



# プログラミング言語 第十回

---

担当: 篠沢 佳久  
栗原 聡

2019年 6月24日



# 本日の内容

---

- 一次元配列の復習
- 多重ループ
- ネスト(入れ子)構造
  
- 練習問題①～⑤

# ネスト(入れ子)

- ネストする: 入れ子にすること



箱根の十二卵(田中一幸氏作) 左端は実際の鶏卵のLL玉ほど。右端は13番目のヒヨコ

<http://dadandmam.whitesnow.jp/moiwayama/?p=8348>



# 配列の復習

---

一次元配列と繰り返し



# 配列の宣言①

---

- 要素が分かっている場合
  - 配列名 = [ 値1, 値2, ... , 値n ]
- 要素が分かっていない場合
  - 配列名 = []
  - appendを用いて要素を追加
- 要素数のみ分かっている場合
  - 配列名=[値]\*要素数
  - 配列の要素に値を代入

## 配列の宣言②

要素が分かっている場合

$a = [4, 6, 7, 9, 10]$

	a	
0	4	a[ 0 ]
1	6	a[ 1 ]
2	7	a[ 2 ]
3	9	a[ 3 ]
4	10	a[ 4 ]

## 配列の宣言②

要素数が分かっている場合

a= [0]\*5

a=[0,0,0,0,0]

a[0]=4

a[1]=6

a[2]=7

a[3]=9

a[4]=10

要素が分かっていない場合

a= []

aが配列と宣言

a.append(4)

a.append(6)

a.append(7)

a.append(9)

a.append(10)

# 配列の要素の参照方法①

## 要素番号を用いて一つずつ取り出す

forを用いる場合

	a
0	0
1	2
2	4
3	6
4	8

a[ 0 ]

a[ 1 ]

a[ 2 ]

a[ 3 ]

a[ 4 ]

この順番に要素を取り出したい

```
for i in range(len(配列)):  
    配列[ i ]の処理
```

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

i は0,1,2,3,4 と代入されるため  
a[ 0 ], a[ 1 ], a[ 2 ], a[ 3 ], a[ 4 ]  
となる





## 配列の要素の参照方法②

### 要素を直接一つずつ取り出す

```
a = [1,3,5,7,9]
for x in a:
    print( x )
```

xには1,3,5,7,9と代入される

```
for x in [1,3,5,7,9]:
    print( x )
```

```
>python sample.py
1
3
5
7
9
```



# 一次元配列のプログラム例

```
name = [ "A" , "B" , "C" , "D" , "E" ]
```

```
test = [ 85 , 60 , 5 , 100 , 50 ]
```

配列name

文字列型

	name
0	A
1	B
2	C
3	D
4	E

配列 test

整数型

	test
0	85
1	60
2	5
3	100
4	50



# 配列の要素への代入

```
name = [ "A" , "B" , "C" , "D" , "E" ]
```

```
test = [ 85 , 60 , 5 , 100 , 50 ]
```

```
name[ 3 ] = "d"
```

```
test[ 3 ] = 90
```

```
print( name )
```

```
print( test )
```

```
>python sample.py  
["A", "B", "C", "d", "E"]  
[85, 60, 5, 90, 50]
```



## 最後の要素への追加①

```
name = [ "A" , "B" , "C" , "D" , "E" ]  
test = [ 85 , 60 , 5 , 100 , 50 ]
```

```
name.append( "F" )  
test.append( 70 )
```

```
print( name )  
print( test )
```

```
>python sample.py  
["A", "B", "C", "D", "E", "F"]  
[85, 60, 5, 100, 50, 70]
```

## 最後の要素への追加②

```
name.append( "F" )
```

配列name

文字列型

	name
0	A
1	B
2	C
3	D
4	E
5	F

```
test.append( 70 )
```

配列 test

整数型

	test
0	85
1	60
2	5
3	100
4	50
5	70



# 平均点を求める①

```
name = [ "A" , "B" , "C" , "D" , "E" ]  
test = [ 85 , 60 , 5 , 100 , 50 ]
```

```
sum = 0
```

```
for i in range( len(test) ):  
    sum += test[ i ]
```

for を用いた方法

iは0,1,2,3,4

→ test[0],test[1],test[2],test[3],test[4]

```
print( "Average --> " , int( sum / len(test) ) )
```

```
>python sample.py  
Average --> 60
```

## 平均点を求める②

```
name = [ "A" , "B" , "C" , "D" , "E" ]  
test = [ 85 , 60 , 5 , 100 , 50 ]
```

```
sum = 0
```

```
i = 0
```

```
while i < len(test):
```

```
    sum += test[i]
```

```
    i += 1
```

```
print( "Average --> " , int( sum / len(test) ) )
```

while を用いた方法

iは0,1,2,3,4

→ test[0],test[1],test[2],test[3],test[4]

```
>python sample.py  
Average --> 60
```



## 平均点を求める③

```
name = [ "A" , "B" , "C" , "D" , "E" ]  
test = [ 85 , 60 , 5, 100 , 50 ]
```

```
sum = 0
```

配列の要素を直接参照

```
for x in test:
```

xは85,60,5,100,50

```
    sum += x
```

```
print( "Average --> " , int( sum / len(test) ) )
```

前々頁との違いに注意して下さい

```
>python sample.py  
Average --> 60
```



# 平均点未満の名前を出力①

```
name = [ "A" , "B" , "C" , "D" , "E" ]
```

```
test = [ 85 , 60 , 5 , 100 , 50 ]
```

```
sum = 0
```

```
for x in test:
```

```
    sum += x
```

```
average = int( sum / len(test) )
```

```
print( "平均点未満は..." )
```

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

```
    if average > test[ i ]:
```

```
        print( name[ i ] , ": " , test[ i ] , "点" )
```

平均点を求める

```
>python sample.py  
平均点未満は...
```

```
C: 5点
```

```
E: 50点
```

## 平均点未満の名前を出力②

```
name = [ "A" , "B" , "C" , "D" , "E" ]
```

```
test = [ 85 , 60 , 5 , 100 , 50 ]
```

```
sum = 0
```

```
for x in test:
```

```
    sum += x
```

```
average = int( sum / len(test) )
```

```
print( "平均点未満は..." )
```

```
for i , x in enumerate(test):
```

```
    if average > x:
```

```
        print( name[ i ] , ": " , x , "点" )
```

