Assignments 05

Week 5 of Programming for Data Scientists: Python

Problems

05-A	Cosmic Culture
05-B	Antimatter action
05-C	Coin Class
05-D	Diplomacy database
05-E	Weird walks

05-A Cosmic Culture

Time limit: 10s

The Planetary Union has decided to launch a new mission for the spaceship Avis. Under the command of Dr. David Poole and H.G. Shatner, the goal of this campaign is to discover new worlds, new civilizations, and to go where no man has gone before. But before the mission can begin, preparations must be made, so the Avis is currently docked at the UBIK space station, 42,000 miles above Earth.

To improve interplanetary cultural exchange, Admiral Dooku Serenno has ordered to fill the database with books to provide new civilizations with information about life on Earth. Unfortunately, the board computer HAL 9000 has been behaving a bit strangely since the last update, so you have to check that all the literature information fits together. A book in the system has a title, an author, a genre, and a time of publication. To speed things up, you get a basic class `Book` that you have to extend to print out all the information:

```
class Book:
    def __init__(self, title, author, genre, time_of_publication):
        pass

    def age(self):
        pass

    def __str__(self):
        pass

book_list = input().split(", ")
book = Book(*book_list)
print(book)
```

Your task is to complete the class, ensuring that the system functions as intended. (Replace the `pass` statements with your own code.) The code is also available for download on the judge system.

Input

One line containing a string describing a book in the format: "`<title>, <author>, <genre>, <time_of_publication>`". They are separated by commas and spaces.

- `<title>`, `<author>`, and `<genre>` are strings composed of printable ASCII characters.
- `<time_of_publication>` is a 4-digit integer denoting the year the book was published.

Output

Print a single line: "<title> by <author> is now <age> years old.", where <age> is calculated based on the current year. Use the provided class to process and print the information.

Input 1

```
Fight Club, Chuck Palahniuk, satirical novel, 1996
```

Output 1

```
Fight Club by Chuck Palahniuk is now 28 years old.
```

Input 2

```
The Great Gatsby, F. Scott Fitzgerald, novel, 1925
```

Output 2

```
The Great Gatsby by F. Scott Fitzgerald is now 99 years old.
```

05-B Antimatter action

Time limit: 10s

A few days into the mission, the Avis has just passed Tetra, moon of the planet venus, when the last updates to the ship's antimatter reactor are installed. At the suggestion of Chief Engineer Jean-Luc Conger, Dr Poole begins a benchmark test of the new power unit and decides to reach the planet Eloi, a few light years away behind the Kessel dust cloud, as quickly as possible (in the hope of setting a new record and completing the Kessel run in less than 12 parsecs).

The test is partially successful: The ship reaches the planet, but unfortunately due to the poor workmanship of the T800-layer (antimatter shielding material) surrounding the reactor, some particles escaped the engine and damaged a hard drive in the computer standing next to it, causing some memory loss. Can you help the ship's technician Winston Smith restoring it?

You have been provided with a Python code by the ship's technician Winston Smith, however, some parts of the code are missing. It is supposed to describe how inheritance works in Python, he explains, but he is not sure how to complete it. Your task is to fill in the missing fragments to complete the code. Ensure that you do not modify or remove the code that is already present.

Below is the provided code:

```
class ParentClass:
def __init__(self):
    self.__private_attribute = 0
    self._protected_attribute = 1
    self.public_attribute = 2

    def __private_method(self):
        print("ParentClass.__private_method() called")

    def _protected_method(self):
        print("ParentClass._protected_method() called")

    def public_method(self):
        print("ParentClass.public_method() called")

class ChildClass(ADD CODE HERE):
    def __init__(self):
        # ADD CODE HERE

    def __private_method(self):
        print("ChildClass.__private_method() called")

    def _protected_method(self):
        print("ChildClass._protected_method() called")
```

```
        def public_method(self):
            print("ChildClass.public_method() called")

    def main():
        child = ChildClass()
        print(child.public_attribute)
        child.public_method()
        child._protected_method()
        child._ParentClass__private_method()
        child._ChildClass__private_method()

    if __name__ == "__main__":
        main()
```

The code is also available for download on the judge system. You need to complete the class to make it functional. Try inheriting from the parent class and see if you can call the methods and access the attributes of the parent class.

Input

/No input/

Output

A single run of the Python code should produce the correct output, if the code is completed correctly.

Input 1

```
```

Output 1

```
2
ChildClass.public_method() called
ChildClass._protected_method() called
ParentClass.__private_method() called
ChildClass.__private_method() called
```

05-C Coin Class

Time limit: 10s

After exactly 28 days, 6 hours, 42 minutes and twelve seconds the Avis enters the Voigt-Kampff-system, a solar system consisting of two interesting planets. Parracis is an unpopulated desert planet controlled by its neighbour planet which has the largest known reserves of Zeta-Reticulium a rare mineral which is essential for powering the antimatter reactors and thus intergalactical traveling in general.

