## Урок 3-4. Строки. Индексация

**Задание 4**

Иногда программисты вынуждены решать специфические задачи, которые могут и не иметь какого-то функционального смысла – так им сказали. Вот и мы Вас просим решить такую задачу.

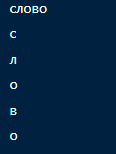
Напишите программу, которая для введенного слова последовательно выводит каждую его букву в новой строке.

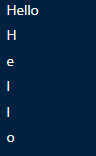
|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| слово | с  л  о  в  о |
| Hello | H  e  l  l  o |

word = input()

for s in word:

print(s)





## Урок 5-6. Строки. Применение срезов

**Задание 3**

Плохо, если в Вашем пароле есть подстрока «qwerty», но ещё хуже, если она не просто есть, но и является началом Вашего пароля.

Напишите программу, которая бы проверяла качество Вашего пароля. При этом программа должна выводить строку «ОЧЕНЬ ПЛОХО», если пароль начинается с подстроки «qwerty», «ПЛОХО» – если в пароле есть «qwerty», но пароль не начинается с этого слова, и «ХОРОШО» во всех остальных случаях.

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| qwerty123456 | ОЧЕНЬ ПЛОХО |
| 123qwerty456 | ПЛОХО |

SUBS = 'qwerty'  
password = input()  
if password[:6] == SUBS:  
 print('ОЧЕНЬ ПЛОХО')  
elif SUBS in password:  
 print('ПЛОХО')  
else:  
 print('ХОРОШО')

** **

## Урок 7-8. Строки. Применение методов

**Задание 5 (\*)**

Шифр Цезаря называют в честь Юлия Цезаря, который, согласно «Жизни двенадцати цезарей» Светония, использовал его со сдвигом 3, чтобы защищать военные сообщения.

Вам необходимо написать программу, которая кодирует сообщение шифром Цезаря. Шифр Цезаря заменяет каждую букву в тексте на букву, которая отстоит в алфавите на некоторое фиксированное число позиций.

Так, например, результатом шифрования сообщения «собака» со сдвигом на 3 будет «фсдгнг».

Программа принимает последовательность строчных русских букв и сдвиг (сдвиг может быть отрицательным). После выполнения шифрования программа выводит полученную строку.

Мы будем не учитывать букву `ё` в русском алфавите. И напоминаем, что символы расположены циклически, то есть, после буквы `я` следует буква `а`.

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| собака  3 | фсдгнг |
| фсдгнг  -3 | собака |

word = input() # a-1072, я-1103

step = int(input())

new\_word = ""

for letter in word:

cod = ord(letter)

new\_cod = cod + step

if new\_cod > 1103:

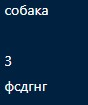
new\_cod = new\_cod - 1103 + ord('a')

elif new\_cod < 1072:

new\_cod = 1103 - (ord('a') - new\_cod)

new\_word += chr(new\_cod)

print(new\_word)





## Урок 9-10. Основы теории множеств. Создание и вывод множества, особенности коллекции

**Задание 2**

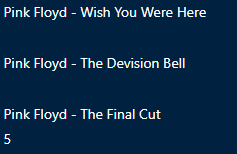
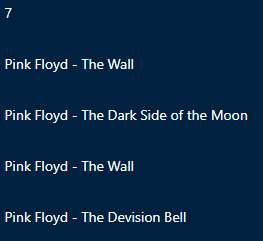
Коллекционер задумал оптимизировать свою коллекцию – избавиться от дубликатов. То есть, он хочет, чтобы экземпляры в его коллекции не повторялись. Но перед эти он решил посчитать, сколько различных предметов у него останется после оптимизации, чтобы понять, сможет ли он их разместить.

Напишите программу, которая при анализе коллекции бы отвечала на вопрос, сколько предметов останется после проведенной оптимизации.

Сначала описывается коллекция коллекционера – вводится целое число n – количество предметов в коллекции, после чего в n-строках вводятся названия предметов. Выводится одно число – сколько предметов в коллекции останется после оптимизации.

