[1][1]	a[1][2]
a[2][0]	a[2][1]	a[2][2]



二次元配列の要素の参照②

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

```
print( a[ 0 ][ 0 ] )  
print( a[ 0 ][ 1 ] )  
print( a[ 0 ][ 2 ] )
```

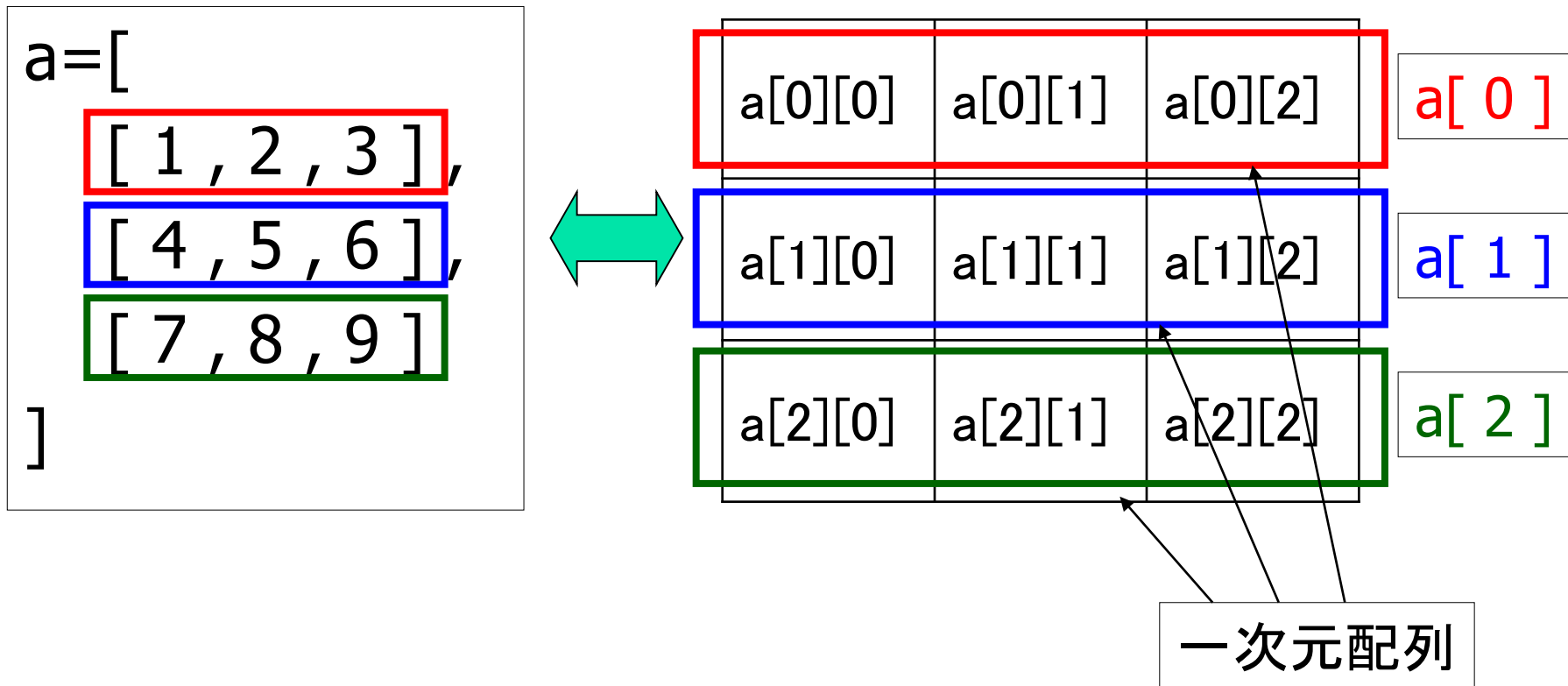
```
print( a[ 1 ][ 0 ] )  
print( a[ 1 ][ 1 ] )  
print( a[ 1 ][ 2 ] )
```

```
print( a[ 2 ][ 0 ] )  
print( a[ 2 ][ 1 ] )  
print( a[ 2 ][ 2 ] )
```

```
>python sample.py
```

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

二次元配列の要素の参照③



二次元配列の要素の参照④

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

```
print( a[ 0 ] )  
print( a[ 1 ] )  
print( a[ 2 ] )
```

```
>python sample.py
```

```
[1, 2, 3]
```

a[0]

```
[4, 5, 6]
```

a[1]

```
[7, 8, 9]
```

a[2]

二次元配列の要素の参照⑤

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

>python sample.py

3

3

3

3

配列a の要素数

a[0], a[1], a[2] の要素数

二次元配列の要素の参照⑤

```
a=[  
    [ 1 ],  
    [ 4 , 5 ],  
    [ 7 , 8 , 9 ]  
]
```

配列の要素数が異なっても良い

```
print( len(a) )  
print( len(a[ 0 ]) )  
print( len(a[ 1 ]) )  
print( len(a[ 2 ]) )
```

配列a の要素数

```
>python sample.py
```

3

1

2

3

a[0], a[1], a[2] の要素数

二次元配列の要素の参照⑥

要素数

len(a)	1	2	3
	4	5	6
	7	8	9

a[0]

a[1]

a[2]

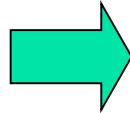
len(a[0])
len(a[1])
len(a[2])

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


二次元配列の宣言①

- 要素の値が分かっている場合

1	2	3
4	5	6
7	8	9
10	11	12



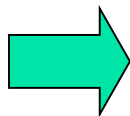
```
a=[  
  [ 1 , 2 , 3 ],  
  [ 4 , 5 , 6 ],  
  [ 7 , 8 , 9 ],  
  [ 10 , 11 , 12 ]  
]
```

二次元配列の宣言②

- 要素数のみ分かっている場合

3列

4行



```
a =[0]*4
```

```
a[ 0 ] = [0]*3
```

```
a[ 1 ] = [0]*3
```

```
a[ 2 ] = [0]*3
```

```
a[ 3 ] = [0]*3
```

二次元配列の宣言②

要素数4の一次元配列a を作成

```
a = [0]*4  
print( a )
```

```
>python sample.py  
[0, 0, 0, 0]
```

0の値を持つ配列a

a	0	a[0]
	0	a[1]
	0	a[2]
	0	a[3]

二次元配列の宣言②

```
a = [0]*4  
a[ 0 ] = [0]*3  
print( a )
```

```
>python sample.py  
[[0, 0, 0], 0, 0, 0]
```

