Задачи урока 1 (классы и объекты)

Пример 1. Класс Time12 для работы с часами по 12-часовому формату

```
class Time12():
    """
    Класс для работы с часами в 12-часовом формате (без AM/PM)
    """

    # конструктор класса имеет специальное имя __init__, по умолчанию сделаем
время 00:00
    def __init__(self, h=0, m=0):       # здесь определяем переменные
объекта:
        self.h = h                      # часы от 0 до 11
        self.m = m                      # минуты от 0 до 59

    # тут же определяем другие методы (функции) класса

    def print(self):
        """ Печатает время в 12-часовом формате (без AM/PM)"""
        print("%02d:%02d" % (self.h, self.m))

    # настоящие программисты не пишут функцию print, а переопределяют
(расскажут позже) функцию __str__(self)
    def __str__(self):
        return '%02d:%02d' % (self.h, self.m)


    def set(self, h24=0, m24=0):
        """ Устанавливает время, время можно задать в 24-часовом формате """
        if h24 < 0 or h24 >= 24 or m24 < 0 or m24 >= 60:
            raise ValueError("hour=%d min=%d" % (h24, m24))
        self.m = m24
        self.h = h24 % 12

    def add(self, dt):
        """ Увеличивает время на dt. dt типа Time12"""
        m = self.m + dt.m       # self.m - минуты ЭТОГО объекта, dt.m -
минуты переданного объекта типа Time12,
                                # m - новая переменная, локальная этого
метода (не путать с аттрибутом self.m)
        self.m = m % 60
        self.h = (self.h + dt.h + m//60) % 12

    def round(self):
        """Возвращает время (объект Time12), округленное вверх до 0 минут"""
        # можно ли тут писать t = self или не надо?
        t = Time12()            # чтобы вернуть объект, его нужно создать,
t.m = 0
        t.h = self.h
        if self.m > 0:
            t.h = (t.h + 1) % 12
        return t
```

```
# тут закончилось описание класса, его полей (аттрибутов) и методов.

if __name__ == "__main__":
    # как можно использовать этот класс (и его нужно протестировать!)
    t1 = Time12(9, 20)     # создаем объект t1 класса Time12 со значениями
9:20
    t1.print()              # вызываем метод print объекта t1

    t2 = Time12()           # создаем другой объект t2 класса Time12 со
значениями 0:0
    t2.print()              # 00:00

    t2.set(2, 55)
    print(t2)               # 02:55 - неявно вызвали функцию Time12.__str__

    t1.add(t2)
    t1.print()              # 00:15
    t2.print()              # 02:55

    t3 = t1.round()
    t3.print()              # 01:00
```

Задача 1. Дописать класс Time12

В классе Time12 дописать функцию вычитания и сравнения времени:

```
    def sub(self, dt):
        """ сколько времени было на dt (типа Time12) раньше"""
        # тут нужно написать код

    def compare(self, t):
        """ сравнивает время объекта с переданным временем t
        Возвращает:
        1 если текущее время больше аргумента t
        0 если времена совпадают
        -1 если текущее время меньше аргумента t
        """
        # тут нужно написать код
```

Проверить и отладить функции

Задача 2.

Сначала у Шляпсика и Бяксика были нормальные одинаковые часы. Когда они показывали время t0, Шляпсик решил переводить их на t1 минут вперед каждые 30 минут, а Бяксик стал переводить из на t2 минут назад каждый час.

Написать программу, которая выясняет через сколько суток, часы Шляпсика и Бяксика покажут одинаковое время и какое это будет время.

Не нужно выводить формул, решаем задачу через циклы

Задача 3. class Point

На координатной плоскости у нас есть точки. Каждая точка задана координатами **x** и **y** .

Опишем класс Point для этих точек.

```
class Point:
    """ класс работы с точками на плоскости XY"""
    def __init__(self, x=0, y=0):
```

```
        self.x = x
        self.y = y

    def read(self):
        """ прочитать значение точки с клавиатуры в формате 1 число на
строку"""
        # тут нужно написать код

    def shift(self, dx=0, dy=0):
        """ сдвинуть точку на dx, dy """
        # тут нужно написать код

    def dx(self, a):
        """ возвращает на сколько нужно передвинуть точку по оси X до точки
a"""
        # тут нужно написать код

    def dy(self, a):
        """ возвращает на сколько нужно передвинуть точку по оси X до точки
a"""
        # тут нужно написать код

    def __str__(self):
        """ возвращает строку с координатами точки в формате x y (на одной
строке через пробел)"""
        # тут нужно написать код
```

Класс Point создать в отдельном файле point.py

Реализовать все функции класса. **Проверить их**.

Как взаимодействуют объекты

Объект может:

- передавать сообщения обращаясь к методам и атрибутам другого объекта (если есть доступ)
- создавать другие объекты
- объект может быть частью другого объекта
- удалять созданные объекты, к которым он имеет доступ

Найдите пример каждого пункта в решении следующей задачи.

Задача 4. class Rect

Прямоугольники имеют стороны параллельные осям координат.

Класс **Point** возьмем из предыдущей задачи.

Прямоугольник задается двумя точками: верхняя левая и правая нижняя.

Значит, в описании прямоугольника должен участвовать класс Point. То есть в объект типа прямоугольник будет иметь атрибуты типа точка.

Опишем class Rect в файле rect.py

Можно добавить свои методы.

```
# тут должен быть нужный import
```

```
class Rect:
    """ прямоугольник со сторонами, параллельными осям X и У"""

    def __init__(self, lt=Point(), rb=Point()):
        self.lt = lt
        self.rb = rb

    def center(self):
        """возвращает точку - центр прямоугольника"""
        pass

    def set(self, lt, rb):
        """ задает точки """
        pass


    def read(self):
        """ читает прямоугольник с клавиатуры, по 1 числу на строку"""
        pass

    def __str__(self):
        return ''

    def moveTo(self, a):
        """сдвигает прямоугольник так, чтобы его центр был в точке a"""
        pass

    def flipH(self):
        """ отображает прямоугольник относительно оси ОУ"""
        pass

    def flipV(self):
        """ отображает прямоугольник относительно оси ОХ"""
        pass

    def rot90(self):
        """ поворачивает прямоугольник на 90 градусов относительно его центра
"""
        pass

    def cross(self, rect):
        """ Возвращает прямоугольник, пересечение текущего и rect. Если не
пересекаются, возвращает None"""
        return None
```

Задача 5. class Line

Продумать и реализовать функциональность класса **Line**. Использовать его для нахождения пересекающихся прямоугольников

Задача 6. (дополнительная) - пересекающиеся прямоугольники.

Даны 2 или более точек. Создать динамический массив всех возможных прямоугольников из этих точек. Прямоугольник задается парой точек (левая верхняя и правая нижняя). Стороны прямоугольников параллельны осям координат. Подсчитать сколько всего прямоугольников будет видно (учитывая пересечения).

Объединение прямоугольников в один НЕ делать.

Вариант 1: совпадающие прямоугольники считать разными.

Вариант 2: совпадающие прямоугольники считать 1 прямоугольником.

Вариант 3: добавлять объединения прямоугольников (стоящие рядом можно объединить в один).

Авторы задач: Овсянникова Т.В.