Исполнитель робот

Напишите класс **Robot**, который инициализируется с начальными координатами – положением Робота на плоскости, обе координаты заключены в пределах от 0 до 100.

Робот может передвигаться на одну клетку вверх (N), вниз (S), вправо (E), влево (W). Выйти за границы плоскости Робот не может.

Метод *move()* принимает строку – последовательность команд перемещения робота, каждая буква строки соответствует перемещению на единичный интервал в направлении, указанном буквой. Метод возвращает кортеж координат – конечное положение Робота после перемещения.

Метод *path()* вызывается без аргументов и возвращает список кортежей координат точек, по которым перемещался Робот при последнем вызове метода *move*. Если метод не вызывался, возвращает список с одним кортежем – начальным положением Робота.

Пример

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| from solution import Robot  r = Robot((0, 0))  print(r.move('NENW'))  print(\*r.path()) | (0, 2)  (0, 0) (0, 1) (1, 1) (1, 2) (0, 2) |

Абстрактный кот

При наследовании классов класс-наследник обычно имеет больший функционал, чем родительский класс. А у котов все наоборот. Кошка по сравнению с котенком умеет ловить мышей (котенок пока нет) и мяукает громче (CAPS LOCK`ом). Поэтому логично кошку наследовать от котенка…

Напишите классы:

Класс **АбстрактныйКот** (**AbstractCat**), который инициализируется без аргументов. Умеет:  
– *eat* – есть. За каждые 10 единиц еды вес увеличивается на 1 единицу, пока не станет 100, дальше не растет. Если при одном приеме пищи количество еды не кратно 10, остаток запасается, а потом суммируется с новой едой.

– возвращать строковое представление в виде *<Имя класса> (вес)*.

Класс **Котенок** (**Kitten**), наследуется от кота с аргументом вес. Умеет мяукать тоненько:  
– *meow* – возвращает строку *"meow..."*  
Еще умеет спать – sleep – возвращает строку *Snore*, повторенную столько раз, сколько число 5 помещается в весе.

Класс **Кошка** (**Cat**), наследуется от котенка с аргументами вес и кличка. Умеет мяукать громко (*meow*):  
*"MEOW..."*   
и возвращать свое имя (*get\_name*). Также умеет ловить мышей:  
– *catch\_mice* – возвращает строку: *Got it!*

### **Пример 1**

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| from solution import \*  abscat = AbstractCat()  abscat.eat(125)  abscat.eat(17)  print(abscat)  kit = Kitten(21)  print(kit.sleep())  cat = Cat(83, 'Molly')  print(cat.meow())  print(cat.get\_name()) | AbstractCat (14)  SnoreSnoreSnoreSnore  MEOW...  Molly |

### **Пример 2**

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| from solution import \*  kit = Kitten(15)  kit.eat(24)  print(kit)  cat = Cat(41, 'Molly')  print(cat.catch\_mice())  print(cat) | Kitten (17)  Got it!  Cat (41) |

# **C. Корзина с грушами**

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Итак, у вас есть корзина с грушами и несколько детей. Нужно поделить груши так, чтобы никому из детей не было обидно, то есть поровну. (Теперь понятно, почему в задаче дети?)

Напишите класс **PearsBasket**, экземпляр которого при инициализации получает целое число – количество груш в корзине.

В классе должны быть методы:

* *pb // n*– деление нацело, возвращает список объектов класса со значениями количества груш в каждой корзинке, если есть остаток – он должен находиться в дополнительной последней корзинке.
* *pb % n*– получение остатка от деления, возвращает число: остаток от деления.
* *pb\_1 + pb\_2*– складываются две корзинки, получается новая корзина;
* *pb\_1 - n*– число вычитается из корзинки, если там есть такое количество груш; если нет – вычитается сколько есть, остается 0;
* *\_\_str\_\_*– возвращает количество груш в корзине;
* *\_\_repr\_\_*– возвращает строку в формате **PearsBasket(<количество>)**.

## Пример

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| from solution import PearsBasket  pb = PearsBasket(17)  array = pb // 4  print(array)  pb\_2 = PearsBasket(13)  pb\_3 = pb + pb\_2  print(pb\_3)  print(pb\_3 % 7)  pb - 2  print([pb]) | [PearsBasket(4), PearsBasket(4), PearsBasket(4), PearsBasket(4), PearsBasket(1)]  30  2  [PearsBasket(15)] |

# **D. Гонки**

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

В городе Тарантасске проходят гонки без ограничений. Это значит, что одновременно соревноваться могут гонщики на автомобилях, мотоциклах и даже гужевых повозках. Скорости автомобиля и мотоцикла зависят от того, каким бензином они заправлены - 98 позволяет ехать с полной скоростью, 95 уменьшает скорость на 10% для машин и на 20% для мотоциклов, 92-й - на 20% и 40% соответственно. Гужевые повозки едут с постояной скоростью.  
Длина кольцевой трассы равна M. Определить, какое транспортное средство будет ближе к финишу спустя время t после начала гонок (возможно, проходя при этом не первый круг). Если несколько средств будут одинаково близки, вывести нужно то, у которого меньше номер. Для описания транспортных средств используйте наследование классов.

