



プログラミング言語 第七回

担当: 篠沢 佳久
栗原 聡

2019年 5月27日



本日の内容

- 繰り返し(3)
 - whileによる繰り返し
 - while 論理式:
- 疑似乱数
- 繰り返しについての練習問題
- 第二回レポート課題



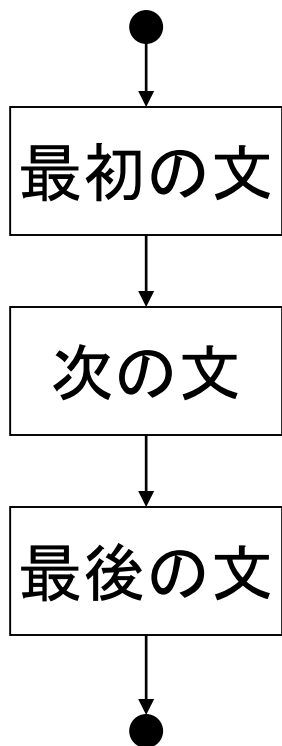
前回の復習

制御構造

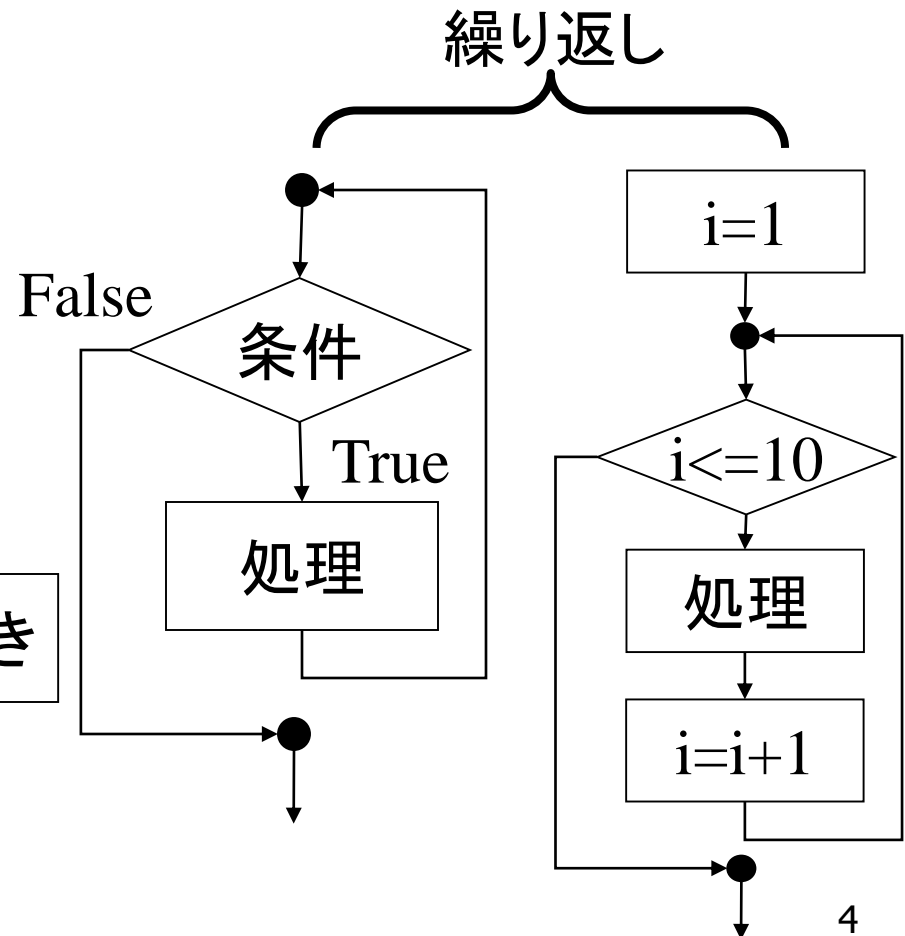
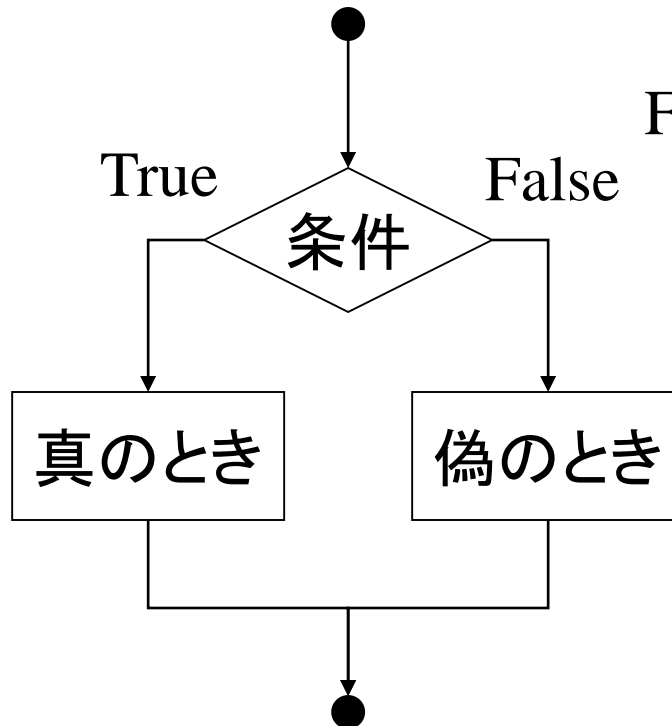
繰り返し(1)(2)

制御構造(復習)

文の並び(接続)



If文(分岐)





回数の決まった繰り返し

for文



回数の決まった繰り返し①

```
for 変数 in range(繰り返し回数):  
    式
```

```
for i in range(5):  
    print( "python" )
```