配列a[0]に要素が3の配列を作成

a[0]	0	0	0
a[1]	0		
a[2]	0		
a[3]	0		

a[0] のみ3個の要素を持つ配列

二次元配列の宣言②

```
a = [0]*4  
a[ 0 ] = [0]*3  
a[ 1 ] = [0]*3  
  
print( a )
```

a[0], a[1]
3個の要素を持つ配列

配列a[1]に要素が3の配列を作成

a[0]	0	0	0
a[1]	0	0	0
a[2]	0		
a[3]	0		

```
>python sample.py  
[[0, 0, 0], [0, 0, 0], 0, 0]
```

二次元配列の宣言②

```
a = [0]*4  
a[ 0 ] = [0]*3  
a[ 1 ] = [0]*3  
a[ 2 ] = [0]*3  
  
print( a )
```

a[0], a[1], a[2]
3個の要素を持つ配列

配列a[2]に要素が3の配列を作成

a[0]	0	0	0
a[1]	0	0	0
a[2]	0	0	0
a[3]	0		

```
>python sample.py  
[[0, 0, 0], [0, 0, 0], [0, 0, 0], 0]
```

二次元配列の宣言②

```
a = [0]*4  
a[ 0 ] = [0]*3  
a[ 1 ] = [0]*3  
a[ 2 ] = [0]*3  
a[ 3 ] = [0]*3  
  
print( a )
```

配列a[3]に要素が3の配列を作成

a[0]	0	0	0
a[1]	0	0	0
a[2]	0	0	0
a[3]	0	0	0

```
>python sample.py  
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```



二次元配列の要素への代入

```
a =[0]*4  
a[ 0 ] = [0]*3  
a[ 1 ] = [0]*3  
a[ 2 ] = [0]*3  
a[ 3 ] = [0]*3
```

```
a[ 0 ][ 0 ] = 1  
a[ 0 ][ 1 ] = 2  
a[ 0 ][ 2 ] = 3  
a[ 1 ][ 0 ] = 4  
a[ 1 ][ 1 ] = 5  
a[ 1 ][ 2 ] = 6
```

```
a[ 2 ][ 0 ] = 7  
a[ 2 ][ 1 ] = 8  
a[ 2 ][ 2 ] = 9
```

```
a[ 3 ][ 0 ] = 10  
a[ 3 ][ 1 ] = 11  
a[ 3 ][ 2 ] = 12
```

```
print( a )
```

```
>python sample.py
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
```


二次元配列の宣言③

aは配列と宣言

```
a = []
```

```
a.append([])
```

```
a[0].append(1)
```

```
a[0].append(2)
```

```
a[0].append(3)
```

```
a.append([])
```

```
a[1].append(4)
```

```
a[1].append(5)
```

```
a[1].append(6)
```

```
a.append([])
```

```
a[2].append(7)
```

```
a[2].append(8)
```

```
a[2].append(9)
```

```
a.append([])
```

```
a[3].append(10)
```

```
a[3].append(11)
```

```
a[3].append(12)
```

```
print( a )
```

a[0], a[1], a[2]
, a[3]を配列として
追加

```
>python sample.py
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
```



二次元配列の宣言③

```
a = []
```

```
print( a )
```

配列a を作成

```
a = []
```

```
a.append([])
```

```
print( a )
```

配列a[0] を作成

```
>python sample.py  
[]
```

```
>python sample.py  
[[]]
```

0番目の要素を配列として追加



二次元配列の宣言③

```
a = []  
a.append([])  
a[0].append(1)  
a[0].append(2)  
a[0].append(3)  
  
print( a )
```

```
>python sample.py  
[[1, 2, 3]]
```

```
a = []  
a.append([])  
a[0].append(1)  
a[0].append(2)  
a[0].append(3)  
  
a.append([])  
  
print( a )
```

```
>python sample.py  
[[1, 2, 3], []]
```



二次元配列の宣言③

```
a = []  
a.append([])  
a[0].append(1)  
a[0].append(2)  
a[0].append(3)
```

```
a.append([])  
a[1].append(4)  
a[1].append(5)  
a[1].append(6)
```

```
print( a )
```

```
>python sample.py  
[[1, 2, 3], [4, 5, 6]]
```



二次元配列の宣言③

```
a = []  
a.append([])  
a[0].append(1)  
a[0].append(2)  
a[0].append(3)  
  
a.append([])  
a[1].append(4)  
a[1].append(5)  
a[1].append(6)
```

```
a.append([])  
print( a )
```

```
a[2].append(7)  
a[2].append(8)  
a[2].append(9)
```

```
print( a )
```

```
>python sample.py  
[[1, 2, 3], [4, 5, 6], []]  
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```



二次元配列の宣言③

```
a = []  
a.append([])  
a[0].append(1)  
a[0].append(2)  
a[0].append(3)
```

```
a.append([])  
a[1].append(4)  
a[1].append(5)  
a[1].append(6)
```

```
a.append([])  
a[2].append(7)  
a[2].append(8)  
a[2].append(9)
```

```
a.append([])  
print( a )
```

```
a[3].append(7)  
a[3].append(8)  
a[3].append(9)  
print( a )
```

```
>python sample.py  
[[1, 2, 3], [4, 5, 6], [7, 8, 9], []]  
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [7, 8, 9]]
```

二次元配列の宣言④

配列の宣言をしない場合

```
a = []
```

```
a[ 0 ][ 0 ] = 1
```

```
a[ 0 ][ 1 ] = 2
```

```
a[ 0 ][ 2 ] = 3
```

```
a[ 1 ][ 0 ] = 4
```

```
a[ 1 ][ 1 ] = 5
```

```
a[ 1 ][ 2 ] = 6
```

```
a[ 2 ][ 0 ] = 7
```

```
a[ 2 ][ 1 ] = 8
```

```
a[ 2 ][ 2 ] = 9
```

```
a[ 3 ][ 0 ] = 10
```

```
a[ 3 ][ 1 ] = 11
```

```
a[ 3 ][ 2 ] = 12
```

```
print( a )
```

```
>python sample.py
```

```
Traceback (most recent call last):
```

```
File "C:¥Users¥shino¥Desktop¥sample.py", line 3, in <module>
```

```
a[ 0 ][ 0 ] = 1
```

```
IndexError: list index out of range
```

