**Антипаттерны в коде:**

**- Blind faith**

1) Нет валидации вводимых данных. Соответственно, можно на страницах редакторов курсов или студентов попробовать настроить валидацию вводимых данных, чтобы отсечб некорректный ввод

**- Cryptic Code**

1) Убраны все непонятные сокращения, в каких то местах переменные с более длинным названием, но так понятнее

**- Hard Code**

1) Порт, по которому подключаться, был явно указан в функции, вытащил его как отдельный параметр.

**- Специфические для Python антипаттерны в коде**

1) Сделал более техничное создание словаря в части создания API для курсов

**Антипаттерны в коде:**

**- Singletonitis**

1) Есть класс на основе Синглтона (логгер). Там я немного переиграл, чтобы можно было создавать 2 объекта логгера. Но, мне кажется, конкретно в этом проекте можно убрать насследование от синглтона, здесь много логгеров не планируется по умолчанию

**- Privatization**

1) Изначально были защищенные атрибуты в классе Subject (Наблюдаемый). Я сделал их публичными

**Методологические антипаттерны:**

**- Improbability factor**

1) Я так полагаю, это относится к антипаттерну «Слепая Вера»

**- Premature optimization**

1) есть пару проблемных кусков кода, я его отметил комментариями.

**- Reinventing wheel**

1) в моем случае, изобретение колеса - это создание api через модуль json, а можно было через jsonpickle, но я пытался немного другой формат вывода сделать

**Архитектурные антипаттерны:**

**- Abstract Inversion**

**- Big ball of mud**

1) есть определенная анархия в действиях, как вариант,отдельным модулем вынесу обработку запросов.

2) Также планирую к завершению работы над проектом вытащить из классов действия, связанные с другими классами, чтобы сделать их между собой более независимыми. Подумаю, как это сделать

**- Input kludge**

1) Отсутствует валидация вводимых данных. Надо добавлять