pythonを5回表示するプログラム

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

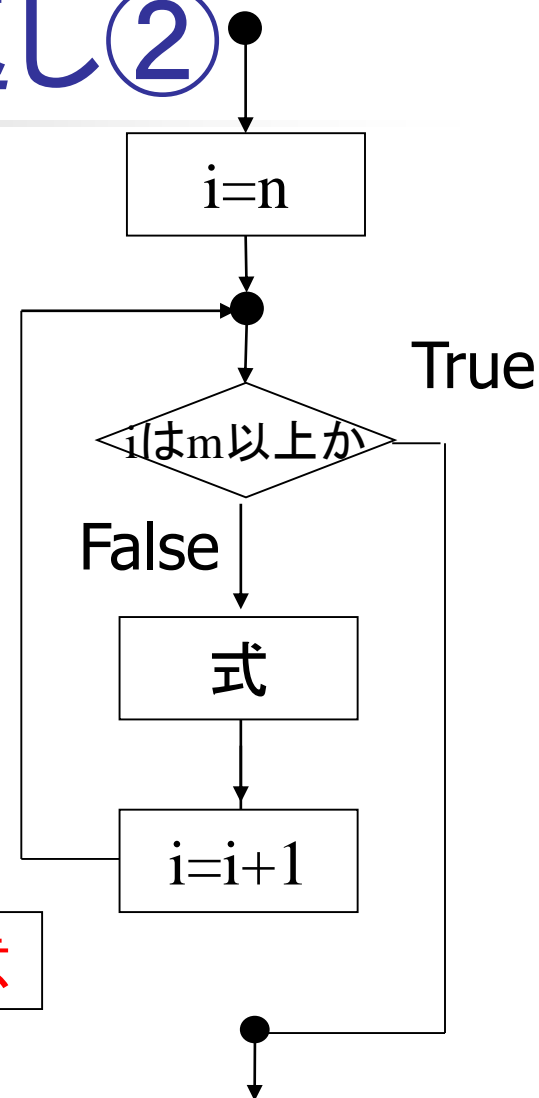
回数の決まった繰り返し②

```
for i in range(n,m):  
    式
```

($m-n-1$)回, 式が繰り返される

変数*i*には*n*から*m*-1まで代入される

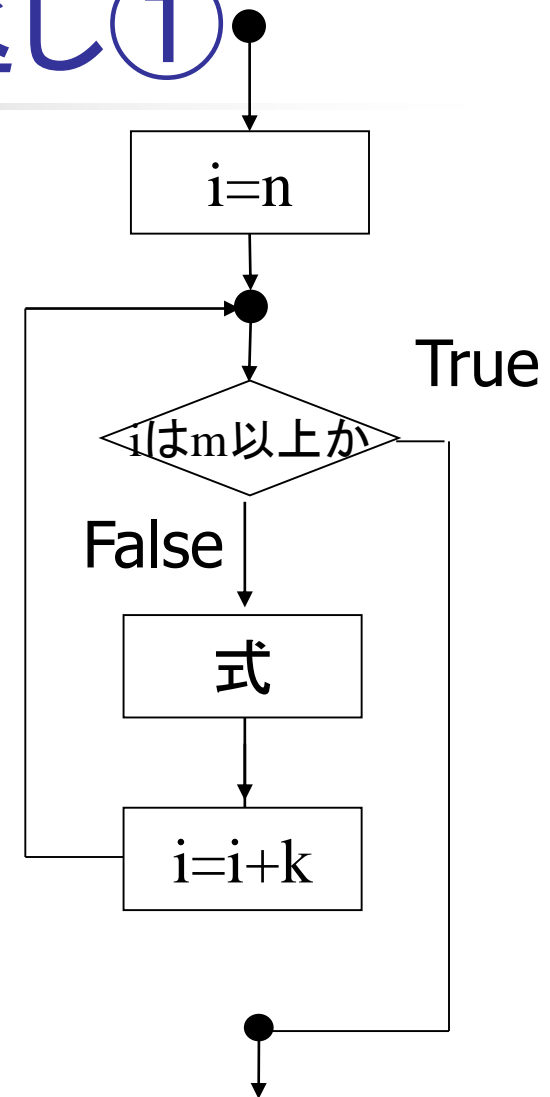
変数*i*は*m*までの代入でないことに注意



範囲の決まった繰り返し①

```
for i in range(n,m,k):  
    式
```

変数*i*には*n*から*k*ずつ加算されながら
、*m*を超えない値まで代入



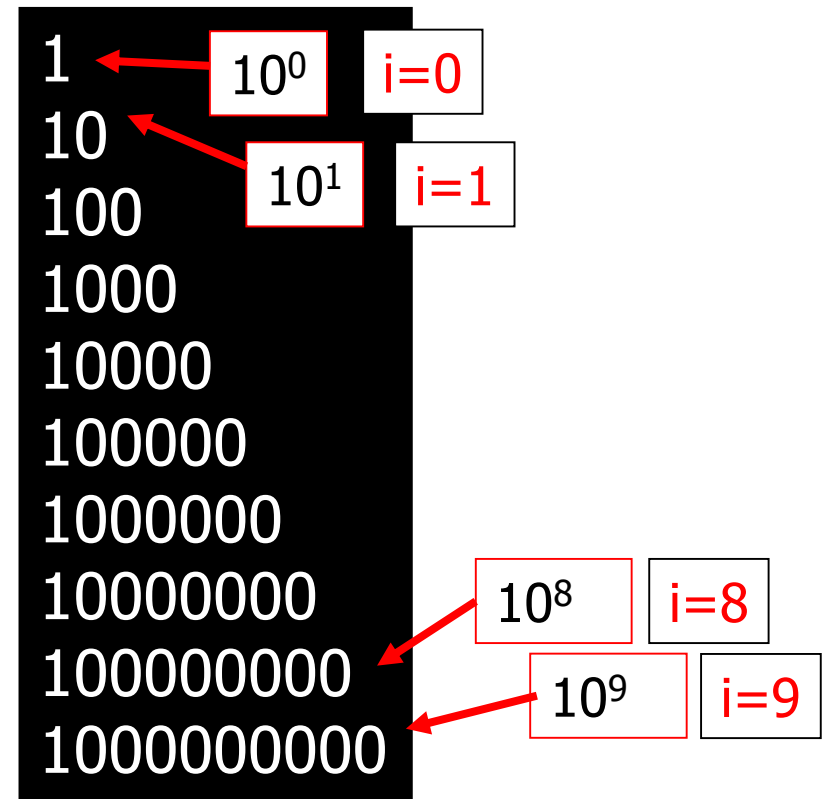
範囲の決まった繰り返し②

10のべき乗を表示するプログラム

```
for i in range(10):  
    print( 10 ** i )
```

```
for i in range(0,10):  
    print( 10 ** i )
```

```
for i in range(0,10,1):  
    print( 10 ** i )
```





無限の繰り返し①

while True:
式

式が永久に実行される
停止するために **break** を
用いる

while True:
if 論理式:
 break
次の式

論理式を満たした場合の
み停止する (whileブロック
の次の式を実行する)