平均点を求める

```
>python sample.py  
平均点未満は...
```

```
C: 5点  
E: 50点
```

iは0,1,2,3,4

xは85,60,5,100,50



# 要素と要素番号を同時に取り出す方法

```
配列名=[値1,値2,...,値n]
```

```
for i , x in enumerate(配列名):  
    print( i , x )
```

x に値1,値2,...値nが代入される

```
a=[1,3,5,7,9]
```

```
for i , x in enumerate(a):  
    print( i , x )
```

i=0,1,2,...,n-1が代入される

i=0,1,2,3,4

x に1,3,5,7,9が代入される



# 最高点とその名前を出力①

```
name = [ "A" , "B" , "C" , "D" , "E" ]  
test = [ 85 , 60 , 5, 100 , 50 ]
```

```
max = 0
```

```
for i in range(1,len(test)):
```

```
    if test[ max ] < test[ i ]:
```

```
        max = i
```

```
print( "最高点は" , name[ max ] , " の " , test[ max ] , "点です" )
```

変数max

最高点の要素番号を格納する  
初期値として(仮に)0としておく

現在の最高点と比較

```
>python sample.py  
最高点はD の 100点です
```

## 最高点とその名前を出力②

```
name = [ "A" , "B" , "C" , "D" , "E" ]  
test = [ 85 , 60 , 5, 100 , 50 ]
```

```
top = 0
```

```
max = test[top]
```

```
for i , x in enumerate(test):
```

```
    if max < x:
```

```
        max = x
```

```
        top = i
```

```
print( "最高点は" , name[ top ] , " の " , max , "点です" )
```

変数top

最高点の番号, 初期値として0とする

変数max

最高点を格納する

初期値として85(Aの点数)とする

現在の最高点と比較

```
>python sample.py  
最高点はD の 100点です
```

# 最低点とその名前を出力

```
name = [ "A" , "B" , "C" , "D" , "E" ]  
test = [ 85 , 60 , 5 , 100 , 50 ]
```

```
min = 0
```

```
for i in range(1,len(test)):
```

```
    if test[ min ] > test[ i ]:
```

```
        min = i
```

```
print( "最低点は" , name[ min ] , " の " , test[ min ] , "点です" )
```

min

最低点の要素番号を格納する

現在の最低点と比較

```
>python sample.py  
最低点はC の 5点です
```

# 検索①

名前に対応した点数を求める

```
name = [ "A" , "B" , "C" , "D" , "E" ]  
test = [ 85 , 60 , 5 , 100 , 50 ]
```

```
search = "B" ← Bの点数を検索  
for i in range(len(name)):  
    if name[ i ] == search:  
        print( search , " の点は " , test[ i ] , " 点です" )  
        break
```

```
>python sample.py  
B の点は 60点です
```

# 配列の要素の検索

```
if name[ i ] == search:
```

配列name

	name
0	A
1	B
2	C
3	D
4	E

"B" == search

search と一致した場合, breakにより  
forのループから抜け出るため, 以降  
は照合しない



## 検索②

得点に対応した点数を求める

```
name = [ "A" , "B" , "C" , "D" , "E" ]  
test = [ 85 , 60 , 5 , 100 , 50 ]
```

```
search = 60
```

← 60点の名前を検索

```
for i in range(len(test)):  
    if test[ i ] == search:  
        print( search , " 点は " , name[ i ] , "です" )  
        break
```

```
>python sample.py  
60 点は B です
```



# コマンドライン引数

---

# コマンドライン引数①

- Pythonプログラムにデータを渡すことができます

```
>python sample.py 2 3  
2 + 3 = 5  
>python sample.py 8 2  
8 + 2 = 10
```

引数

例えば、プログラム名の後に二つの整数値も書き、これらの値をプログラムで読み込み、合計するといったことができます

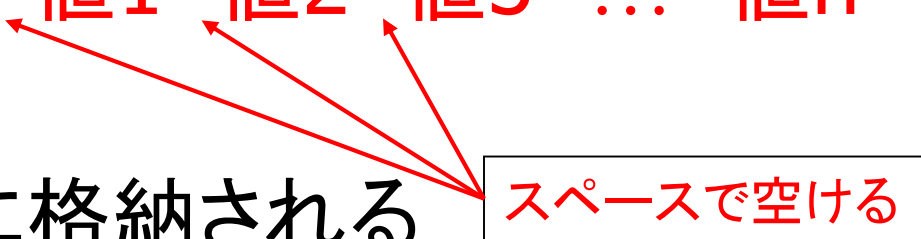
## コマンドライン引数②

スペースで空ける

- python プログラム 値1 値2 値3
  - 値1, 値2, ... を引数と呼ぶ
- 値1は sys.argv[1] に格納される
- 値2は sys.argv[2] に格納される
- 値3は sys.argv[3] に格納される
- sys.argv[1], sys.argv[2], sys.argv[3]は文字列型
- sys.argv[0]にはプログラム名が入ります
- import sysを忘れずに書く



## コマンドライン引数③

- python プログラム 値1 値2 値3 ... 値n
  - 値1は `sys.argv[1]` に格納される
  - 値2は `sys.argv[2]` に格納される
  - 値nは `sys.argv[n]` に格納される
  - 引数の個数は `len(sys.argv)`
- 

# コマンドライン引数④

```
import sys
print( sys.argv[ 0 ] )
print( sys.argv[ 1 ] )
print( sys.argv[ 2 ] )
print( sys.argv[ 3 ] )
print( len(sys.argv) )
```

```
>python sample.py 2 4 6
sample.py
2
4
6
4
```

len(sys.argv)=4

文字列型

配列sys.argv

	sys.argv
0	sample.py
1	2
2	4
3	6

自動的に配列sys.argvに引数は代入される

# コマンドライン引数: プログラム

```
import sys
print( sys.argv[1] , "+" , sys.argv[2] , "=" , end=" " )
print( int( sys.argv[1] ) + int( sys.argv[2] ) )
```

