

Solutions to first 50 projecteuler.net problems by Sanakreon

EULER 1

```
print sum(xrange(0,1000,3))+sum(xrange(0,1000,5))-sum(xrange(0,1000,15))
```

EULER 2

```
#Only those members of fibonacci sequence(f(i)) are even, which satisfy i%3==2
a=1
b=2
fibsum=0
while b<4000000:
    a,b=b,a+b
    if a%2==0:
        fibsum+=a
print fibsum
```

EULER 3

```
N=600851475143
max=1
for i in xrange(3,int(N**0.5)+1,2):
    while N % i ==0:
        if max<i: max=i
        N=N//i
print max
```

EULER 4

```
def is_palindrome(num):
    A=list(str(num))
    for i in xrange(len(A)//2):
        if A[i]!=A[-1-i]:
            return False
    return True

maximum=0
for i in xrange(999,99,-1):
    for j in xrange(i,1000):
        k=i*j
        if is_palindrome(k) and k>maximum:
            maximum=k
print maximum
```

EULER 5

```
#The answer for 20 is simple to calculate without programming,
#basically it is:
#2**4 * 3**2 * 5*7*11*13*17*19
#So I am writing a general program for N

#Python 2.7.1
```

```

from math import log
from operator import mul
N=1000

#Find all prime numbers less or equal to N
L=range(N+1)
i=1
while i <= int(N**0.5):
    i+=1
    if L[i]==1: continue
    j=i+i
    while j<=N:
        L[j]=1
        j+=i

L=[i for i in L[1:] if i!=1]
P=[]

#Calculate maximum powers of each prime number
dont_take_log_anymore=False
for i in L:
    if dont_take_log_anymore :
        P.append(1)
    else:
        power=int(log(N,i))
        if power==1:
            dont_take_log_anymore=True
            P.append(power)

Final=[L[i]**P[i] for i in range(len(L))]
print reduce(mul,Final)

```

EULER 6

Ofcourse there are direct formulae for this also

```

N=100
sumsqr = sum([i*i for i in range(1,101)])
sqrsum = sum(range(1,101))**2
print sqrsum - sumsqr

```

EULER 7

*#By Rosser's theorem: $n \cdot \ln(n) + n \cdot (\ln(\ln(n)) - 1) < P(n) < n \cdot \ln(n) + n \cdot \ln(\ln(n))$
*#n>=6**

```

N=10001
from math import log

end=int(N*log(N)+N*log(log(N)))

def erat(N):
    L=range(N+1)
    i=1
    while i <= int(N**0.5):
        i+=1
        if L[i]==1: continue
        j=i+i

```

```
while j<=N:
    L[j]=1
    j+=i

return [i for i in L[1:] if i!=1]

print erat(end)[N-1]
```

EULER 8

```
from operator import mul
```

```
S="73167176531330624919225119674426574742355349194934\  
96983520312774506326239578318016984801869478851843\  
85861560789112949495459501737958331952853208805511\  
12540698747158523863050715693290963295227443043557\  
66896648950445244523161731856403098711121722383113\  
62229893423380308135336276614282806444486645238749\  
30358907296290491560440772390713810515859307960866\  
70172427121883998797908792274921901699720888093776\  
65727333001053367881220235421809751254540594752243\  
52584907711670556013604839586446706324415722155397\  
53697817977846174064955149290862569321978468622482\  
83972241375657056057490261407972968652414535100474\  
82166370484403199890008895243450658541227588666881\  
16427171479924442928230863465674813919123162824586\  
17866458359124566529476545682848912883142607690042\  
24219022671055626321111109370544217506941658960408\  
07198403850962455444362981230987879927244284909188\  
84580156166097919133875499200524063689912560717606\  
05886116467109405077541002256983155200055935729725\  
71636269561882670428252483600823257530420752963450"
```

```
L=[int(i) for i in S]
maximum=-1
for i in xrange(len(L[:4])):
    product=reduce(mul,L[i:i+5])
    if product > maximum:
        maximum=product
print maximum
```

EULER 9

```
# c can not be more than 500
# c can not be less than 250*sqrt(2) which is about 353
# b can not be more than sqrt(2) times smaller than c
# b can not be larger than c
# a is defined by b and c
def main():
    for c in xrange(353,500):
        for b in xrange(int(c/1.42)-1,c):
            a=1000-c-b
            if a*a+b*b==c*c:
                print a*b*c
                return

if __name__=="__main__":
    main()
```