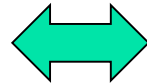
無限の繰り返し②

while True:

if 論理式:

break

次の式



while True:

if 論理式: break

次の式

無限の繰り返し②

10回で「こんにちは」の表示をやめるには？

```
i = 0
```

```
while True:
```

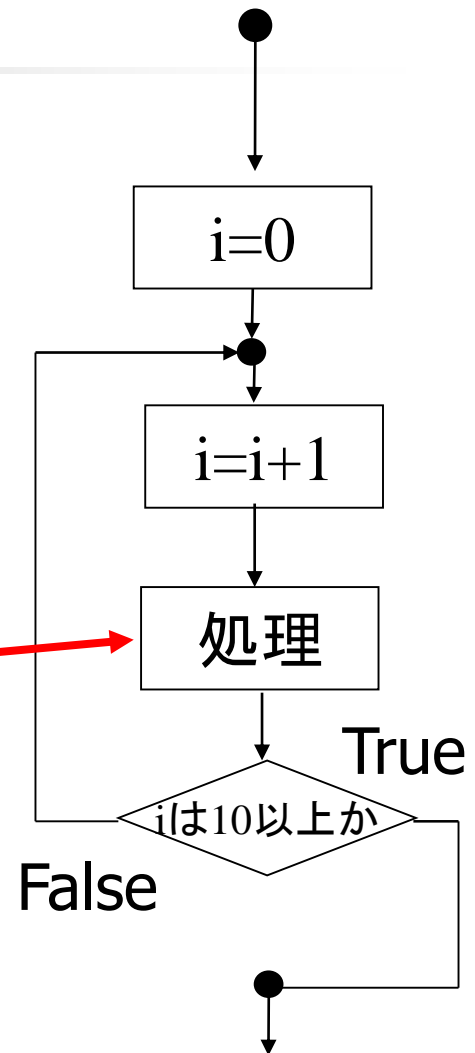
```
    i = i+1
```

```
    print( i , "回目のこんにちは" )
```

```
    if i >= 10:
```

```
        break
```

iが10以上となった場合、停止



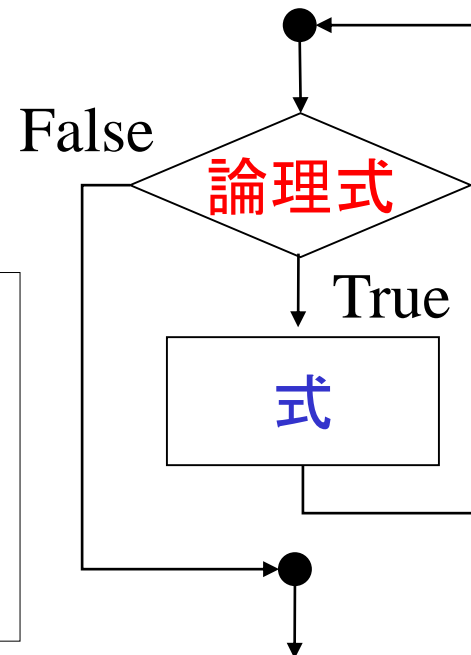


whileによる繰り返し

whileによる繰り返し

while 論理式:
式

論理式がTrueである場合は式を実行し、False の場合は式を実行しない
すなわち論理式がTrueである限りは式を繰り返し実行する



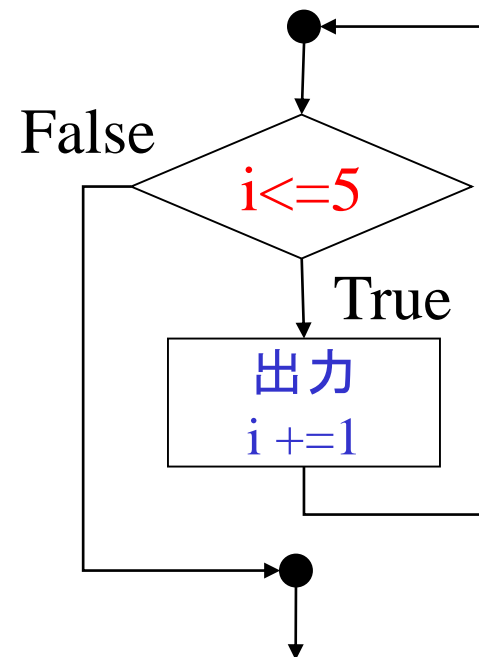
while による繰り返し(例①)

```
i = 1
while i <= 5:
    print( i , "回目です。" )
    i += 1
```

iが5以下である限り, 式を実行

i = i+1

1 回目です。
2 回目です。
3 回目です。
4 回目です。
5 回目です。



while による繰り返し(例①')

```
i = 1
while i <= 5:
    print( i , "回目です。" )
    i += 1
```

1回目です。
2回目です。
3回目です。
4回目です。
5回目です。

- ① 初期値 i=1
- ② 条件判定 → True
- ③ print による出力
- ④ i に1を加算 (i=2)
- ⑤ 条件判定 → True
- ⑥ printによる出力

- ⑦ i に1を加算 (i=3)
- ⑧ 条件判定 → True
- ⑨ printによる出力
- ⑩ i に1を加算 (i=4)
- ⑪ 条件判定 → True
- ⑫ printによる出力

while による繰り返し(例①')

```
i = 1
while i <= 5:
    print( i , "回目です。" )
    i += 1
```

1回目です。
2回目です。
3回目です。
4回目です。
5回目です。

- ⑬ i に1を加算 (i=5)
- ⑭ 条件判定 → True
- ⑮ printによる出力
- ⑯ i に1を加算 (i=6)
- ⑰ 条件判定 → False
- ⑱ 終了



これまで学んだ方法で書くと①

for文の場合

```
for i in range(5):  
    print( i+1 , "回目です。" )
```

```
for i in range(1,6):  
    print( i , "回目です。" )
```



これまで学んだ方法で書くと②

「while True:」(無限ループ)の場合

```
i = 1
while True:
    print( i , "回目です。" )
    if i >= 5:
        break
    i += 1
```

while による繰り返し(例②)

```
total = 0
```

```
i = 1
```

```
while i <= 10:
```

```
    total += i
```

```
    i += 1
```

```
print( total )
```

i=1 から開始

i<=10 の場合, 以下を実行



i=11 となった場合, print文を実行

i に1を加算

```
>python sample.py  
55
```

while による繰り返し(例②')

```
total = 0
```

```
i = 1
```

```
while total <= 100:
```

```
    total += i
```

```
    i += 1
```

```
print( i , total )
```

i=1 から開始

total<=100 の場合, 以下を実行



total>100となった場合, print文を実行

i に1を加算

```
>python sample.py  
15 105
```



while による繰り返しの例

- while による繰り返し(例③~⑥)の出力結果を考えなさい
- 実行する前に結果を予想して下さい



while による繰り返し(例③)

