問題集

Simple

変数、リスト、辞書などの定義、if elif else、for・・・くらいの知識で解けるもの。

■ Hello, world!

```
print 'Hello, world!'
```

■ 1 から 50 までの和を計算して表示

```
# 普通の方法
s = 0
for x in range(1,51):
    s += x
print s

#sum を使う
print sum(range(1,51))

#generator 式を使う
print sum(x for x in range(1,51))

#reduce を使う
print reduce(lambda x,y: x+y, range(1,51))

# 頭を使う
print 50*51/2
```

■ 2 つの自然数の最大公約数を求める（ユークリッドの互除法）

```
a,b = input(), input()

# 再帰関数
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a%b)
print gcd(a,b)
```

■ 2 つの自然数の最小公倍数を求める。

```
# 上の gcd を用いて
print a*b/gcd(a,b)
```

■ 10000 以下の素数を表示する。

```
# 怠惰な方法
for i in range(2,10000):
    if 0 not in [i%j for j in range(2,i/2+1)]:
        print i

# 少しちゃんとした方法
from math import sqrt
for i in range(2,10000):
    if 0 not in [i%j for j in range(2, int(sqrt(i))+1)]:
        print i

# 上のよりもっと速い方法
# 2 に対して 2 より大きい偶数を削除し、3 に対して 3 より大きく 3 で割れる数を排除し・・・
# としていくので無駄なチェックが減っている
MAX = 10000
LIST = range(2, MAX + 1)
```

```
for i in range(2, int(MAX ** 0.5)):
     LIST = [x for x in LIST if (x == i or x % i != 0)]
for j in LIST:
     print j


# エラトステネスの篩
# (コードは煩雑そうに見えるが、速度が圧倒的に速い。
#  大きな数までの素数表が必要な場合に必要な方法。)
a = range(10000)
a[1] = 0 # 1 は素数ではない
for p in a:
    if not p:
        continue
    elif p > 100: # 100 = sqrt(10000)
        break
    else:
        for multi in xrange(p+p, 10000, p):
            a[multi] = 0

# この時点で、a は、素数 p 番目には p が、それ以外には 0 が入ったリストになっている。
a = [x for x in a if x]
for x in a: print x


# list を生成せずに generator で無限に出力
# ( エラトステネスの篩とほぼ等価で高速 )
from itertools import ifilter, count
def primes():
  g = count(2)
  while True:
    p = g.next()
    yield p
    g = ifilter(lambda n, p=p: n % p, g)

>>> for i, p in enumerate(primes()):
...  print '(%d:%d)' % (i, p)
...  raw_input()
...
(0:2)
(1:3)
(2:5)
```

■ フィボナッチ数 (1,1,2,3,5,8,13,...) を 2**31 より小さい範囲まで表示する。

```
a=b=1
while a < 2**31:
    print a
    a, b = b, a+b
```

■ 無限にフィボナッチ級数を返し続ける generator。

```
def fib():
  a = b = 1
  while True:
    yield a
    a, b = b, a+b

>>> f = fib()
>>> for t in xrange(15):
>>>   print f.next(),
>>> 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

generator を使わずに

```
a = b = 1
c = 2
while a < c:
  print a,
  a, b = b, a + b
  c += a
```

■ n**(n**n) の一の位の数字を表示する。(n が 1000 程度でもそれなりの速度で動作し

ますが、n **の剰余と合同を使えば** pow **使わずに済みます。下記参照**)

```
pow(n, pow(n, n), 10)
```

■ n**(n**n) **の一の位の数字を表示する。**(pow **不使用**)

```
>>> c=[(0,), (1,), (2,4,8,6,), (3,9,7,1,), (4,6,), (5,), (6,), (7,9,3,1,), (8,4,2,6,), (9,1,)]
>>> [c[n%10][(n%2)*n%len(c[n%10])-1] for n in xrange(20)]
[0, 1, 6, 7, 6, 5, 6, 3, 6, 9, 0, 1, 6, 3, 6, 5, 6, 7, 6, 9]
```

Normal

文字列やリストのメソッド、関数定義、イテレータ、ファイル入出力、基本的な正規表現などが使える人向け。

■ **テキストファイル** 'text.txt' **から数字を読み込み大きい数から順に並べて画面に表示する**