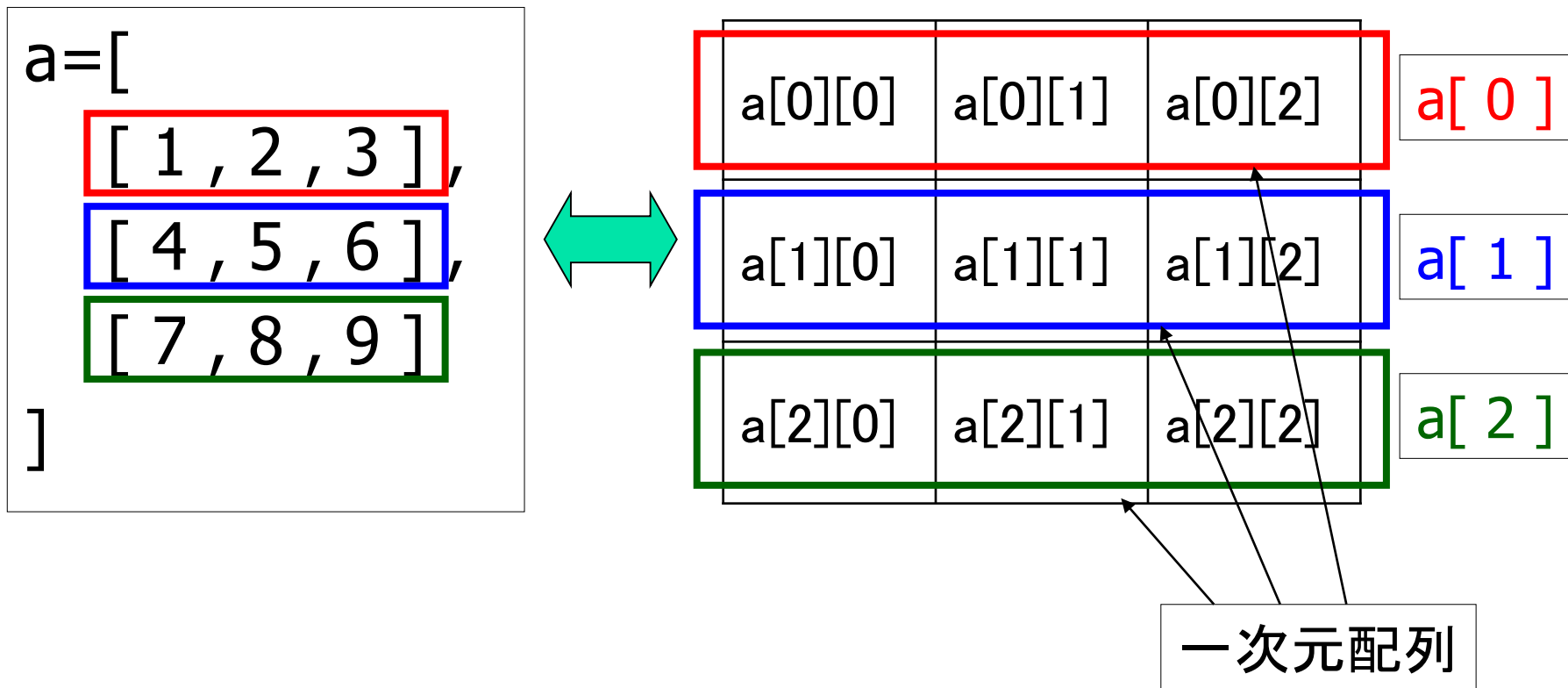


二次元配列の宣言のまとめ

- 要素の値が分かっている場合
- 要素数のみ分かっている場合
- 要素の値, 要素数も分からない場合

二次元配列のまとめ①

■ 一次元配列の一次元配列





二次元配列のまとめ②

- 各要素は要素数が異なっても良い
- 各要素の値は、整数、小数、文字型、配列（多次元配列）でもよい

```
a=[  
    [ 1 ],  
    [ 4 , 5 ],  
    [ 7 , 8 , 9 ]  
]
```

```
a=[  
    [ 1 ],  
    [ "A" , "B" , 3.141 ],  
    [ 2.0 , "3" , [ 4 , 5 ] ]  
]
```



配列



二次元配列と繰り返し①

二重ループ中での要素の参照

二重ループ(復習)

```
for i in range(4):  
    for j in range(3):  
        print( i , "+" , j , "=" , i + j )
```

>python sample.py

0 + 0 = 0
0 + 1 = 1
0 + 2 = 2

i = 0 の時, j = 0~2

1 + 0 = 1
1 + 1 = 2
1 + 2 = 3

i = 1 の時, j = 0~2

2 + 0 = 2
2 + 1 = 3
2 + 2 = 4

i = 2 の時, j = 0~2

3 + 0 = 3
3 + 1 = 4
3 + 2 = 5

i = 3 の時, j = 0~2

i=0

```
for j in range(3):  
    print( i , "+" , j , "=" , i + j )
```

i=1

```
for j in range(3):  
    print( i , "+" , j , "=" , i + j )
```

i=2

```
for j in range(3):  
    print( i , "+" , j , "=" , i + j )
```

i=3

```
for j in range(3):  
    print( i , "+" , j , "=" , i + j )
```

二次元配列の要素の参照①

要素番号(インデックス)を用いての参照

```
a=[  
    [ 1, 2, 3 ],  
    [ 4, 5, 6 ],  
    [ 7, 8, 9 ],  
    [10, 11, 12]  
]
```

```
for i in range(4):  
    for j in range(3):  
        print( " a[" , i , "][" , j , "] =" , a[ i ][ j ] )
```

$i=0, j=0\sim 2$

$i=1, j=0\sim 2$

$i=2, j=0\sim 2$

$i=3, j=0\sim 2$

>python sample.py

```
a[ 0 ][ 0 ] = 1  
a[ 0 ][ 1 ] = 2  
a[ 0 ][ 2 ] = 3
```

```
a[ 1 ][ 0 ] = 4  
a[ 1 ][ 1 ] = 5  
a[ 1 ][ 2 ] = 6
```

```
a[ 2 ][ 0 ] = 7  
a[ 2 ][ 1 ] = 8  
a[ 2 ][ 2 ] = 9
```

```
a[ 3 ][ 0 ] = 10  
a[ 3 ][ 1 ] = 11  
a[ 3 ][ 2 ] = 12
```

i

j

```
for j in range(3):  
    print( " a[ 0 ][", j , "]" = " , a[ 0 ][ j ] )
```

```
for j in range(3):  
    print( " a[ 1 ][", j , "]" = " , a[ 1 ][ j ] )
```

```
for j in range(3):  
    print( " a[ 2 ][", j , "]" = " , a[ 2 ][ j ] )
```

```
for j in range(3):  
    print( " a[ 3 ][", j , "]" = " , a[ 3 ][ j ] )
```

