Geekbrains

**Программа для составления учебного расписания для школьников средней ступени обучения (пилотная версия)**

Программа:

Разработчик - Data Engineer

Технологическая специализация

Специализация:

Data Science в медицине

ФИО:

Тушина Анастасия Константиновна

Минск

2024

**Содержание**

Введение………………………………………………………………………...3

Теоретическая часть……………………………………………………………7

Практическая часть…………………………………………………………….20

Заключение……………………………………………………………………..39

Список использованной литературы…………………………………………..42

**Введение**

Данные диплом посвящен проблеме *генерации расписания для школьников средней ступени обучения*. Представленный диплом посвящен проблеме генерации оптимального расписания. Проблема лежит в плоскости понятия «исследования операций».

Исследование операций – дисциплина, занимающаяся разработкой и практическим применением методов наиболее эффективного управления различными организационными и производственными системами.

В различных областях практической деятельности организации производства и снабжения, эксплуатации транспорта, расстановке кадров, в бытовом обслуживании, здравоохранении, связи и т.д. часто возникают задачи, сходные между собой по постановке, обладающие рядом общих признаков и решаемые сходными методами, которые удобно объединять под общим названием «задач исследования операций».

Например, организуется какое-то целенаправленное мероприятие (система действий), которое можно организовать тем или другим способом, то есть выбрать решение из ряда возможных вариантов. Каждый вариант обладает какими-то преимуществами и какими-то недостатками, причем, в силу сложности обстановки, не сразу ясно, какой из них предпочтительнее других и почему. С целью прояснить обстановку и сравнить между собой по ряду признаков различные варианты решения организуется серия математических расчетов. Их задача – помочь произвести критический анализ ситуации и принять оптимальное решение.

Этот подход можно применить и к несложной на первый взгляд задаче – составление школьного расписания. Порой составить качественное расписание для нескольких классов, которые подходят под заданные условия, бывает трудоемко и неэффективно. В данной работе приведена попытка разработки алгоритма к задаче составления оптимального расписания для учеников средней школы.

Актуальность проекта определяется преимущественно «ручным» составлением расписания практически во всех учебных заведениях: школах, гимназиях, лицеях, университетах, академиях, учреждениях постдипломного образования. Процесс должен учитывать: количество направлений, учебных предметов, учебные часы, количество учащихся в классах/группах, количество учителей/преподавателей, а также особенности календаря: отпуска/больничные преподавательского состава, праздничные, выходные дни, а также каникулярное время.

Современные концепции и нормативные документы требуют от педагогики как науки разработки и анализа концептуально новых подходов к образовательному процессу. В концепции цифровой трансформации процессов в системе образования указаны основные цели: подготовка обучающихся к жизни в цифровом обществе; темы образования к работе в условиях быстрых изменений: внедрение инновационных технологий, изменение образовательных парадигм, гибкое формирование требований и программ; оптимизация процессов, протекающих в системе образования; обеспечение качества и мобильности предоставляемых образовательных услуг на всех уровнях образования.

Сегодня у современных участников образовательного процесса есть возможность использования карманных персональных компьютеров, интерактивных планшетов, смартфонов, мобильных приложений, социальных сетей и различных иных средств (3-D очки дополненной и виртуальной реальности, умные часы, фитнес браслеты, диктофоны, шагомеры, электронные книги и т.д.), которые можно использовать для организации электронного обучения. Увеличение скорости передачи данных, повышение доступности сети Интернет и расширение диапазона мобильных устройств диктуют целесообразность их применения. Поэтому наличие расписания на стендах отделений или на стенде при входе в учреждение образования вызывает вопрос, а отсутствие возможности узнать свое расписание с помощью современных гаджетов вызывает определенные неудобства.

Одной из наиболее распространенных задач в планировании и оптимизации образовательного процесса в учреждениях образования является составление расписания. От хорошо составленного расписания, в высокой степени зависит насколько эффективно работают преподаватели, усваивается учебный материал обучающимися, рационально используются интеллектуальная и материальная базы учреждения образования. Автоматизации составления расписания – достаточно классическая задача в системах управления учебным процессом учреждения образования, но на данный момент нет единого, общепринятого способа ее решения.

В представленной работе не решаются все описанные проблемы, но предлагается одно из решений проблем подбора и нахождения оптимального расписания.

*Цель диплома* – написать прототип программы, в которой применяется алгоритм построения расписания для школьников средней ступени обучения.

*Задачи:*

1. Изучить литературу, касающуюся темы исследования.

2. Рассмотреть основные виды и методы составления программы

3. Получить практический навык использования программного обеспечения и анализа данных для разработки программы.

4. Создать рабочий прототип программы.

*Критерии эффективности продукта:*

1. Продукт должен являться рабочей программой, на вход которой поступает набор параметров, а на выходе – расписание (критерий пригодности системы).

**Теоретическая часть**

Этапы написания программы

1. **Постановка задачи.** Задача формулируется на естественном языке. Определяются цели. Подготавливается техническое задание на разработку программы.
2. **Обоснованный выбор средств разработки (программирования).** Разрабатываются форматы ввода исходных данных и отображения результатов.
3. **Выбор метода решения задачи.** Анализ возможности использования ранее разработанного и доступного для программиста программного обеспечения.
4. **Разработка алгоритма решения задачи.** Декомпозиция задачи на подзадачи. Определение последовательности решения подзадач. Разработка структуры программы.
5. **Обоснование выбора средств программирования.** Выбор языка программирования и системы (среды) программирования.
6. **Кодирование средствами выбранного языка программирования.**
7. **Верификация и проверка корректности.**  
   Аналитическое доказательство правильности программы.
8. **Тестирование программы.** Разработка тестов и контрольных примеров. Сопоставление реальных и ожидаемых результатов.
9. **Отладка программы в случае обнаружения ошибок.** Локализация обнаруженных ошибок. Коррекция ошибок. Возврат к этапу тестирования.
10. **Разработка документации.** Текстовое описание программы. Разработка инструкций пользователю – лицу, применяющему разработанную программу в своей работе. Разработка инструкций по эксплуатации, содержащих информацию, требующуюся программистам, ответственным за нормальное функционирование программы.
11. **Опытная эксплуатация.** Уточнение требований заказчика к представлению исходных данных и результатов работы программы. При необходимости возврат к предыдущим этапам.
12. **Промышленная эксплуатация.** Сопровождение программы. Обработка требований к новым версиям программы.

Характеристика научной дисциплины исследование операций.

Как представлено ранее, исследование операций (ИО) – область науки, основанная на применении современных разделов математики и тесно связанная с экономикой, кибернетикой, теорией управления системами и другими науками. Целью ИО является получение оптимальных решений задач, возникающих в процессе функционирования или создания сложных систем, включающих коллективы людей, машины и средства вычислительной техники. ИО – это комплекс научных методов, призванных решать задачи организационного управления.

Для полного понимания процессов следует оговорить стандартные определения.

Операция – это организуемое мероприятие (система действий), объединенное единым замыслом, обеспеченное средствами, ресурсами, усилиями и направленное к достижению определенной сформулированной цели.

Цель — это описание желаемого состояния или запланированного результата деятельности, на создание (получение) которого направлены ресурсы, усилия. Чаще всего цель или цели (дерево целей) оформляются в форме требуемых значений различных показателей (в ТТЗ на НИР, ОКР и др.).

Модель — условный образ реального объекта (операции), отражающий его существенные свойства — физический или абстрактный: скульптура, барельеф, игрушки, фотография, карта, схема, рисунок, формула, уравнение, система уравнений, неравенств, граф, сеть, совокупность формул, схем и текста.

Решение – это какой-то конкретный выбор из полного ряда возможностей (ограничения задачи не учитываются).

Допустимые решения - те решения, которые удовлетворяют наложенным в задаче ограничениям.

Оптимальными решениями – являются те допустимые решения, которые по тем или иным соображениям предпочтительнее других. Поэтому основной задачей исследования операций является предварительное количественное обоснование оптимальных решений. Предъявление требования к вероятности выполнения задачи одно из главных при организации исследования. Необходимость принятия решения диктуется наличием цели, которую нужно достичь, например, выполнить плановое задание, выбрать тип прибора, построить план перевозок. Если же цель поставлена или фактическое состояние не соответствует желаемому, то имеет место проблемная ситуация, процесс выработки плана устранения которой и составляет сущность задачи принятия решений. Результат операции зависит от способа ее проведения, организации, т.е. от выбора определенных параметров.

Принятие решений обычно выходит за рамки исследования операций и относится к компетенции ответственного лица, чаще – группы лиц, которым предоставлено право окончательного выбора и на которых возложена ответственность за этот выбор. Делая выбор, они могут учитывать, наряду с рекомендациями, вытекающими из математического расчета, еще ряд соображений (количественного и качественного характера), которые этим расчетом не были учтены.

Эффективность операции — степень ее приспособленности к выполнению задачи, часто измеряется вероятностью выполнения задачи, количественно выражается значением показателя целевой функции.

Чтобы сравнивать между собой по эффективности разные системы, решения нужно выбрать, т.е. назначить в модели какой-то количественно измеряемый показатель, так называемый показатель эффективности – его часто называют «целевой функцией» задачи. Этот показатель выбирается так, чтобы он отражал целевую направленность операции. «Лучшим» будет считаться то решение, которое в максимальной степени способствует достижению поставленной цели.

Критерий эффективности – правило, обеспечивающее фиксирование итогового успешного результата исследования. Рассматриваются три типа критерии эффективности: критерий пригодности системы (система удовлетворяет требованиям, предъявляемым к ней); критерий сравнения систем (из двух или более выбирается та, показатель эффективности которой предпочтительнее); критерий оптимальности (из всех возможных значений показателя эффективности выбирается экстремальное при условии допустимости решения).

Для решения поставленной задачи был выбран код на языке Python.

Python — это высокоуровневый язык программирования, с помощью которого создают сайты, разрабатывают приложения, автоматизируют процессы анализа или визуализации данных. Python не был разработан для конкретных целей, поэтому подходит как для создания алгоритма рекомендаций видеосервиса, так и для разработки программного обеспечения для самоуправляемых автомобилей или управления космическими аппаратами на других планетах. Python — стабильный и распространённый язык. Он используется во многих проектах и в различных качествах: как основной язык программирования или для создания расширений и интеграции приложений. На Python реализовано большое количество проектов, также он активно используется для создания прототипов будущих программ.

К основным характеристикам Python относят:

* Объектную ориентированность. Python — это объектно-ориентированный язык программирования, в его основе лежат понятия объекта и класса. Объект — экземпляр класса, объединяющий данные и способы взаимодействия с ними. Объектная ориентированность Python помогает структурировать код и повысить его читабельность, что особенно важно при реализации крупных проектов.
* Читабельность кода. В Python каждый блок кода выделяется отступами. Это позволяет существенно облегчить визуальное восприятие.
* Интерпретируемость. Код на Python можно запустить на любой платформе, если на ней установлен интерпретатор, с помощью которого последовательно исполняются операторы. Если в процессе программа-интерпретатор сталкивается с ошибкой, исполнение прекращается. Это позволяет разработчику кода быстро найти и исправить недочёт.
* Динамическую типизацию. Когда переменной присваивается значение, это автоматически связывает её с определённым типом. Процесс написания кода на Python значительно ускоряется, но повышается риск возникновения ошибок.

Основные сферы применения Python.