EULER 10

```
N= 2000000
```

```
def erat(N):
    L=range(N+1)
    i=1
    while i <= int(N**0.5):
        i+=1
        if L[i]==1: continue
        j=i+i
        while j<=N:
            L[j]=1
            j+=i

    return [i for i in L[1:] if i!=1]
print sum(erat(N))
```

EULER 11

```
L =[\
"08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08",\
"49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00",\
"81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65",\
"52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91",\
"22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80",\
"24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50",\
"32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70",\
"67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21",\
"24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72",\
"21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95",\
"78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92",\
"16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57",\
"86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58",\
"19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40",\
"04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66",\
"88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69",\
"04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36",\
"20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16",\
"20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54",\
"01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48"]
```

```
for i,line in enumerate(L):
    L[i]=[int(x) for x in line.split()]

product= lambda a,b,c,d : a*b*c*d
N=20
h=max(product(*L[i][j:j+4]) for i in range(N) for j in range(N-4))
v=max(product(*L[i+k][j] for k in [0,1,2,3]) for i in range(N-4) for j in range(N))
d1=max(product(*L[i+k][j+k] for k in [0,1,2,3]) for i in range(N-4) for j in range(N-4))
d2=max(product(*L[i+k][j-k] for k in [0,1,2,3]) for i in range(N-4) for j in range(4, N))
print( max(h,v,d1,d2))
```

EULER 12

```
from functools import reduce
from math import sqrt

def num_of_divisors(N):
    L=[]
    i=2
    till=int(sqrt(N))+1
    while i<=till:
        k=0
        while N % i ==0:
            N=N//i
            k+=1
        if k!=0:
            till=int(sqrt(N))+1
            L.append(k)
        i+=1
    if N!=1:
        L.append(1)
    return reduce(lambda x,y: (x)*(y+1), L, 1)

triangular=0
i=0
while True:
    i+=1
    triangular+=i
    if num_of_divisors(triangular)>500:
        break

print (triangular)

EULER 13

L=[37107287533902102798797998220837590246510135740250,
46376937677490009712648124896970078050417018260538,
74324986199524741059474233309513058123726617309629,
91942213363574161572522430563301811072406154908250,
23067588207539346171171980310421047513778063246676,
89261670696623633820136378418383684178734361726757,
28112879812849979408065481931592621691275889832738,
44274228917432520321923589422876796487670272189318,
47451445736001306439091167216856844588711603153276,
70386486105843025439939619828917593665686757934951,
62176457141856560629502157223196586755079324193331,
64906352462741904929101432445813822663347944758178,
92575867718337217661963751590579239728245598838407,
58203565325359399008402633568948830189458628227828,
80181199384826282014278194139940567587151170094390,
35398664372827112653829987240784473053190104293586,
86515506006295864861532075273371959191420517255829,
71693888707715466499115593487603532921714970056938,
54370070576826684624621495650076471787294438377604,
53282654108756828443191190634694037855217779295145,
36123272525000296071075082563815656710885258350721,
45876576172410976447339110607218265236877223636045,
17423706905851860660448207621209813287860733969412,
81142660418086830619328460811191061556940512689692,
51934325451728388641918047049293215058642563049483,
62467221648435076201727918039944693004732956340691,
15732444386908125794514089057706229429197107928209,
```