二次元配列の要素の参照①'

要素番号(インデックス)を用いての参照

```
a=[  
  [ 1, 2, 3 ],  
  [ 4, 5, 6 ],  
  [ 7, 8, 9 ],  
  [10, 11, 12 ]  
]
```

```
for i in range(3):  
  for j in range(4):  
    print( " a[" , j , "][" , i , "]" = " , a[ j ][ i ] )
```

i=0, j=0~3

i=1, j=0~3

i=2, j=0~3

>python sample.py


```
a[ 0 ][ 0 ] = 1  
a[ 1 ][ 0 ] = 4  
a[ 2 ][ 0 ] = 7  
a[ 3 ][ 0 ] = 10
```

```
a[ 0 ][ 1 ] = 2  
a[ 1 ][ 1 ] = 5  
a[ 2 ][ 1 ] = 8  
a[ 3 ][ 1 ] = 11
```

```
a[ 0 ][ 2 ] = 3  
a[ 1 ][ 2 ] = 6  
a[ 2 ][ 2 ] = 9  
a[ 3 ][ 2 ] = 12
```

j

i



```
for j in range(4):  
    print( " a[" , j , "]"[ 0 ] =" , a[ j ][ 0 ] )
```

```
for j in range(4):  
    print( " a[" , j , "]"[ 1 ] =" , a[ j ][ 1 ] )
```

```
for j in range(4):  
    print( " a[" , j , "]"[ 2 ] =" , a[ j ][ 2 ] )
```

二次元配列の要素の参照②

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [10 ,11 ,12 ]  
]
```

len(a)の値は4

```
for i in range(len(a)):  
    for j in range(len(a[i])):  
        print( " a[" , i , "][" , j , "]" = " , a[ i ][ j ] )
```

len(a[0]), len(a[1]), len(a[2]), len(a[3])は全て3

```
>python sample.py
```

```
a[ 0 ][ 0 ] = 1  
a[ 0 ][ 1 ] = 2  
a[ 0 ][ 2 ] = 3  
a[ 1 ][ 0 ] = 4  
a[ 1 ][ 1 ] = 5  
a[ 1 ][ 2 ] = 6  
a[ 2 ][ 0 ] = 7  
a[ 2 ][ 1 ] = 8  
a[ 2 ][ 2 ] = 9  
a[ 3 ][ 0 ] = 10  
a[ 3 ][ 1 ] = 11  
a[ 3 ][ 2 ] = 12
```



二次元配列の要素の参照③

whileを用いての参照

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [ 10 , 11 , 12 ]  
]  
  
i = 0  
while i < len(a):  
    j = 0  
    while j < len(a[i]):  
        print( " a[" , i , "][" , j , "]" = " , a[ i ][ j ] )  
        j+=1  
    i+=1
```

```
>python sample.py
```

```
a[ 0 ][ 0 ] = 1  
a[ 0 ][ 1 ] = 2  
a[ 0 ][ 2 ] = 3  
a[ 1 ][ 0 ] = 4  
a[ 1 ][ 1 ] = 5  
a[ 1 ][ 2 ] = 6  
a[ 2 ][ 0 ] = 7  
a[ 2 ][ 1 ] = 8  
a[ 2 ][ 2 ] = 9  
a[ 3 ][ 0 ] = 10  
a[ 3 ][ 1 ] = 11  
a[ 3 ][ 2 ] = 12
```

二次元配列の要素の参照④

要素番号(インデックス)ではなく直接、二次元配列の要素を参照するには？

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [10 , 11 , 12 ]  
]
```

```
for x in a:  
    print( x )
```

変数 x には順番に「配列」
a[0], a[1], a[2], a[3] が代入される

```
>python sample.py  
[1, 2, 3] ← a[ 0 ]  
[4, 5, 6] ← a[ 1 ]  
[7, 8, 9] ← a[ 2 ]  
[10, 11, 12] ← a[ 3 ]
```

二次元配列の要素の参照④'

```
a=[  
    [ 1, 2, 3 ],  
    [ 4, 5, 6 ],  
    [ 7, 8, 9 ],  
    [10, 11, 12 ]  
]
```

```
for j in a[ 0 ]:  
    print( j )
```

```
for j in a[ 1 ]:  
    print( j )
```

```
for j in a[ 2 ]:  
    print( j )
```

```
for j in a[ 3 ]:  
    print( j )
```

>python sample.py

1
2
3

a[0]

4
5
6

a[1]

7
8
9

a[2]

10
11
12

a[3]

二次元配列の要素の参照④"

```
a=[
    [ 1, 2, 3 ],
    [ 4, 5, 6 ],
    [ 7, 8, 9 ],
    [10, 11, 12]
]
```

```
for x in a:
    print( x )
    for j in x:
        print( j )
```

jには配列の値が代入されます

```
>python sample.py
```

```
[1, 2, 3]
```

```
1
2
3
```

x にa[0]が代入された場合

```
[4, 5, 6]
```

```
4
5
6
```

x にa[1]が代入された場合

```
[7, 8, 9]
```

```
7
8
9
```

x にa[2]が代入された場合

```
[10, 11, 12]
```

```
10
11
12
```

x にa[3]が代入された場合

二次元配列の要素の参照④"

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [ 10 , 11 , 12 ]  
]
```

```
for x in a:  
    print( x )  
    for j in x:  
        print( j )
```

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [ 10 , 11 , 12 ]  
]
```

```
for j in a[ 0 ]:  
    print( j )
```

```
for j in a[ 1 ]:  
    print( j )
```

```
for j in a[ 2 ]:  
    print( j )
```

```
for j in a[ 3 ]:  
    print( j )
```

二次元配列の要素の参照④'''

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [10 ,11 ,12 ]  
]
```

```
for x in a:  
    print( x )  
    for j in range(len(x)):  
        print( x[ j ] )
```

jには0,1,2が代入されます

```
>python sample.py
```

```
[1, 2, 3]
```

```
1  
2  
3
```

x にa[0]が代入された場合

```
[4, 5, 6]
```

```
4  
5  
6
```

x にa[1]が代入された場合

```
[7, 8, 9]
```

```
7  
8  
9
```

x にa[2]が代入された場合

```
[10, 11, 12]
```

```
10  
11  
12
```

x にa[3]が代入された場合

二次元配列の要素の参照④"と④"' 違いをよく理解して下さい

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [10 ,11 ,12 ]  
]
```

x にはa[0],a[1],a[2],a[3]が代入されます