どのような出力結果になるでしょうか

```
i = 10
while i != 0:
    print( i )
    i -= 1
```

i = i-1



```
i = 10
while i > 0:
    print( i )
    i -= 1
```



while による繰り返し(例④)

どのような出力結果になるでしょうか

```
i = 0
while i < 10:
    print( i * 2 )
    i += 1
```

```
i = 0
while i <= 10:
    print( i * 2 )
    i += 1
```




while による繰り返し(例⑤)

どのような出力結果になるでしょうか

```
i = 1000
while i > 0:
    print( i )
    i = int( i / 2 )
```

```
i = 2
while i < 100000:
    print( i )
    i = i * 2
```



while による繰り返し(例⑥)

```
i = 1  
while i <= 5:  
    print( i , "回目です。" )
```

実行させるとどうなるでしょうか？



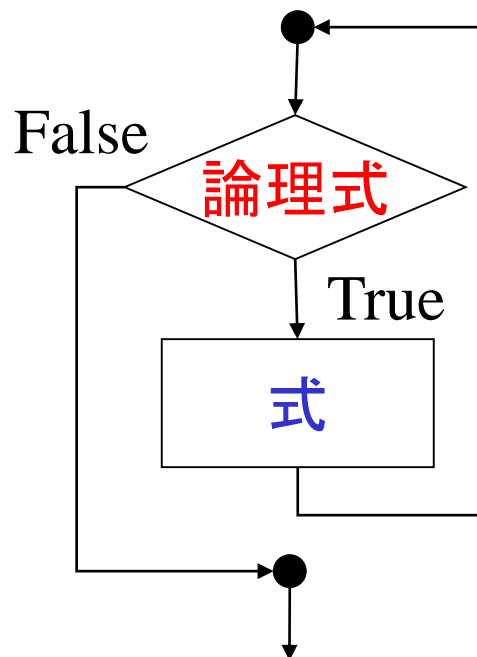
while による繰り返し(例⑥)

```
i = 1  
while i <= 5:  
    print( i , "回目です。" )
```

i の値は1から変化しない

そのため論理式は常にTrue であるため, print文が永久に実行される

while による繰り返しにおける注意



論理式がTrueである限り式は実行され続ける。そのため、論理式がFalseとなり終了するように条件を設定しなければならない



forとwhileの違い

- 同じである. ただし, コンセプトには結構な違いがある
 - for: 各回には, 制御変数(カウンター変数)の値の違いしかない
 - while: 各回には, 制御変数以外の変数で違いがある. または, 制御変数を自分で作ってあげないといけない



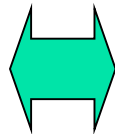
forとwhileの違い(続)

- Whileを使うのは,
 - ファイルの読み込み(後述)のように, 読んでみないとわからない場合
 - 次のテーマ
 - むずかし〜い計算なので, 計算してみないとわからない場合
 - Collatz-角谷の予想(練習問題⑤)
 - 素数を求める
 - 次回でてくる配列を用いる計算の中にある前の行為の結果が, 影響を及ぼす場合

例: forとwhileの違い①

"繰り返し回数"が分っている場合

```
for i in range(10):  
    print( i )
```



```
i=0  
while i<10:  
    print( i )  
    i = i + 1
```

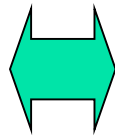
制御変数
(カウンター変数)

制御変数を自分で指定しなければならない

例: forとwhileの違い②

"繰り返し回数"が分っている場合

```
total = 0
for i in range(10):
    total += i
print( total )
```



```
i=0
total = 0
while i<10:
    total += i
    i = i + 1
print( total )
```

制御変数



例: forとwhileの違い③

```
i = 1000
while i > 0:
    print( i )
    i = int( i / 2 )
```

```
i = 2
while i < 100000:
    print( i )
    i = i * 2
```

"停止条件"が分っているが"繰り返し回数"は分らない
→ 変数 i の値はループ内の処理によって変わっていく
→ for では書きづらい



例: forとwhileの違い④

偶数を10個求めるプログラム

```
i = 1
count = 0
while count < 10:
    if i % 2 == 0:
        print( i )
        count += 1
    i += 1
```

```
>python sample.py
2
4
6
8
10
12
14
16
18
20
```

停止条件は「10個偶数が求まった時」
→実際に何回ループを回せばよいかは分からない

例: forとwhileの違い⑤

偶数を10個求めるプログラム

```
import random
count = 0
while count < 10:
    x = random.randint( 1,100 )
    if x % 2 == 0:
        print( x )
        count += 1
```

`random.randint(a,b)`
0以上b以下の整数を生成

```
>python sample.py
22
60
34
38
88
30
56
62
94
34
```

停止条件は「10個偶数が求まった時」
→実際に何回ループを回せばよいかは分からない

例: forとwhileの違い⑥

偶数を10個求めるプログラム

```
i = 1
count = 0
while count < 10:
    if i % 2 == 0:
        print( i )
        count += 1
    i += 1
```

```
i = 1
count = 0
while count != 10:
    if i % 2 == 0:
        print( i )
        count += 1
    i += 1
```

動作は同じであるが、停止条件が一意なため、間違えて無限ループのプログラムとして書いてしまいやすいので注意



例: forとwhileの違い⑦

"繰り返し回数" が
分かっている場合

```
import random
s = 0.0
for i in range(10):
    r = random.random()
    s += r
a = s/10.0
print( "平均=" , a )
```

random.random()
0以上1未満の乱数を生成

"繰り返し回数"は分からない
"停止条件" は分かっている場合

```
import random
s = 0.0
n = 0
while s<10.0:
    r = random.random()
    s += r
    n += 1
print( n , "回目で10を越えた" )
```