文字列型のため型変換(整数)が必要

```
>python sample.py 3 5
3 + 5 = 8
```

	sys.argv
0	sample.py
1	3
2	5

# コマンドライン引数: プログラム

```
import sys
print( sys.argv[1] , "+" , sys.argv[2] , "=" , end=" " )
print( int( sys.argv[1] ) + int( sys.argv[2] ) )
```

文字列型のため型変換(整数)が必要

```
>python sample.py 100 200
100 + 200 = 300
```

	sys.argv
0	sample.py
1	100
2	200



# コマンドライン引数⑤

```
>python sample.py 2 3
```

```
2 + 3 = 5
```

```
>python sample.py 8 2
```

```
8 + 2 = 10
```

引数

配列sys.argv

文字列型

	sys.argv
0	sample.py
1	2
2	3

	sys.argv
0	sample.py
1	8
2	2



# 浮動小数点数にすると

```
import sys
print( sys.argv[1] , "+" , sys.argv[2] , "=" , end=" " )
print( float( sys.argv[1] ) + float( sys.argv[2] ) )
```

文字列型のため型変換(小数)が必要

```
>python sample.py 3 5
3 + 5 = 8.0
>python sample.py 100 200
100 + 200 = 300.0
```



## 引数を3個にした場合

```
import sys
print( sys.argv[1] , "+" , sys.argv[2] , "+" , sys.argv[3] , "=" , end=" " )
print( int( sys.argv[1] ) + int( sys.argv[2] ) + + int( sys.argv[3] ) )
```

```
>python sample.py 3 5 7
3 + 5 + 7 = 15
>python sample.py 100 200 300
100 + 200 + 300 = 600
```

# 検索③

引数で名前を入力し, 対応する点数を出力

```
import sys
name = [ "A" , "B" , "C" , "D" , "E" ]
test = [ 85 , 60 , 5 , 100 , 50 ]

search = sys.argv[1] ← コマンドライン引数

for i in range(len(name)):
    if name[ i ] == search:
        print( search , "の点は" , test[ i ] , "点です" )
        break
```

# 検索③の実行結果

文字列型

>python sample.py A  
A の点は 85点です

sys.argv[1] には"A" が代入

>python sample.py B  
B の点は 60点です

>python sample.py C  
C の点は 5点です

>python sample.py D  
D の点は 100点です

>python sample.py E  
E の点は 50点です

search = sys.argv[1]

search には "A" が代入

## 検索④

引数で点数を入力し, 対応する名前を出力

```
import sys
name = [ "A" , "B" , "C" , "D" , "E" ]
test = [ 85 , 60 , 5 , 100 , 50 ]

score = sys.argv[1] ← コマンドライン引数
val = int(score)
for i in range(len(test)):
    if test[ i ] == val:
        print( val , "点は" , name[ i ] , "です" )
        break
```

# 検索④の実行結果

文字列型

```
>python sample.py 100
```

100 点は D です

```
>python sample.py 85
```

85 点は A です

```
>python sample.py 50
```

50 点は E です

```
>python sample.py 35
```

sys.argv[ 1 ] には "100" が代入



```
score = sys.argv[ 1 ]
```

score には "100" が代入



```
val = int(score)
```

val には 整数100が代入

35点の名前は存在しない



# 二重ループ

---





# 一重ループ

$$y = x^2$$

```
for x in range(10):  
    y = x*x  
    print( x , ":" , y )
```

```
for x in range(0,10):  
    y = x*x  
    print( x , ":" , y )
```

```
> python sample.py  
0: 0  
1: 1  
2: 4  
3: 9  
4: 16  
5: 25  
6: 36  
7: 49  
8: 64  
9: 81
```



# 二重ループの必要性①

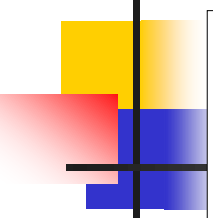
$$z = x^2 + y^2$$

$0 \leq x < 10, 0 \leq y < 10$  の範囲で値を求めるには？

$x=0$  の時,  $y$ の値を0から9まで変えて  $z$  を求める

```
x = 0
for y in range(0,10):
    z = x*x + y*y
    print( x , y , ":" , z )
```

x=1 の時, yの値を0から9まで変えて z を求める



```
x = 1
for y in range(0,10):
    z = x*x + y*y
    print( x , y , ":" , z )
```

以下同様にx=9 まで同じことを繰り返し z を求める

```
x = 9
for y in range(0,10):
    z = x*x + y*y
    print( x , y , ":" , z )
```

## 二重ループの必要性②

x = 0

```
for y in range(0,10):  
    z = x*x + y*y  
    print( x , y , ":" , z )
```

x = 1

```
for y in range(0,10):  
    z = x*x + y*y  
    print( x , y , ":" , z )
```

x = 9

```
for y in range(0,10):  
    z = x*x + y*y  
    print( x , y , ":" , z )
```

xの値も0から9まで一つ  
ずつ増やしていけばよい



# 二重ループ

①のループによって, xは0から9まで変わる



①のループ

```
for x in range(0,10):
```

```
    for y in range(0,10):
```

```
        z = x*x + y*y
```

```
        print( " x =" , x , "y =" , y , ": z=" , z )
```

②のループ



②のループによって, yは0から9まで変わる



# 二重ループの出力結果①

①のループ中  $x=0$  として  
②のループの処理を行なう

```
>python sample.py  
x = 0 y = 0: z= 0  
x = 0 y = 1: z= 1  
x = 0 y = 2: z= 4  
x = 0 y = 3: z= 9  
x = 0 y = 4: z= 16  
x = 0 y = 5: z= 25  
x = 0 y = 6: z= 36  
x = 0 y = 7: z= 49  
x = 0 y = 8: z= 64  
x = 0 y = 9: z= 81
```