```
a = open('text.txt').read().split()
a.sort()
for i in a[::-1]: print i
# 別解
print sorted(open('text.txt').read().split(), key = int, reverse = True)
#sorted 関数を使うと、ソートのための一時的なリストを作る必要が無くなるので便利
```

■ **標準ライブラリを使って** sin 60 °**を求める**

```
from math import sin, pi
print sin(pi/3)
```

■ **摂氏を入力すると、華氏で出力。華氏を入力すると、摂氏で出力。動作例** : "41F" **を入力すると** "5C"**、**"5C" **を入力すると** "41F" **と表示する**

```
>>> def convt():
...  import re
...  r = re.compile(r'^([\.\d]+)(C|F)$', re.I)
...  t = raw_input('temperature(C or F)= (ex 5C, 41F, etc)? ')
...  q = r.match(t)
...  if q:
...   if q.group(2).upper() == 'C':
...    print '%fF' % (float(q.group(1)) * 9 / 5 + 32)
...   if q.group(2).upper() == 'F':
...    print '%fC' % ((float(q.group(1)) - 32) * 5 / 9)
...
>>> convt()
temperature(C or F)= (ex 5C, 41F, etc)? 5c
41.000000F
>>> convt()
temperature(C or F)= (ex 5C, 41F, etc)? 41f
5.000000C
>>> convt()
temperature(C or F)= (ex 5C, 41F, etc)? 2.5c
36.500000F
>>> convt()
temperature(C or F)= (ex 5C, 41F, etc)? 20.5f
-6.388889C
```

■ 16 **進数ダンプを行う**

```
print ''.join('%02x '%ord(x) for x in given_string)
```

■ **テキストファイル** 'text.txt' **の中に** Python **と言う文字がいくつ含まれるか調べる**

```
print open('text.txt').read().count('Python')
```

■ パスワード生成する

```
import string
import random

print ''.join([string.printable[n] for n in [
 random.randint(0, len(string.printable) - 1) for r in xrange(20)]])
```

■文字候補 'python' から n 文字の文字列を生成する (重複なしの場合)

```
>>> def perm(n, m):
...  if m<1:
...   yield()
...  else:
...   for r in perm(n, m-1):
...    for x in xrange(n):
...     if x not in r:
...      yield r + (x,)
...
>>> def create_word_list_p(n, s):
...  return [''.join(s[i] for i in t) for t in perm(len(s), n)]
...
>>> create_word_list_p(3, 'python')
['pyt', 'pyh', 'pyo', 'pyn', 'pty', 'pth', 'pto', 'ptn', 'phy', 'pht',
 'pho', 'phn', 'poy', 'pot', 'poh', 'pon', 'pny', 'pnt', 'pnh', 'pno',
 ( 省略 )
 'npy', 'npt', 'nph', 'npo', 'nyp', 'nyt', 'nyh', 'nyo', 'ntp', 'nty',
 'nth', 'nto', 'nhp', 'nhy', 'nht', 'nho', 'nop', 'noy', 'not', 'noh']
```

■文字候補 'python' から n 文字の文字列を生成する (重複ありの場合)

```
>>> def combi(a, b):
...  if isinstance(a[0], tuple):
...   return [x + (y,) for x in a for y in b]
...  else:
...   return [(x, y) for x in a for y in b]
...
>>> def combination(*l):
...  return reduce(combi, l)
...
>>> combination([1,2,3],[4,5],[6,7])
[(1, 4, 6), (1, 4, 7), (1, 5, 6), (1, 5, 7),
 (2, 4, 6), (2, 4, 7), (2, 5, 6), (2, 5, 7),
 (3, 4, 6), (3, 4, 7), (3, 5, 6), (3, 5, 7)]
>>> def create_word_list(n, s):
...  return [''.join(s[i] for i in t) for t in combination(*[xrange(len(s)) for j in xrange(n)])]
...
>>> create_word_list(3, 'python')
['ppp', 'ppy', 'ppt', 'pph', 'ppo', 'ppn',
 'pyp', 'pyy', 'pyt', 'pyh', 'pyo', 'pyn',
 'ptp', 'pty', 'ptt', 'pth', 'pto', 'ptn',
 'php', 'phy', 'pht', 'phh', 'pho', 'phn',
 'pop', 'poy', 'pot', 'poh', 'poo', 'pon',
 'pnp', 'pny', 'pnt', 'pnh', 'pno', 'pnn',
 ( 省略 )
 'npp', 'npy', 'npt', 'nph', 'npo', 'npn',
 'nyp', 'nyy', 'nyt', 'nyh', 'nyo', 'nyn',
 'ntp', 'nty', 'ntt', 'nth', 'nto', 'ntn',
 'nhp', 'nhy', 'nht', 'nhh', 'nho', 'nhn',
 'nop', 'noy', 'not', 'noh', 'noo', 'non',
 'nnp', 'nny', 'nnt', 'nnh', 'nno', 'nnn']
```

■ リスト A = ['AA', 'AB', 'AC'] , B = ['BA', 'BB', 'BC'] から [('AA', 'BA'), ('AB', 'BA'), ... , ('AC', 'BB'), ('AC', 'BC')] のように，それぞれから 1 つづつ取ってきたリストを簡単に生成する