擬似乱数（詳細は後ほど）

- 平均，不偏分散，不偏標準偏差を求めてみよう

```
import random
import math
s = 0.0
s2= 0.0
n =10
for i in range(n):
    r = random.random()
    s += r
    s2 += r*r
    print( r , "は" , i , "番目の乱数です." )
a  = s/n
v  = (s2 - a*a*n)/(n-1)
sd = math.sqrt( v )
print( "平均=" , a , "分散=" , v , "標準偏差=" , sd )
```

random.random()
0以上1未満の乱数を生成



擬似乱数（詳細は後ほど）

実行結果

```
> python sample.py
0.9342183253647653 は 0 番目の乱数です.
0.17464151855331933 は 1 番目の乱数です.
0.042984728731218724 は 2 番目の乱数です.
0.783219542598958 は 3 番目の乱数です.
0.9851461205355044 は 4 番目の乱数です.
0.043461396973162536 は 5 番目の乱数です.
0.4956017448990262 は 6 番目の乱数です.
0.9952302150843787 は 7 番目の乱数です.
0.28483068587097293 は 8 番目の乱数です.
0.9527685456573891 は 9 番目の乱数です.
平均= 0.5692102824268696, 分散= 0.1639936761847672, 標準偏差= 0.4049613267767272
```

for～else文

```
for i in range(5):  
    print( i )  
else:  
    print( "End" )
```

0
1
2
3
4
End

for文の繰り返し
終了後に実行

```
for i in range(10):  
    print( i )  
    if i >= 5:  
        break  
else:  
    print( "End" )
```

0
1
2
3
4
5

breakによって
終了した場合は
実行されない

while~else文

```
i = 1
while i<=5:
    print( i )
    i = i+1
else:
    print( "End" )
```

1
2
3
4
5
End

for文の繰り返し
終了後に実行

```
i = 1
while i<=10:
    print( i )
    i = i+1
    if i >5:
        break
else:
    print( "End" )
```

0
1
2
3
4
5

breakによって
終了した場合は
実行されない



break文①

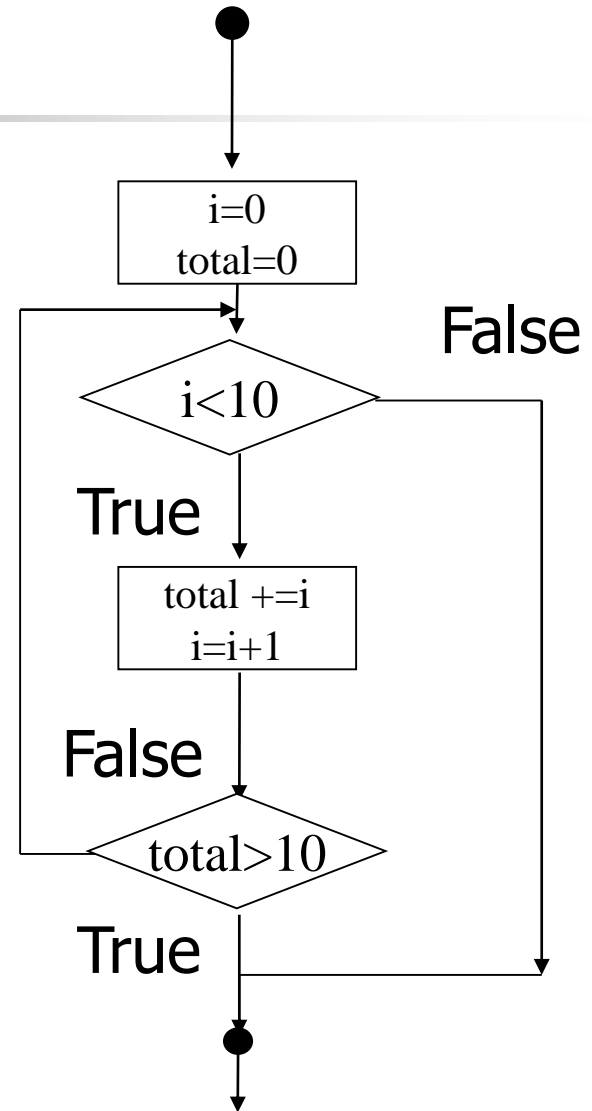
- 繰り返し(ループ)の中で使用
- そのbreakが所属するループを1つ抜ける

break文②

```
i = 0
total = 0
while i < 10:
    total += i
    i += 1
    if total > 10:
        break
print( i , total )
```

totalが10より大きい場合、停止

```
> python sample.py
6 15
```






break文③

True となっているため、無限に実行される

```
for i in range(100):  
    print( i )  
    if i >= 10:  
        break
```

for文中においてもbreakにより停止できる

i=0



```
while True do  
    print( i ):  
    i=i+1  
    if i > 10:  
        break
```

条件式をTrueとすることによって無限ループとなる



break文④

```
i = 1000
while i > 0:
    print( i )
    i = int( i / 2 )
```

二つのプログラムとも同じ動作をします

```
i = 1000
while True:
    print( i )
    i = int( i / 2 )
    if i <= 0:
        break
```

```
>python sample.py
1000
500
250
125
62
31
15
7
3
1
```



break文④

```
i = 1000
while i > 0:
    print( i )
    i = int( i / 2 )
```

ループの停止条件
→ $i > 0$ を満たさない時

```
i = 1000
while True:
    print( i )
    i = int( i / 2 )
    if i <= 0:
        break
```

ループの停止条件
→ なし(無限ループ)

break文によってループから抜ける



break文⑤

```
for i in range(10):  
    print( i )  
    if i > 5:  
        break
```

```
for i in range(10,100):  
    print( i )  
    if i > 20:  
        break
```

for文においてもbreakでループを抜け出ることができる



continue

- continueはもっとも内側のループの次の繰り返しにジャンプします
- while であれば,「継続条件」の判定の直前が再開場所です

continue①

```
for i in range(6):  
    if ( i==1 or i==4 ):  
        continue  
  
    print( "iは " , i )
```