Biff is the opposite of Parracis. The planet is densely populated and its citizens live in relative Since the way to the deeper universe would lead the Avis in the general direction of the Voigt-Kampff system, it was decided that this would be the right place to stock up on reserves for the discovery, so H.G. Shatner has to fly down to the planet alone (Biffians are suspicious by nature...) and climb the mountain leading from the landing zone to the Intergalctical Contact Center in the capital (... and also unfriendly and unwelcoming) where he can buy the substance.

In preparation for this journey, the correct amount of Coins, since Biff is the only planet using exclusively coins - has to be collected. Unfortunately, the coin system is rather complicated: only coins that are rectangles, parallelograms or trapeziums are accepted. Biff's government calls these coins quadrilaterals.

The value of a coin is determined by is base area (that's right: the we are not interested in the height or material of a coin). Can you help the team to complete the class given below? The `area` method is abstract, ensuring that any subclass must implement it. Try creating a subclass for each type of quadrilateral and implement the 'area' method for each.

```
from abc import ABC, abstractmethod

class Quadrilateral(ABC):
    def __init__(self, a, b, h):
        self.a = int(a)
        self.b = int(b)
        self.h = int(h)

    @abstractmethod
    def area(self):
        pass
```

The code is also available for download on the judge system. The team of the Avis has already found out the following formulas for calculating the area of each type of quadrilateral:

- Rectangle: $A = a \cdot b$
- Parallelogram: $A = b \cdot h$
- Trapezium: $A = \frac{(a+b) \cdot h}{2}$

Input

The first line will contain a string of quadrilateral details separated by commas. Each detail is given in the format: "type a b h" (without quotes).

Note: For rectangles, ‘h‘ will always be ‘0‘ as it’s not needed for calculating the area.

Output

Output the value of each quadrilateral in the same order as input. The output should be an integer. It is guaranteed that the area will be an integer.

Input 1

```
rectangle 4 5 0, parallelogram 0 5 4, trapezium 4 5 4
```

Output 1

```
[20, 20, 18]
```

Input 2

```
rectangle 582 867 0, rectangle 261 120 0, parallelogram 0 667 388
```

Output 2

```
[504594, 31320, 258796]
```

Input 3

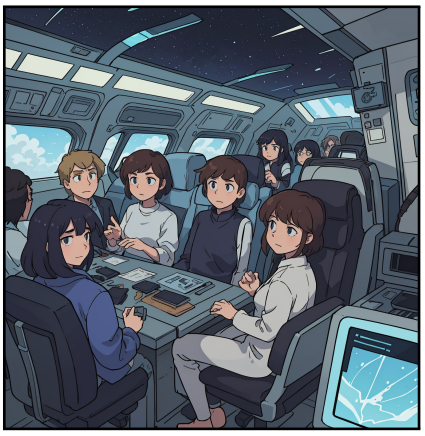
```
parallelogram 0 382 273, rectangle 190 887 0, rectangle 346 514 0, trapezium 82
```

Output 3

```
[104286, 168530, 177844, 120204, 281982]
```

05-D Diplomacy database

Time limit: 10s



On their travels, the crew of the Avis are always keen to establish diplomatic relations with species and cultures in order to make space a peaceful place. This is complicated by the fact that the universe is really big[citation needed]. Since it is frowned upon to start new relations with societies (and even alliances) that already have diplomatic relations, it is important to keep track of this.[1]

To help with this often confusing and complicated task, the Union plans to implement the Track Relations and Other Nexuses System (or TRON for short). The program will take the name, years since first contact and nexus for potential new partners as input, and should return a list of this information, plus True/False depending on whether it is the same nexus as in the first input.

The module dataclasses provides a decorator that can be used to automatically generate special methods for a class. This is useful if you need standard methods like `__init__` or `__repr__` but don't want to write them yourself.

You need to create a class representation of these relations. The class, named `Relation`, should have the following attributes:

- `name`: The name of the species or culture.
- `age`: The number of years since first contact. (An integer)
- `nexus`: The nexus of the species or culture.

The class should also have methods to compare two objects of the class (based on their attributes) and another method to return a string representation of the object in a format similar to its initialization.

Given a list of relations as input, your task is to return a list where, for each relation, it checks whether the relation is the same (equal) as the first diplomatic relation in the list (True/False) followed by its string representation. For a relation to be equal, all attributes must be equal.

Input

A single string containing a list of relations. Each is defined by its name, age, and nexus separated by spaces. Different relations are separated by a comma and a space.