```
for x in a:  
    print( x )  
    for j in x:  
        print( j )
```

jには配列の値が代入されます

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [10 ,11 ,12 ]  
]
```

```
for x in a:  
    print( x )  
    for j in range(len(x)):  
        print( x[ j ] )
```

jには0,1,2が代入されます

二次元配列の要素の参照⑤

```
a = [  
    [ 1 ],  
    [ 2, 3 ],  
    [ 4, 5, 6 ],  
    [ 7, 8, 9, 10 ]  
]
```

```
for i in range(len(a)):  
    for j in range(len(a[i])):  
        print( " a[" , i , "][" , j , "]" = " , a[ i ][ j ] )
```

配列の要素数が異なってもよい

```
>python sample.py
```

```
a[ 0 ][ 0 ] = 1
```

```
a[ 0 ]
```

```
a[ 1 ][ 0 ] = 2
```

```
a[ 1 ][ 1 ] = 3
```

```
a[ 1 ]
```

```
a[ 2 ][ 0 ] = 4
```

```
a[ 2 ][ 1 ] = 5
```

```
a[ 2 ][ 2 ] = 6
```

```
a[ 2 ]
```

```
a[ 3 ][ 0 ] = 7
```

```
a[ 3 ][ 1 ] = 8
```

```
a[ 3 ][ 2 ] = 9
```

```
a[ 3 ][ 3 ] = 10
```

```
a[ 3 ]
```

二次元配列の要素の参照⑤'

```
a = [  
    [ 1 ],  
    [ 2, 3 ],  
    [ 4, 5, 6 ],  
    [ 7, 8, 9, 10 ]  
]  
  
for x in a:  
    print( x )  
    print( len(x) )
```

```
>python sample.py
```

[1]

1

a[0]

[2, 3]

2

a[1]

[4, 5, 6]

3

a[2]

[7, 8, 9, 10]

4

a[3]

変数 x には a[0], a[1], a[2], a[3] が代入されていく

二次元配列の要素の参照⑤"

```
a = [  
    [ 1 ],  
    [ 2, 3 ],  
    [ 4, 5, 6 ],  
    [ 7, 8, 9, 10 ]  
]
```

```
for x in a:  
    for j in range(len(x)):  
        print( x[ j ] )
```

>python sample.py

1

a[0]

2

3

a[1]

4

5

6

a[2]

7

8

9

10

a[3]

変数 x には a[0], a[1], a[2], a[3] が代入されていく

二次元配列の要素の参照⑤'''

```
a = [  
    [ 1 ],  
    [ 2, 3 ],  
    [ 4, 5, 6 ],  
    [ 7, 8, 9, 10 ]  
]
```

```
for i in a:  
    for j in i:  
        print( j )
```

>python sample.py

1

a[0]

2

3

a[1]

4

5

6

a[2]

7

8

9

10

a[3]

変数 i には a[0], a[1], a[2], a[3] が代入されていく
変数 j には a[i]の要素が代入されていく

二次元配列の要素への代入①

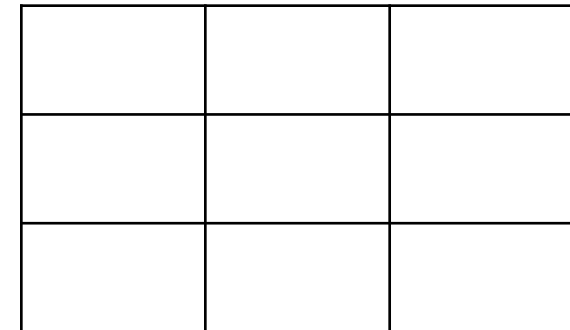
```
a=[0]*3
```

```
a[ 0 ] = [0]*3
```

```
a[ 1 ] = [0]*3
```

```
a[ 2 ] = [0]*3
```

3×3の要素(値は0)を持つ
二次元配列を宣言




```
count = 1
```

```
for i in range(3):
```

```
    for j in range(3):
```

```
        a[ i ][ j ] = count
```

```
        count += 1
```

代入式

```
print( a )
```

```
>python sample.py
```

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

二次元配列の要素への代入①'

二次元配列の宣言の方法

```
a=[0]*3
```

要素数を3個と指定

```
count = 1
```

```
for i in range(3):
```

```
    a[ i ] = [0]*3
```

```
    for j in range(3):
```

```
        a[ i ][ j ] = count
```

```
        count += 1
```

```
print( a )
```

```
a=[]
```

配列の要素を追加

```
count = 1
```

```
for i in range(3):
```

```
    a.append([])
```

```
    for j in range(3):
```

```
        a[ i ].append(count)
```

```
        count += 1
```

```
print( a )
```

二次元配列の要素への代入①"

二次元配列の宣言の方法

```
a=[]
```

```
count = 1
```

```
for i in range(3):
```

```
    a.append([])
```

```
    a[ i ] =[0]*3
```

```
    for j in range(3):
```

```
        a[ i ][ j ] = count
```

```
        count += 1
```

```
print( a )
```

要素数を3個と指定

```
>python sample.py
```

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




二次元配列の要素への代入②

```
import random  
a=[]
```

```
for i in range(3):  
    a.append([])  
    for j in range(3):  
        a[ i ].append( random.randint(0, 10) )  
print( a )
```

0から9の乱数を代入



```
>python sample.py  
[[2, 7, 5], [7, 2, 9], [2, 7, 4]]
```

二次元配列の要素への代入③

```
a=[1,2,3]

b=[]
for i in range(3):
    b.append([])
    for j in range(3):
        b[ i ].append( a[ j ] )

print( b )
```

配列を追加

```
>python sample.py
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
```

二次元配列の要素への代入④

```
a=[1,2,3]
```

```
b=[]
```

```
for i in range(3):
```

```
    b.append([])
```

```
    b[i] = a
```

```
    print( b )
```

```
print( b )
```

b[0],b[1],b[2]にaを代入(?)

```
>python sample.py
```

```
[[1, 2, 3]]
```

```
[[1, 2, 3], [1, 2, 3]]
```

```
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
```

```
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
```

二次元配列の要素への代入④'

```
a=[1,2,3]
```

```
b=[]
```

```
for i in range(3):
```

```
    b.append([])
```

```
    b[i] = a
```

b[0],b[1],b[2]にaを代入(?)

```
print( b )
```

配列aの要素を変えるとbも変わることに注意！

```
a[0]=100
```

```
print( b )
```

```
>python sample.py
```

```
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
```

```
[[100, 2, 3], [100, 2, 3], [100, 2, 3]]
```