* Веб-разработка. Python благодаря следующим плюсам языка:
* У Python множество библиотечных инструментов, пакетов услуг и удобных фреймворков. Например — гибкий, масштабируемый веб-фреймворк Django включает пакет услуг, в который входит панель управления, интерфейс БД, структура каталога. Из-за большого количества функций необходимо добавить отдельные библиотеки. Это позволит настроить маршрутизацию, создать механизмы шаблонов, выполнить аутентификацию пользователей.
* Python — популярный язык программирования, поэтому у него регулярно появляются новые библиотеки, обновляются функции. Благодаря широкому распространению у Python активное комьюнити, а также множество документации, инструкций и туториалов.
* Благодаря быстрому прототипированию разработчики на Python могут оперативнее реализовать свои идеи — на все итерации уходит значительно меньше времени. Такая скорость делает пайтон особенно удобным для стартапов, ускоряя выход на рынок.
  + Data Science

У Python есть набор инструментов, необходимых для визуализации и анализа данных, например

* Библиотека SciPy помогает выполнять научные вычисления, pandas — аналитику данных, Matplotlib — строить графики.
* Фильтры принимают правила и списки, сравнивают элементы с правилами фильтрации и возвращают новые множества исходных списков. Функция карты также принимает список, но создаёт его новый вариант, выполняя определённую операцию над каждым элементом.
* Лямбда-выражения помогают при работе с операциями, которые выполняются всего один раз. Даже односоставные лямбда-выражения могут делать всё, что заложено в функцию.
* Для визуализации данных, вывода массивов NumPy помогут функции linspace и arrange.
* Сводные таблицы помогают перевести данные из определённых столбцов в многомерное представление.
* Тестирование
* Pytest — специальная экосистема для тестирования, в основе которой лежат плагины. Она предназначена для написания тестов различных типов — от модульных до функциональных.
* Модуль unittest — одна из стандартных библиотек Python, содержащая необходимые инструменты для тестирования кода. Unittest помогает проверять корректность работы отдельных частей функций.
* Doctest — модуль, предназначенный для тестирования документации.
* Tox — библиотека, помогающая тестировать код в разном окружении.
* Mock — модуль, позволяющий упростить модульные тесты на Python.
* Линтер flake8 помогает найти стилистические ошибки в коде.
* Приложения. На Python можно создавать десктопные и мобильные приложения, особенно актуален для работы над графическим интерфейсом. В основном язык используется для серверной части.
  + Игры. Python часто используется как вспомогательный инструмент при разработке масштабных игр:
  + на нём прописывают логику игры: обработку событий, запуск сцен, организацию взаимодействия персонажей;
  + на Python пишут внутриигровые скрипты;
  + его используют для создания элементов управления и интерфейса;
  + помогает в тестировании.

Как и все языки программирования, Python имеет свои преимущества и недостатки.

Преимущества:

* Лёгкость освоения. Python прост, логичен, имеет понятный синтаксис, поэтому подходит даже для новичков. Для его изучения не нужны знания английского.
* Простота визуального воспитания.Блоки кода отделяются отступами, за счёт меньшего объёма код также воспринимается проще.
* Кроссплатформенность. Программы, написанные на Python, могут запускаться и функционировать на всех типах операционных систем. Отличия можно узнать заранее, поскольку они подробно описаны в документации.
* Скорость разработки. Чтобы написать программу на Python нужно значительно меньше кода, чем при разработке, например — на Java.
* Универсальность. Благодаря интерпретируемости используется практически на всех платформах для различных задач — от тестирования до научных исследований. Код удобно писать даже в стандартных текстовых редакторах.
* Множество инструментов. Стандартные библиотеки Python способны решать даже сложные задачи. Установка дополнительных модулей, созданных для конкретных целей, помогает при разработке специальных проектов.
* Масштабируемость. Возможность адаптации высокоуровневой логики позволяет проектам, разработанным на Python, масштабироваться и расширяться.

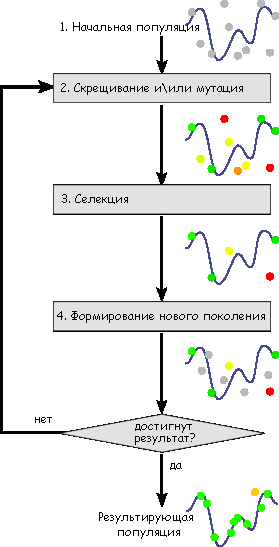
Недостатки:

* Медленная работа. Этот недостаток лишает пайтон-разработчиков возможности создавать высокопроизводительные проекты только на Python. Необходимо задействовать другие языки программирования. Программное обеспечение работает медленно из-за устаревшего кода Global Interpreter Lock, сохранившимся её с первой версии языка.
* Трудность переноса проектов на другие системы. Проблема возникает из-за зависимости языка программирования от библиотек.
* Ресурсоёмкость. Python не подходит для проектов, требующих больших объёмов памяти.

Для написания проекта был выбран алгоритм, напоминающий генетический. Этот алгоритм близок и понятен автору диплома, так как основной вид деятельности – медицина.

Генетический алгоритм - эвристический алгоритм поиска (ответ может быть не идеально точным, но достаточно точным для поставленной задачи), который решает задачи с использованием механизма, похожего на естественный отбор в природе.

Схема работы генетического алгоритма:



Генетические алгоритмы применяются для решения следующих задач:

* Оптимизация функций
* Оптимизация запросов в базах данных
* Разнообразные задачи на графах (задача коммивояжера, раскраска, нахождение паросочетаний)
* Настройка и обучение искусственной нейронной сети
* Задачи компоновки
* Составление расписаний
* Игровые стратегии
* Теория приближений
* Искусственная жизнь
* Биоинформатика (фолдинг белков)
* Синтез конечных автоматов
* Настройка ПИД регуляторов
* Черный ящик
* Обработка сигналов

Основные понятия:

* Днк - набор параметров, который описывает, как выглядит и ведет себя объект (в данном случае - расписание)
* Ген - параметр, который описывает один признак объекта (к примеру, один урок)
* Популяция - набор объектов, каждый со своим днк.
* Приспособленность (fitness) - значение, которое определяет, насколько хорошо данное днк подходит к поставленной задаче (насколько близок объект к идеальному)
* Скрещивание - процесс, во время которого из двух различных объектов создается 1 "ребенок".
* Мутация - процесс, благодаря которому в Днк может случайно измениться один из параметров в лучшую или худшую сторону.
* Селекция - отбор лучших объектов в новую популяцию.

Главная структура алгоритма выглядит так:

* Задать идеал, к которому стремится алгоритм.
* Создать начальную популяцию днк.
* Цикл:

1. Скрещивание.

2. Мутация.

3. Вычисление fitness’а для каждого объекта.

4. Селекция.

Цикл продолжается до тех пор, пока не будет получен достаточно близкий к идеалу объект, либо количество популяций достигнет установленного ограничения.

В последние годы предпринимаются множественные попытки совершенствования планирования учебного процесса путем построения алгоритмов оптимизации задач планирования учебной работы с использованием вычислительной техники и программного обеспечения.

Ручное решение задачи составления расписания занятий требует больших затрат времени, квалифицированных специалистов, в то же время результат такого решения часто получается далеко не оптимальным. После ввода исходной информации требуется её согласование, в то время как невозможность получения требуемого расписания может быть определена ещё на этапе анализа. Во время составления расписания возможно возникновение тупиковых ситуаций. Всё это требует изменения исходных данных и ослабления ограничений, и здесь без человека не обойтись. Без внесения данных изменений расписание не будет иметь практической ценности. Также следует учесть тот момент, что расписание может меняться и во время его использования, т.е. после составления, и здесь весьма важен человеческий фактор. В этом плане важна поддержка данного процесса автоматизированными методами и процедурами. Основное преимущество состоит в том, что автоматизированное составление устраняет массу рутинной работы, такой как: поиск возможных вариантов внесения очередных элементов в расписание, проверку выполнения требований, поиск случайных ошибок в готовом расписании, оформление расписания на бумаге в виде различных таблиц (для преподавателей, групп, покабинетного), оставляя человеку больше времени на более интеллектуальные действия. Компьютер в данном случае также является инструментом, существенно усиливающим способности человека, т.к. человек не в состоянии перебрать и проанализировать такое же количество вариантов расписаний, как компьютер.

Для решения существующих проблем требуется построение гибкой и легко адаптируемой системы на основе новых принципов, с использованием современных веб-технологий. Необходима система, составляющая расписание в соответствии с выбранными критериями и заданными требованиями. Данные возможности должны осуществляться также без изменения исходного кода системы. Для покрытия наиболее типичных случаев необходимо создание нескольких типовых алгоритмов, реализующих составление расписаний. Данная система должна иметь возможность дополнения и изменения существующей базы данных и пользовательского интерфейса. Всё это давало бы возможность задавать в каждом учебном учреждении требования, отвечающие его условиям, и с помощью подбора и настройки подходящего алгоритма получать требуемое расписание.

**Практическая часть**

В общей постановке задача составления расписания представляет собой процесс распределения некоторого конечного набора событий во времени в условиях ресурсных и других ограничений.

В качестве исходных данных для составления расписания выступают перечень специальностей, списки групп студентов, дисциплин, а также учебная нагрузка на группы. Имеющийся ресурс выражен в качестве списка профессорско-преподавательского состава, которые могут читать некоторые предметы в заданных группах, списка аудиторий, количества учебных дней в неделю, максимального количества занятий в день.

Необходимо составить оптимальное расписание занятий для каждой группы по дням недели, распределить учебные аудитории в каждый из дней недели, распределить нагрузку на преподавателей по дням с учетом их пожеланий. То есть наиболее удобным образом распределить имеющиеся ресурсы в соответствии с запросами и выделить время для занятий каждой группы.

Существование удачно составленных расписаний говорит о том, что задача составления оптимального расписания разрешима, или хотя бы о том, что для нее существуют допустимые решения.

Продуктом данного диплома является программа на языке Python, построенная по принципу объектно-ориентированного программирования, которая, используя генетический алгоритм, по заданным параметрам *(количество уроков, набор уроков, дни недели, ...)* создает приближенное к идеальному расписание. Расписания сохраняются в формате CSV и удовлетворяют поставленным условиям *(к примеру, недопустимо 3 одинаковых урока подряд, или недопустимо одни и те же уроки у разных классов в одно и то же время)* и параметрам.

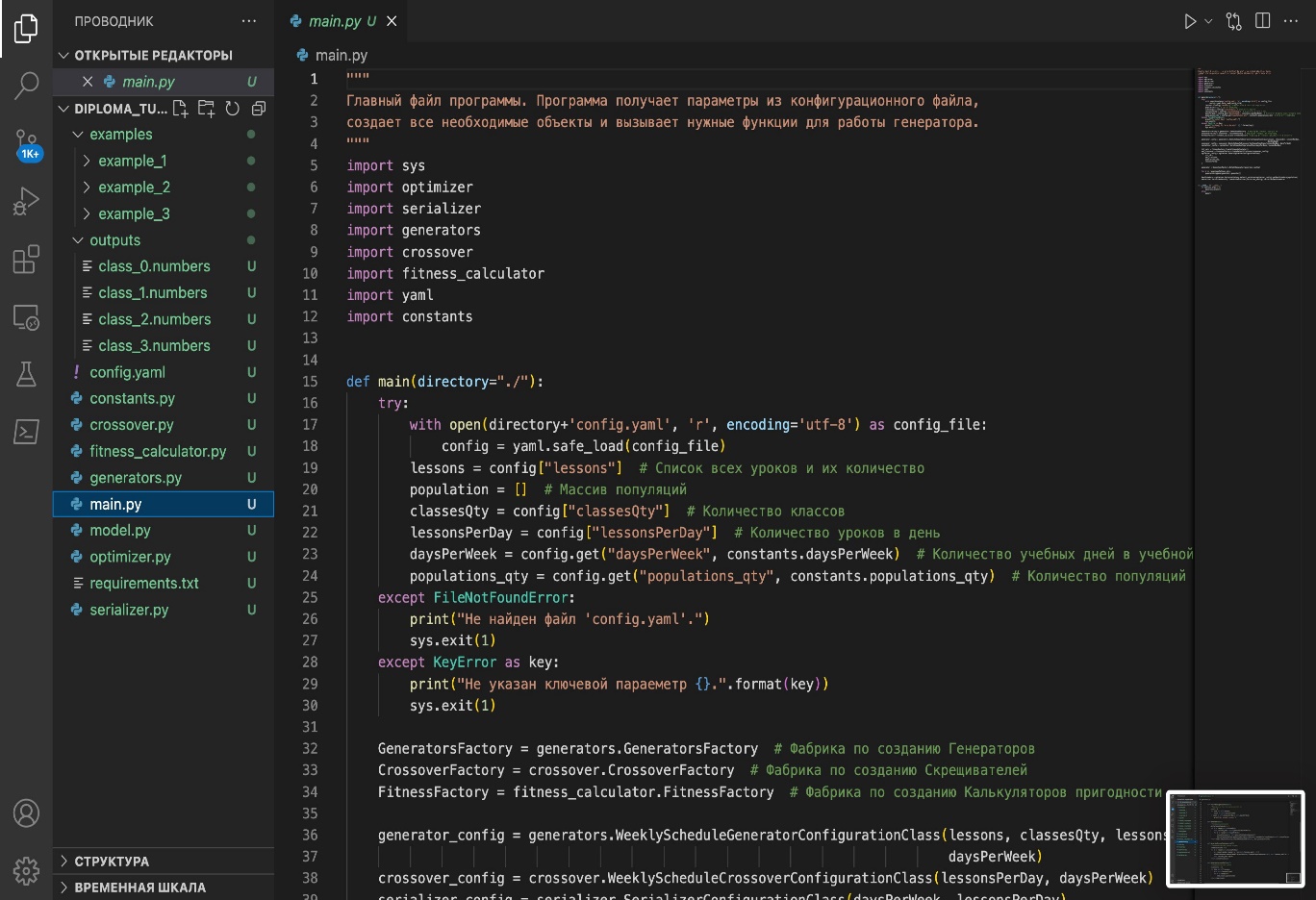
Основные определения:

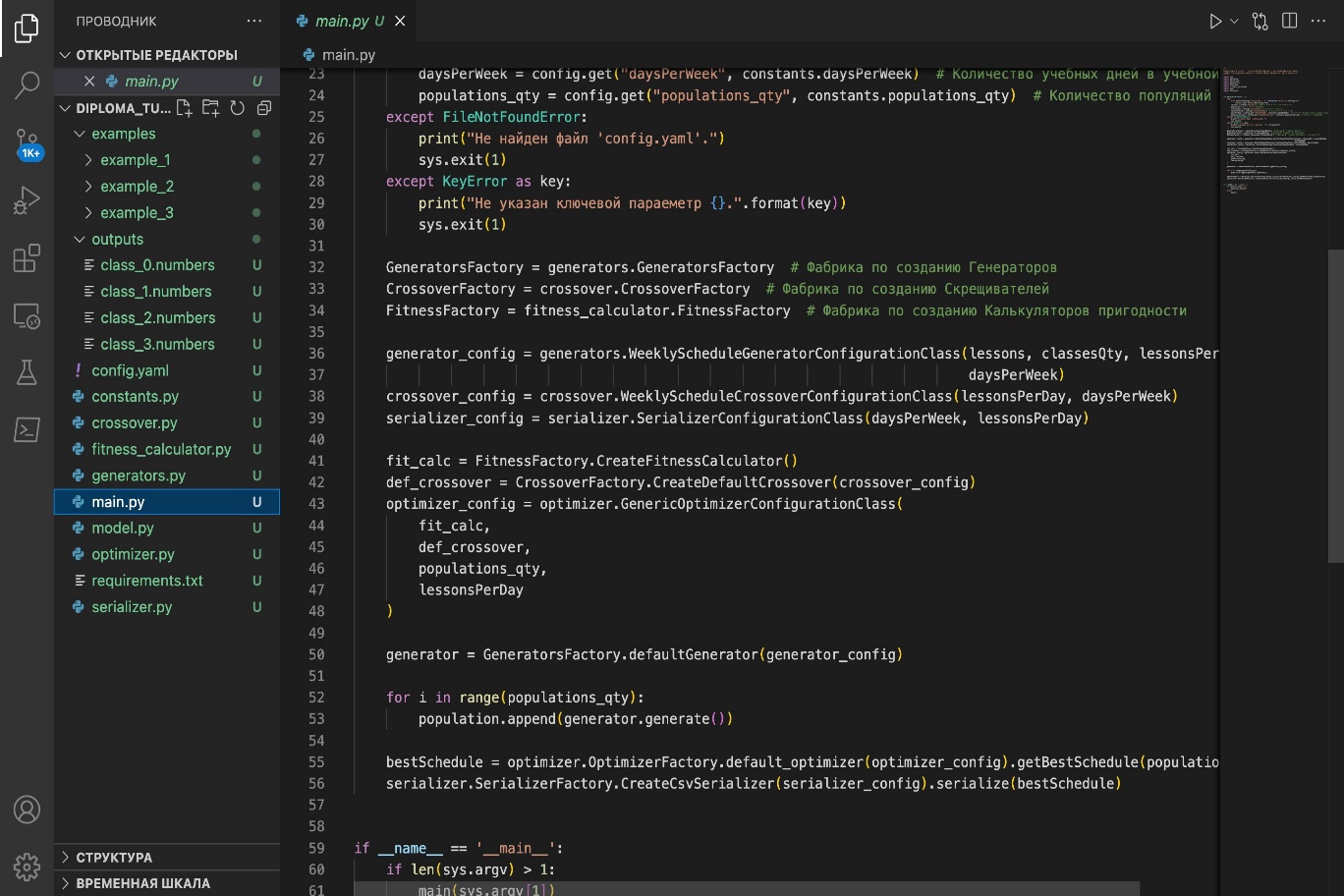
Объектно-ориентированное программирование (ООП) – совокупность методов, основанные на представлении программы в виде набора объектов, каждый из которых является экземпляром определенного класса, а классы наследуются друг от друга.

* Расписание - план школьных уроков за определенный промежуток времени (день, неделя, месяц, ...)
* Урок - промежуток времени, во время которого ученики изучают 1 определенный предмет (математика, русский)
* Генератор –фабрика класса.

Список файлов и классов:

* main - основной файл программы. Здесь программа получает параметры из конфигурационного файла, создает все необходимые объекты и вызывает нужные функции для работы генератора.