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| 7  Pink Floyd - The Wall  Pink Floyd - The Dark Side of the Moon  Pink Floyd - The Wall  Pink Floyd - The Devision Bell  Pink Floyd - Wish You Were Here  Pink Floyd - The Devision Bell  Pink Floyd - The Final Cut | 5 |
| 3  Pink Floyd - The Wall  Pink Floyd - The Wall  Pink Floyd - The Wall | 1 |

n = int(input())  
collection = set()  
for i in range(n):  
 item = input()  
 collection.add(item)  
print(len(collection))





## Урок 11-12. Множества. Пересечение, объединение, разность, симметрическая разность

**Задание 5**

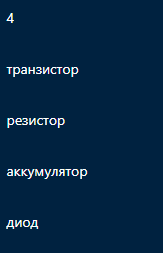
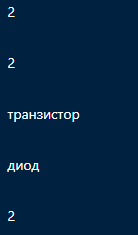
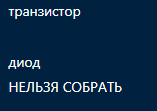
Желания схемотехника растут всё больше. Вот он уже хочет узнать, сможет ли из имеющихся видов деталей собрать n схем! Имейте ввиду, если вдруг так случилось, что один вид компонент используется в одной схеме, то он не может использоваться в другой.

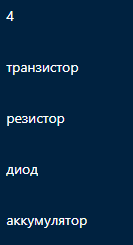
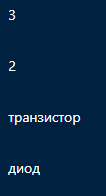
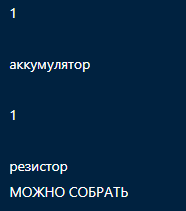
Напишите программу, которая проверяла, сможет ли схемотехник собрать желаемые схемы.

Сначала вводится число k – количество имеющихся видов компонент у молодого схемотехника. После чего происходит ввод (описание) компонент. Далее вводится число n – количество схем, которые хочет собрать наш герой. Затем описывается каждая конкретная схема: ni – количество элементов для конкретной схемы, после чего описываются элементы данной схемы. Программа выводит «МОЖНО СОБРАТЬ», если видов деталей, имеющихся у схемотехника, достаточно для построения схемы; иначе программа выводит слово «НЕЛЬЗЯ СОБРАТЬ».

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| 4  транзистор  резистор  аккумулятор  диод  2  2  транзистор  диод  2  транзистор  диод | НЕЛЬЗЯ СОБРАТЬ |
| 4  транзистор  резистор  диод  аккумулятор  3  2  транзистор  диод  1  аккумулятор  1  резистор | МОЖНО СОБРАТЬ |

flag = True  
detail = set()  
schema = set()  
k = int(input())  
for i in range(k):  
 detail.add(input())  
n = int(input())  
for i in range(n):  
 t = int(input())  
 schema.clear()  
 for j in range(t):  
 schema.add(input())  
 if detail & schema == schema:  
 detail = detail - schema  
 else:  
 flag = False  
 print('НЕЛЬЗЯ СОБРАТЬ')  
 break  
if flag:  
 print('МОЖНО СОБРАТЬ')

## Урок 13-14. Множества. Применение методов

**Задание 2**

Давайте в очередной раз представим себя лингвистами и решим задачу, не имеющую никакого отношения к лингвистике.

Предположим, что условие задачи весьма просто – нам дано два слова и необходимо найти количество общих букв в этих словах. При этом если в первом слове буква `а` встречается 3 раза, а во втором слове – 2 раза, мы будем считать, что `а` является общей буквой и будем её учитывать один раз.

Написанная Вами программа должна в двух строках принимать по одному слову в каждой и выводить одно число – количество общих букв.

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| красота  краснота | 6 |
| переправа  право | 4 |

word1 = set(input())

word2 = set(input())

print(len(word1 & word2))

**** ****

## Задания для Уроков 18-19-20.

## Списки. Выделение памяти, пустой список. Заполнение присваиванием и вывод

**Задание 1**

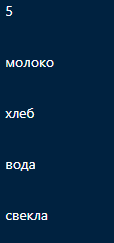
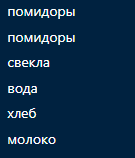
Может ли поход в магазин быть увлекательным? Мы не знаем… Но мы точно знаем, что поход в магазин может быть отличным контекстом задачи – одна задача о рюкзаке чего стоит!