55037687525678773091862540744969844508330393682126,
18336384825330154686196124348767681297534375946515,
80386287592878490201521685554828717201219257766954,
78182833757993103614740356856449095527097864797581,
16726320100436897842553539920931837441497806860984,
48403098129077791799088218795327364475675590848030,
87086987551392711854517078544161852424320693150332,
59959406895756536782107074926966537676326235447210,
69793950679652694742597709739166693763042633987085,
41052684708299085211399427365734116182760315001271,
65378607361501080857009149939512557028198746004375,
35829035317434717326932123578154982629742552737307,
94953759765105305946966067683156574377167401875275,
88902802571733229619176668713819931811048770190271,
25267680276078003013678680992525463401061632866526,
36270218540497705585629946580636237993140746255962,
24074486908231174977792365466257246923322810917141,
91430288197103288597806669760892938638285025333403,
34413065578016127815921815005561868836468420090470,
23053081172816430487623791969842487255036638784583,
11487696932154902810424020138335124462181441773470,
63783299490636259666498587618221225225512486764533,
67720186971698544312419572409913959008952310058822,
95548255300263520781532296796249481641953868218774,
76085327132285723110424803456124867697064507995236,
37774242535411291684276865538926205024910326572967,
23701913275725675285653248258265463092207058596522,
29798860272258331913126375147341994889534765745501,
18495701454879288984856827726077713721403798879715,
38298203783031473527721580348144513491373226651381,
34829543829199918180278916522431027392251122869539,
40957953066405232632538044100059654939159879593635,
29746152185502371307642255121183693803580388584903,
41698116222072977186158236678424689157993532961922,
62467957194401269043877107275048102390895523597457,
23189706772547915061505504953922979530901129967519,
86188088225875314529584099251203829009407770775672,
11306739708304724483816533873502340845647058077308,
82959174767140363198008187129011875491310547126581,
97623331044818386269515456334926366572897563400500,
42846280183517070527831839425882145521227251250327,
55121603546981200581762165212827652751691296897789,
32238195734329339946437501907836945765883352399886,
75506164965184775180738168837861091527357929701337,
62177842752192623401942399639168044983993173312731,
32924185707147349566916674687634660915035914677504,
99518671430235219628894890102423325116913619626622,
73267460800591547471830798392868535206946944540724,
76841822524674417161514036427982273348055556214818,
97142617910342598647204516893989422179826088076852,
87783646182799346313767754307809363333018982642090,
10848802521674670883215120185883543223812876952786,
71329612474782464538636993009049310363619763878039,
62184073572399794223406235393808339651327408011116,
66627891981488087797941876876144230030984490851411,
60661826293682836764744779239180335110989069790714,
85786944089552990653640447425576083659976645795096,
66024396409905389607120198219976047599490197230297,
64913982680032973156037120041377903785566085089252,
16730939319872750275468906903707539413042652315011,

```
94809377245048795150954100921645863754710598436791,  
78639167021187492431995700641917969777599028300699,  
15368713711936614952811305876380278410754449733078,  
40789923115535562561142322423255033685442488917353,  
44889911501440648020369068063960672322193204149535,  
41503128880339536053299340368006977710650566631954,  
81234880673210146739058568557934581403627822703280,  
82616570773948327592232845941706525094512325230608,  
22918802058777319719839450180888072429661980811197,  
77158542502016545090413245809786882778948721859617,  
72107838435069186155435662884062257473692284509516,  
20849603980134001723930671666823555245252804609722,  
53503534226472524250874054075591789781264330331690]
```

```
from functools import reduce  
from operator import add  
S=reduce(add,L,0)  
print( int(str(S)[0:10]))
```

EULER 14

```
D={}  
D[1]=1  
def fly(n):  
    global D  
    if n in D:  
        return D[n]  
  
    if n%2==0:  
        result = fly(n//2)+1  
    else:  
        result = fly(n*3+1)+1  
  
    D[n]=result  
    return result  
maximum=0  
maxi=0  
for i in range(1,1000001):  
    temp=fly(i)  
    if temp>maximum:  
        maximum=temp  
        maxi=i  
  
print(maximum,maxi)
```

EULER 15

```
N=21  
L=[[0 for j in range(N+1)] for i in range(N+1)]  
L[0][1]=1  
for i in range(1,N+1):  
    for j in range(1,N+1):  
        L[i][j]=L[i-1][j]+L[i][j-1]  
print (L[N][N])
```

EULER 16

```
print (sum(int(x) for x in str(2 <<999)))
```

EULER 17

```
Words="one,two,three,four,five,six,seven,eight,nine,ten,eleven,twelve,thirteen,\
fourteen,fifteen,sixteen,seventeen,eighteen,nineteen,twenty,thirty,forty,fifty,sixty,\
seventy,eighty,ninety,hundredand,onethousand".split(",")
L=[len(word) for word in Words]
Digits=L[0:9]
Teens=L[9:19]
Decas=L[19:27]

ten = sum(Digits)
hundred = sum(Digits+Teens+Decas)+sum((L[i]*9+ten) for i in range(19,27) )
thousand = sum((L[j]+L[27])*99+hundred for j in range(0,9))+hundred
thousand+=L[28]

thousand+=ten+7*9 #one hundred, two hundred..
print( thousand)
```