## Формат ввода

Программа получает на вход строку с натуральными числами N, M и t - количеством транспортных средств, длиной трассы и временем гонки соответственно.  
Дальше следуют N строк. В каждой записано три натуральных числа - индивидуальный номер транспортного средства, тип средства (1 - автомобиль, 2 - мотоцикл, 3 - гужевая повозка) и скорость данного средства. Если это автомобиль или мотоцикл, то в строке также есть четвёртое число - марка бензина, которым заправлено средство.

## Формат вывода

Программа должна вывести одно число - номер транспортного средства, которое спустя время t окажется ближе всего к финишу.

### **Пример 1**

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 3 100 1  34 1 80 98  19 2 90 95  14 3 30 | 34 |

### **Пример 2**

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 2 60 2  87 1 100 98  71 2 50 95 | 71 |

## Примечания

В первом тесте ближе всего к финишу в момент t будет автомобиль. Во втором автомобиль и мотоцикл окажутся в одной точке (автомобиль проедет на два круга больше), следовательно, нужно вывести номер мотоцикла, так как он меньше.

# **E. Переговоры**

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

– Вот наши условия, – наконец нарушил молчание Сильвер. – Вы отдадите нам карту, чтобы мы могли заняться поисками клада. Тогда мы оставим вам половину припасов и первому же встреченному кораблю сообщим, что вы нуждаетесь в помощи.

– И это все? – осведомился капитан Смоллетт.

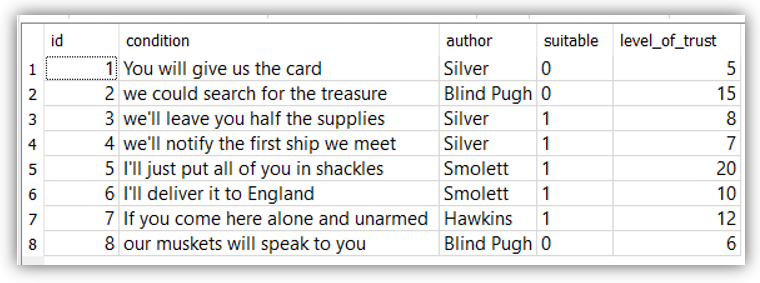
– Это мое последнее слово, будь я проклят! – ответил Джон. – Если вы откажетесь, то с вами будут говорить наши мушкеты.

– Превосходно, – сказал капитан. – А теперь выслушайте меня. Если вы явитесь сюда поодиночке и без оружия, то я всего лишь закую всех вас в кандалы и доставлю в Англию, где вас будут судить по закону.

Напишите программу, которая выберет условия, на которые можно согласиться.

Вашему решению будет доступен файл базы данных с таблицей *Talks*формата (показана БД для примеров):

*id, condition, author, suitable, level\_of\_trust*



## Формат ввода

Вводится имя файла БД, затем два условия отбора, которые должны выполняться одновременно, и поле, по которому происходит сортировка выводимых значений.

## Формат вывода

Вывести подходящие условия (*condition*), отсортированные по указанному полю. Каждое с новой строки.

### **Пример 1**

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| conversations.db  author = 'Silver'  level\_of\_trust BETWEEN 6 AND 15  id | we'll leave you half the supplies  we'll notify the first ship we meet |

### **Пример 2**

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| discussion.db  suitable = 1  level\_of\_trust > 10  author | If you come here alone and unarmed  I'll just put all of you in shackles |

## Примечания

Файлы из примеров можно загрузить [здесь](https://yastatic.net/s3/lyceum/task_files/2020_E2L2_fOkUl.zip).

Используется СУБД sqlite.

Данные необходимо упорядочивать по возрастанию (ASC).

# **F. Один шаг от истины**

|  |  |
| --- | --- |
| Ограничение времени | 10 секунд |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод |
| Вывод | truth.csv |

Все время казалось, что мы стоим на шаг от расшифровки непрерывно увеличивающегося моря записей. Океан – источник электрических, магнитных, гравитационных импульсов – говорил как бы языком математики. Все это склоняло ученых к выводу, что перед ними мыслящее существо, что-то вроде гигантски разросшегося, покрывшего целую планету протоплазменного моря-мозга, которое тратит время на неестественные по своему размаху теоретические исследования сути всего существующего, а то, что выхватывают наши аппараты, составляет лишь оборванные, случайно подслушанные обрывки этого, продолжающегося вечно в глубинах океана, перерастающего всякую возможность нашего понимания, гигантского монолога.