①のループ中  $x=1$  として  
②のループの処理を行なう

```
x = 1 y = 0: z= 1  
x = 1 y = 1: z= 2  
x = 1 y = 2: z= 5  
x = 1 y = 3: z= 10  
x = 1 y = 4: z= 17  
x = 1 y = 5: z= 26  
x = 1 y = 6: z= 37  
x = 1 y = 7: z= 50  
x = 1 y = 8: z= 65  
x = 1 y = 9: z= 82
```



## 二重ループの出力結果②

①のループ中  $x=9$  として

②のループの処理を行ない終了する

```
x = 9 y = 0: z= 81
x = 9 y = 1: z= 82
x = 9 y = 2: z= 85
x = 9 y = 3: z= 90
x = 9 y = 4: z= 97
x = 9 y = 5: z= 106
x = 9 y = 6: z= 117
x = 9 y = 7: z= 130
x = 9 y = 8: z= 145
x = 9 y = 9: z= 162
```

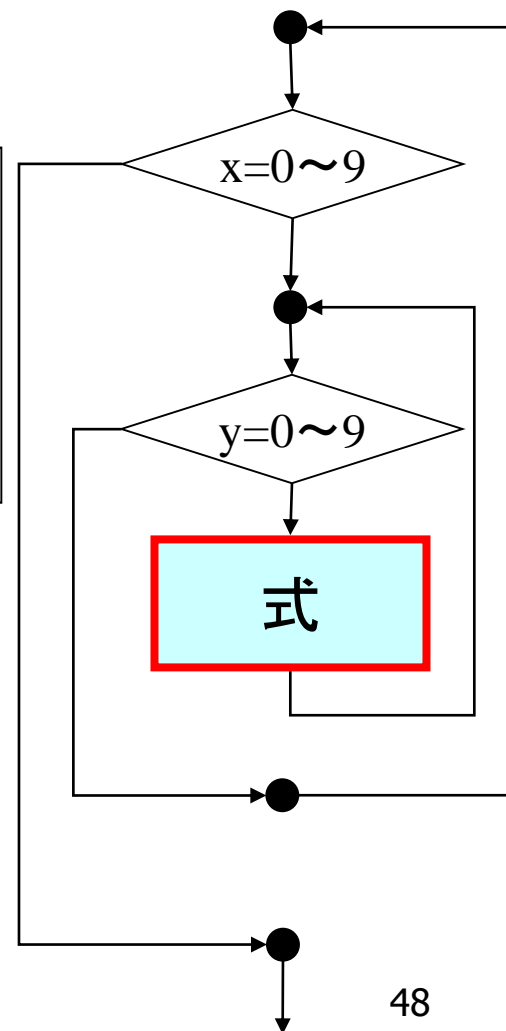
# 二重ループのまとめ①

forを用いた場合

```
for x in range(10):  
    for y in range(10):  
        式
```

```
for x in range(0,10):  
    for y in range(0,10):  
        式
```

外側と内側の制御変数は異なる名前にする  
(この場合は, x と y)

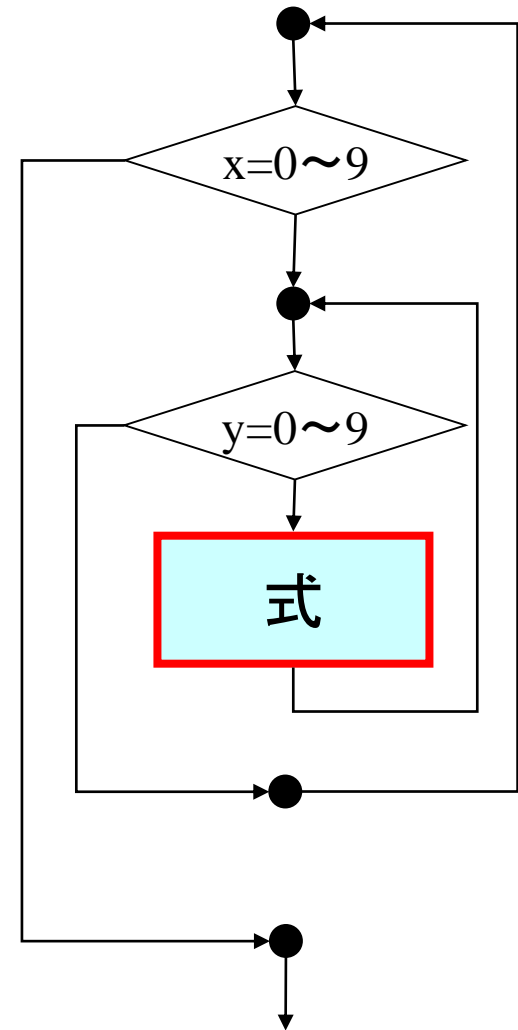




## 二重ループのまとめ②

while を用いた場合

```
x = 0
while x < 10:
    y = 0
    while y < 10:
        式
        y += 1
    x += 1
```

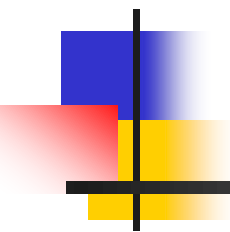


```
for x in range(0,10):  
    for y in range(0,10):  
        z = x*x + y*y  
        print( " x =" , x , "y =" , y , ": z=" , z )
```



while 文で書いた場合

```
x = 0  
while x < 10:  
    y = 0  
    while y < 10:  
        z = x * x + y * y  
        print( " x =" , x , "y =" , y , ": z=" , z )  
        y += 1  
    x += 1
```



## 二重ループの例

---



# 二重ループの例①

## 九九の表の表示プログラム

```
for x in range(1,10):  
    for y in range(1,10):  
        print( "{0} × {1}={2:2d} ".format( x , y , x * y ) , end="" )  
    print()
```

2桁で表示

改行なし

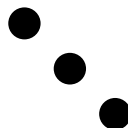
**x=1**

```
for y in range(1,10):  
    print( "{0} × {1}={2:2d} ".format( x , y , x * y ) , end="" )  
print()
```