EULER 18

```
L=[
[75],
[95, 64],
[17, 47, 82],
[18, 35, 87, 10],
[20, 4, 82, 47, 65],
[19, 1, 23, 75, 3, 34],
[88, 2, 77, 73, 7, 63, 67],
[99, 65, 4, 28, 6, 16, 70, 92],
[41, 41, 26, 56, 83, 40, 80, 70, 33],
[41, 48, 72, 33, 47, 32, 37, 16, 94, 29],
[53, 71, 44, 65, 25, 43, 91, 52, 97, 51, 14],
[70, 11, 33, 28, 77, 73, 17, 78, 39, 68, 17, 57],
[91, 71, 52, 38, 17, 14, 91, 43, 58, 50, 27, 29, 48],
[63, 66, 4, 68, 89, 53, 67, 30, 73, 16, 69, 87, 40, 31],
[4, 62, 98, 27, 23, 9, 70, 98, 73, 93, 38, 53, 60, 4, 23]]

for i in range(13,-1,-1):
    for j in range(len(L[i])):
        L[i][j]+=max(L[i+1][j],L[i+1][j+1])
print(L[0][0])
```

EULER 19

```
dayofweek=1

from itertools import cycle

def isLeap(n):
    if n%400 ==0:
        return True
    elif n%100==0:
        return False
    else:
```

```

        return n%4==0

Year=1901
Month = cycle([1,2,3,4,5,6,7,8,9,10,11,12])

sundays=0
m=next(Month)
while Year<2001:
    if m==4 or m==6 or m==9 or m==11:
        dayofweek=(dayofweek+2)%7
    elif m==2:
        if isLeap(Year):
            dayofweek=(dayofweek+1)%7
        else:
            dayofweek=(dayofweek+0)%7
    else:
        dayofweek=(dayofweek+3)%7

    if dayofweek==6:
        sundays+=1
    m=next(Month)
    if m==1:
        Year+=1
print (sundays)

EULER 20

from math import factorial

print (sum(int(c) for c in str(factorial(100))))

EULER 21

from collections import Counter
from functools import reduce
from math import sqrt
from itertools import chain,combinations
from operator import mul

def powerset(iterable):
    s = list(iterable)
    return chain.from_iterable(combinations(s, r) for r in range(len(s)+1))
S={}
def sum_of_divisors(M):
    N=M
    global S
    if N in S:
        return S[N]
    D=Counter()
    i=2
    till=int(sqrt(N))+1
    while i<=till:
        k=0
        while N % i ==0:
            N=N//i
            k+=1
        if k!=0:
            till=int(sqrt(N))+1
            D[i]=k
        i+=1

```

```

if N!=1:
    D[N]=1
S[N] = sum(reduce(mul, t,1) for t in set(powerset(Counter(D).elements())))-M
return S[N] #,D,set(powerset(List(Counter(D).elements())))

s=0
for i in range(2,10000):
    temp=sum_of_divisors(i)
    if i==sum_of_divisors(temp) and i!=temp:
        s+=i+temp
        print(i,temp)
print (s//2)

EULER 22

with open("names.txt","r") as f:
    L=f.read().split(",")
L.sort()

print(sum((i+1)*sum(ord(c)-ord("A")+1 for c in word if c.isalpha()) for i,word in
enumerate(L)))

EULER 23

from collections import Counter
from functools import reduce
from math import sqrt
from itertools import chain,combinations
from operator import mul

def powerset(iterable):
    "powerset([1,2,3]) --> () (1,) (2,) (3,) (1,2) (1,3) (2,3) (1,2,3)"
    s = list(iterable)
    return chain.from_iterable(combinations(s, r) for r in range(len(s)+1))
S={}

def sum_of_divisors(M):
    N=M
    global S
    if N in S:
        return S[N]
    D=Counter()
    i=2
    till=int(sqrt(N))+1
    while i<=till:
        k=0
        while N % i ==0:
            N=N//i
            k+=1
        if k!=0:
            till=int(sqrt(N))+1
            D[i]=k
        i+=1
    if N!=1:
        D[N]=1
    S[N] = sum(reduce(mul, t,1) for t in set(powerset(Counter(D).elements())))-M
    return S[N]

def cant_be_expressed(n):

```

```
i=0
length=len(A)
while i<length and A[i]<(n-11):
    if n-A[i] in Sa:
        return False
    i+=1
return True

A=[] #Abundant numbers list
for i in range(2,28123):
    if sum_of_divisors(i)>i:
        A.append(i)
Sa=set(A) #Abundant numbers set
Sum=0

print("starting main part")
for i in range(1,28123):
    if cant_be_expressed(i):
        Sum+=i
        if i<500:
            print(i)
print(Sum)

EULER 24

from itertools import permutations

p=permutations([0,1,2,3,4,5,6,7,8,9],10)

for i in range(1000000):
    k=next(p)
    print(k)

EULER 25

a=1
b=1
n=2
while len(str(b))<1000:
    a,b=b,a+b
    n+=1
print (n)

EULER 26

m=0
max_period_num=0
for p in range(3,1000,2):
    if p%5==0:
        continue

    period=1
    while pow(10,period,p)!=1: #Calculating multiplicative order
        period+=1
    if period>m:
```

```

        m=period
        max_period_num=p
print (m,max_period_num)

