Ποιο είναι το άθροισμα της τετραγωνικής ρίζας όλων των αριθμών από το 1 μέχρι το 3000 που διαιρούνται με το 3 αλλά όχι με το 6;

In [5]:
```
import math

def f():
    s = 0
    for i in range(1, 3001):
        if not (i%3==0 and i%6!=0):
            continue

        s += math.sqrt(i)

    return s

f()
```

Out[5]: 18257.565447000314

In [7]:
```
import math

def f():
    s = 0
    for i in range(1, 3001):
        if i%3==0 and i%6!=0:
            s += math.sqrt(i)

    return s

f()
```

Out[7]: 18257.565447000314

In [4]:
```
sum(math.sqrt(i) for i in range(1,3001) if i%3==0 and i%6!=0 )
```

Out[4]: 18257.565447000314

Όλα τα ζευγάρια κ,λ θετικών ακεραίων που είναι μικρότεροι ή ίσοι με το 10 όπου το λ διαιρεί ακριβώς το κ είναι:

(4, 2), (6, 2), (6, 3), (8, 2), (8, 4), (9, 3), (10, 2), (10, 5)

Το άθροισμα των διαφορών αυτώ των ζευγαριών είναι:

(4 - 2) + (6 - 2) + (6 - 3) + (8 - 2) + (8 - 4) + (9 - 3) + (10 - 2) + (10 - 5) = 38

Ποιο είναι το αντίστοιχο άθροισμα διαφορών αν πάρουμε όλα τα ζευγάρια κ,λ που είναι μικρότεροι η ίσοι με το 1000;

In [12]:
```
s = 0
for x in range(1,1001):
    #print (x)
    for y in range(2,x):
        if x%y == 0:
            #print (x,y)
            s += x-y
print (s)
```

```
2465073
```

In [13]:
```
sum(x-y for x in range(1,1001) for y in range(2,x) if x%y == 0 )
```

Out[13]:
```
2465073
```

In [17]:
```
a = ['kwstas', 'mitsos', 'elenh', 'maria']

for x in range(len(a)):
    #for y in range(x+1, len(a)):
    for y in range(x, len(a)):
                    print (a[x], a[y])
```

```
kwstas kwstas
kwstas mitsos
kwstas elenh
kwstas maria
mitsos mitsos
mitsos elenh
mitsos maria
elenh elenh
elenh maria
maria maria
```

In [19]:
```
list(range(5,10))
```

Out[19]:
```
[5, 6, 7, 8, 9]
```

Φτιάξτε μία συνάρτηση με το όνομα f η οποία να επιστρέφει μία συνάρτηση η οποία να επιτρέφει μία συνάρτηση η οποία να επιστρέφει μία λίστα της οποίας το 2ο στοιχείο να είναι μία συνάρτηση η οποία να επιστρέφει το string 'mitsos'. Θα πρέπει δηλαδή να μπορώ να κάνω:

```
f()()()[1]()
```

Αυτό πρέπει να Τυπώνει: 'Μήτσος'

In [26]:
```
def t():
    return "mhtsos"

def f():
    def g():
        def h():
            #return [1, t]
            return [1, t]

        return h

    return g
```

In [27]:
```
f()()()[1]()
```

Out[27]: 'mhtsos'

In [28]:
```
f = lambda : lambda : lambda : [1, lambda : 'mhtsos']
```

In [29]:
```
f()()()[1]()
```

Out[29]: 'mhtsos'

In [30]:
```
def f(x):
    return x/2
```

In [31]:
```
f(10)
```

Out[31]: 5.0

In [ ]:
```
f = lambda x : x/2
```

In [32]:
```
(lambda x : x/2)(100) # lambda , Nameless functions
```

Out[32]: 50.0

In [33]:
```
def f():
    return 5
```

In [34]:
```
f = lambda : 5
```

In [35]:
```
f()
```

Out[35]: 5

In [37]:
```
(lambda : 5)()
```

Out[37]: 5

Φτιάξτε μία συνάρτηση με τον όνομα f η οποία να επιστρέφει μία λίστα με 10 στοιχεία.