Вы идете в магазин и хотите купить определенные продукты. Но Вы очень хотите купить их в определенном порядке -- в обратном.

Программа принимает на вход в первой строке число n – количество продуктов, после чего вводятся сами продукты – каждый в отдельной строке. Программа должна вывести продукты в обратном порядке.

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| 5  молоко  хлеб  вода  свекла  помидоры | помидоры  свекла  вода  хлеб  молоко |

products = list()  
n = int(input())  
for i in range(n):  
 products.append(input())  
for i in range(n):  
 print(products[n-i-1])

## Урок 21-22. Списки. Применение функции map, методов split, append, extend

**Задание 2**

Солдаты на построении были выстроены в шеренгу по росту. Необходимо определить, сколько солдат среди выстроившихся обладают не повторяющимся ростом.

Вам необходимо написать программу, которая принимает в качестве входных данных одну строку, составленную из произвольного количества чисел (значение роста каждого солдата). Гарантируется, что числа разделены пробелом и находятся в порядке невозрастания. Программа должна вывести одно число – ответ на вопрос задачи.

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| 181 178 178 175 174 174 | 2 |
| 181 181 178 178 175 175 174 174 | 0 |

height = list(map(int, input().split()))

set\_height = set(height)

k = 0

for h in set\_height:

if height.count(h) == 1:

k += 1

print(k)

****

****

## Урок 23-24. Списки. Применение срезов

**Задание 3 (\*)**

Все очень просто. Вы, как филолог, хотите проверить, не меняется ли смысл при перестановке слов. Но чтобы не затуманивать свой разум, Вы хотите, не думаю об изначальном предложении, прочитать уже предложение с перестановкой.

Ваша задача – написать программу, которая считала бы предложение из (возможно) большого количества слов и вывела бы данное предложение, в котором последнее слово стало первым, а первое – последним.

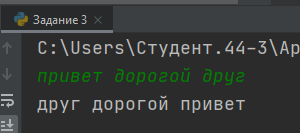
|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| привет дорогой друг | друг дорогой привет |
| казнить нельзя помиловать нельзя | нельзя нельзя помиловать казнить |

sp = input().split()

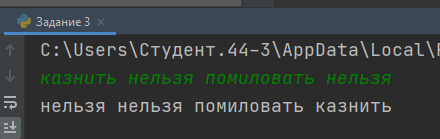
sp[0], sp[-1] = sp[-1], sp[0]

print(' '.join(sp))

Тест 1.



Тест 2.



## Урок 27-28. Списки. Применение методов (сортировки, поиска, удаления, замены и др.)

**Задание 3**

Используя функции max, min, find и rfind, а так же явно не используйте циклы, решите задачу, с которой Вы уже встречались.

Вот то, к чему так настырно шёл Аркадий. Он всегда мечтал, что научится менять местами самый левый минимальный элемент и самый правый максимальный элемент. Давайте ему в этом поможем.

Напишите программу, которая принимает последовательность чисел, после чего меняет в этой последовательности самый левый минимальный элемент и самый правый максимальный элемент местами, и выводит получившийся список в одну строку.

Программа принимает в одной строке последовательность целых чисел, разделённых пробелом. Программа выводит изменённую согласно условию задачи последовательность.

|  |  |
| --- | --- |
| **Ввод** | **Вывод** |
| 2 -1 4 2 -1 4 2 | 2 4 4 2 -1 -1 2 |
| -5 -5 3 3 | 3 -5 3 -5 |

numbers = list(map(int, input().split()))

mn = min(numbers)

i\_mn = numbers.index(mn)

numbers.reverse()

mx = max(numbers)

i\_mx = -numbers.index(mx) - 1

numbers.reverse()

numbers[i\_mn], numbers[i\_mx] = numbers[i\_mx], numbers[i\_mn]

print(' '.join(map(str, numbers)))

