Объекты в духе Python

Представление объекта

- `repr()` — вернуть строку, представляющую объект в виде, удобном для разработчика.
- `str()` — вернуть строку, представляющую объект в виде, удобном для пользователя.

Для поддержки repr и str мы должны реализовать методы `__repr__` и `__str__`. Существует еще два специальным метода для поддержки альтернативных представлений объектов `__bytes__` и `__format__`

И снова класс вектора

```
import math
from array import array


class Vector2d:
    """
    self.typecode -- аттрибут класса для преобразования экземпляра Vector2d
последовательность байтов и обратно.
    """
    typecode = 'd'

    def __init__(self, x, y):
        """
        Преобразование x, y в тип float в методе инициализации позволяет на ранней
стадии обнаруживать ошибки.
        Это полезно когда конструктор класса вызывается с не подходящими аргументами
        """
        self.x = float(x)
        self.y = float(y)

    def __iter__(self):
        """
        Наличие __iter__ делает объект итерируемым.
        Благодаря ему работает распаковка (пр.: x, y = ny_vector).
        В данном случае реализация при помощи генераторного выражения, который отдает
компоненты поочередно.
        """
        return (i for i in (self.x, self.y))

    def __repr__(self):
        class_name = type(self).__name__
        return f"{class_name} ({self.x}, {self.y})"

    def __str__(self):
        return str(tuple(self))

    def __bytes__(self):
        return (bytes([ord(self.typecode)]) +
                bytes(array(self.typecode, self)))

    def __eq__(self, other):
        return tuple(self) == tuple(other)

    def __abs__(self):
        return math.hypot(self.x, self.y)

    def __bool__(self):
        return bool(abs(self))
```