```

EULER 27

```

from EulerLib import is_prime

# 168th prime is equal to 997
# b must be a positive prime less than 1000
# so there are only 168 possible values for b

P=[] #List of first 1000 prime numbers
for i in range(2,10000):
    if is_prime(i):
        if len(P)<1000:
            P.append(i)
        else:
            break
ma=mb=0
max_prime=1
for b in P[0:168]:
    for a in range(-999,1000):
        if a==0 : continue
        len_prime=1
        for i in range(1,1000):
            if not is_prime(i*i+i*a+b):
                break
            else:
                len_prime+=1
        if len_prime>max_prime:
            ma=a
            mb=b
            max_prime=len_prime

print(ma,mb,ma*mb,max_prime)

```

EULER 28

```

#Basic series are like this, for each diagonal direction:
#1+3*3+5*5+7*7+9*9+.. (2n-1)^2 = n(2n-1)(2n+1)/3
#1+3*3-2+5*5-4+7*7-6+... =
#1+3*3-4+5*5-8+7*7-12+..
#1+3*3-6+5*5-12+7*7-18+..
# Summing them together:
# 4*n(2n-1)(2n+1)/3 - 6*n(n-1)-3, where N=2*n-1

N=1001 #rows and columns each
n=(N+1)//2
print (4*n*(2*n-1)*(2*n+1)//3 - 6*n*(n-1)-3)

```

EULER 29

```

from EulerLib import decomposition

L=[]
print (decomposition(50))
for i in range(2,101):
    L.append(list(decomposition(i).items()))

```

```
print (L)

LL=set()
for a in range(len(L)):
    for b in range(2,101):
        for k in range(len(L[a])):
            LL.add(tuple([ (L[a][k][0],L[a][k][1]*b) for k in range(len(L[a]))]))

print (len(LL))

EULER 30

N=9**5*6 #Limit

ssum=0
for i in range(2,N):
    s=0
    for c in str(i):
        s+=int(c)**5
    if s==i:
        ssum+=s

print (ssum)

EULER 31

def F125(N):
    ssum=0
    for k in range(0,N+1,5):
        ssum+=k//2+1
    return ssum

def F(L,N):
    if len(L)==3:
        return F125(N)
    else:
        ssum=0
        for i in range((N//L[-1])+1):
            ssum+=F(L[:-1],N-i*L[-1])
        return ssum

print(F([1,2,5,10,20,50,100,200],200))

EULER 32

def is_pandigital(a,b,c):
    s="".join([str(a),str(b),str(c)])
    return sorted(s)==list("123456789")

L=[]
for i in range(1000,10000):
    for j in range(2,100):
        if i%j==0:
            if is_pandigital(j,i//j,i):
                L.append([j,i//j,i])

print(L,sum(set(x[2]for x in L)))

EULER 33
```

```

from EulerLib import Reduce,product
from collections import Counter

L=[]
for i in range(10,100):
    for j in range(10,i):
        si,sj=str(i),str(j)
        t=Counter(si) & Counter(sj)
        if sum(t.values())==1 and t!=Counter({"0":1}):
            z1=int("".join(list((Counter(si)-t).elements()))))
            z2=int("".join(list((Counter(sj)-t).elements()))))
            if Reduce(z2,z1)==Reduce(j,i):
                L.append(Reduce(j,i))

print(Reduce(product([x[0] for x in L]),product([x[1] for x in L]))[1])

```

EULER 34

```

from math import factorial
from itertools import combinations_with_replacement

Factorial={x : factorial(int(x)) for x in "0123456789"} #precomputing factorials
result=0

#There are no 1 digit results.
#The upper limit is 2540160(9!*7), which is 7 digits.
for n in range(2,7):
    for i in combinations_with_replacement("0123456789",n):
        s=sorted("".join(i))
        t =sum(Factorial[k] for k in s )
        if s==sorted(str(t)):
            result+=t

print(result)

```

EULER 35

```

from itertools import combinations_with_replacement, permutations
from EulerLib import is_prime

def rotations(s):
    return [s[i:]+s[:i] for i in range(1,len(s))]

def main():
    count=len([2,3,5,7]) #digit solutions
    for digits in range(2,7):
        for i in combinations_with_replacement("1379",digits):
            s=["".join(k) for k in set(permutations(i))]
            s=[ k for k in s if is_prime(int(k))]
            for j in s:
                if all([(k in s) for k in rotations(j)]):
                    count+=1

    print(count)