(復習) 注意: 配列の要素の参照例

「配列のコピー」にはなりません！

```
a=[4,2,1,6,7]
```

```
x=a
```

```
print( a )
```

```
print( x )
```

```
a[0]=10
```

```
print( a )
```

```
print( x )
```

配列xにコピー？

配列aだけ変更

しかし、x も変っている！

```
>python sample.py
```

```
[4, 2, 1, 6, 7]
```

```
[4, 2, 1, 6, 7]
```

```
[10, 2, 1, 6, 7]
```

```
[10, 2, 1, 6, 7]
```

(復習) 注意': 配列の要素の参照例

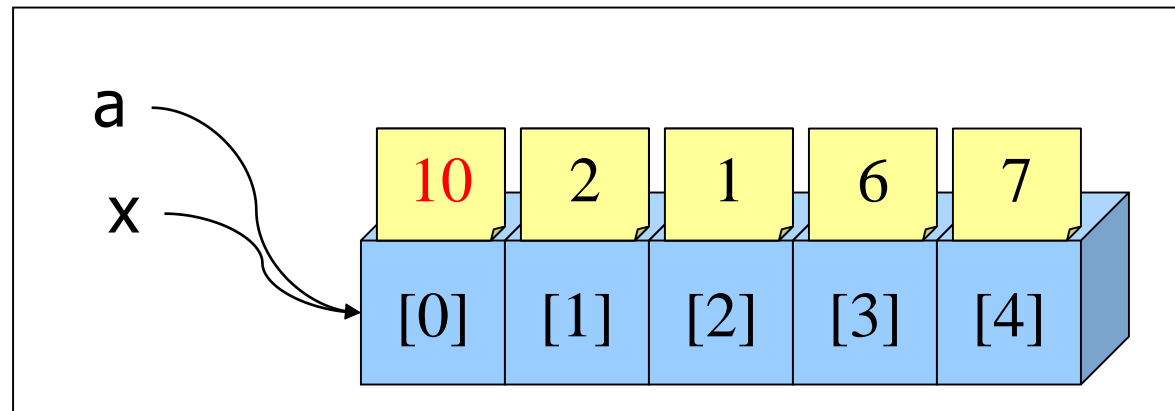
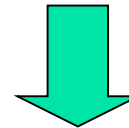
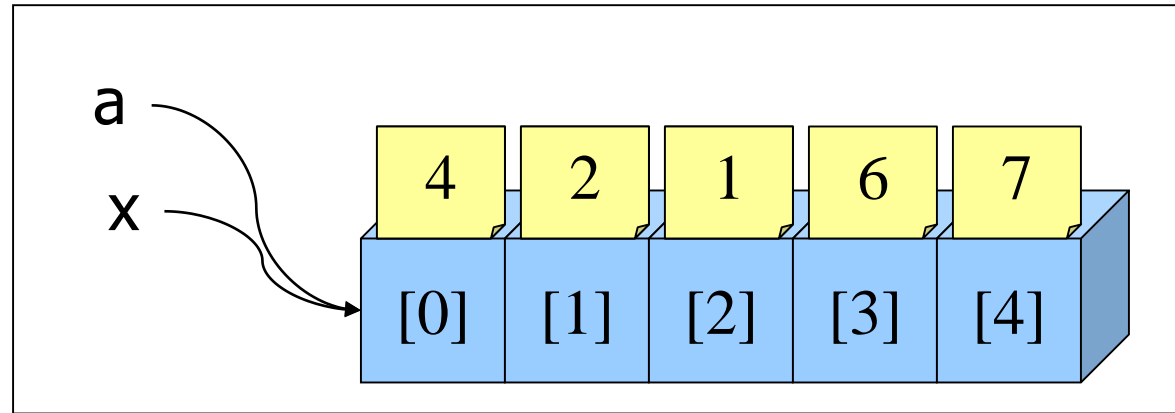
```
a=[4,2,1,6,7]
```

```
x=a
```

```
print( a, x )
```

```
a[0]=10
```

```
print( a, x )
```



二次元配列の要素への代入⑤

```
a=[]  
  
for i in range(3):  
    a.append([])  
    for j in range(3):  
        work = int(input())  
        a[ i ].append( work )  
  
print( a )
```

キーボード入力

>python sample.py

3
4
5
6
7
1
2
3
4

[[3, 4, 5], [6, 7, 1], [2, 3, 4]]

二次元配列のコピー①

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [ 10 , 11 , 12 ]  
]
```

配列 b の宣言

```
b = []  
for i in range(len(a)):  
    b.append([])  
    for j in range(len(a[i])):  
        b[ i ].append( a[i][j] )
```

```
print( b )
```

```
>python sample.py  
[[1, 2, 3], [4, 5, 6], [7, 8, 9],  
 [10, 11, 12]]
```


二次元配列のコピー①'

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [ 10 , 11 , 12 ]  
]
```

```
b = []
```

```
for i in range(len(a)):
```

```
    b.append([])
```

```
    b[i] = a[i]
```

```
print( b )
```

配列 b の宣言

b[i] に a[i] を代入(?)

```
>python sample.py  
[[1, 2, 3], [4, 5, 6], [7, 8, 9],  
[10, 11, 12]]
```

二次元配列のコピー①

```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [10 ,11 ,12 ]  
]
```

```
b = []
```

```
for i in range(len(a)):
```

```
    b.append([])
```

```
    b[i] = a[i]
```

```
print( b )
```

```
a[0][0] = 100
```

```
print( b )
```

配列 b の宣言

b[i] に a[i] を代入(?)

