**7 kyu**

**Permutation Average**

18292% of 39173 of315[Nmistrata](https://www.codewars.com/users/Nmistrata)

Python

* [TRAIN AGAIN](https://www.codewars.com/kata/permutation-average/train/python)
* [NEXT KATA](https://www.codewars.com/trainer/python)

Details

[Solutions](https://www.codewars.com/kata/permutation-average/solutions/python)

[Discourse (6)](https://www.codewars.com/kata/permutation-average/discuss/python)

* Add to Collection
* |
* Share this kata:

A number is simply made up of digits.  
The number 1256 is made up of the digits 1, 2, 5, and 6.  
For 1256 there are 24 distinct permuations of the digits:  
1256, 1265, 1625, 1652, 1562, 1526, 2156, 2165, 2615, 2651, 2561, 2516,  
5126, 5162, 5216, 5261, 5621, 5612, 6125, 6152, 6251, 6215, 6521, 6512.

Your goal is to write a program that takes a number, n, and returns the average value of all distinct permutations of the digits in n. Your answer should be rounded to the nearest integer. For the example above the return value would be 3889.

n will never be negative

A few examples:

permutation\_average(2)

return 2

permutation\_average(25)

>>> 25 + 52 = 77

>>> 77 / 2 = 38.5

return 39

permutation\_average(20)

>>> 20 + 02 = 22

>>> 22 / 2 = 11

return 11

permutation\_average(737)

>>> 737 + 377 + 773 = 1887

>>> 1887 / 3 = 629

return 629

Note: Your program should be able to handle numbers up to 6 digits long

<https://www.codewars.com/kata/permutation-average/python>

# Hello World program in Python

'''

def nextPermutation(array):

#array = list(str(a))

i = len(array) - 1

while(i > 0 and array[i-1] >= array[i]): i-=1

if(i <= 0): return False

j = len(array) - 1

while(array[j] <= array[i-1]): j-=1

temp = array[i-1]

array[i-1] = array[j]

array[j] = temp

j = len(array) - 1

while(i < j):

temp = array[i]

array[i] = array[j]

array[j] = temp

i+=1

j-=1

return True

'''

'''

def permutations(iterable, r=None):

# permutations('ABCD', 2) --> AB AC AD BA BC BD CA CB CD DA DB DC

# permutations(range(3)) --> 012 021 102 120 201 210

pool = tuple(iterable)

n = len(pool)

r = n if r is None else r

if r > n:

return

indices = range(n)

cycles = range(n, n-r, -1)

yield tuple(pool[i] for i in indices[:r])

while n:

for i in reversed(range(r)):

cycles[i] -= 1

if cycles[i] == 0:

indices[i:] = indices[i+1:] + indices[i:i+1]

cycles[i] = n - i

else:

j = cycles[i]

indices[i], indices[-j] = indices[-j], indices[i]

yield tuple(pool[i] for i in indices[:r])

break

else:

return

'''

from itertools import permutations

import math

def permutation\_average(n):

a = str(n)

sum = 0

cont = 0

perms = list(set(permutations(a)))

for item in perms:

#print(item)

sum += int(''.join(item))

return int(round(sum/len(perms)))

print(permutation\_average(222))

#print(int(round(56.666)))