```

EULER 36

```

ssum=0
for i in range(1,1000):
    s=str(i)
    num1=int(s[:-1]+s[::-1])
    num2=int(s+s[::-1])
    b1="{0:b}".format(num1)
    b2="{0:b}".format(num2)
    if b1==b1[::-1]:
        ssum+=num1
    if b2==b2[::-1]:
        ssum+=num2

```

```
print(ssum)
```

EULER 37

```
from EulerLib import is_prime
```

```
def truncations(s):
    llen=len(s)
    return [s[i:] for i in range(1,llen)]

```

```
count=0
ssum=0
L=[]

```

```
def F(depth,s):
    global count,ssum,L
    if not is_prime(int(s)):
        return None
    if depth==N:
        if all(is_prime(int(x)) for x in truncations(s)) and len(s)>1:
            L.append(int(s))
            count+=1
            ssum+=int(s)
    else:
        for i in "1379":
            F(depth+1,s+i)

```

```
for N in range(1,50):
    for i in "2357":
        F(1,i)

```

```
print(count,L,ssum)
```

EULER 38

```

# 9 > n > 1
# n=9 -> (nine 1s) -> 1(solution)
# n=8 -> (seven 1s, one 2) -> None
# n=7 -> (five 1s, two 2s) -> None
# n=6 -> (three 1s, three 2s) -> 3
# n=5 -> (one 1, 4 2s) -> 7,8, (9)(solution)
# n=4 -> (three 2s, one 3 ) ->
# 33>=x>=25
# n=3 -> (three 3s)
# 100<=x<=333
# n=2 -> (one 4, one 5)

```

```

# 9999>=x>=5000

#for n=2, we check only numbers in range 9999>=x>=5000
import operator
L=[[123456789,1,9],[918273645,9,5]] # solutions for n=1 and n=5
def F(n,I):
    global L
    R=range(1,n+1)
    for x in I:
        s=""
        s="".join(str(x*i) for i in R)
        if "".join(sorted(s))=="123456789":
            L.append([int(s),x,n])

F(2,range(5000,10000))
F(3,range(100,334))
F(4,range(25,34))

prini+=1

EULER 39

import operator
N=1000
L=[0 for i in range(N)]

for p in range(1,N):
    for a in range(1,p):
        b = (p*p-2*a*p)/(2*p-2*a) #Algebra magic
        if b<2:
            continue
        t=(p-a-b)
        if a*a+b*b==t*t:
            L[p]+=1

print(max(enumerate(L), key=operator.itemgetter(1)))

EULER 40

#F(n) = F(n-1)+(n-1)*9*10^(n-2) : 1, 10, 190, 2890, 38890, 488890
from EulerLib import memoize
from bisect import bisect

N=100 # Till what digit numbers shall we investigate

@memoize
def F(n):
    "returns the position at which numbers with n digits start"
    if n==1:
        return 1
    else:
        return F(n-1)+(n-1)*9*10**(n-2)

L=[F(n) for n in range(1,N+1)]

product = 1
for n in (10**x for x in range(7)):
    digits = bisect(L,n)
    startplace = L[digits-1]

```

```

lastnum = 10**(digits - 1) - 1

s = str(lastnum + (n - startplace) // digits + 1)
place=(n - startplace)% digits
product*=int(s[place])
print(s[place],s,place)
print(product)
# print("n={},digits={},startplace={},lastnum={},s={}".format(n,digits,startplace,
lastnum,s))

##string=""
##for i in range(1,250000):
##    string+=str(i)
##
##for i in [10**x for x in range(7)]+[1,9,10,189,190,1989,1990,19989,19990,199989,
199990]:
##    print(i,string[i-1])
##

```

EULER 41

```

from itertools import permutations
from EulerLib import is_prime

maximum=0
for n in range(2,10):
    for i in permutations(str(i) for i in range(1,n)):
        t=int("".join(i))
        if is_prime(t):
            if t>maximum:
                maximum=t

print(maximum)

```

EULER 42

```

L=[]
with open("words.txt") as f:
    L=f.read().split(",")
S={(i*(i+1))/2 for i in range(1000)} #Set of triangular numbers
ordA=ord("A")
T=[sum( ord(char)-ordA+1 for char in word[1:-1] ) for word in L] #number for each word

print(T)
print(sum(1 for i in range(len(L)) if T[i] in S))