配列aの要素を変えるとbも変わることに注意！

```
>python sample.py
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
```

```
[[100, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
```

二次元配列のコピー②

```
a=[
    [ 1 , 2 , 3 ],
    [ 4 , 5 , 6 ],
    [ 7 , 8 , 9 ],
    [ 10 , 11 , 12 ]
]
```

```
for i in range(len(b)):
    for j in range(len(b[i])):
        print( b[i][j] , end=" " )
    print()
```

```
b = [0]*len(a[0])
for i in range(len(b)):
    b[i]=[0]*len(a)
```

3 × 4の配列の作成

```
for i in range(len(a)):
    for j in range(len(a[i])):
```

行列の転置

```
>python sample.py
1 4 7 10
2 5 8 11
3 6 9 12
```

練習問題の答えです…



ファイル操作

ファイルからの読み込み, 書き込み

ファイルからの読み込み①

読み込みたいファイル名を記述

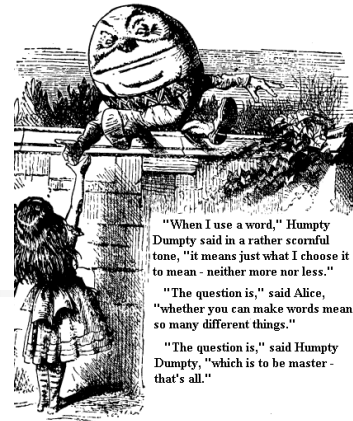
```
f = open( "ファイル名" )  
for line in f:  
    print( line )  
f.close()
```

文字列変数 line に
ファイルから一行読
み込まれる

最後に「閉じる」こと
を忘れずに

ファイルの末尾まで読み込
まれるとfor文から抜け出る

ファイルからの読み込み②



```
f = open("HumptyDumpty.txt")
for line in f:
    print( line )
f.close()
```

ファイルから一行ずつ読み込む

一行出力

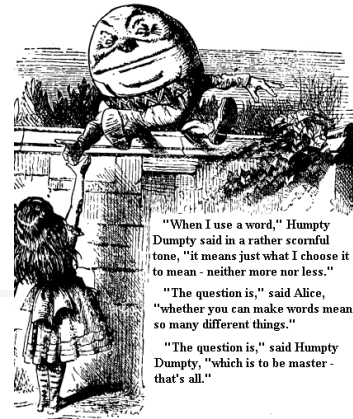
HumptyDumpty.txt

Humpty Dumpty sat on a wall.
Humpty Dumpty had a great fall.
All the king's horses and all the king's men
Couldn't put Humpty together again.

```
>python sample.py
Humpty Dumpty sat on a wall.
Humpty Dumpty had a great fall.
All the king's horses and all the king's men
Couldn't put Humpty together again.
```

一行空いてしまう→改行が含まれている

ファイルからの読み込み③



```
f = open("HumptyDumpty.txt")
for line in f:
    print( line.strip() )
f.close()
```

文字列の両端から改行
、空白文字を削除

```
>python sample.py
Humpty Dumpty sat on a wall.
Humpty Dumpty had a great fall.
All the king's horses and all the king's men
Couldn't put Humpty together again.
```

ファイルからの読み込み④

読み込みたいファイル名を記述

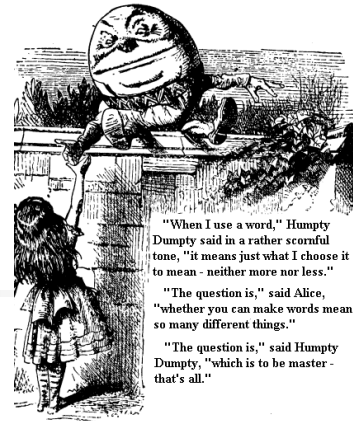
```
with open("ファイル名") as f:  
    for line in f:  
        print( line )
```

close()は不要

文字列変数 line に
ファイルから一行読
み込まれる

ファイルの末尾まで読み込
まれるとfor文から抜け出る

ファイルからの読み込み④



```
with open("HumptyDumpty.txt") as f:  
    for line in f:  
        print( line.strip() )
```

```
>python sample.py  
Humpty Dumpty sat on a wall.  
Humpty Dumpty had a great fall.  
All the king's horses and all the king's men  
Couldn't put Humpty together again.
```

ファイルからの読み込み⑤

```
sum = 0
f = open( "file.txt" )
for line in f:
    x = int( line.strip() )
    print( x , end=" " )
    sum += x
f.close()
print( "¥n 合計:" , sum )
```

file.txt

1
2
3
4
5
6
7
8
9
10

改行

```
>python sample.py
1 2 3 4 5 6 7 8 9 10
合計:55
```

ファイルからの読み込み⑥

```
sum = 0
f = open( "file1.txt" )
for line in f:
    x = line.strip().split(",")
    print( x[ 0 ] , x[ 1 ] )
    sum += int( x[ 1 ] )
print( " 合計:" , sum )
```

file1.txt

A,1
B,2
C,3
D,4
E,5
F,6
G,7
H,8
I,9
J,10

「,」で「A,1」「B,2」「C,3」を分割
x[0]にはA,B,C,D,...,I,J
x[1]には1,2,3,4,...,9,10

ファイルからの読み込み⑥'

```
sum = 0
f = open( "file1.txt" )
for line in f:
    x = line.strip().split(",")
    print( x[ 0 ] , x[ 1 ] )
    sum += int( x[ 1 ] )
print( " 合計:" , sum )
```

「,」で分割し, x[0]とx[1]に代入する

```
>python sample.py
A 1
B 2
C 3
D 4
E 5
F 6
G 7
H 8
I 9
J 10
合計:55
```



ファイルへの書き込み①

```
f = open( "text.txt" , "w" )  
f.write( "python¥n" )  
f.write( "ruby¥n" )  
f.write( "java¥n" )  
f.write( "C++¥n" )  
f.close()
```

文字列で書き込む

```
>python sample.py
```

```
>type text.txt  
python  
ruby  
java  
C++
```

type

ファイルの内容を見るコマンド



ファイルへの書き込み②

```
f = open( "text.txt" , "w" )  
for i in range(10):  
    f.write( str(i)+"¥n" )  
f.close()
```

文字列で書き込む

type
ファイルの内容を見るコマンド

```
>python sample.py
```

```
>type text.txt
```

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

ファイルへの書き込み③

```
f = open( "text.txt" , "w" )  
f.write( "python¥n" )  
f.write( "ruby¥n" )  
f.write( "java¥n" )  
f.write( "C++¥n" )  
f.close()
```

text.txt という名前のファイル(書き込み用)を準備

text.txt に文字列で書き込む

ファイルを閉じる(忘れずに！)

```
open( "ファイル名" , "モード" )
```

モード(省略した場合は「r」)
w 書き込み用
r 読み込み用
a 追加書き込み用

ファイルへの書き込み③

```
s = open( "text.txt" , "w" )  
f = open("HumptyDumpty.txt")  
for line in f:  
    s.write( line )  
s.close()  
f.close()
```

f 読み込み用

s 書き込み用

```
>python sample.rb
```

```
>type text.txt
```

```
Humpty Dumpty sat on a wall.
```

```
Humpty Dumpty had a great fall.
```

```
All the king's horses and all the king's men  
Couldn't put Humpty together again.
```




