ООП Python

продолжение

```
class Planet:
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return self.name

earth = Planet("Earth")
print(earth)
```

```
class Planet:
    def __init__(self, name):
        self.name = name

    def __repr__(self):
        return f"Planet {self.name}"

solar_system = []
planet_names = ["Mercury", "Venus", "Earth",
                "Mars", "Jupiter", "Saturn", "Uranus", "Neptune"]

for name in planet_names:
    planet = Planet(name)
    solar_system.append(planet)

print(solar_system)
```

```
class Planet:
    count = 0
    def __init__(self, name, population = None):
        self.name = name
        self.population = population or []
        Planet.count+=1

earth = Planet("Earth")
mars = Planet("Mars")

print(Planet.count)
#ищет метод у текущего объекта, если не находит ищет этот метод у класса
print(mars.count)
```

```
class Planet:  
    #деструктор класса  
    def __del__(self):  
        print("Goodbye")
```

```
class Planet:
    count = 1
    def __init__(self, name, population = None):
        self.name = name
        self.population = population or []

planet = Planet("Earth")
print(planet.__dict__) #посмотреть свойства
```

```
class Human:

    def __init__(self, name, age = 0):
        self.name = name
        self.age = age

class Planet:
    def __init__(self, name, population = None):
        self.name = name
        self.population = population or []
    def add_human(self, human):
        print(f"Welcome to {self.name}, {human.name}!")
        self.population.append(human)

mars = Planet("Mars")

nata = Human("Nata")
mars.add_human(nata)
```

```
# механизмами защиты атрибутов, которые определены внутри класса  
# есть такое соглашение, что если атрибут либо метод названы с  
# символа нижнего подчёркивания, то ими пользоваться не рекомендуется  
# потому, что в дальнейших версиях той или иной библиотеки программист,  
# который пишет может либо отказаться от этих атрибутов или методов,  
# начинающихся с символа нижнего подчеркивания, либо поменять  
# их поведение каким-то образом.
```

```
class Human:  
    def __init__(self, name, age = 0):  
        self._name = name  
        self._age = age  
    def _say(self, text):  
        print(text)  
    def say_name(self):  
        self._say(f"Hello, I am {self._name}")  
    def say_how_old(self):  
        self._say(f"I am {self._age} yars old")
```

```
nata = Human("Nata", age = 29)  
nata.say_how_old()  
nata.say_name()
```

```
#не рекомендуется!  
print(nata._name)
```

```
#не рекомендуется!  
print(nata._say("Whatevet we want"))
```

Задание:

- 1) Класс Дробное число со знаком (Fractions). Число должно быть представлено двумя полями: целая часть - длинное целое со знаком, дробная часть - беззнаковое короткое целое. Реализовать арифметические операции сложения, вычитания, умножения и операции сравнения. В функции main проверить эти методы.
- 2) Класс Равнобокая трапеция, члены класса: координаты 4-х точек. Предусмотреть в классе методы: проверка, является ли фигура равнобокой трапецией; вычисления и вывода сведений о фигуре: длины сторон, периметр. В функции main продемонстрировать работу с классом: дано N (придумайте сами количество) трапеций, найти количество трапеций, у которых площадь больше средней площади.

PS: Трапеция равнобокая, если у нее равны боковые стороны!

Чтобы найти длину стороны по координатам, есть формула:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Задание:

1) Описать класс «Каталог библиотеки». Каждая запись каталога содержит информацию о книге – название, автор, количество экземпляров, количество экземпляров «на руках». Поля приватные. Написать сеттеры и геттеры. Создать список книг каталога. Формирование печати каталога, поиска книг в библиотеке (вывести сообщение есть такая книга или нет такой), добавления книг в библиотеку, удаления книг из нее.