```

EULER 43

```

ssum=0
Stack=[] # tuples of (location, num_string)
digits={"1","2","3","4","5","6","7","8","9","0"}
Result=[]

ListOfDicts=[] # contains paths from each 2 digits to next one, depending on location
#Generate ListOfDicts
for division in [2,3,5,7,11,13,17]:
    Dict={}

```

```

for j in range(0,100):
    strj=str(j) if j>=10 else "0"+str(j)
    List=[]
    for n in range(4):
        mod= (j*10) % division
        digit=division*n-mod
        if 10>digit>=0:
            List.append(str(digit))
    Dict[strj]=List
    ListOfDicts.append(Dict)

for i in range(100):
    Stack.append((0,str(i) if i>=10 else "0"+str(i)))

while Stack:
    location, num_string=Stack.pop()
    if location==7:
        first_digit_set=digits - set(num_string)
        if len(first_digit_set)==1:
            if first_digit_set!={0}:
                number=int(list(first_digit_set)[0]+num_string)
                ssum+=number
                Result.append(number)
    else:
        for i in ListOfDicts[location][num_string[-2:]]:
            Stack.append((location+1,num_string+i))

print(Result,ssum)

```

EULER 44

```

#I only find the first D, but it also seems to be the minimal one
pentagonal_nums=[]
for i in range(1,1000):
    pentagonal_nums.append((i*(3*i-1))//2)

pent_set=set(pentagonal_nums)
for i in pentagonal_nums:
    for j in pentagonal_nums[0:i]:
        if ((i+j) in pent_set) and ((j-i) in pent_set):
            print (j-i)

```

EULER 45

```

# We will check each hexagonal number for whether it is also triangular and pentagonal

from math import sqrt
from EulerLib import float_equal

def is_triangular(x):
    num=(1/2) * (sqrt(8*x+1)-1 )
    return float_equal(num,int(num))

def is_pentagonal(x):
    num=(1/6) * (sqrt(24*x+1)+1)
    return float_equal(num,int(num))

```

```
for i in range(100000):
    num=i*(2*i-1)
    if is_triangular(num) and is_pentagonal(num):
        print (num)
```

EULER 46

```
from EulerLib import is_prime,float_equal,memoize
import math
```

N=10000 #Till where to check

```
@memoize
def memoized_is_prime(x):
    return is_prime(x)

for num in range(3,N,2):
    found_decomposition=False
    if not memoized_is_prime(num):
        for i in range(1,int(math.sqrt(num))+1):
            t=2*i*i
            if memoized_is_prime(num-t):
                found_decomposition=True
                break
        if not found_decomposition:
            print(num,i,t)
```

EULER 47

```
from collections import deque
from EulerLib import Time
```

```
def mesh(N):
    L=[0 for i in range(N)]
    for i in range(2,N//2):
        if L[i]==0:
            L[i+i::i] = [x+1 for x in L[i+i::i]]
    return L
```

```
def main():
    N=150000 # Till where to check
    SEQ=4 #how many consequent integers to track, and how many unique divisors
           #should they have
    Num_of_unique_divisors=mesh(N)
    Deck=deque([],SEQ) # maxlen is SEQ, so deque is always of maximum size SEQ
    for num in range(2,N):
        t=Num_of_unique_divisors[num]
        Deck.append(t)
        if all(i==SEQ for i in Deck) and len(Deck)==SEQ:
            print(num-SEQ+1)
```

Time(main)

EULER 48

```
digits=10**10
ssum=0
```

```
for i in range(1,1001):
    num=pow(i,i,digits)
    ssum+=num
```

```
print(str(ssum)[-10:])
```

EULER 49

```
from bisect import bisect
from EulerLib import sieve
from itertools import permutations
```

```
P=sieve(10000)
start_index = bisect(P.primes,1000)
```

```
Result=set()
for num in P.primes[start_index:]:
    perms=[int("".join(i)) for i in permutations(str(num)) if i[0]!="0"]
    perms_set=set(filter(lambda x: P.is_prime[x], perms))
    perms=sorted(list(perms_set))
    for i,a1 in enumerate(perms):
        for a2 in perms[i+1:]:
            if (2*a2-a1) in perms_set:
                Result.add((a1,a2,2*a2-a1))
```

```
print(Result)
```