**x=1,2,...9と変わっていく**

**x=2**

```
for y in range(1,10):  
    print( "{0} × {1}={2:2d} ".format( x , y , x * y ) , end="" )  
print()
```



**x=9**

```
for y in range(1,10):  
    print( "{0} × {1}={2:2d} ".format( x , y , x * y ) , end="" )  
print()
```

# 二重ループの例①

前頁の実行画面

xを1とし, yを1から9まで変える

```
C:\Windows\system32\cmd.exe
C:\Users\shino\Desktop>python sample.py
1×1= 1 1×2= 2 1×3= 3 1×4= 4 1×5= 5 1×6= 6 1×7= 7 1×8= 8 1×9= 9
2×1= 2 2×2= 4 2×3= 6 2×4= 8 2×5=10 2×6=12 2×7=14 2×8=16 2×9=18
3×1= 3 3×2= 6 3×3= 9 3×4=12 3×5=15 3×6=18 3×7=21 3×8=24 3×9=27
4×1= 4 4×2= 8 4×3=12 4×4=16 4×5=20 4×6=24 4×7=28 4×8=32 4×9=36
5×1= 5 5×2=10 5×3=15 5×4=20 5×5=25 5×6=30 5×7=35 5×8=40 5×9=45
6×1= 6 6×2=12 6×3=18 6×4=24 6×5=30 6×6=36 6×7=42 6×8=48 6×9=54
7×1= 7 7×2=14 7×3=21 7×4=28 7×5=35 7×6=42 7×7=49 7×8=56 7×9=63
8×1= 8 8×2=16 8×3=24 8×4=32 8×5=40 8×6=48 8×7=56 8×8=64 8×9=72
9×1= 9 9×2=18 9×3=27 9×4=36 9×5=45 9×6=54 9×7=63 9×8=72 9×9=81
```

xを9とし, yを1から9まで変える

## 二重ループの例②

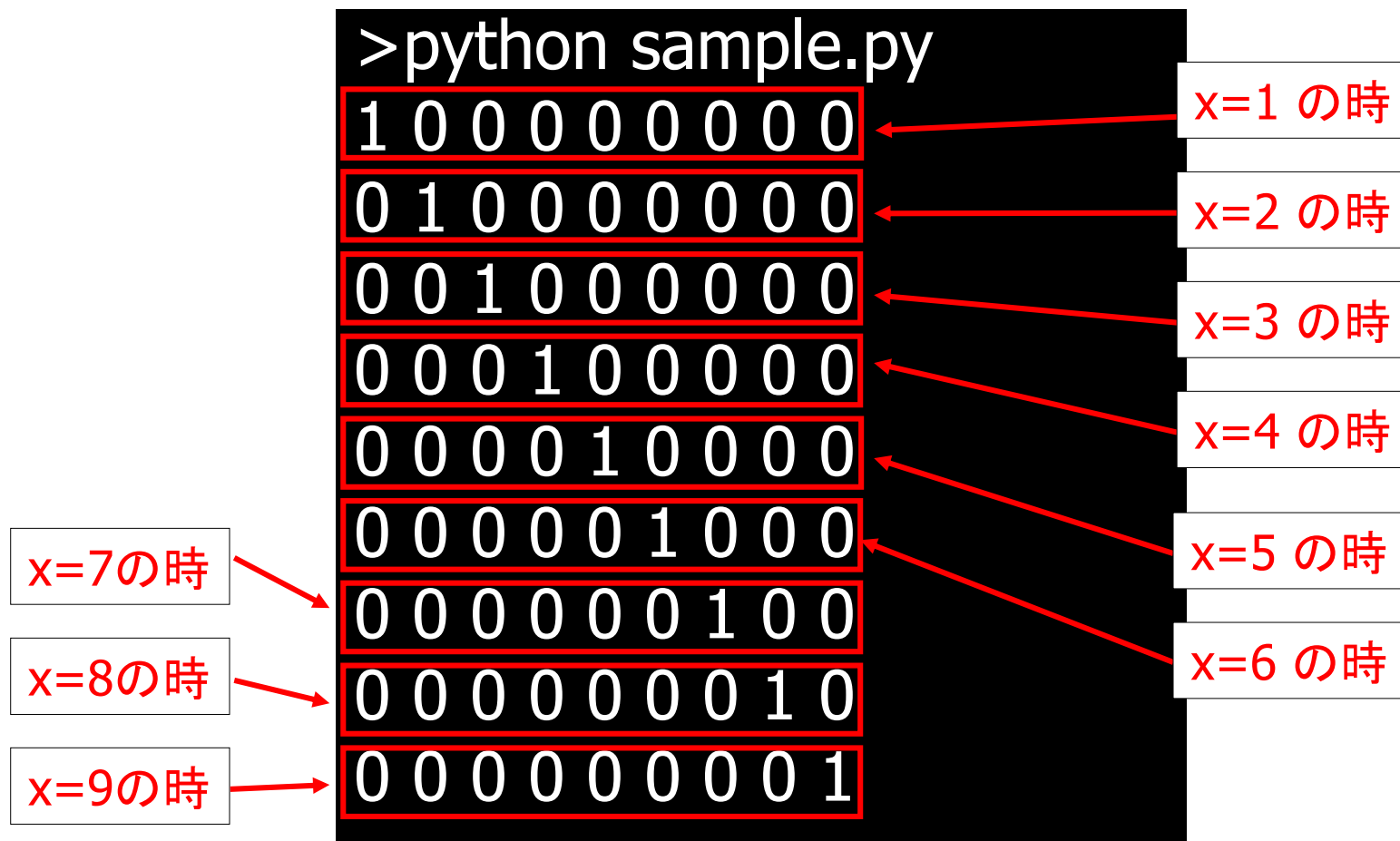
### 対角行列の表示プログラム

```
for x in range(0,10):  
    for y in range(0,10):  
        if x == y:  
            print( "1 " , end="" )  
        else:  
            print( "0 " , end="" )  
    print()
```

xとyの値が同じ→"1 "  
異なる場合は→"0 "

```
>python sample.py  
1 0 0 0 0 0 0 0 0  
0 1 0 0 0 0 0 0 0  
0 0 1 0 0 0 0 0 0  
0 0 0 1 0 0 0 0 0  
0 0 0 0 1 0 0 0 0  
0 0 0 0 0 1 0 0 0  
0 0 0 0 0 0 1 0 0  
0 0 0 0 0 0 0 1 0  
0 0 0 0 0 0 0 0 1
```

