#1. Найти все числа от 1 до 1000, которые делятся на 17 без остатка

result = 0

for i in range (1,1001):

if i % 17 == 0:

result += 1

print(result)

#2. Найти все числа от 1 до 1000, которые содержат в себе цифру 2

print(\*(i for i in range(0, 1001) if '2' in str(i)))

#3. Найти все числа от 1 до 10000, которые являются палиндромом

for i in range(1,10001):

for n in range(1,10001):

number = str(n\*i)

if (number == number[::-1]):

palindromic = number

break

print(palindromic)

#4. Посчитать количество пробелов в строке

string = "A simple sentence has the most basic elements that make it a sentence: a subject a verb and a completed thought"

print(string.count(" "))

#5. Есть любая последовательность непробельных символов латинского алфавита, удалить все гласные из этого слова

vowels = ["a", "e", "i", "o", "u", "y"]

input\_str = "HelloEarth"

result = [c for c in input\_str if c.lower() not in vowels]

result\_str = "".join(result)

print (result\_str)

#6. На входе строка со словами, разделенными через 1 пробел. Найти все слова, длина которых не больше 5

text = "'Maty', 'Gahjuro', 'Haus', 'Ikurdfsss', 'Glecho'"

print([e for e in eval('('+text+')') if len(e)<=5])

#7. На входе строка со словами, разделенными через 1 пробел. Получить словарь, где в качестве ключа используется само слово, а в значении длина этого слова.

string = "A simple sentence has the most basic elements that make it a sentence: a subject a verb and a completed thought"

new = string.split()

result = {i: len(str(i)) for i in new}

print(result)

#8. На входе предложение со всеми пробельными и непробельными символами латинского алфавита. Получить словарь используемых букв в строке, то есть на выходе список уникальных букв.

#9. На входе список чисел, получить список квадратов этих чисел

numbers = [2, 3, 4, 6]

def square(x):

return x\*x

print(list(map(square, numbers)))

#10. На входе список координат, например, [(1, 1), (2, 3), (5, 3)]. Найти все точки, которые принадлежат прямой y = 5 \* x - 2.

На выходе получить словарь из самой точки и расстоянии до этой точки из начала координат (0, 0)

# 11. Возвести в квадрат все четные числа от 2 до 27. На выходе список.

def square(x):

return x \* x

arr = []

for x in range(2, 27):

if x % 2 == 0:

arr.append(square(x))

print(arr)

#12. На входе список из координат точек на плоскости. Найти расстояние до самой удаленной точку от начала координат (0, 0) в первой четверти

coords = [(1, 1), (2, 2), (3, 3), (4, 4), (-1, -1), (-1, 5)]

result\_square\_distance = {point: point[0] \*\* 2 + point[1] \*\* 2 for point in coords if point[0] > 0 and point[1] > 0}

print(max(result\_square\_distance))

#13. На входе два списка чисел nums\_first = [1, 2, 3, 5, 8] и nums\_second = [2, 4, 8, 16, 32]. Получить пары сумм и разниц, [(3, -1), (6, -2), (11, -5), ...]

nums\_first = [1, 2, 3, 5, 8]

nums\_second = [2, 4, 8, 16, 32]

plus\_numbers = [x+y for x, y in zip(nums\_first, nums\_second)]

minus\_numbers = [x-y for x, y in zip(nums\_first, nums\_second)]

print((list(zip (plus\_numbers, minus\_numbers))))

#14. На входе список строк из чисел, например, ['43141', '32441', '431', '4154', '43121']. Найти четные квадраты этих чисел. Ответ записать снова в список из строк, то есть сформировать обратно список строк, но уже отфильтровать все четные квадраты

numbers = ['43141', '32441', '431', '4154', '43121']

numm = map(int, numbers)

def square(x):

return x\*x

print(list(map(square, numm)))