Напишите программу, которая немного систематизирует полученные от Океана импульсы.

## Формат ввода

Через аргументы командной строки передаются адрес сервера и порт (*host*, *port*). Также могут быть переданы опциональные аргументы:

*--not\_mult*– не рассматривать значения, кратные данному числу, по умолчанию 100;

*--smallest*– рассматривать только числа, не меньшие данного, по умолчанию 0.

## Формат вывода

На сервере находится список словарей, в которых по одинаковым ключам находятся списки целых чисел. Для каждого ключа нужно получить и записать в файл **truth.csv**строку данных (заголовки не записывать, разделитель точка с запятой):

* *ключ;*
* *наибольшее значение по данному ключу;*
* *наименьшее значение;*
* *среднее, округленное до 2 знаков после запятой;*
* *сумма.*

Строки записывать в алфавитном порядке ключей.

## Пример

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| Пример запуска:  python3 solution.py 127.0.0.1 5000 --smallest -2  Данные с сервера:  [  {  "electric": [  62,  45,  15,  44,  20  ],  "magnetic": [  47,  10,  43,  61  ],  "gravitational": [  5,  41,  63,  -9,  9,  12  ]  },  {  "electric": [  1,  82,  56,  48  ],  "magnetic": [  -2,  58,  96,  14,  94  ],  "gravitational": [  -8,  98,  -5  ]  },  {  "electric": [  68,  81,  40,  -2,  -8  ],  "magnetic": [  74,  -12,  67,  54  ],  "gravitational": [  85,  -6  ]  }  ] | electric;82;-2;43.08;560  gravitational;98;5;44.71;313  magnetic;96;-2;51.33;616 |

## Примечания

Общение с сервером происходит по протоколу HTTP.

Набрать здесьОтправить файл

# **G. Клуб любителей винила**

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | input.db |
| Вывод | стандартный вывод или output.txt |

В очередном заседании клуба любителей виниловых пластинок разгорелись нешуточные страсти – Михаил и Владимир поспорили, кто сможет перечислить как можно больше пар исполнитель-альбом, которые были записаны на студиях «Граммофон» или «Скрипичный ключ». При этом по правилам клуба принято считать, что пластинки с рейтингом «4 и ниже» не стоят внимания членов клуба, а упоминать пластинки с рейтингом «10 из 10» считается плохим тоном.

Полемика Михаила и Владимира грозит расколом клуба.

К счастью, у вас под рукой оказался ноутбук и доступ к базе данных рейтингов.

Даны таблицы со следующим содержимым:

musicians (   
    id **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,   
    name **VARCHAR**(20)   
);   
labels (   
    id **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,   
    name **VARCHAR**(20)   
);   
albums (   
    id **INTEGER** **PRIMARY** **KEY** AUTOINCREMENT,   
    name **VARCHAR**(20),   
    rating **INTEGER**,   
    musician\_id **INTEGER**,   
    label\_id **INTEGER**   
);

Нужно вывести все строки «музыкант» – «лэйбл» – «альбом» – «рейтинг», такие, что лэйбл равен «Граммофон» или «Скрипичный ключ», и альбомы имеют рейтинг от 5 до 9включительно.

Сделать вывод в порядке возрастания имени музыканта.

На следующей базе данных:

|  |  |
| --- | --- |
| musicians | |
| id | name |
| 1 | Петя |
| 2 | Маша |
| labels | |
| id | name |
| 1 | Граммофон |
| 2 | Скрипичный ключ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | albums | | |  |
| id | name | rating | musician\_id | label\_id |
| 1 | Дебютный альбом | 8 | 1 | 1 |
| 2 | Этюды | 10 | 2 | 1 |
| 3 | Осень | 6 | 2 | 2 |

Результат должен быть:

|  |  |  |  |
| --- | --- | --- | --- |
| Маша | Скрипичный ключ | Осень | 6 |
| Петя | Граммофон | Дебютный альбом | 8 |

## Примечания

В качестве ответа надо написать запрос на языке SQL.   
  
Пример какого-то корректного запроса на языке SQL, выводящего названия альбомов и их рейтинг в порядке возрастания рейтинга:

**SELECT**   
    albums.name,   
    albums.rating   
**FROM**   
    albums   
**ORDER** **BY**   
    albums.rating

Набрать здесьОтправить файл