Το πρώτο στοιχείο της λίστας θα είναι μία συνάρτηση η οποία θα παίρνει σαν όρισμα έναν αριθμό και θα επιστρέφει το άθροισμά του με το 1.

Το δεύτερο στοιχείο της λίστας θα είναι μία συνάρτηση η οποία θα παίρνει σαν όρισμα έναν αριθμό και θα επιστρέφει το άθροισμά του με το 2

κ.ο.κ.

Το 10ο στοιχείο της λίστας θα είναι μία συνάρτηση η οποία θα παίρνει σαν όρισμα έναν αριθμό και θα επιστρέφει το άθροισμά του με το 10.

Θα πρέπει δηλαδή να μπορώ να γράψω:

```
f()[5](4) --> Τυπώνει 4+6 = 10
f()[0](2) --> Τυπώνει 1+2 = 3
```

In [49]:
```
def f():

    def g(y):
        return lambda x : x+y

    l = []
    for x in range(1, 11):

        l.append(g(x))


    return l

f()[0](1)
```

Out[49]: 2

Έστω το παρακάτω string:

'++-+--+-+++++++++++++-+-++++++-++--++-++++-+---++-++-+--++---++-+-++-------+-+++---+---++-+-+-

Φτιάξτε μια συνάρτηση που θα παίρνει μία παράμετρο ένα string. Στη συνέχεια θα επιστρέφει τη θέση στο string που βρίσκεται η μεγαλύτερη ακολουθία από συνεχόμενα '+'. Αν η μεγαλύτερη ακολουθία υπάρχει παραπάνω από μία φορά, επιστρέφει τη πιο μικρή θέση. Για παράδειγμα:

'+--+-+-----+-+--+----+--+---+++-+-+---+-----++----+-++--++++++-+-----++---+--+-+-+-++++--++-------+
--> 56
'+-+-++--+----+-+---+-+-+---+----+-++-+-+--++++-++++-+++--++++--+----+----+-+++--+-++-+---++-+-----
--> 42

In [75]:
```
def f(x):


    y = x.split('-')
    m = max(y, key=len)
    return x.index(m)

a =
'+--+-+-----+-+--+----+--+---+++-+-+---+-----++----+-++--++++++-+--
---++---+--+-+-+-+++++--++-------+'

f(a)
```

Out[75]: 56

In [76]:
```
f = lambda x: x.index(max(x.split('-'), key=len))
```

In [77]:
```
f(a)
```

Out[77]: 56

In [78]:
```
l = ['a', 'bbbb', 'cc']
max(l)
```

Out[78]: 'cc'

In [79]:
```
l = ['a', 'bbbb', 'cc']
max(l, key=len)
```

Out[79]: 'bbbb'

In [80]:
```
l = ['a', 'bbbb', 'cc']
max(l, key = lambda x: len(x))
```

Out[80]: 'bbbb'

In [ ]:

In [65]:
```
'+' < '++'
```

Out[65]: True

In [67]:
```
'abcdefg'.index('ef')
```

Out[67]: 4

In [ ]:

In [59]:
```
'11112111111112111111'.split('2')
```

Out[59]: ['1111', '11111111', '111111']

In [60]:
```
'asdf asdf asdf asdf as'.split(' ')
```

Out[60]: ['asdf', 'asdf', 'asdf', 'asdf', 'as']

Έστω το παρακάτω string:

'++-+--+-+++++++++++++-+-+++++-++--++-++++-+---++-++-+--++---++-+-++-------+-+++---+---++-+-+-

Και έστω η παρακάτω λέξη:

ZABARAKTRANEMIA

Φτιάξτε μία συνάρτηση η οποία θα παίρνει δύο παραμέτρους. Ένα string μόνο με '+' και '-' και ένα άλλο string με οποιουσδίποτε χαρακτήρες. Η συνάρτησή σας θα πρέπει να αντιστοιχείσει το 2ο string στο 1ο. Με τον εξής τρόπο: Όπου υπάρχει το πρώτο '-' αντιστοιχεί το 1ο γράμμα του 2ου string. Όπου υπάρχει το δεύτερο '-' αντιστοιχεί το 2ο γράμμα του 2ου string κτλ. Αν υπάρχει '+' τότε θα πρέπει να βάζει κενό. Δεν χρειάζεται να βάζετε κενά μόλις τελειώσει το 2ο string. Για παράδειγμα:

ΕΙΣΟΔΟΣ:
'++-+--+-+++++++++++++-+-+++++-++--++-++++-+---++-++-+--++---++-+-++-------+-+++---+---++-+-+-
'ZABARAKATRANEMIA'

ΕΞΟΔΟΣ Z AB A R A K AT R A NEM I A

ΕΞΗΓΗΣΗ
'++-+--+-+++++++++++++-+-+++++-++--++-++++-+---++-++-+--++---++-+-++-------+-+++---+---++-+-+-
Z AB A R A K AT R A NEM I A

In [82]:
```
def f(l,m):

    r = ''
    m_letter = 0
    for x in l:
        if x == '+':
            r += ' '
        elif x == '-':
            r += m[m_letter]
            m_letter += 1

        if m_letter == len(m):
            break
    return r

a =
'++-+--+-+++++++++++++-+-+++++-++--++-++++-+---++-++-+--++---++-+-+
+-------+-+++---+---++-+-+++-+-+++'
b = 'ZABARAKATRANEMIA'

f(a,b)
```

Out[82]: ' Z AB A   R A   K AT R   A NEM I A'

In [90]:
```
def f(l):

    r = ''
    for x in range(0, len(l[0])):
        letter = l[0][x]
        #print (letter)

        if True:
            if all(y[x] == letter for y in l[1:]):
                r += letter
            else:
                break

        if False:
            ok = True
            for y in l[1:]:
                if y[x]!=letter:
                    ok = False
            if ok:
                r += letter
            else:
                break

    return r


l = [
   'Alekos',
   'Aleksandra',
   'Aleksandros'
]

f(l)
```

Out[90]:
```
'Alek'
```

Φτιάξτε μία συνάρτηση η οποία θα παίρνει 1 παραμέτρο, ν ακέραιος. Η συνάρτηση θα πρέπει να επιστρέφει ένα string το οποίο θα είναι το ανάπτυγμα της έκφρασης (α+β)^ν. Παραδείγματα:

n=10 Δηλαδή (α+β)^10

```
--> a^10 + 10a^9b + 45a^8b^2 + 120a^7b^3 + 210a^6b^4 + 252a^5b^5 + 210a
^4b^6 + 120a^3b^7 + 45a^2b^8 + 10ab^9 + b^10
```

Χρησιμοποιείστε τη φόρμουλα που περιγράφεται εδώ: https://en.wikipedia.org/wiki/Binomial_theorem (https://en.wikipedia.org/wiki/Binomial_theorem) για να βρείτε τους παράγοντες του κάθε γινομένου.

In [276]:
```
def fact(x):

    if x==0:
        return 1
    if x==1:
        return 1
    return x*fact(x-1)

def combinations(n,k):
    return int(fact(n)/ (fact(k)*fact(n-k)))

def f(n):

    ret = []
    for x in range(n+1):
        s1 = str(combinations(n,x))
        if s1 == '1':
            s1 = ''

        if x==0:
            s2 = ''
        elif x == 1:
            s2 = 'a'
        else:
            s2 = 'a^{}'.format(x)

        if n-x==0:
            s3 = ''
        elif n-x == 1:
            s3 = 'b'
        else:
            s3 = 'b^{}'.format(n-x)

        ret.append(s1+s2+s3)

    return '+'.join(ret)



#f(2) # (a+b)^2 --> a^2 + 2ab + b^2
#f(3) # (a+b)^3 --> a^2 + 3a^2b + 3ab^2 + b^3
f(10)
```

