1. В первой строке введены 3 символа (подстрока). Затем вводятся еще несколько строк.

Среди всех строк (кроме первой) найти такую строку, в которой первая подстрока встречается наименьшее, но ненулевое количество раз.  Если таких строк несколько, выбрать ту, что ближе к началу ввода.

В найденной строке нужно определить количество вхождений символов в строку.

Вывести полученные значения в порядке возрастания количества вхождений и по алфавиту (при их равенстве) в следующем формате:

***символ – количество, символ – количество, символ – количество***

|  |  |
| --- | --- |
| **Ввод**  MIG  YKGMIGFCQIACWMDERDMIGR  CXATHLOMIGCX  YWMIGIS  CSVVXQWOQOLHDKNOFYPRB | **Вывод**  A - 1, G - 1, H - 1, I - 1, L - 1, M - 1, O - 1, T - 1, C - 2, X - 2 |

1. Вводится имя файла csv с данными об именах и городах. Заголовки файла (разделители точки):

*id, имя, пункт назначения, расстояние*  
*id, name, destination, distance*

**Формат вывода**

В файл **res.json**запишите список словарей с ключами  
*city*(пункт назначения)  
*name*(имя)  
в алфавитном порядке городов, в случае одинакового города – в алфавитном порядке имен.

**Пример 1**

| Ввод | Вывод |
| --- | --- |
| # Содержимое stdin:  schedule.csv  # Содержимое файла schedule.csv:  id.name.destination.distance  1.Carnatic.Yokohama.1308  2.Magnolia.Hong Kong.2707  3.Claremont.Klang.508  4.Savannah.Klang.508 | # Содержимое файла res.json:  [  {  "city": "Hong Kong",  "name": "Magnolia"  },  {  "city": "Klang",  "name": "Claremont"  },  {  "city": "Klang",  "name": "Savannah"  },  {  "city": "Yokohama",  "name": "Carnatic"  }  ] |

**Пример 2**

| Ввод | Вывод |
| --- | --- |
| # Содержимое stdin:  trips.csv  # Содержимое файла trips.csv:  id.name.destination.distance  1.Sirius.Shenzhen.1707  2.Archimedes.Busan.2707  3.Stockton.Qingdao.2707  4.Amazon.Qingdao.1707  5.East.Shenzhen.2707 | # Содержимое файла res.json:  [  {  "city": "Busan",  "name": "Archimedes"  },  {  "city": "Qingdao",  "name": "Amazon"  },  {  "city": "Qingdao",  "name": "Stockton"  },  {  "city": "Shenzhen",  "name": "East"  },  {  "city": "Shenzhen",  "name": "Sirius"  }  ] |

1. Напишите функцию для выбора подходящих судов для морского путешествия.

Функция func() принимает произвольное число позиционных аргументов – городов, и два именованных параметра: *address* и *port* сервера.  
На сервере по указанному адресу находится словарь: название корабля – список портов по пути следования.

Функция возвращает список кораблей, в маршруте которых присутствуют города из аргументов функции; список отсортирован по убыванию количества общих городов, затем по алфавиту названий кораблей.

**Пример 1**

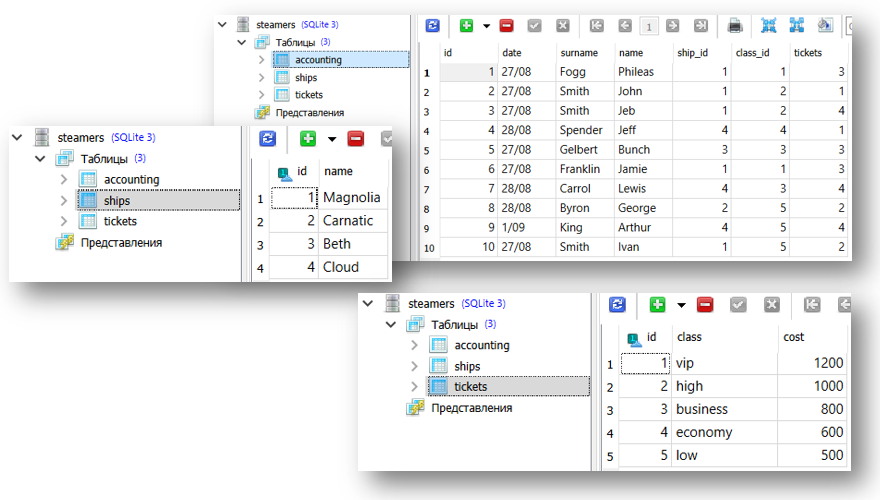
| Ввод | Вывод |
| --- | --- |
| # Пример вызова функции:  data = ["Kiama", "Nelson", "Eden"]  print(func(\*data, address="127.0.0.1", port=5000))  # Данные на сервере:  {  "Carnatic": ["Bellambi", "Kiama", "Eden"],  "Magnolia": ["Kembla", "Kiama"],  "Beth": ["Sidney", "Nelson", "Kiama"],  "Win": ["Darwin", "Kembla"]  } | [  "Beth",  "Carnatic",  "Magnolia"  ] |