「または」です

出力結果

iは 0
iは 2
iは 3
iは 5

i が1または4の場合

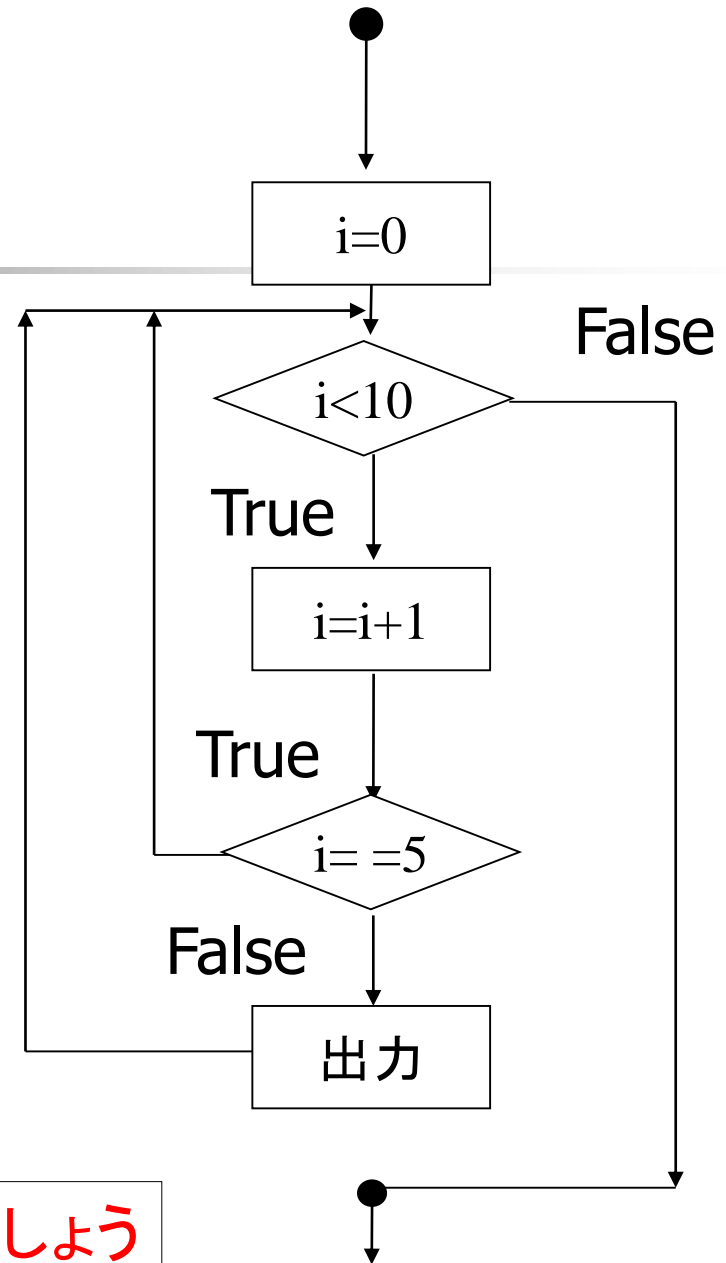
→ continueによってループの先頭に戻る

→ print文は実行されない

continue②

```
i=0
while i < 10:
    i += 1
    if i == 5:
        continue
    print( i )
```

出力はどうなるでしょう





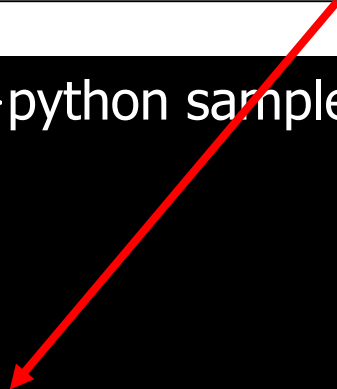
continue②

```
i=0
while i < 10:
    i += 1
    if i == 5:
        continue

    print( i )
```

i=5の場合, print文は実行されない

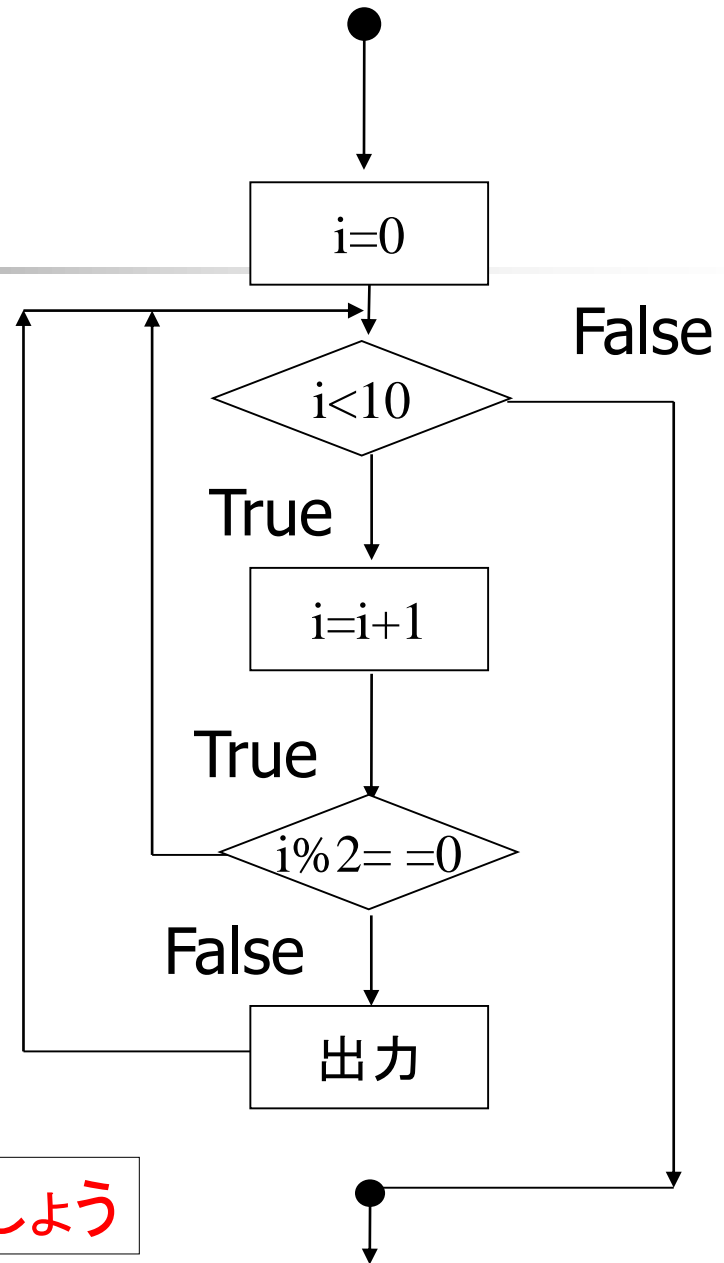
```
>python sample.py
1
2
3
4
6
7
8
9
10
```



continue③

```
i=0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue
    print( i )
```