# 二重ループの例②の出力結果



>python sample.py

	1 0 0 0 0 0 0 0 0	x=1 の時
	0 1 0 0 0 0 0 0 0	x=2 の時
	0 0 1 0 0 0 0 0 0	x=3 の時
	0 0 0 1 0 0 0 0 0	x=4 の時
	0 0 0 0 1 0 0 0 0	x=5 の時
x=7の時	0 0 0 0 0 1 0 0 0	x=6 の時
x=8の時	0 0 0 0 0 0 1 0 0	
x=9の時	0 0 0 0 0 0 0 1 0	
	0 0 0 0 0 0 0 0 1	



# 二重ループの例②の出力結果

x=y=1の場合

x=y=2の場合

x=y=3の場合

x=y=4の場合

x=y=5の場合

```
>python sample.py
```

1 0 0 0 0 0 0 0 0

0 1 0 0 0 0 0 0 0

0 0 1 0 0 0 0 0 0

0 0 0 1 0 0 0 0 0

0 0 0 0 1 0 0 0 0

0 0 0 0 0 1 0 0 0

0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 1 0

0 0 0 0 0 0 0 0 1

x=y=6の場合

x=y=7の場合

x=y=8の場合

x=y=9の場合

## 二重ループの例③

```
for x in range(1,10):  
    for y in range(1,10):  
        if x == (10-y):  
            print( "1 " , end="")  
        else:  
            print( "0 " , end="")  
    print()
```

xと(10-y)の値が同じ→"1 "  
異なる場合は→"0 "

```
>python sample.py  
0 0 0 0 0 0 0 0 1  
0 0 0 0 0 0 0 1 0  
0 0 0 0 0 0 1 0 0  
0 0 0 0 0 1 0 0 0  
0 0 0 0 1 0 0 0 0  
0 0 0 1 0 0 0 0 0  
0 0 1 0 0 0 0 0 0  
0 1 0 0 0 0 0 0 0  
1 0 0 0 0 0 0 0 0
```