練習問題

練習問題①～⑤



練習問題①

- 配列 a を 3×3 の二次元配列とする
- 二重ループを用いて下記のような要素を持つ配列を作成しなさい

```
a=[  
    [ 1 , 0 , 0 ],  
    [ 0 , 1 , 0 ],  
    [ 0 , 0 , 1 ]  
]
```

```
a=[  
    [ 0 , 0 , 1 ],  
    [ 0 , 1 , 0 ],  
    [ 1 , 0 , 0 ]  
]
```

```
>python 11-1-1.py  
[[1, 0, 0], [0, 1, 0], [0, 0, 1]]
```

```
>python 11-1-2.py  
[[0, 0, 1], [0, 1, 0], [1, 0, 0]]
```



練習問題②

- 配列 a を 3×3 の二次元配列とする
- 二重ループを用いて下記のような要素を持つ配列を作成しなさい(スライド54ページを参照)

```
a=[  
    [ 0 , 2 , 0 ],  
    [ 0 , 5 , 0 ],  
    [ 0 , 8 , 0 ]  
]
```

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

```
>python 11-2-1.py  
[[0, 2, 0], [0, 5, 0], [0, 8, 0]]
```

```
>python 11-2-2.py  
[[1, 2, 0], [4, 0, 6], [0, 8, 9]]
```



練習問題③

- 二次元配列 `a` の転置行列を出力するプログラムを作成しなさい

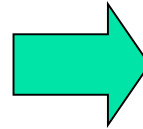
```
a=[  
    [ 1 , 2 , 3 ],  
    [ 4 , 5 , 6 ],  
    [ 7 , 8 , 9 ],  
    [ 10 , 11 , 12 ]  
]
```

```
>python 11-3.py  
1 4 7 10  
2 5 8 11  
3 6 9 12
```

行列の転置のプログラム(ヒント)

配列a

$\langle 0,0 \rangle$	$\langle 0,1 \rangle$	$\langle 0,2 \rangle$
$\langle 1,0 \rangle$	$\langle 1,1 \rangle$	$\langle 1,2 \rangle$
$\langle 2,0 \rangle$	$\langle 2,1 \rangle$	$\langle 2,2 \rangle$
$\langle 3,0 \rangle$	$\langle 3,1 \rangle$	$\langle 3,2 \rangle$



配列a^t

$\langle 0,0 \rangle$	$\langle 1,0 \rangle$	$\langle 2,0 \rangle$	$\langle 3,0 \rangle$
$\langle 0,1 \rangle$	$\langle 1,1 \rangle$	$\langle 2,1 \rangle$	$\langle 3,1 \rangle$
$\langle 0,2 \rangle$	$\langle 1,2 \rangle$	$\langle 2,2 \rangle$	$\langle 3,2 \rangle$



練習問題④

- 二次元配列 a , b の和の行列 c と差の行列 d を求めるプログラムを二重ループを用いて書きなさい

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

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

```
>python 11-4-1.py
```

```
c --->
```

```
[[10, 10, 10], [10, 10, 10], [10, 10, 10]]
```

```
d --->
```

```
[[ -8, -6, -4], [-2, 0, 2], [4, 6, 8]]
```



練習問題④

- (さらに頑張れる人へ) 二次元配列 a , b の積の行列 e を求めるプログラムを三重ループ(二重ループでもよい)を用いて書きなさい

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

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

```
>python 11-4-2.py  
e --->  
30 24 18  
84 69 54  
138 114 90
```

念のため：行列の積

$e[i][k]$ の計算

配列a

i 行目

$\langle 0, 0 \rangle$	$\langle 0, 1 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 3 \rangle$
$\langle 1, 0 \rangle$	$\langle 1, 1 \rangle$	$\langle 1, 2 \rangle$	$\langle 1, 3 \rangle$
$\langle 2, 0 \rangle$	$\langle 2, 1 \rangle$	$\langle 2, 2 \rangle$	$\langle 2, 3 \rangle$
$\langle 3, 0 \rangle$	$\langle 3, 1 \rangle$	$\langle 3, 2 \rangle$	$\langle 3, 3 \rangle$

\times

配列b

k 列目

$\langle 0, 0 \rangle$	$\langle 0, 1 \rangle$	$\langle 0, 2 \rangle$	$\langle 0, 3 \rangle$
$\langle 1, 0 \rangle$	$\langle 1, 1 \rangle$	$\langle 1, 2 \rangle$	$\langle 1, 3 \rangle$
$\langle 2, 0 \rangle$	$\langle 2, 1 \rangle$	$\langle 2, 2 \rangle$	$\langle 2, 3 \rangle$
$\langle 3, 0 \rangle$	$\langle 3, 1 \rangle$	$\langle 3, 2 \rangle$	$\langle 3, 3 \rangle$

$$e[i][k] = a[i][0] * b[0][k] + a[i][1] * b[1][k] + a[i][2] * b[2][k] + a[i][3] * b[3][k]$$



行列の積のプログラム(ヒント)

三重ループ

```
for i in range(len(a)):
    for k in range(len(a[i])):
        sum = 0
        for j in range(len(b)):
            sum に a[i][j] と b[j][k] の積を足す
        e[i][k] に sum を代入
```



練習問題⑤

- HumptyDumpty.txtの内容が最後の行から印字されるようにサンプルプログラムを改良しなさい

```
>python 11-5.py  
Couldn't put Humpty together again.  
All the king's horses and all the king's men  
Humpty Dumpty had a great fall.  
Humpty Dumpty sat on a wall.
```

- HumptyDumpty.txtはプログラムと同じフォルダーに置いて下さい



練習問題⑤

```
f = open("HumptyDumpty.txt")
for line in f:
    print( line.strip() )
f.close()
```

プログラムを実行した場合

```
>python sample.py
Humpty Dumpty sat on a wall.
Humpty Dumpty had a great fall.
All the king's horses and all the king's men
Couldn't put Humpty together again.
```

HumptyDumpty.txt

```
Humpty Dumpty sat on a wall.
Humpty Dumpty had a great fall.
All the king's horses and all the king's men
Couldn't put Humpty together again.
```



練習問題⑤(ヒント)

- ファイルから、一行ずつ読み込んだ文字列を配列に格納する
- 読み込み終了後、配列の最後の要素から印字する



練習問題

- 練習問題①から⑤を(できるだけ)(頑張って)行ないなさい.
- プログラムと実行結果をワープロに貼り付けて, keio.jp から提出して下さい.