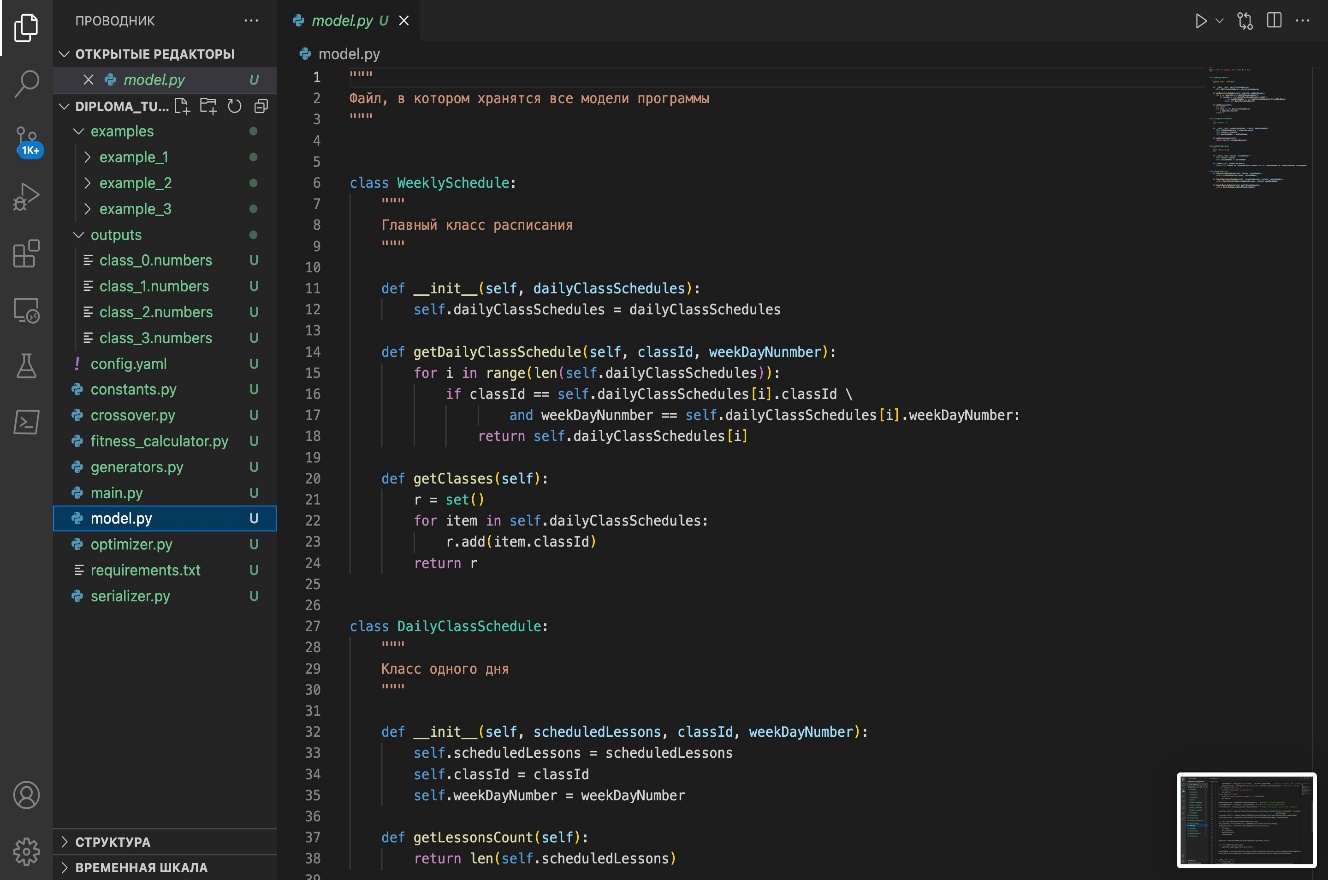


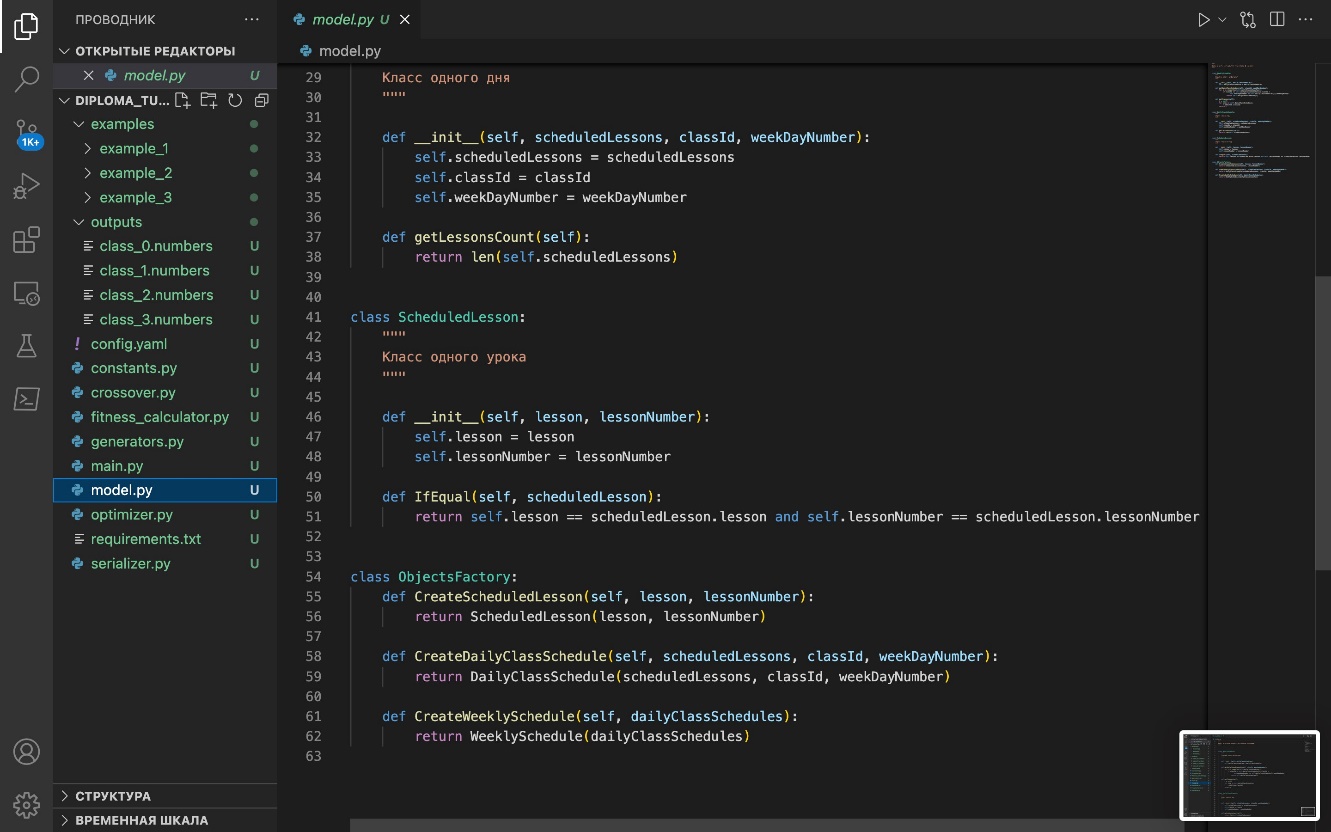
* model - файл со всеми моделями

1. WeeklySchedule - расписание на неделю.

2. DailyClassSchedule - расписание на один день для одного класса.

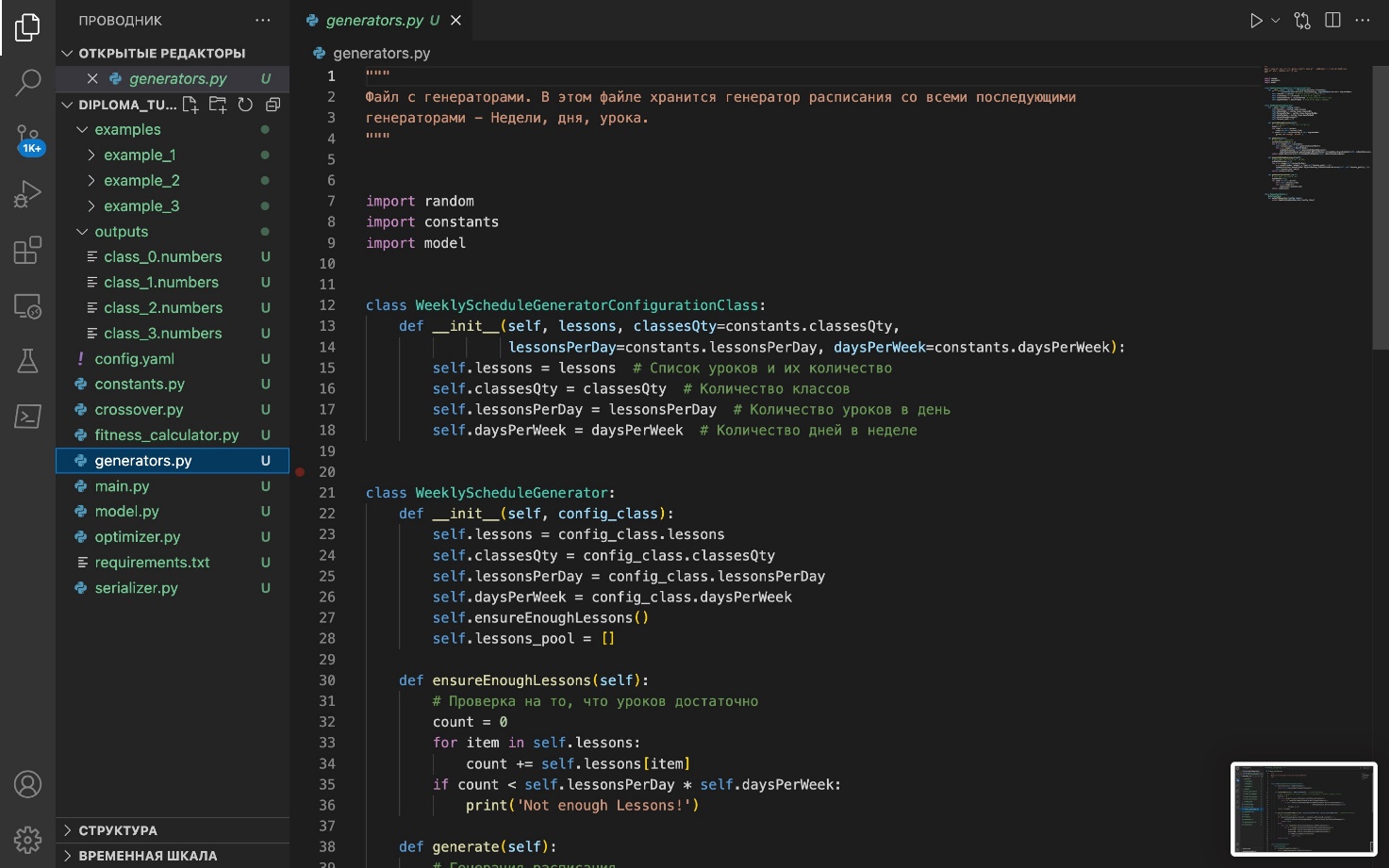
3. ScheduledLesson - запланированный урок.





* generators - файл, в котором хранятся все генераторы.

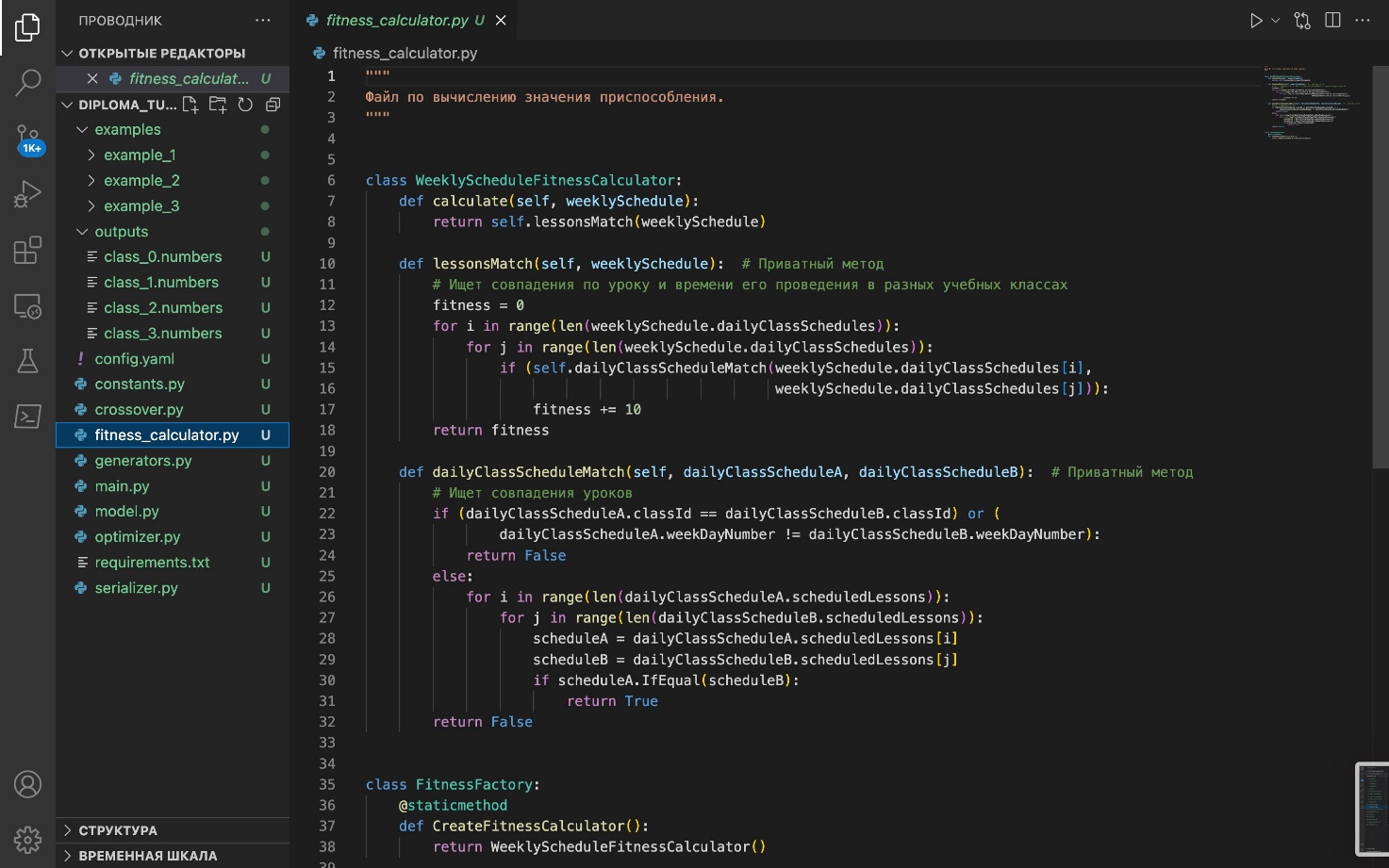
1. WeeklyScheduleGenerator - генератор расписания на неделю.





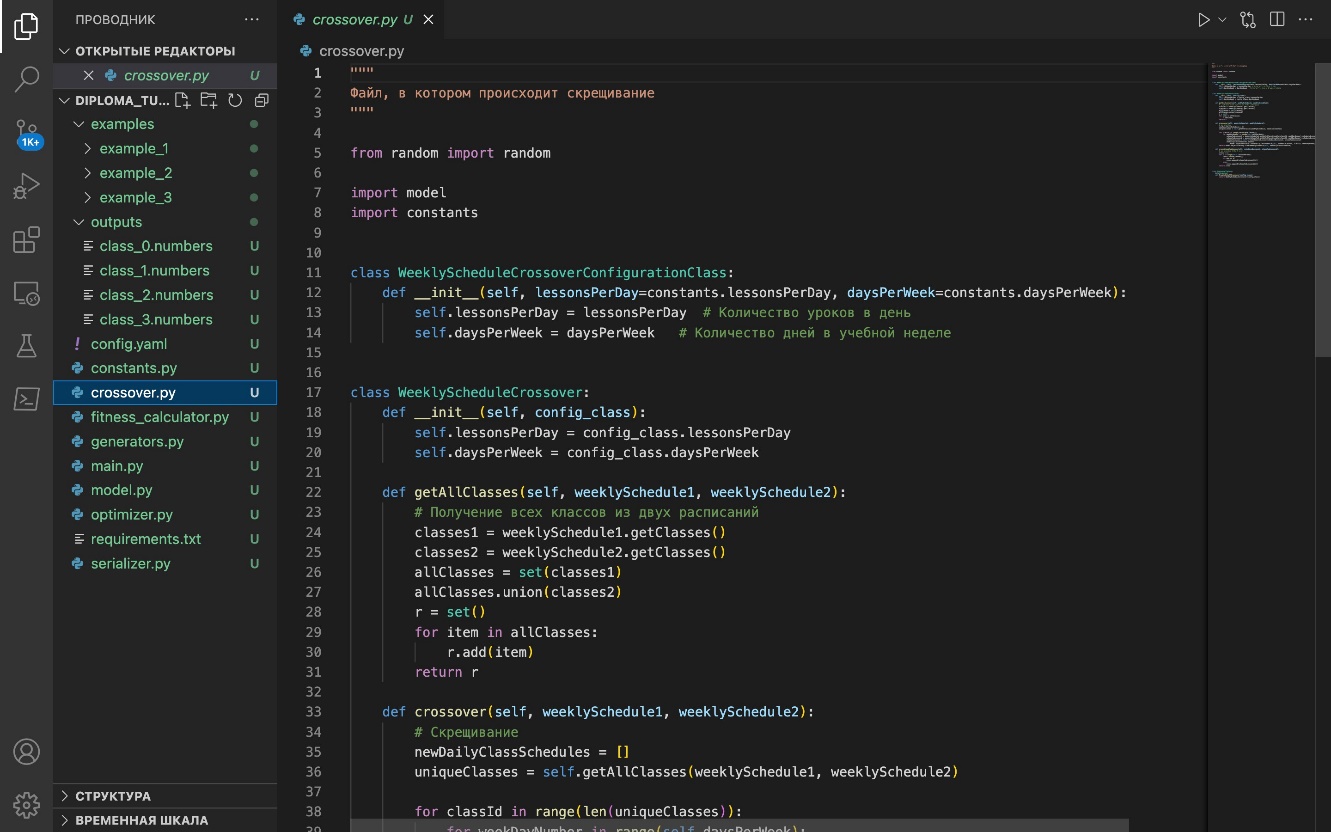
* fitness\_calculator - файл с классами по вычислению fitness'а.