出力はどうなるでしょう





continue③

```
i=0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue

    print( i )
```

iが偶数の場合, print文は実行されない

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

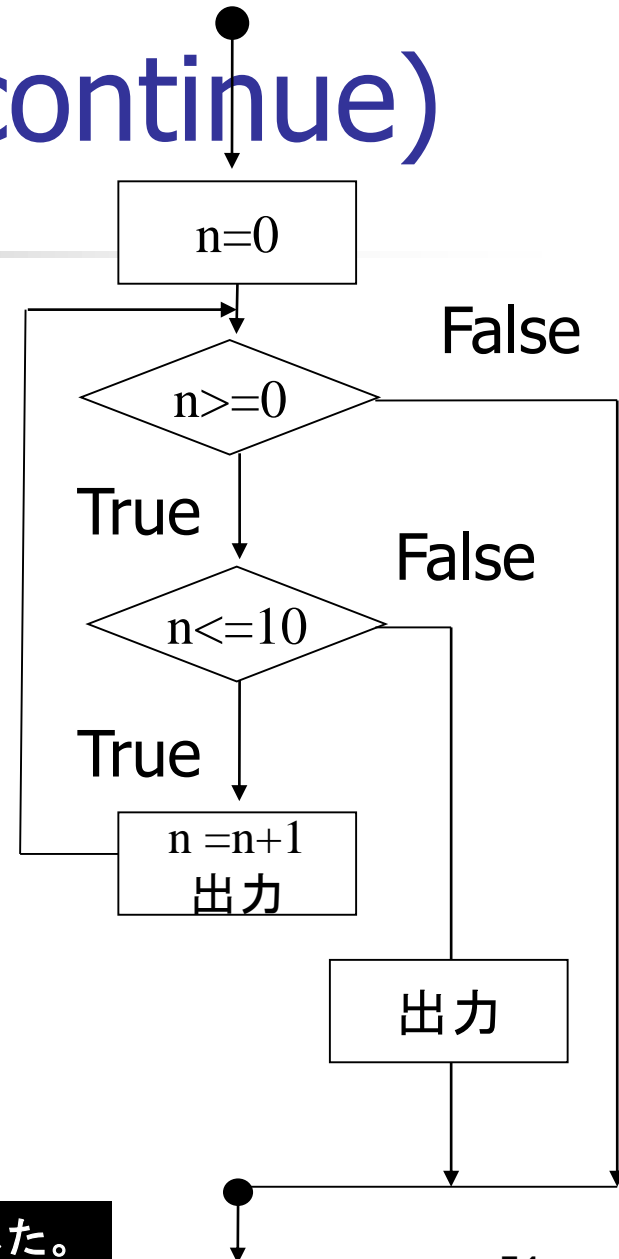
プログラム例(break,continue)

```
n = 0
while n >= 0:
    if n <= 10:
        print( n, end=" ")
        n += 1
        continue
    else:
        print( "変数 n は10を越えました。" )
        break
```

ループの先頭であるwhileに戻る

ループwhileを抜ける

0 1 2 3 4 5 6 7 8 9 10 変数 n は10を越えました。





繰り返しのまとめ

for, while



繰り返しのまとめ①

- 繰り返し回数が分かっている場合
 - for, while
- 繰り返し回数が分かっていない場合
 - while
- ただし、同じ動作をするにしても、いろいろな書き方があります



繰り返しのまとめ②

同じ動作をするプログラム

```
for i in range(10):  
    print( i )
```

```
for i in range(0,10):  
    print( i )
```

```
i = 0  
while True:  
    if i >= 10:  
        break  
    print( i )  
    i += 1
```

```
i = 0  
while i < 10:  
    print( i )  
    i += 1
```

```
>python sample.py
```

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



繰り返しのまとめ③

同じ動作をするプログラム

```
i = 2
while i < 100:
    print( i )
    i = i*2
```

```
i = 1
while True:
    i = i*2
    if i > 100:
        break
```

```
i = 1
while 2 ** i < 100:
    print( 2**i )
    i = i+1
```

```
>python sample.py
2
4
8
16
32
64
```



疑似乱数

疑似乱数のプログラム例



擬似乱数①

- コンピュータが計算して作り出す乱数. 本当の乱数ではないが, かなり本物に近い
- `random.random()`
 - 0以上1未満の値(小数)が一樣ランダムに生成される
- `random.uniform(a,b)`
 - a以上b未満の値(小数)が一樣ランダムに生成される
- `random.randint(a,b)`
 - a以上b以下の値(整数)が一樣ランダムに生成される
- `import random`を先頭に書くことを忘れずに

擬似乱数②

```
import random
for i in range(30):
    if random.random() > 0.5:
        print( "1" , end="" )
    else:
        print( "0" , end="" )
```

import randomを必ず書く

0以上1未満の乱数を発生

改行しない

実行する度に結果は異なります

111111001001110001110110000110



擬似乱数②

import randomを必ず書く

import random

for i in range(30):

print(random.randint(0,1) , end="")

0以上1以下の整数を発生

実行する度に結果は異なります

111111001001110001110110000110



擬似乱数③

```
import random
```

```
s = 0
```

```
while True:
```

```
    s = random.randint(1,6)
```

```
    print( s )
```

```
    if s == 6:
```

```
        break
```

1から6の乱数(整数)を生成

```
>python sample.py
```

```
4  
1  
5  
4  
5  
4  
5  
6
```

6の場合, 停止



擬似乱数④

```
import random
s = 0
while True:
    s = random.uniform(1,6)
    print( s )
    if s >= 5.0:
        break
```

1から6の乱数(小数)を生成

```
> python sample.py
4.384231592209254
2.9536332421136353
2.3703083582301656
4.983698163040744
3.7466176974037033
2.7180904882511197
2.9904581478042473
5.090845709448202
```

5.0以上の場合, 停止



擬似乱数⑤

randint(0,10)によって0から10の整数
randint(10)/10.0によって0.0から1.0ま
で0.1刻みの小数を生成

```
import random
for i in range(5):
    s = random.randint(0,10)/10.0
    print( s )
```

```
>python sample.py
0.5
0.0
0.8
0.5
0.2
```

どういう乱数でしょうか

```
random.randint(0,10)*100

random.randint(0,100)/100.0

random.randint(0,5)/10.0+0.5
```



疑似乱数を用いた例①

```
import random
total = 0
for i in range(5):
    x = random.randint(0,99)
    print( x )
    total += x
print( " 合計は" , total )
```

0から99の整数を生成

```
>python sample.py
72
29
88
4
98
合計は291
```



疑似乱数を用いた例②

```
import random
max = 0
for i in range(5):
    x = random.randint(1,100)
    print( x )
    if max < x:
        max = x
print( " 最大値は" , max )
```

最大値を求めるプログラム

```
>python sample.py
90
88
38
79
68
最大値は90
```



疑似乱数を用いた例③

```
import random
min = 999
for i in range(5):
    x = random.randint(1,100)
    print( x )
    if min > x:
        min = x
print( " 最小値は" , min )
```

