

def fun\_many(data):

a = len(data)

count = 0

listFinal = []

for i in range(a):

ist = [0]\*3

ist[0] = data[i]

for j in range(a):

ist[1] = data[j]

for k in range(a):

ist[2] = data[k]

listFinal.append(list(ist))

count+=1

print(listFinal)

print(count)

def fun\_single(data):

a =len(data)

count = 0

listFinal = []

for i in range (a):

ist = [0] \* 3

ist[0] = data[i]

for j in range(a):

if data[j] != ist[0]:

ist[1] = data[j]

for k in range(a):

if data[k] != ist[0] and data[k] != ist[1]:

ist[2] = data[k]

listFinal.append(list(ist))

count+=1

print(listFinal)

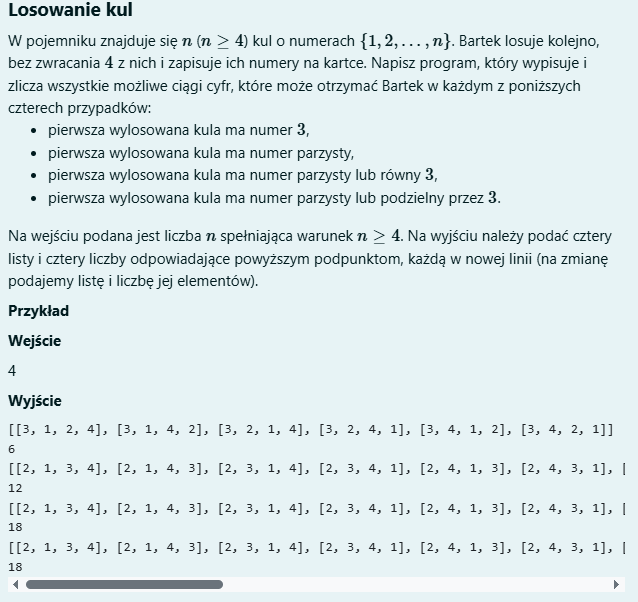
print(count)

data = input()

data = sorted(data.split(", "))

fun\_many(data)

fun\_single(data)



import itertools

def losowanie\_kul(n):

kule = list(range(1, n + 1))

permutacje = list(itertools.permutations(kule, 4))

# Konwersja krotek na listy

permutacje = [list(p) for p in permutacje]

# Warunki

war1 = [p for p in permutacje if p[0] == 3]

war2 = [p for p in permutacje if p[0] % 2 == 0]

war3 = [p for p in permutacje if p[0] % 2 == 0 or p[0] == 3]

war4 = [p for p in permutacje if p[0] % 2 == 0 or p[0] % 3 == 0]

# Wyświetlenie wyników

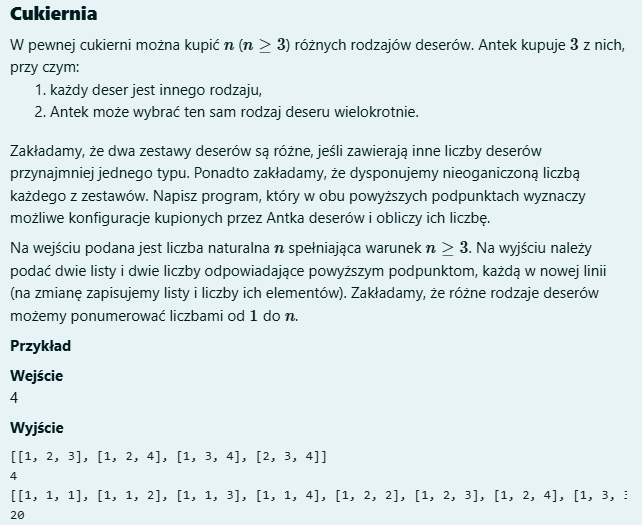
for lista in [war1, war2, war3, war4]:

print(lista)

print(len(lista))

n = int(input())

losowanie\_kul(n)



import itertools

def cukiernia(n):

# 1. Każdy deser innego rodzaju (kombinacje bez powtórzeń)

wariant1 = list(itertools.combinations(range(1, n + 1), 3))

wariant1 = [list(p) for p in wariant1]

# 2. Z powtórzeniami, ale bez znaczenia kolejności (multizbiory)

wariant2 = list(itertools.combinations\_with\_replacement(range(1, n + 1), 3))

wariant2 = [list(p) for p in wariant2]

# Wynik

print(wariant1)

print(len(wariant1))

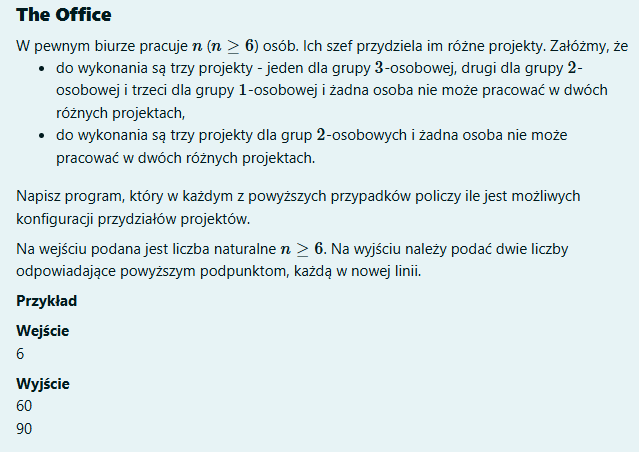
print(wariant2)

print(len(wariant2))

# Przykład użycia

n = int(input())

cukiernia(n)



import itertools

def the\_office\_corrected(n):

osoby = list(range(1, n + 1))

# --- Wariant 1: 3 + 2 + 1 ---

konfiguracje\_1 = []

for trojka in itertools.combinations(osoby, 3):

pozostali\_po\_3 = [x for x in osoby if x not in trojka]

for dwojka in itertools.combinations(pozostali\_po\_3, 2):

pozostaly = [x for x in pozostali\_po\_3 if x not in dwojka]

for jedynka in itertools.combinations(pozostaly, 1):

konfiguracje\_1.append([list(trojka), list(dwojka), list(jedynka)])

# --- Wariant 2: trzy rozróżnialne pary (projekty) ---

konfiguracje\_2 = []

for szostka in itertools.combinations(osoby, 6):

osoby\_szostka = list(szostka)

# Wszystkie możliwe permutacje osób

for perm in itertools.permutations(osoby\_szostka):

para1 = tuple(sorted(perm[0:2]))

para2 = tuple(sorted(perm[2:4]))

para3 = tuple(sorted(perm[4:6]))

projekt = [para1, para2, para3]

konfiguracje\_2.append(tuple(projekt))

# Usuwamy powtórzenia (te same 3 pary mogą wystąpić w różnych permutacjach)

konfiguracje\_2\_unikalne = set(konfiguracje\_2)

print(len(konfiguracje\_1))

print(len(konfiguracje\_2\_unikalne))

# Test

n = int(input())

the\_office\_corrected(n)