[1] Around 42 years ago, the crew of the Millennium Hawk offered diplomatic relations to representatives of the planet Kaylonia, a cybernic species closely allied with the cyborgs of the planet Bork, with whom the Planetary Union was at that time in dispute. A war, which would have had catastrophic consequences, especially for the biological side, was averted in the last second when Klaatu B. Nikto, President of the PU, was able to convince the Cyber Alliance that the Hawk crew didn't officially represent the Union. Since that incident, relations with both the Borks and the Kaylons have been frosty, but peaceful.

Output

Print a list, where each element is either ‘True‘ or ‘False‘ (depending on if it is identical to the first relation in the list) followed by the string representation of the object. You might need to call the `str` function on the object to get the correct string representation. Obvioulsy, the first element of the list should be ‘True‘.

Input 1

```
Lems 3771 ephedra, Asimovians 1122 ephedra
```

Output 1

```
[True, "Relation(name='Lems', age=3771, nexus='ephedra')", False, "Relation(nam
```

Input 2

```
Pappa 4242 orville, Portas 1984 verne, Pappa 4242 orville
```

Output 2

```
[True, "Relation(name='Pappa', age=4242, nexus='orville')", False, "Relation(na
```

05-E Weird walks

Time limit: 10s

THIS PROBLEM IS MORE COMPLEX.

The Avis left the known universe years ago, and is discovering strange new worlds as it travels deeper into space. The latest example is a comet near Nebuchadnezzar's Nebula: its scans show that it is filled with large amounts of dark matter, making it extremely dense and the gravity strong enough that you could stand on its surface without floating into space. But the strangest thing of all is that, due to its extremely fast rotation, the comet is torus-shaped (i.e. it looks like a doughnut). To everyone's surprise, when this comet is scanned, it turns out to be inhabited.

The inhabitants appear to be visitors from Planet V. As these guys are always interesting to watch and generally friendly individuals, Dr Poole decides to stay in orbit for a while to study the processes taking place on the surface. The background is simple: V is inhabited by three distinct species.

The first, known in the local language as Lems, move at one distance unit per hour, the second - the Asimovians - are three times faster and the third, the Julvers, who are medium speed, 2 units per hour.

- Asimovians: 3 units/hour
- Lems: 1 unit/hour
- Julvers: 2 units/hour

As each species has a different way of moving forward, the Avis observes the comet for a while and finds out that

- The Asimovians move right in the first hour, up in the second, right in the third and so on.
- The Lems move left and to down alternately every hour. The first hour it moves to the left. The second hour it moves down...
- Finally, the Julvers seem to move clockwise. The first hour they move to the right, the second to the bottom, the third to the left...

If two Vs try to occupy the same square, the one already there stays and the new one does not move. However, they will move differently in the next hour.

In a single hour, the visitors move in the order in which they are listed in the input. Given the positions of the different species and a number of hours, you have to simulate the movement on this comet to predict where they will land and have a chat with them.

The comet can be mapped as a 10x10 grid, so to map out the comet, you can use a 10x10 grid with the top left corner being $(0, 0)$ and the bottom right corner being $(9, 9)$. As the comet is torus-shaped, the left and right edges are connected, as are the top and bottom edges. This means that if an species moves off the left edge, it will reappear on the right edge, and if it moves off the top edge, it will reappear on the bottom edge.

Input

The input consists of a single line. The line starts with an integer h, the number of hours to simulate. This is followed by a comma and then a list of species with their initial positions. Each species' description consists of its name (either "asimovians", "lems", or "julvers") and two integers x and y ($0 \leq x, y < 10$) separated by spaces, indicating the species' starting position in the grid. The list is comma-separated.

Be aware that x denotes the horizontal position from left to right, and y denotes the vertical position from top to bottom. In particular, top left corner is $(0, 0)$.

Output

Output the state of the world as a 10x10 grid after h hours of simulation. Represent an empty unit by a minus sign (-). Represent species with the first letter of their names (a for asimovians, l for lems, and j for julvers).

Input 1

```
6, asimovians 0 0, lems 1 1, julvers 2 5
```

Output 1

```
- - - - - - - - - -
- - - - - - - - - a
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - l -
- - - - - - - - - -
- - - - - - - - - -
- - - - j - - - - -
- - - - - - - - - -
- - - - - - - - - -
```

Input 2

```
18, julvers 1 4, asimovians 7 7
```

Output 2

```
- a - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - j - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
```

Input 3

```
100, lems 6 0, lems 8 7, lems 4 7
```

Output 3

```
- - - - - - l - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - l - - - l -
- - - - - - - - - -
- - - - - - - - - -
```

Input 4

```
2, lems 4 2, julvers 3 2, asimovians 2 2, asimovians 8 3, julvers 4 0, lems 2 9
```

Output 4

```
- a - - j - - l - -
a - - - a j - j - a
- - - - l - a l - -
- a - - - - l - - -
- - - - l j - - - -
- - - - l a - j l j
- - j - l - - - - j
- - - - j - - - - a
- - - - - - - - l j
- l a - l - - - - -
```