# 二重ループの例③の出力結果

```
>python sample.py
```

0 0 0 0 0 0 0 0 1

x=1 の時

0 0 0 0 0 0 0 1 0

x=2 の時

0 0 0 0 0 0 1 0 0

x=3 の時

0 0 0 0 0 1 0 0 0

x=4 の時

0 0 0 0 1 0 0 0 0

x=5 の時

0 0 0 1 0 0 0 0 0

x=7の時

0 0 1 0 0 0 0 0 0

x=6 の時

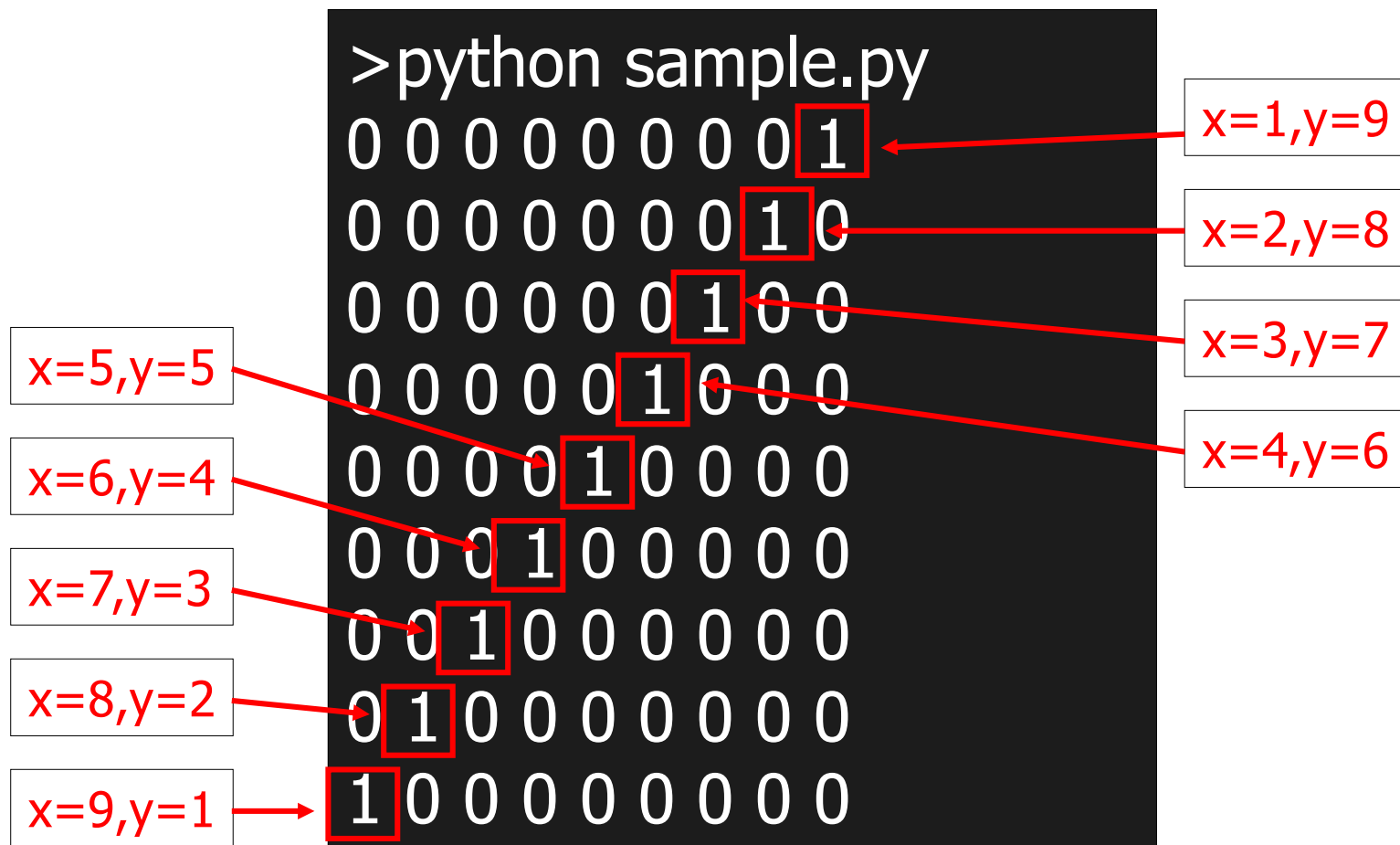
x=8の時

0 1 0 0 0 0 0 0 0

x=9の時

1 0 0 0 0 0 0 0 0

# 二重ループの例③の出力結果





## 二重ループの例④

```
for i in range(1,10):  
    for j in range(1,i+1):  
        print( j , end="" )  
    print()
```

j は 1から i まで変わる

```
>python sample.py  
1  
12  
123  
1234  
12345  
123456  
1234567  
12345678  
123456789
```

# 二重ループの例④の出力結果

```
>python sample.py
```

1

i=1の時, j=1

12

i=2の時, j=1~2

123

i=3の時, j=1~3

1234

12345

⋮

123456

1234567

12345678

i=8の時, j=1~8

123456789

i=9の時, j=1~9



## 二重ループの例⑤

```
for i in range(1,10):  
    for j in range(1,(11-i)):  
        print( j , end="" )  
    print ()
```

j は 1から 10-i まで変わる

```
>python sample.py  
123456789  
12345678  
1234567  
123456  
12345  
1234  
123  
12  
1
```

# 二重ループの例⑤の出力結果

```
>python sample.py
```

```
123456789
```

i=1の時, j=1~9

```
12345678
```

i=2の時, j=1~8

```
1234567
```

i=3の時, j=1~7

```
123456
```

```
12345
```

⋮

```
1234
```

```
123
```

```
12
```

i=8の時, j=1~2

```
1
```

i=9の時, j=1

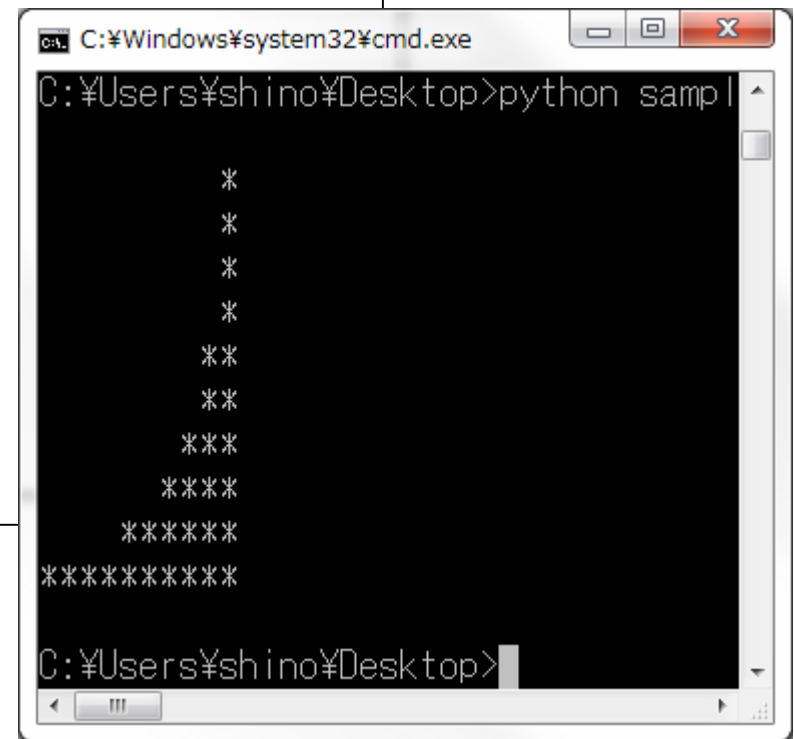


## 二重ループの例⑥

```
import math
for i in range(0,11):
    d = int( math.sqrt( 100 - i*i ))
    for j in range(1,d+1):
        print( " ", end="" )
    for j in range(d+1,11):
        print( "*", end="" )
    print()
```

d回は" "(空白)を表示

10-d回は"\*"を表示



```
C:\Windows\system32\cmd.exe
C:\Users\shino\Desktop>python samp1

          *
          *
          *
          *
         **
         **
        ***
        ***
       ****
       ****
      *****
      *****

C:\Users\shino\Desktop>
```



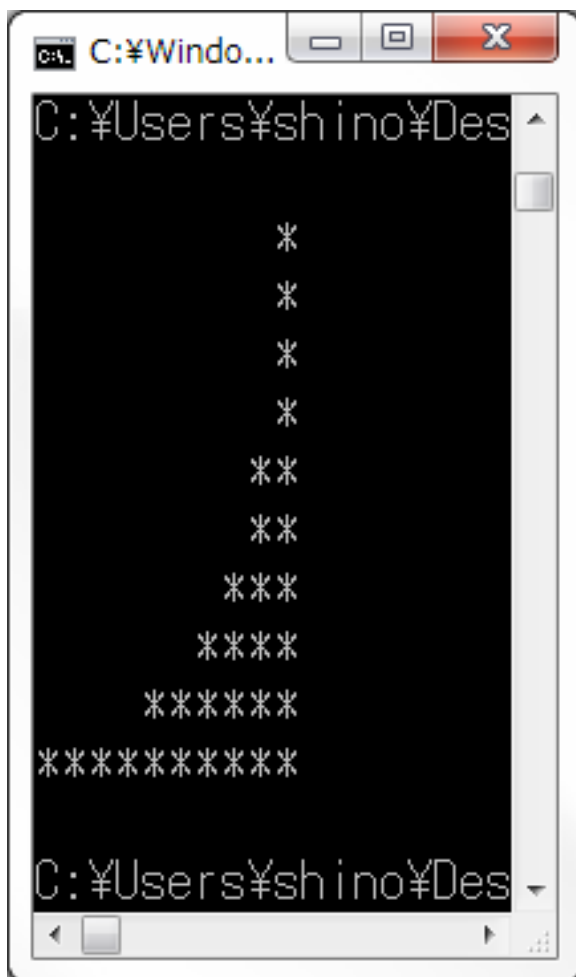
## 二重ループの例⑥

```
import math
for i in range(0,11):
    d = int(math.sqrt( 100 - i*i ))
    print( d )
```