Out[276]:
```
'b^10+10ab^9+45a^2b^8+120a^3b^7+210a^4b^6+252a^5b^5+210a^6b^4+120a
^7b^3+45a^8b^2+10a^9b+a^10'
```

Έστω το παρακάτω string:

'aahtoootoootttuuuu----------o'

Υπάρχει ενας τρόπος να "συμπιέσουμε" αυτό το string, γράφοντας κάθε γράμμα που επαναλαμβάνετε σκολουθούμενο με το πόσες φορές επαναλμβάνεται. Αν το υλοποιήσουμε τότε το string αυτό μπορει να γραφεί ως εξής:

'a2h1t1o3t1o3t2u4-11o1'

Φτιάξτε μία συνάρτηση με τον όνομα compress η οποία θα παίρνει ως παράμετρο ένα string s. Η συνάρτηση θα επιστρέφει ένα άλλο string το οποίο θα είναι η συμπίεση του s σύμφωνα με τον τρόπο που παρουσιάστηκε. Υποθέστε ότι το s δεν θα έχει αριθμούς μέσα.

Φτιάξτε μία συνάρτηση με τον όνομα decompress η οποία θα παίρνει ένα συμπιεσμένο string και θα επιστρέφει το αρχικό, αποσυμπιεσμένο string.

In [278]:
```
def compress(x):
    ret = ''

    rest = x

    while rest:

        first_letter = rest[0]

        counter = 0
        while rest[counter]==first_letter:
            counter += 1
            if counter == len(rest):
                break

        #print (counter)
        ret += '{}{}'.format(first_letter, str(counter))

        rest = rest[counter:]
        #print ('rest:', rest)

    return ret
    #print (ret)


compress('aahtooootoootttuuuu----------o') # Must return a2h1t1o3t1o3t2u4-11o1
```

Out[278]: 'a2h1t1o3t1o3t3u4-11o1'

In [280]:
```
def decompress(x):

    ret = ''

    while x:
        first_letter = x[0]
        next_letter = 1

        num_str = ''
        while x[next_letter] in '0123456789':
            num_str += x[next_letter]
            next_letter += 1
            if next_letter == len(x):
                break

        num = int(num_str)

        ret += first_letter * num
        #print (ret)
        rest = x[next_letter:]
        #print (rest)
        x = rest

    return ret


decompress('a2h1t1o3t1o3t2u4-11o1') # aahtooootoootttuuuu----------
o
```

Out[280]:
```
'aahtoootooottuuuu----------o'
```

Ένας παίκτης παίζει 100 παρτίδες πόκερ. Σε κάθε παρτίδα σημειώνει πόσα λεφτά κέρδισε ή έχασε. Τα ποσά αυτά τα σημειώνει σε μία λίστα. π.χ.:

```
[31, -28, 14, -12, -4, 44, 47, 2, -48, -5, -43, 32, 0, -4, 24, -46, -1
2, 38, -38, -27, -23, -26, 10, 42, 26, -20, -43, -50,
2, 42, 32, 17, -33, 5, 42, 28, 2, 12, 9, -33, 22, 10, 3, 34, 12, 17, 2
1, 17, 24, 22, 21, -35, 33, 12, -43, 49, -17, 3, -2,
-25, -29, -35, -26, -25, -22, -33, 10, 26, -41, 29, 6, -10, 15, -28, -2
3, -35, -1, -16, 24, -45, -50, -17, 20, 12, -32, 48,
-48, 2, -41, 4, 5, 29, -36, -46, -6, -17, -18, 16, 42, 42]
```

Σε ποια παρτίδα ο παίκτης είχε τα περισσότερα χρήματα; Δηλαδή στη πρώτη παρτίδα ο παίκτης είχε 31. Στη δεύτερη είχε 31-28=3. Στη τρίτη είχε 31-28+14=17. κτλ. Όπως βλέπετε υπάρχουν παρτίδες που ο παίκτης έχασε χρήματα.