最小値を求めるプログラム

```
>python sample.py
62
66
63
14
77
最小値は14
```

疑似乱数を用いた例④

```
import random
a = 0
b = 0
for i in range(10000):
    x = random.randint(0,1)
    if x == 0:
        a += 1
    else:
        b += 1

print( " 0の出現回数 " , a , "回" )
print( " 1の出現回数 " , b , "回" )
```

乱数が0の場合, aに1を加算
乱数が1の場合, bに1を加算

```
>python sample.py
0の出現回数 4927回
1の出現回数 5073回
```

疑似乱数を用いた例⑤

```
import random
ans=rand(1,100)
while True:
    print( "数字を入力して下さい" )
    input_data = int(input(">"))
    if ans == input_data:
        print( "正解" )
    elif input_data > ans:
    else:
        print( "正解はその値よりも大きい" )
```

1から100の整数を生成

整数の入力

条件式1

条件式2

条件式3

疑似乱数を用いた例⑤

>python sample.py
数字を入力して下さい
>12
正解はその値よりも大きい
数字を入力して下さい
>20
正解はその値よりも大きい
数字を入力して下さい
>30
正解はその値よりも大きい
数字を入力して下さい
>40
正解はその値よりも大きい
数字を入力して下さい

条件式3

>80
正解はその値よりも小さい
数字を入力して下さい
>75
正解はその値よりも大きい
数字を入力して下さい
>76
正解はその値よりも大きい
数字を入力して下さい
>77
正解はその値よりも大きい
数字を入力して下さい
>78
正解はその値よりも大きい
数字を入力して下さい
>79
正解

条件式2

条件式1



練習問題

入出力画面の通りに表示しなくてよいです



練習問題

- 練習問題①から④を行ないなさい.
- (簡単な人は⑤も行ないなさい)
- 「while文を用いて」と指定している場合は, 「while True:」, 「while 論理式:」どちらをも用いてもかまいません
- 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.



練習問題①

- 下記と同じ動作をするプログラムを「while Ture:」と「while 論理式:」を用いて書きなさい

```
sum = 0
for i in range(3,8):
    sum += i*i
print( sum )
```

```
>python 7-1.py
135
```



練習問題②

整数 n ($n > 1$) をキーボードより入力し, n の階乗を印字するプログラムを while 論理式: を用いて書きなさい

```
>python 7-2.py  
整数を入力して下さい>10  
10の階乗は3628800
```



練習問題③

- 1以上100以下の整数に対し、それが3の倍数でも10の倍数でもないときに、印字するプログラムをwhile文を用いて書きなさい。
- ただし、continueを用いて下さい。
- 結果は一部のみ貼り付けて下さい。

```
>python 7-3.py
```

```
1  
2  
4  
5  
7  
8  
11  
13  
14  
16  
17  
19  
22  
23
```



練習問題④

- サイコロを1000回ふった際, サイコロの目の出現回数を求めるプログラムを書きなさい.
- (69ページを参考)

```
>python 7-4.py  
1の出現回数 171  
2の出現回数 171  
3の出現回数 185  
4の出現回数 149  
5の出現回数 149  
6の出現回数 174
```



練習問題⑤

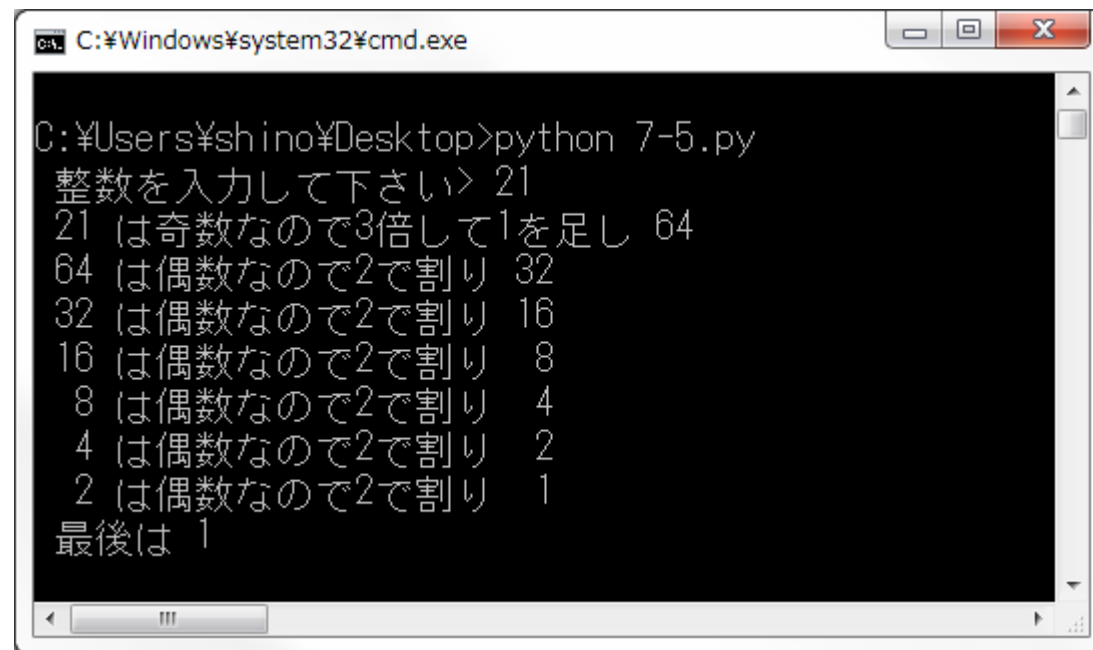
- Collatz-角谷の予想
- 自然数 n を選び,
 - 奇数ならば, 3倍して1を足す
 - 偶数ならば, 2で割る

これを繰り返すと, どんな n を選んでも, いつかは, 1になる

```
3, 10, 5, 16, 8, 4, 2, 1
4, 2, 1
5, 16, 8, 4, 2, 1
6, 3, 10, 5, 16, 8, 4, 2, 1
7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
8, 4, 2, 1
9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16,
8, 4, 2, 1
10, 5, 16, 8, 4, 2, 1
11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
```

練習問題⑤

- 整数 n をキーボードから読み込み，前頁の性質の確認するプログラムをwhile文を用いて作成しなさい.



```
C:\Windows\system32\cmd.exe

C:\Users\shino\Desktop>python 7-5.py
整数を入力して下さい> 21
21 は奇数なので3倍して1を足し 64
64 は偶数なので2で割り 32
32 は偶数なので2で割り 16
16 は偶数なので2で割り 8
8 は偶数なので2で割り 4
4 は偶数なので2で割り 2
2 は偶数なので2で割り 1
最後は 1
```