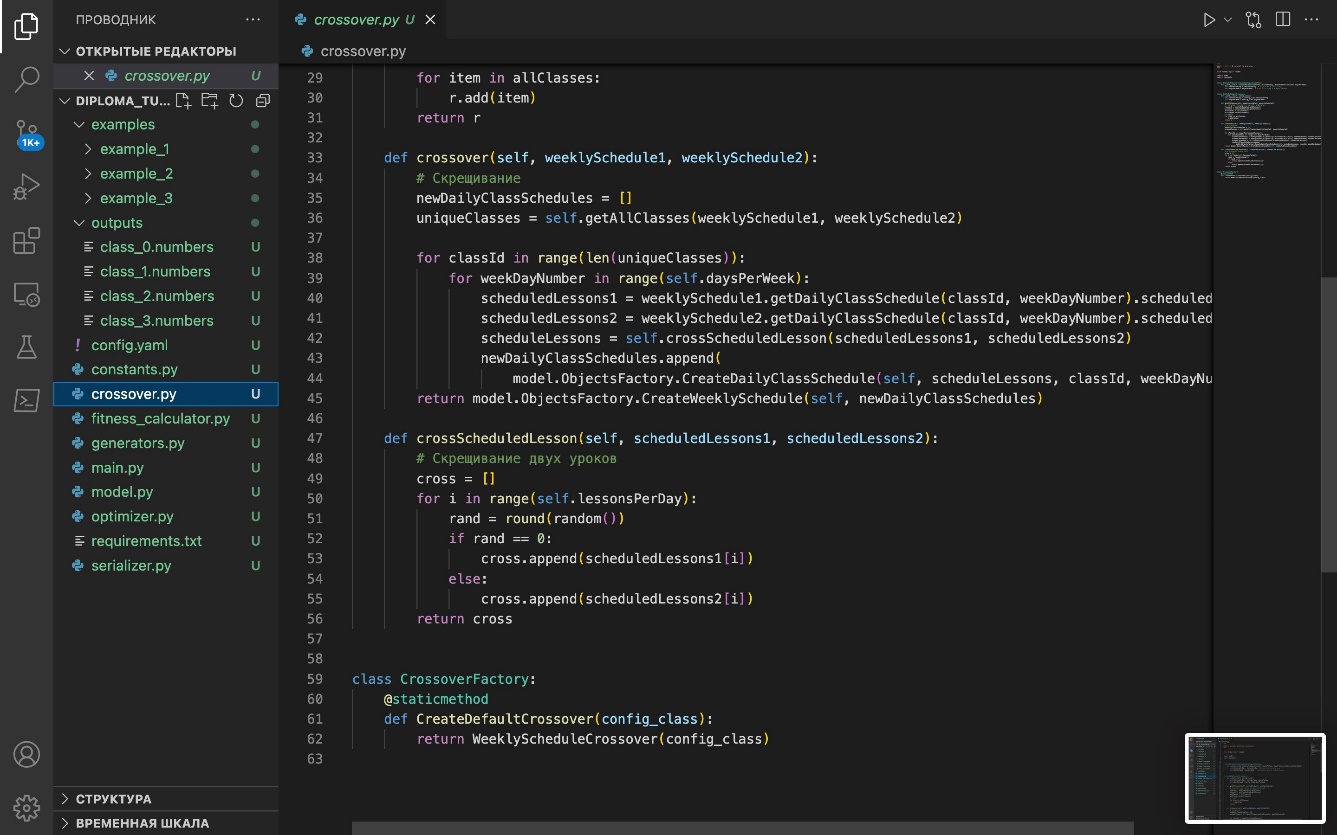
1. WeeklyScheduleFitnessCalculator - класс вычисления fitness'а для одного расписания.



* crossover - файл скрещивания расписанию

1. WeeklyScheduleCrossover - класс по скрещиванию одной популяции.

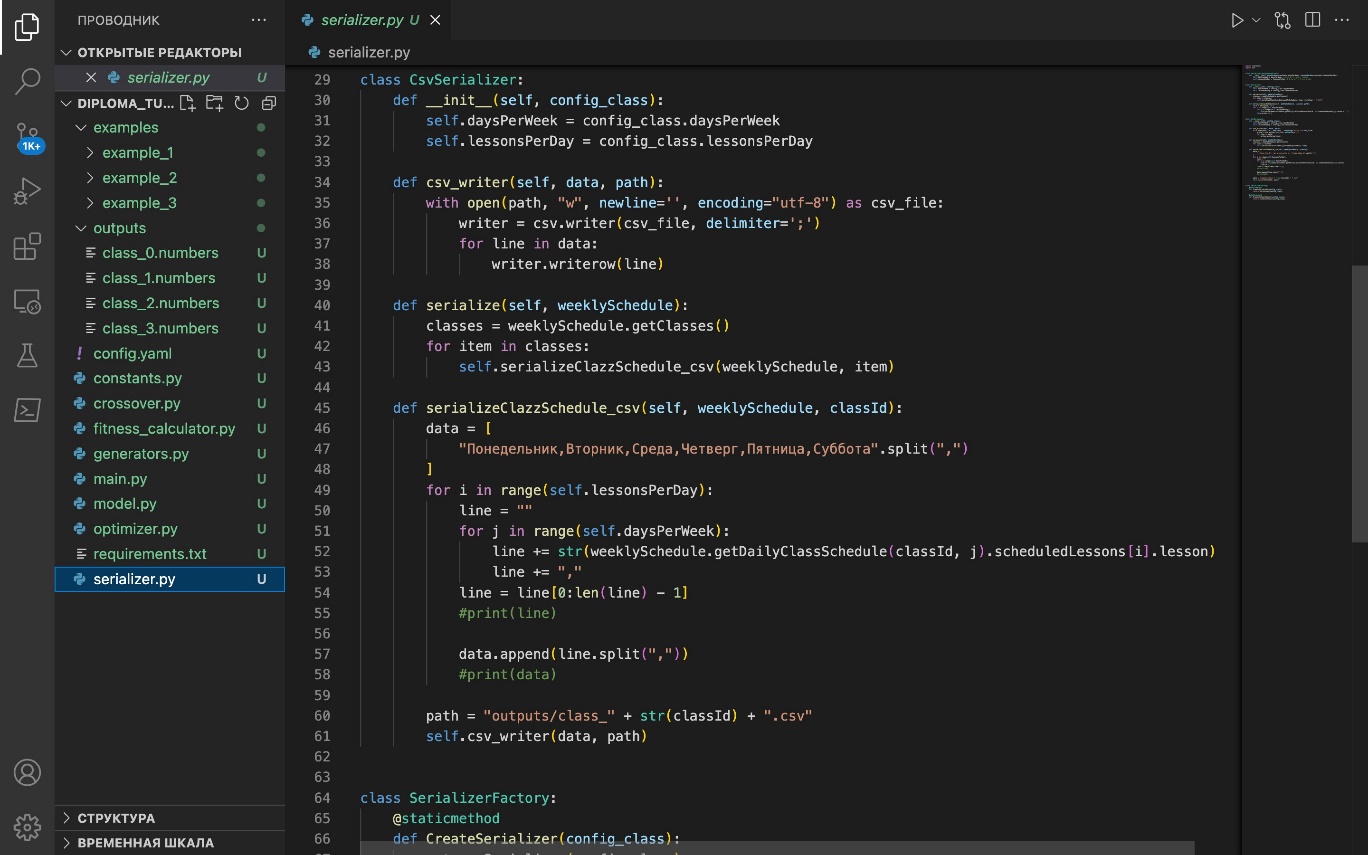


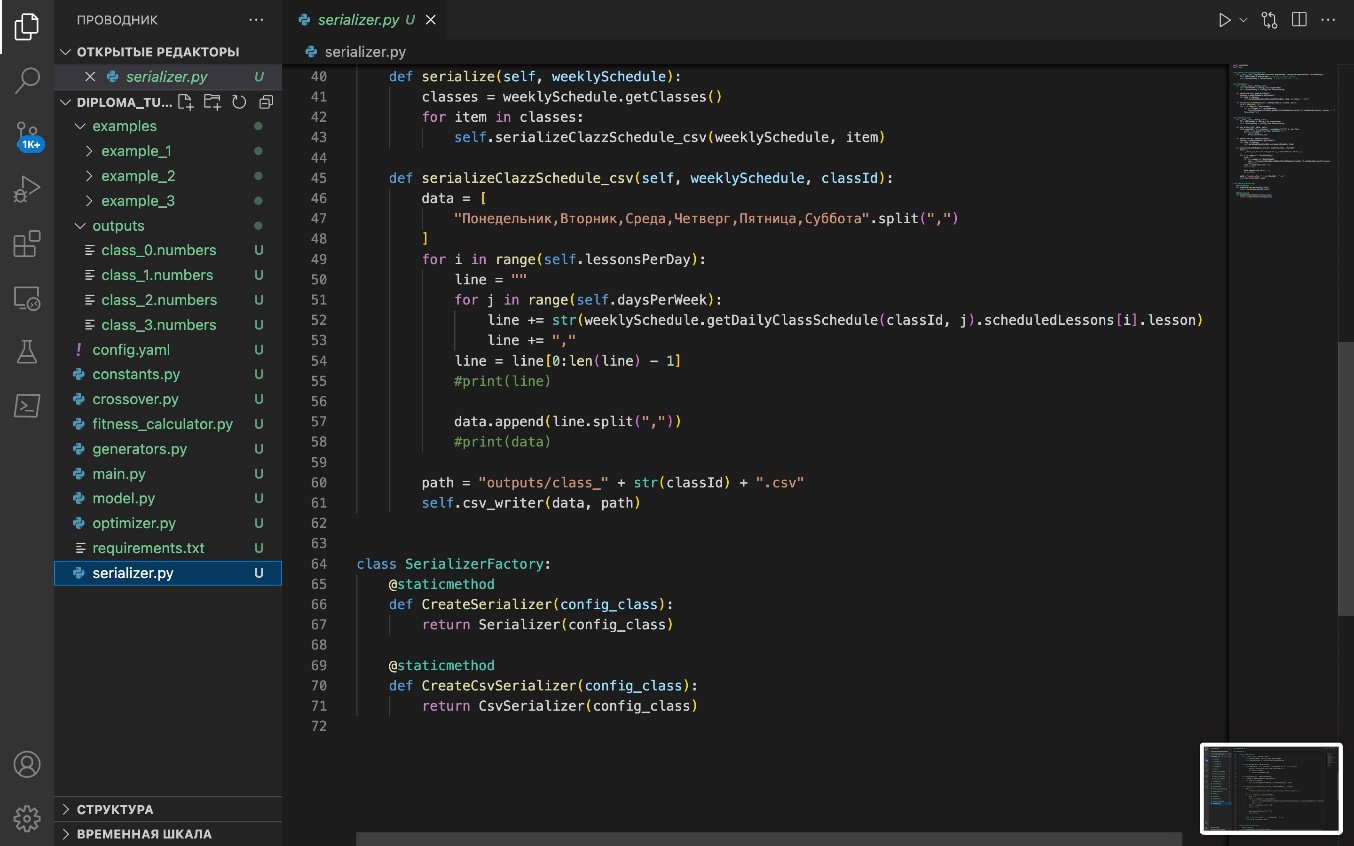


* serializer - файл по сериализации и выводу расписания.