**Пример 2**

| Ввод | Вывод |
| --- | --- |
| # Пример вызова функции:  data = ["Keats", "Daly", "Hurd"]  print(func(\*data, address="127.0.0.1", port=8800))  # Данные на сервере:  {  "Curtis": ["Bradshaw", "Daly"],  "Gove": ["Essington", "Keats", "Daly"],  "Bynoe": ["Langdon", "Hurd", "Bremer"]  } | [  "Gove",  "Bynoe",  "Curtis"  ] |

1. Напишите класс для учёта пассажиров пароходов.

Экземпляр класса **AccountingPassengers**при инициализации принимает имя файла базы данных, структура которой показана на рисунке.



Таблицы и названия полей базы данных:  
*accounting (id, date, surname, name, ship\_id, class\_id, tickets)*  
*ships (id, name)*  
*tickets (id, class, cost)*

Класс обеспечивает реализацию методов:  
*trip()*– метод принимает дату и название корабля, а возвращает список строк: *фамилия, имя, суммарная стоимость купленных билетов*, после фамилии и имени запятая и пробел; список упорядочен по алфавиту фамилий, затем имен;  
*date\_list()*– метод получает дату отправления, а возвращает список кортежей (*название корабля, количество проданных билетов*); порядок обратный алфавитный.

**Пример 1**

| Ввод | Вывод |
| --- | --- |
| ap = AccountingPassengers("steamers.db")  print(\*ap.trip("27/08", "Magnolia"), sep="\n") | Fogg Phileas, 3600  Franklin Jamie, 3600  Smith Ivan, 1000  Smith Jeb, 4000  Smith John, 1000 |

**Пример 2**

| Ввод | Вывод |
| --- | --- |
| ap = AccountingPassengers("steamers.db")  print(\*ap.date\_list("28/08"), sep="\n") | ('Cloud', 5)  ('Carnatic', 2) |

1. Через аргументы командной строки передаются:

порт (*port*) и хост (*server*) сервера,  
имя файла *csv*(*ﬁlename*) с данными.

В файле записана информация о сообщениях, заголовки файла (разделители +):  
*id, фамилия, имя, сообщение, правда или ложь*  
*id, last\_name, ﬁrst\_name, message, validity*

Напишите приложение на *ﬂask*, которое при переходе по адресу *http://server:port/false*возвращает список словарей с ключами:  
*name*– строка Фамилия Имя через пробел,  
*truth*– список сообщений, в которых правда,  
*lie*– сообщения, в которых ложь.  
Словари в списке идут в алфавитном порядке фамилий, сообщения в списках – как в файле.

**Пример**

| Ввод | Вывод |
| --- | --- |
| # Пример запуска:  python3 solution.py --port 5000 --server 127.0.0.1 --filename info.csv  # Содержимое файла info.csv:  id+last\_name+first\_name+message+validity  1+Fix+John+the ship is delayed+lie  2+Smith+Jeb+the repair is finished+truth  3+Fix+John+the ship is sailing+truth  4+Fix+John+the rudder is broken+lie  5+Smith+Jeb+the captain is ill+truth  6+Fix+John+sun is going down+lie  7+Smith+Jeb+departure is delayed+truth | [  {  "name": "Fix John",  "lie": [  "the ship is delayed",  "the rudder is broken",  "sun is going down"  ],  "truth": [  "the ship is sailing"  ]  },  {  "name": "Smith Jeb",  "lie": [],  "truth": [  "the repair is finished",  "the captain is ill",  "departure is delayed"  ]  }  ] |