EULER 50

```
from EulerLib import sieve
```

```
N=1000000
Sieve=sieve(N)
num_primes=len(Sieve.primes) #about 78000 primes are below 1,000,000
```

```
max_len=0
ssum=0
while ssum<N:
    ssum+=Sieve.primes[max_len]
    max_len+=1
max_len-=1 # 547 is the maximum length of the chain of primes
           # for the sum to be less than 1,000,000
```

```
def partial_sums(length):
    "Returns a set of partial sums of length 'length'"
    part_sum=set()
    ssum=0
    for i in range(length):
        ssum+=Sieve.primes[i]
        part_sum.add(ssum)

    for i in range(length,num_primes):
        ssum=ssum+Sieve.primes[i]-Sieve.primes[i-length]
        if ssum<=N:
            part_sum.add(ssum)
    return part_sum
```

```
found_solution=False
```

```
solution=0,0
for length in range(max_len,2,-1):
    for i in partial_sums(length):
        if Sieve.is_prime[i]:
            found_solution=True
            solution=i,length
            break
    if found_solution:
        break

print(solution)
```

EulerLib

```
from collections import Counter
from functools import reduce
from math import sqrt
from itertools import chain, combinations
from operator import mul
import sys
import time

class memoize(object): # A memoization decorator, taken from python.org
    """Decorator that caches a function's return value each time it is called.
    If called later with the same arguments, the cached value is returned, and
    not re-evaluated.
    """
    def __init__(self, func):
        self.func = func
        self.cache = {}
    def __call__(self, *args):
        try:
            return self.cache[args]
        except KeyError:
            value = self.func(*args)
            self.cache[args] = value
            return value
        except TypeError:
            # uncacheable -- for instance, passing a List as an argument.
            # Better to not cache than to blow up entirely.
            return self.func(*args)
    def __repr__(self):
        """Return the function's docstring."""
        return self.func.__doc__
    def __get__(self, obj, objtype):
        """Support instance methods."""
        return functools.partial(self.__call__, obj)

@memoize
def sum_of_divisors(N):
    """Returns sum of all divisors of M, 1 and M included. Caches known values"""
    def powerset(iterable): # This little function is taken from python recipes on
python.org
        s = list(iterable)
        return chain.from_iterable(combinations(s, r) for r in range(len(s)+1))

D=Counter()
```

```
i=2
till=int(sqrt(N))+1
while i<=till:
    k=0
    while N % i ==0:
        N=N//i
        k+=1
    if k!=0:
        till=int(sqrt(N))+1
        D[i]=k
    i+=1
if N!=1:
    D[N]=1
return sum(reduce(mul, t,1) for t in set(powerset(Counter(D).elements())))
```

```
def decomposition(M):
    "Returns primary decomposition of integer in a form of Counter object"
    N=M
    D=Counter()
    i=2
    till=int(sqrt(N))+1
    while i<=till:
        k=0
        while N % i ==0:
            N=N//i
            k+=1
        if k!=0:
            till=int(sqrt(N))+1
            D[i]=k
        i+=1
    if N!=1:
        D[N]=1
    return D
```

```
def is_prime(n):
    if n<=1:
        return False
    if n==2 or n==3 or n==5 or n==7:
        return True
    if n%2==0:
        return False
    for i in range(3,int(sqrt(n))+1,2):
        if n%i==0:
            return False
    return True
```

```
def product(L):
    return reduce(mul, L, 1)
```

```
def GCD(a,b):
    "Greatest Common Divisor by Euclid's algorithm"
    a,b=abs(a),abs(b)
    a,b = max(a,b),min(a,b)

    if a==0:
        return -1
    else:
        while b!=0:
            a,b=b,a%b
```

```
    return a

def Reduce(a,b):
    t=GCD(a,b)
    return (a//t,b//t)

def float_equal(a,b): #Checking for float equality
    return abs(a - b) <= sys.float_info.epsilon

class sieve(object): #Works for N=1,000,000 under a second
    """Has two attributes: primes and grid. primes is a list of primes till N,
    is_prime is a list of booleans: is_prime[i] is True if i is prime.
    """
    def __init__(self,N):
        __IsPrime__=[True for i in range(N+1)]
        __Primes__=[]
        for i in range(2,int(sqrt(N))+1):
            if __IsPrime__[i]:
                __IsPrime__[i+i::i]=[False for x in __IsPrime__[i+i::i]]
        for i in range(2,N+1):
            if __IsPrime__[i]:
                __Primes__.append(i)

        self.primes=__Primes__
        self.is_prime=__IsPrime__
        self.position={prime : i for (i,prime) in enumerate(__Primes__)}

def Time(f):
    t0 = time.time()
    f()
    print(time.time() - t0, "seconds")
```