1. Serializer - класс, который выводит расписание в текстовый файл.

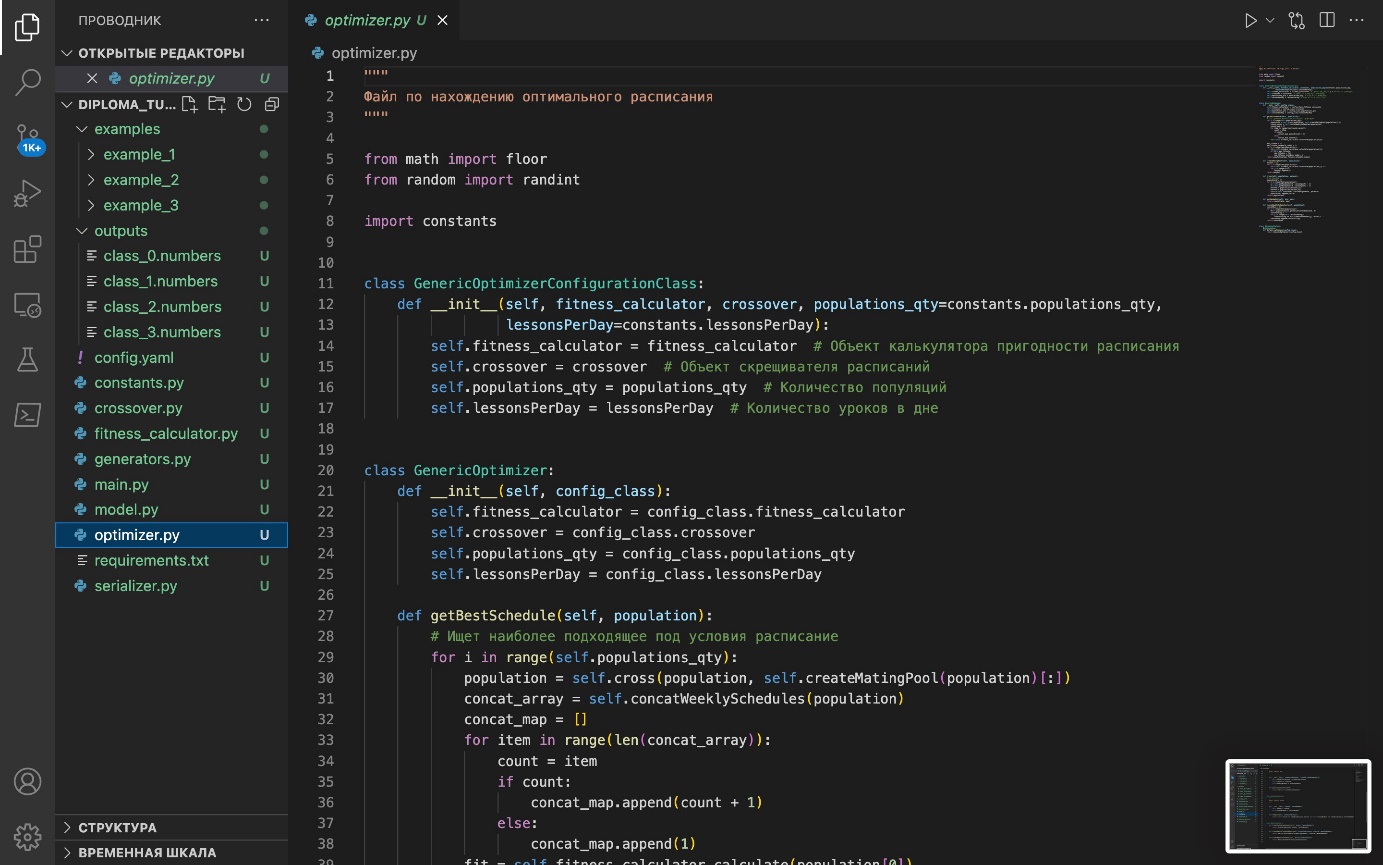
2. Csv\_serializer - класс, который выводит расписание в файл формата *.csv* (для просмотра с помощью электронных таблиц)

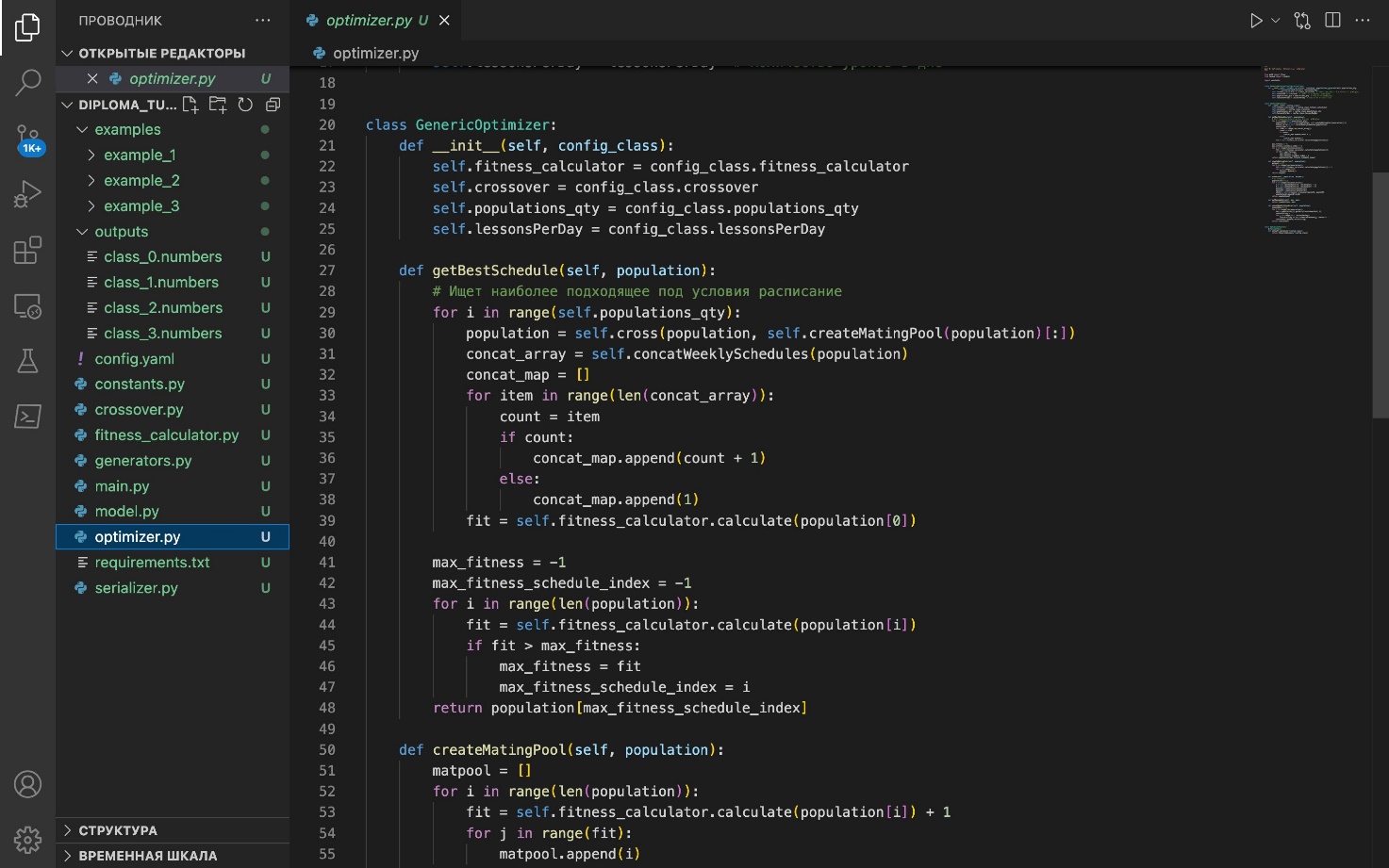


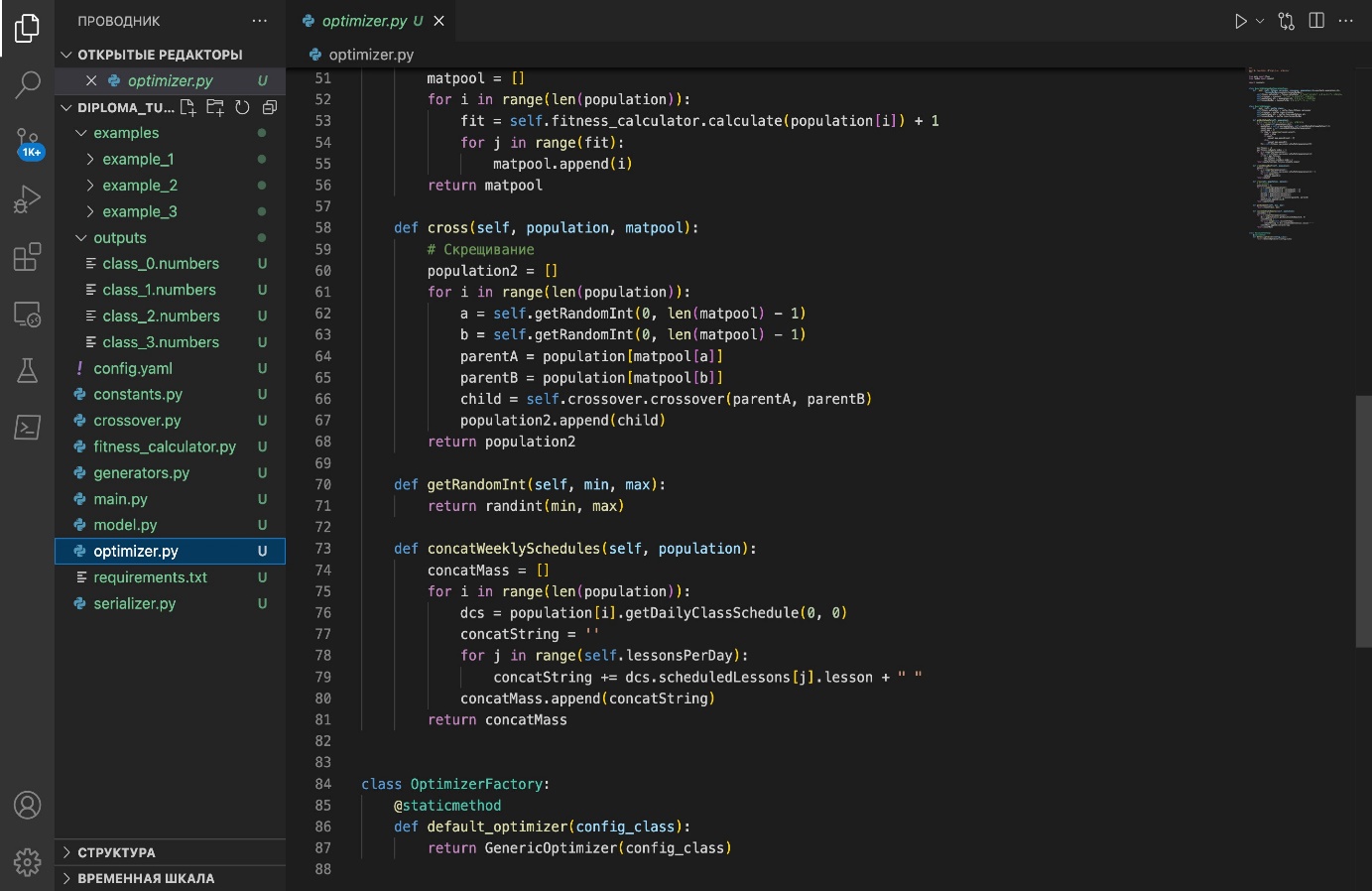


* optimizer - файл, в котором производится поиск оптимального расписания.

1. GenericOptimizer - главный класс программы. В нем производятся все нужные операции для нахождения наиболее схожее с идеальным расписание.