Φτιάξτε μία συνάρτηση η οποία θα παίρνει ως παράμετρο μία λίστα από ακέραιους. Θα επιστρέφει τη θέση στην οποία το άθροισμα όλων των αριθμών από την αρχή μέχρι εκείνη τη θέση είναι ο μεγαλύτερος. Για παράδειγμα η απάντηση στη λίστα που δόθηκε είναι 55 (ξεκινώντας από το 0).

In [281]:
```
def f(l):

    s = 0
    new_list = []
    for index, x in enumerate(l):
        s += x
        new_list.append((s, index))
        #print (s)
    print (max(new_list))

l = [31, -28, 14, -12, -4, 44, 47, 2, -48, -5, -43, 32, 0, -4, 24,
-46, -12, 38, -38, -27, -23, -26, 10, 42, 26, -20, -43, -50,
2, 42, 32, 17, -33, 5, 42, 28, 2, 12, 9, -33, 22, 10, 3, 34, 12, 1
7, 21, 17, 24, 22, 21, -35, 33, 12, -43, 49, -17, 3, -2,
-25, -29, -35, -26, -25, -22, -33, 10, 26, -41, 29, 6, -10, 15, -2
8, -23, -35, -1, -16, 24, -45, -50, -17, 20, 12, -32, 48,
-48, 2, -41, 4, 5, 29, -36, -46, -6, -17, -18, 16, 42, 42]
f(l)
```

```
(225, 55)
```

Θεωρήστε τον ίδιο παίκτη πόκερ. Ποιες είναι οι συνεχόμενες παρτίδες που είχε το μεγαλύτερο κέρδος; Φτιάξτε μία συνάρτηση η οποία θα παίρνει σαν παράμετρο μία λίστα με ακέραιους. Η συνάρτηση θα επιστρέφει το υποσύνολο της λίστας από συνεχόμενες τιμές που έχει το μεγαλύτερο άθροισμα. Η απάντηση στη λίστα του παραδείγματος είναι: (28,56). Δηλαδή το άθροισμα όλων των στοιχείων της λίστα απο το 28 μέχρι το 56 είναι και το μεγαλύτερο δυνατό.

In [148]:
```
def f(l):
    k = []
    for x in range(len(l)):
        for y in range(x+1, len(l)):
            k.append(   (sum(l[x:y+1]), (x,y))     )

    return max(k)

f(l)
```

Out[148]:
```
(344, (28, 55))
```

In [149]:
```
l[28:56]
```

Out[149]:
```
[2,
 42,
 32,
 17,
 -33,
 5,
 42,
 28,
 2,
 12,
 9,
 -33,
 22,
 10,
 3,
 34,
 12,
 17,
 21,
 17,
 24,
 22,
 21,
 -35,
 33,
 12,
 -43,
 49]
```

In [282]:
```
'1' + '1'
```

Out[282]:
```
'11'
```

Ας υποθέσουμε ότι έχουμε ένα γονίδιο το οποίο έχει μέγεθος 100 νουκλεοτίδια. Από το γονίδιο αυτό παίρνουμε 20 ακολουθίες. Η κάθε μία ξεκινάει και τελειώνει στις θέσεις που φαίνονται παρακάτω:

```
[(22, 34), (66, 75), (35, 46), (45, 59), (77, 87), (38, 58), (51, 58),
(81, 90), (52, 70), (53, 65),
(53, 72), (50, 63), (80, 100), (0, 12), (68, 81), (35, 51), (27, 34),
(69, 87), (39, 47), (0, 8)]
```