```
C = reduce(lambda x, y: x+y,[[(x,y) for y in B] for x in A])
```

■ リスト [['Py','PyA'],['Py','PyB'],['Py','PyC'],['Pe','PeA'],['Pe','PeC']] から、辞書 {'Py': ['PyA', 'PyB', 'PyC'], 'Pe': ['PeA', 'PeC']} を生成する

```
a = [ ['Py','PyA'],['Py','PyB'],['Py','PyC'],['Pe','PeA'],['Pe','PeC'] ]
t = {}
for n in a:
    if n[0] in t: t[n[0]].append(n[1])
    else: t[n[0]] = [n[1]]
# 別解
t = {}
for n in a:
    t.setdefault(n[0], []).append(n[1])
```

■ n 個の空リストを要素とするリスト [[],[],・・・,[]] を生成する。

```
a = [[]] * n # 注意：浅いコピー
a = [[] for x in range(n)]
```

Advanced

やや専門的なライブラリが使える。クラス定義してオブジェクト指向プログラミングができる。正規表現をけっこう使える。

■ 2ch のトリップを生成する。(旧)

2009-06-19 より新方式が採用されていますので一部結果が異なります。12 桁トリップ

```
import sys, re
from string import maketrans
if sys.platform == 'win32':
  from fcrypt import crypt
else:
  from crypt import crypt
def trip(key):
  salt = re.sub('[^\.-z]', '.', (key + 'H.')[1:3]).translate(
    maketrans(':;<=>?@[\\]^_`', 'ABCDEFGabcdef'))
  return crypt(key, salt)[-10:]
```

■ 電話帳プログラムを作る。

```
>>> phones = {}
>>> def reg():
...  name = raw_input('name= ')
...  tel = raw_input('tel= ')
...  phones[name] = tel
...
>>> reg()
name= abc
tel= 123-4567
>>> reg()
name= def
tel= 890-1234
>>> phones
{'abc': '123-4567', 'def': '890-1234'}
>>>
```

Maniac

問題読んだだけでは解法が思いつかない。あるいはすぐ思いつく方法が実は間違ってるようなもの。

■ FizzBuzz

解き方いろいろ

■ 乱数を使って円周率の近似値を求める

```
#!/usr/bin/python

# モンテカルロ法による円周率の近似値の計算
#
# 1 セット MAX_I 回の試行を MAX_J セット繰り返す。

from random import *

seed()
MAX_J = 100
MAX_I = 10000
total = 0.0
for j in range(MAX_J):
  score = 0.0
  for i in range(MAX_I):
    x = random()
    y = random()
    r = x*x+y*y
    if (r &lt; 1):
      score += 1
  score = 4*score/MAX_I
  print score
  total += score
print
print total/MAX_J
```

■ Quine (自分自身のソースコードと同じ文を出力するプログラム)

```
x='x=%s;print x%%`x`,';print x%`x`,
```

```
x='x=%s\nprint x%%`x`'
print x%`x`
```

```
x='x=%s;print x%%`x`,;print';print x%`x`,;print 1729
```

■ http://www.geocities.co.jp/SiliconValley-Cupertino/7896/software/turing.html みたいなチューリングマシン

■ brainfuck interpreter

■ bogoYAML parser