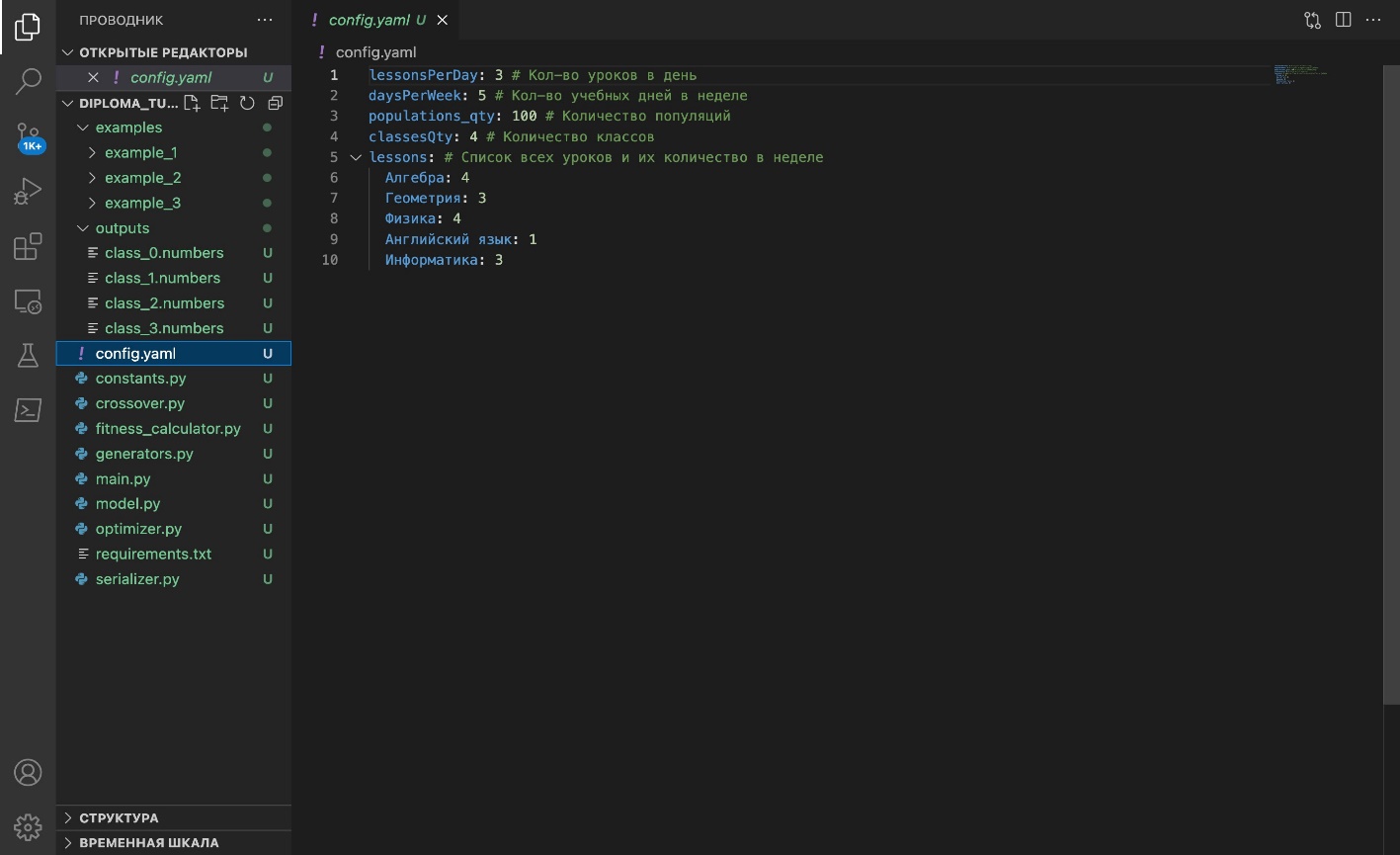


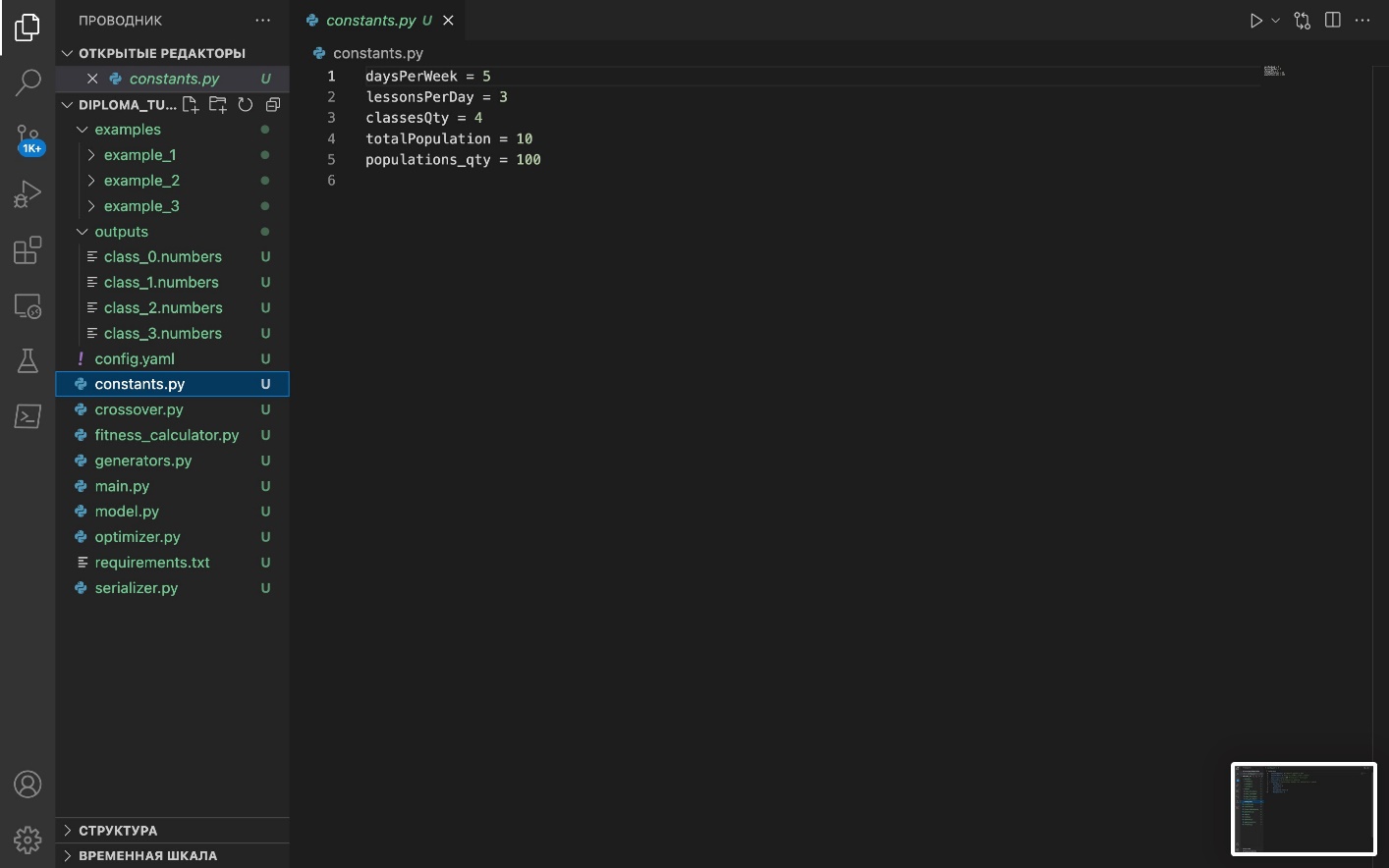


* Общие классы:

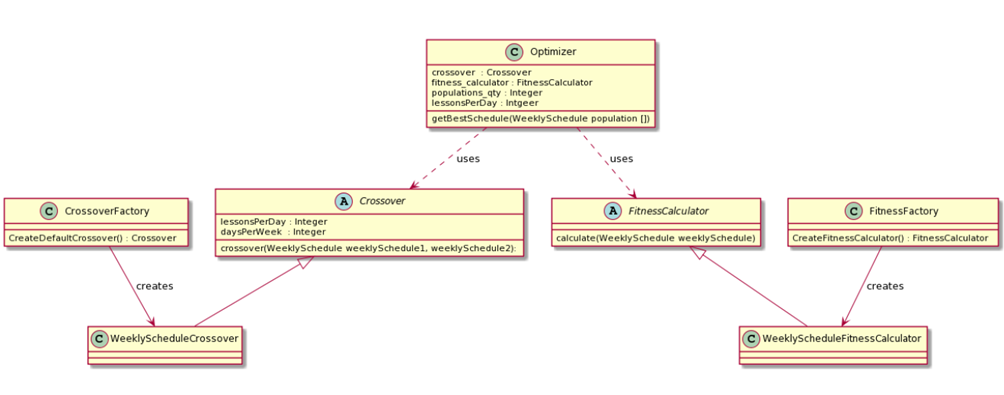
1. Factory - фабрика определенного класса.

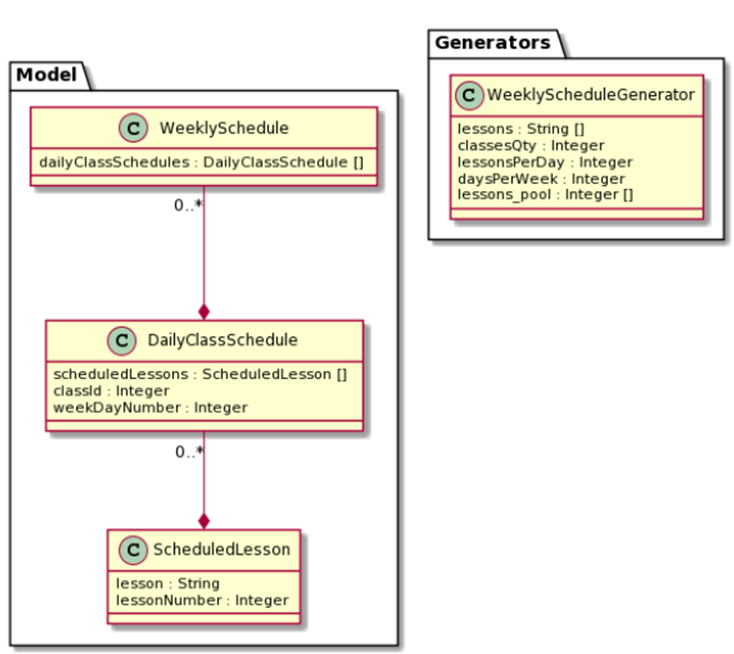
2. ConfigurationClass - класс, в который передаются параметры для успешной работы другого класса.





**Диаграмма программы**





**Инструкция по использованию и установке:**

*Note:* Для нормальной установки и работы программы требуется Python версии 3+ и установленный Pip.

1. Скачать архив с программой из GitHub

2. Открыть командную строку (*win+r*) и перейти в папку программы (с помощью команды *cd путь*)

3. Установить необходимые библиотеки с помощью команды *pip install -r requirements.txt*

4. Запустить файл main.py: *python main.py*

Чтобы использовать конфигурации примеров, добавьте параметр examples/example\_номер/ к команде запуска.

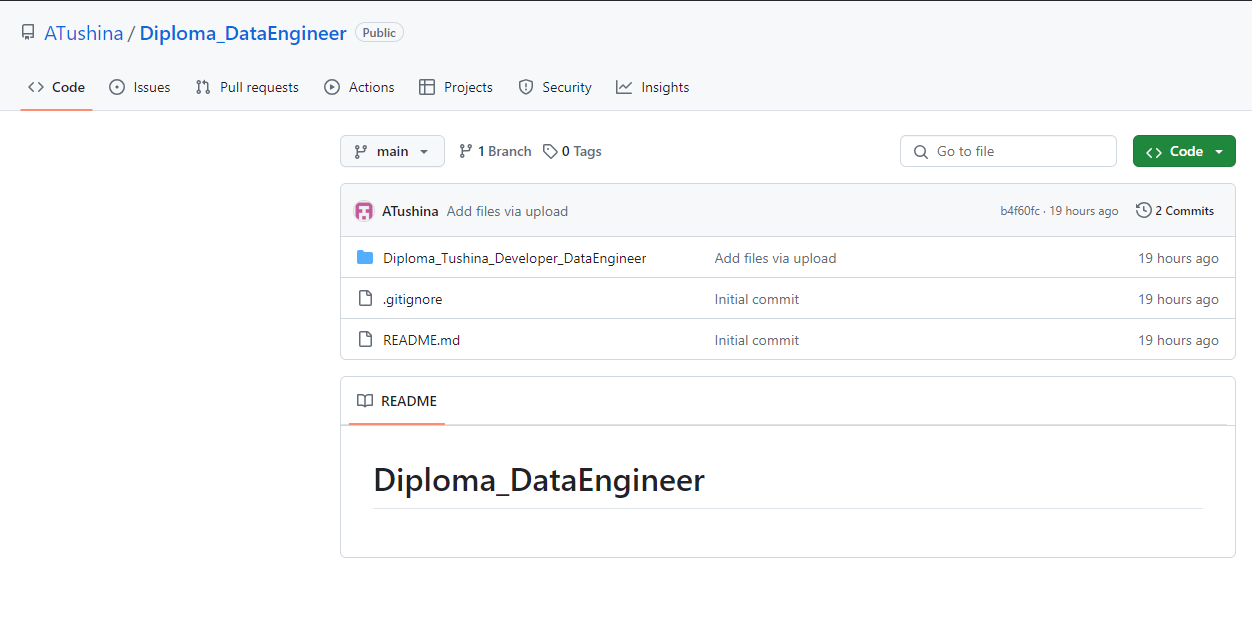
*Пример:* python main.py examples/example\_1/

Настройка параметров производится в конфигурационном файле config.yaml.