Ποιες θέση του γονιδίου έχουν τη μεγαλύτερη "κάλυψη". Δηλαδή η ακολουθία τους έχει διαβαστεί τις περισσότερες φορές; Φτιάξτε μία συνάρτηση η οποία θα παίρνει μία λίστα από ζευγάρια από αριθμούς. Η συνάρτηση θα επιστρέφει μία λίστα με τις θέσεις που έχουν τη μεγαλύτερη κάλυψη. Για το παράδειγμα που έχει δοθεί η απάντηση είναι:

```
[53, 54, 55, 56, 57]
```

In [164]:
```
def f(l):

    d = {}

    for x,y in l:
        for z in range(x,y+1):

            if not z in d:
                d[z] = 0

            d[z] += 1

    max_location = max(d, key=d.get)
    max_coverage =  d[max_location]
    #print (max_coverage)

    if True:
        return [k for k,v in d.items() if v==max_coverage]

    if False:
        new_list = []
        for k,v in d.items():
            #print (k,v)
            if v == max_coverage:
                new_list.append(k)
        return new_list

    #sorted(d.items(), key=lambda x : x[1])

    #print (d)


l = [(22, 34), (66, 75), (35, 46), (45, 59), (77, 87), (38, 58), (5
1, 58), (81, 90), (52, 70), (53, 65),
(53, 72), (50, 63), (80, 100), (0, 12), (68, 81), (35, 51), (27, 3
4), (69, 87), (39, 47), (0, 8)]

f(l)
```

Out[164]:
```
[53, 54, 55, 56, 57, 58]
```

Έστω η λίστα με τα παρακάτω strings:

```
b = [
    'zabarakatranemia',
    'askardamikth',
    'mpampesika',
]
```

Ποια είναι τα γράμματα που υπάρχουν σε όλα τα strings της λίστας; Φτιάξτε μία συνάρτηση που θα παίρνει σαν παράμετρο μία λίστα με strings. Θα επιστρέφει μία λίστα με όλα τα γράμματα που υπάρχουν σε όλα τα strings της παραμέτρου. Για παράδειγμα για τη λίστα που δόθηκε, τα γράμματα που υπάρχουν σε όλα strings είναι: ['m', 'k', 'i', 'a']. Προσοχή η λίστα μπορεί να περιέχει και γράμματα πέρα του λατινικού αλφάβητου (ελληνικα, κινέζικα, αραβικά, ...)

In [175]:
```
def f(l):

    s = set(l[0])


    for x in l[1:]:
        #s = s & set(x)
        s &= set(x)

    return s

b = [
    'zabarakatranemia',
    'askardamikth',
    'mpampesika',
]

f(b)
```

Out[175]:
```
{'a', 'i', 'k', 'm'}
```

In [169]:
```
set([1,2,3,2,1,2,3,2])
```

Out[169]:
```
{1, 2, 3}
```

In [283]:
```
(set('zabarakatranemia') & set('askardamikth')) & set('mpampesika')
```

Out[283]:
```
{'a', 'i', 'k', 'm'}
```

Έστω ο παρακάτω αριθμός:

```
a = 25629456287456291
```

Ποιος είναι ο μεγαλύτερος αριθμός ο οποίος υπάρχει μέσα στον a και αποτελείται από ψηφία τα οποία είναι συνεχόμενα; Για παράδειγμα ο 456 είναι ο μεγαλύτερος αριθμός ο οποίος αποτελείται από συνεχόμενα ψηφία (4,5,6) ο οποίος υπάρχει μέσα στον a. Φτιάξτε μία συνάρτηση που θα παίρνει σαν παράμετρο τον a και θα επιστρέφει τον ζητούμενο αριθμό.