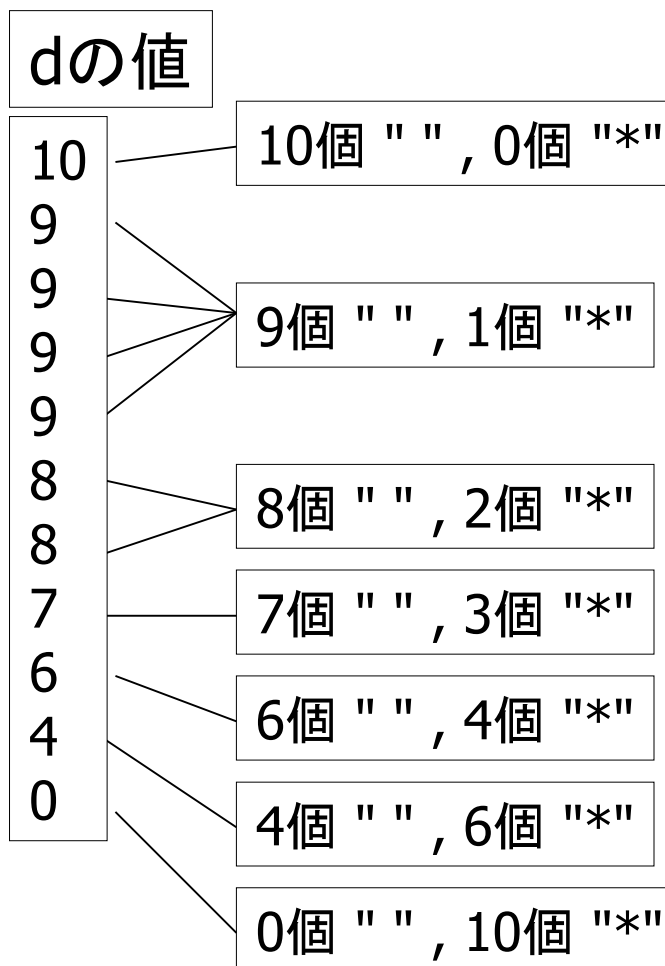
d の値はどう変わっていついてい  
るでしょうか

```
>python sample.py
10
9
9
9
9
9
8
8
7
6
4
0
```

# 二重ループの例⑥の出力結果



```
C:\Users\shino\Desktop>
          *
         *
        *
       *
      **
     **
    ***
   ****
  *****
 *****
C:\Users\shino\Desktop>
```







## 二重ループの例⑦

(ヒント:dの値はどう変わっていくでしょうか)

```
import math
for i in range(0,11):
    d = int(math.sqrt( 400 - 4*i*i ))
    print( d )
```

```
>python sample.py
20
19
19
19
18
17
16
14
12
8
0
```



# 練習問題

---

練習問題①～⑤

(時間があれば⑥もして下さい)



## 練習問題①

- 下記のプログラムにおいて、配列 test の要素値の標準偏差を求めるプログラムを追加しなさい

```
import math
name = [ "A" , "B" , "C" , "D" , "E" ]
test = [ 85 , 60 , 5 , 100 , 50 ]
sum = 0
for x in test:
    sum += x
average = int( sum / len(test) )
print( "平均値 ---> " , average )
```

標準偏差

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - a)^2}{n}}$$

## 練習問題②

- 下記のプログラムにおいて、配列aには1から100の乱数が格納されます.
- $j=1\sim 8$ において、 $(a[j-1]+a[j]+a[j+1])/3$ を求めなさい(移動平均)

```
import random
a = [0]*10
for i in range(10):
    a[ i ] = random.randint(1,100)
    print( a[ i ] , end=" " )
print()
```

```
> python 10-2.py
41 28 97 60 23 31 73 27 78 58
41 28 97 -> 55
28 97 60 -> 61
97 60 23 -> 60
60 23 31 -> 38
23 31 73 -> 42
31 73 27 -> 43
73 27 78 -> 59
27 78 58 -> 54
```



## 練習問題②'

前の問題を二重ループで作成した場合、やらなくて結構です

- 前ページの問題を二重ループを用いたプログラムで行ないなさい
- ヒント
  - $j = 1$  の時,  $a[0]+a[1]+a[2]$  を求めるには？

```
j = 1
sum = 0
for i in range(-1,2):
    sum += a[ j+i ]
```

- $j=1,2,\dots,8$  と変化させる



## 練習問題③

---

- $x, y$  ともに0から10までの整数とする. この場合,
  - ①  $x$ と $y$ の和が10となる組み合わせ
  - ②  $x^2$ と $y^2$ の和が100となる組み合わせを二重ループを用いて求めなさい



## 練習問題④

- 下記のような出力を行なうプログラムを二重ループを用いて書きなさい

```
>python 10-4-1.py
```

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

9行9列

```
>python 10-4-2.py
```

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

# 練習問題④

## ■ ヒント

```
>python 10-4-1.py
```

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

x=4

y=4

```
>python 10-4-2.py
```

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

二重ループの例③④参照



## 練習問題⑤

複数個の整数を引数として読み込みなさい。  
それらの整数の平均値, 最大値, 最小値を  
出力するプログラムを書きなさい。

```
>python 10-5.py 56 23 12 234 25 126 78 11  
平均値 70.625  
最小値 11  
最大値 234
```



## 練習問題⑥

---

- $x, y, z$ とも1から10までの整数とする. 以下の条件に合う $x, y, z$ を全て表示しなさい.

$$x^2 + y^2 + z^2 = 125$$



# 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
- 時間があれば⑥もして下さい
  
- プログラムと実行結果をワープロに貼り付けて, [keio.jp](http://keio.jp) から提出して下さい