Примеры файлов находятся в папке *examples,* конфигурационный файл называется *config.yaml,* Выходные файлы находятся в папке *outputs* и представляют из себя файлы в формате .csv. Файлы такого формата легко открыть и в простом текстовом редакторе, и в редакторе электронных таблиц типа Excel.

**Ссылка на проект**

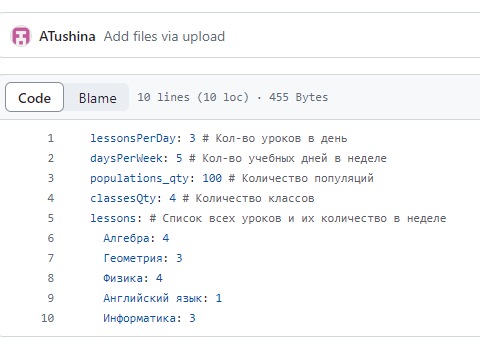
[**https://github.com/ATushina/Diploma\_DataEngineer.git**](https://github.com/ATushina/Diploma_DataEngineer.git)

****

**Примеры использования программы**

Пример 1.

Заданная конфигурация:

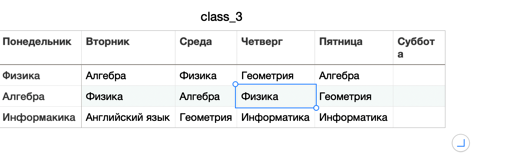


Результаты расписаний:



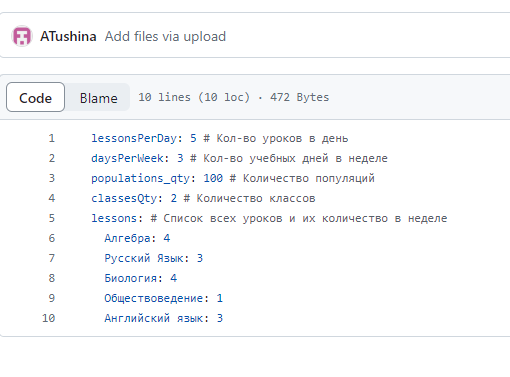




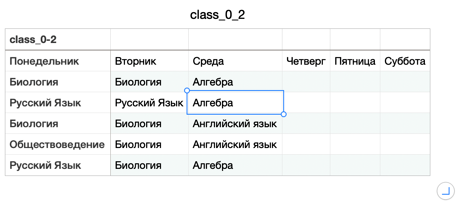


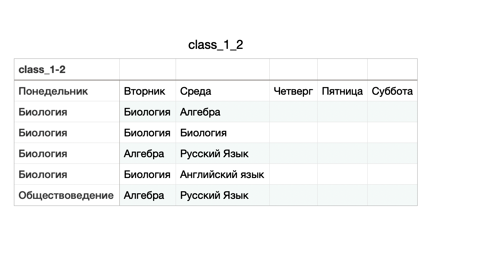
Пример 2.

Заданная конфигурация:



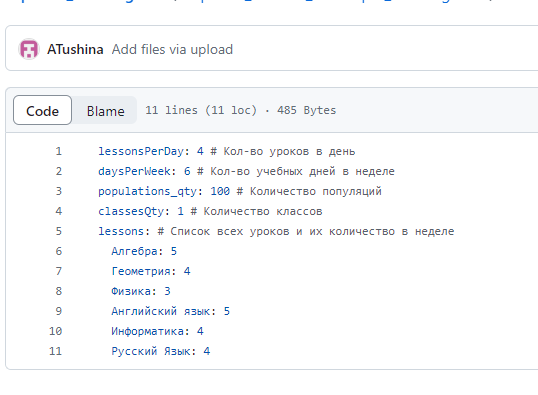
Результаты расписаний:



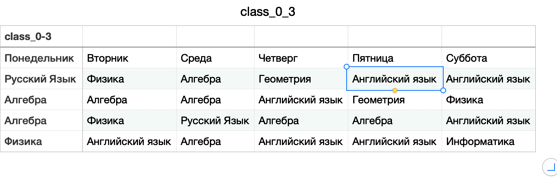


Пример 3.

Заданная конфигурация:



Результаты расписаний:



**Заключение**

Таким образом, во время работы над дипломным проектом удалось выполнить все задачи и достичь поставленной цели. Составлена рабочая программа, способная помочь в составлении расписания для школьников средней ступени обучения. При работе над программой были использованы навыки, полученные за период обучения на платформе GeekBrains. Разработан проект, который в перспективе поможет обеспечить сбор данных, хранение, обработку и анализ больших объёмов данных.

Однако продукт не идеален, и требует доработок: необходимо отладить алгоритм, сделать более удобный User Interface и предоставить пользователю больше информации о процессе работы программы.

По мнению составителя программы в дальнейшем поможет решить следующие проблемы:

1. Составление расписания на всех уровнях обучения: от дошкольных учреждений и учреждений дополнительного образования, младшей, средней, старшей школы, колледжей, гимназий, лицеев до высших учебных заведений (институты, университеты, академии), а также учреждений последипломного образования

2. Программу можно доработать и синхронизировать с учебными планами учреждений образования

3. Программу можно будет настроить с учетом выходных/праздничных дней, каникулярного времени

4. Доработанная программа поможет составить расписание контрольных, итоговых работ, экзаменационных сессий

5. При помощи доработанной программы можно будет учитывать почасовую нагрузку преподавателей и учителей

6. Программу можно развить до составления покабинетного (т.е расписание учебных классов, аудиторий)

7. Программа поможет анализировать количество ставок и физических лиц преподавателей и учителей, с учетом больничных листков, курсов повышения квалификации и отпусков (трудовых или без сохранения заработной платы)

8. Программу также можно доработать до системы учета оценки знаний учащихся: то есть не только составлять расписание, но и получать обратную связь по результатам обучения

9. Таким образом, выбрав определенный вектор развития проекта или же развив все возможные направления, можно добиться отлаженной системы, которая позволит облегчить весь учебно-методический процесс - от составления расписания до учета нагрузки на преподавателя ( с возможным выходом на расчет заработной платы), получить данные об успеваемости учащихся и огромный массив данных для анализа организации учебного процесса. Аналитика полученных данных в перспективе позволит оптимизировать учебный процесс и для учащихся, и для профессорско-преподавательского состава.

Полученные данные можно использовать следующим образом:

* Организовать ETL-процессы, то есть извлечь (extract), трансформировать (transform) и загружать (load) данные в базу данных.
* Очистить данные от дубликатов, ошибок, нерелевантных полей.
* Организовать пайплайн потоков данных.
* Настраивать мониторинг так, чтобы в случае поломки система оповестила о возникшей проблеме.
* Разрабатывать и поддерживать базы данных, чтобы нужные данные легко было найти и извлечь.
* Оптимизировать производительность баз данных, чтобы данные стабильно поступали в систему даже при высокой нагрузке.
* Разрабатывать и поддерживать приложения для работы с данными, такие как API и CLI.
* Составлять расписание, по которому система обработки данных в компании будет работать с информацией.

В результате, разработанная система позволит повысить скорость обработки информации, сократит сроки формирования отчетов и сэкономит время работы пользователя пользователей.

Автоматизация процесса администрирования расписания занятий и гибкость разработанной автоматизированной системы дают преимущества при её использовании в системе образования, улучшая при этом деятельность персонала, а вместе с тем и повышая качество предоставляемого учебным учреждением образования, а также степень удовлетворенности учебным процессом и учащихся, и преподавателей.

**Список использованной литературы**

1. Хемди А. Таха. Введение в исследование операций = Operations Research: An Introduction. — М.: Вильямс, 2007. — 912 с. — ISBN 0-13-032374-8.
2. Грешилов А. А. Математические методы принятия решений. — М.: МГТУ им. Н. Э. Баумана, 2006. — 584 с. — ISBN 5-7038-2893-7.
3. Билл Фрэнкс. Революция в аналитике: Как в эпоху Big Data улучшить ваш бизнес с помощью операционной аналитики = The Analytics Revolution: How to Improve Your Business By Making Analytics Operational In The Big Data Era. — М.: Альпина Паблишер, 2016. — ISBN ISBN 978‑5‑9614‑4132‑1
4. History and License - Python documentation
5. Why was Python created in the first place? — Python Software Foundation.
6. Classes The Python Tutorial — Python Software Foundation.
7. Прохоренок Н., Дронов В. Введение // Python 3. Самое необходимое, 2-е изд.. — БХВ-Петербург, 2019. — С. 11. — 608 с. — ISBN 9785977539944.
8. Yogesh Rana. Python: Simple though an Important Programming language (англ.) // International Research Journal of Engineering and Technology (IRJET). — 2019. — 2 February (vol. 06, iss. 2). — P. 1856—1858. — ISSN 2395-0056.
9. SkipMontanaro. Why is Python a dynamic language and also a strongly typed language - Python Wiki (англ.). wiki.python.org (24 февраля 2012). Дата обращения: 14 марта 2021.
10. Python Developer's Guide — Python Developer's Guide. devguide.python.org.
11. What’s New In Python 3.0 — Python 3.12.0 documentation.
12. Ars Technica report on Unladen Swallow goals (англ.). Дата обращения: 29 октября 2017. Архивировано 1 мая 2012 года.
13. MicroPython - Python for microcontrollers (англ.). micropython.org. Дата обращения: 9 октября 2023. Архивировано 6 июня 2014 года.
14. Яворски, Зиаде, 2021, p. 466.
15. Васильев Денис Алексеевич. Методические особенности изучения языка Python школьниками // Символ науки. — 2017. — № 1.
16. Mark Lutz. Learning Python: Powerful Object-Oriented Programming (англ.). — O'Reilly Media, Inc., 2009-10-06. — P. 7—8. — 1218 p. — ISBN 978-1-4493-7932-2.
17. Сергей Николенко. Генетические алгоритмы (слайды) — лекция № 4 из курса «Самообучающиеся системы»
18. Poli, R., Langdon, W. B., McPhee, N. F. A Field Guide to Genetic Programming — Lulu.com, freely available from the internet, 2008. — ISBN 978-1-4092-0073-4.
19. Банкрашков, А.В. Программирование для детей на языке Python / А.В. Банкрашков. - М.: АСТ, 2018. - 288 c.
20. Вордерман, К. Программирование на Python. Иллюстрированное руководство для детей / К. Вордерман, К. Стили, К. Квигли. - М.: Манн, Иванов и Фербер, 2017. - 346 c.
21. Лутц, М. Программирование на Python. Т. 1 / М. Лутц. - М.: Символ, 2016. - 992 c.
22. Лутц, М. Программирование на Python, II том / М. Лутц. - СПб.: Символ-плюс, 2015. - 992 c.
23. МакГрат, М. Программирование на Python для начинающих / М. МакГрат. - М.: Эксмо, 2015. - 192 c.
24. Мэтиз, Э. Изучаем PYTHON.Программирование игр, визуализация данных, веб-приложения / Э. Мэтиз. - СПб.: Питер, 2017. - 496 c.
25. Саммерфилд, М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. - СПб.: Символ-плюс, 2015. - 608 c.
26. Ротер М. Учитесь видеть бизнес–процессы: Построение карт потоков создания ценности / М. Ротер. – М.: Альпина Паблишер. – 136 c.
27. Тельнов Ю. Ф. Реинжиниринг бизнес–процессов. Компонентная методология [Текст] / Ю. Ф. Тельнов. Изд. 2–е, перераб. и доп. М.: Финансы и статистика. – 320 с.
28. Шерегов Н.А. Теоретические и прикладные аспекты моделирования деятельности компании// электронный научный журнал// Управление экономическими системами – С. 54–58.
29. Елиферов В.Г. Бизнес–процессы: Регламентация и управление: Учебник / В.Г. Елиферов. – М.: НИЦ ИНФРА–М. – 319 c.
30. Репин В.В. Бизнес–процессы. Моделирование, внедрение, управление / В.В. Репин. – М.: Манн, Иванов и Фербер. – 512 c.
31. Долганова О.И. Моделирование бизнес–процессов: Учебник и практикум для академического бакалавриата / О.И. Долганова, Е.В. Виноградова, А.М. Лобанова. – Люберцы: Юрайт. – 289 c.