In [284]:
```
def is_continuous(x):

    s = str(x)
    for k in range(0, len(s)-1):
        #print (s[k], s[k+1])
        #print (s[k], s[k+1], int(s[k+1])-int(s[k]))
        if int(s[k+1])-int(s[k]) != 1 :
            return False

    return True

def f(x):
    s = str(x)

    all_continuous = []

    for x in range(len(s)):
        for y in range(x+1, len(s)):
            current_number = int(s[x:y])
            if is_continuous(current_number):
                all_continuous.append(current_number)

    #print (all_continuous)
    return max(all_continuous)



a = 25629456287456291
f(a)

#print (456, ':')
#is_continuous(456) # True
##print (457, ':')
#is_continuous(457) # False
#print (4568, ':')
#is_continuous(4568) # False
```

Out[284]: 456

Ένα ρομποτάκι κινείται σε έναν διδιάστατο χώρο ξεκινώντας από το σημείο 0,0. Το ρομποτάκι δέχεται σαν είσοδο ένα string το οποιο περιγράφει τα βήματά του. Το string αυτό έχει τα παρακάτω γράμματα:

```
">": πήγαινε δεξιά (π.χ. από το 1,3 --> 2,3)
"<": πήγαινε αριστερά (π.χ. από το 1,3 --> 0,3)
"v": πήγαινε κάτω (π.χ. από το 1,3 --> 1,2)
"^": πήγαινε πάνω (π.χ. από το 1,3 --> 1,4)
```

Για παράδειγμα μία ακολουθία μπορεί να είναι:

^>v<

Δηλαδή πήγαινε πάνω, μετά δεξιά, μετά κάτω μετά αριστερά.

Φτιάξτε μία συνάρτηση η οποία θα δέχεται ένα string το οποίο θα έχει μόνο τους χαρακτήρες: "<", ">", "^", "v". Θα επιστρέφει το πλήθος από τις θέσεις τις οποίες το ρομποτάκι έχει επισκεφθεί μία και μόνο φορά. Θεωρείστε ότι το 0,0 είναι η θέση που ξεκινάει οπότε πριν ακόμα αρχίσει να κινείται, έχει επισκεφθεί αυτή τη θέση 1 φορά (μπορεί όμως στο μέλλον να τη ξαναεπισκεφθεί).

Παραδείγματα:

```
'v>v<vvv<<vv^v<v>vv>v<<<^^^^^<<^<vv>^>v^>^>^>^>^><vvvv<^>^<<^><<<^vvvv>
^>^><^v^><^<>^^>^vvv^<vv>>^>^^<>><>^>vvv>>^vv>^<><>^<v^>^>^><vv^vv^>>
<<^><<v>><>^<^>>vvv>v>>>v<<^<>'
--> 53

'<^<v<>v>^^v^^^<^v^^>>><^>^>v<>^<>>^>^^v^><v<v>>><>v<v^v>^v<>>^><v>^<>v
^>^<>^v^^^v^^>>vv<<^^><^<vvv>^>^^<^>>^^^^^v^<v>vv<>>v^v<^v^^<><^<^vv^>
<>><><>v>vvv^vv^^<<><<vvv><<^v^>'
--> 33
```

In [285]:
```
def f(x):


    pos = {
        (0,0) : 1
    }

    cur_x = 0
    cur_y = 0
    for p in x:
        if p == 'v':
            cur_y += 1
        if p == '^':
            cur_y -= 1
        if p == '<':
            cur_x -= 1
        if p == '>':
            cur_x += 1

        pos_tuple = (cur_x, cur_y)
        if not pos_tuple in pos:
            pos[pos_tuple] = 0

        pos[pos_tuple] += 1

    return sum(v==1 for v in pos.values())


x = 'v>v<vvv<<vv^v<v>vv>v<<<^^^^^<<^<vv>^>v^>^>^>^><vvvv<^>^<<^>
<<<^vvvv>^>^><^v^><^<>^^>^vvv^<vv>>^>^^<>><>^>vvv>>^vv>^<><>^<v^>^>
^><vv^vv^>><<^><<v>><>^<^>>vvv>v>>>v<<^<>'


f(x)
```

Out[285]: 53

In [ ]: