Бюджетное профессиональное образовательное учреждение Вологодской области

«Череповецкий лесомеханический техникум им. В.П. Чкалова»

Специальность 09.02.07 «Информационные системы и программирование»

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПП по ПМ.02 Осуществление интеграции программных модулей

Выполнил студент 2 курса группы ИС-____

_____

подпись _____

место практики

_____

наименование юридического лица, ФИО ИП

Период прохождения: с «___» _____

2024 г.

по «___» _____ 2024 г.

Руководитель практики от

предприятия

должность_____

_____

подпись_____

МП

Руководитель практики

от техникума: Материкова

А.А.

_____

Оценка:_____

«___» _____2024 года

г. Череповец

2024

2024

Содержание

Введение

Задачи:

Задачами данной практики являются подготовка обучающихся осознанному и углубленному изучению дисциплин, привитие им практических умений и получение первичных профессиональных навыков по выбранной специальности.

Цели:

Целями производственной практики (по профилю специальности) являются:

закрепление и совершенствование общих и профессиональных компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам;

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности;

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность В профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях; ОК 04. Эффективно взаимодействовать и работать в коллективе и команде;

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста;

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих

ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения;

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях;

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности;

ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках.

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение

ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

1. Общая характеристика компании

Малленом Системс — это одна из ведущих российских организаций, сосредоточенная на создании и внедрении систем компьютерного зрения и промышленной видеоаналитики, использующих технологии машинного зрения и искусственного интеллекта, включая машинное обучение и глубокие нейронные сети.

Компания была основана в 2011 году группой ученых и программистов СанктПетербургского политехнического университета Петра Великого при поддержке инвестиционной компании «Малленом».

Основная деятельность компании включает реализацию высокотехнологичных ITпроектов в таких отраслях, как транспорт, машиностроение, нефтегазовая, металлургическая, пищевая и фармацевтическая промышленности, а также в алмазодобывающем и атомном секторах.

Продукция Малленом Системс доступна в различных регионах России, странах СНГ и ЕС. Уникальный опыт и ноу-хау компании позволяют быстро разрабатывать новые высокотехнологичные решения.

На протяжении десяти лет компания выступает официальным партнером и интегратором для Cognex — мирового лидера в области машинного зрения и промышленной идентификации.

С 2022 года Малленом Системс представляет на российском и евразийском рынках компанию Hikrobot — дочернюю организацию Hikvision, которая занимается производством оборудования для машинного зрения и мобильных роботов.

Кроме того, компания активно участвует в национальном рейтинге быстрорастущих технологических компаний России «ТехУспех», разработанном РВК.

1.1 Организационная структура компании

1. Высшее руководство

- Генеральный директор: Является центральной фигурой в управлении компанией, отвечая за стратегическое направление и общее

руководство. Генеральный директор формирует видение и миссию организации, а также принимает ключевые решения, касающиеся ее развития и устойчивости на рынке.

- Главный технический директор: Отвечает за технологическое развитие и внедрение инноваций. Он следит за современными трендами в технологии и обеспечивает их интеграцию в процессы компании для поддержания конкурентоспособности.

- Директор по развитию: Занимается стратегическим планированием и реализацией инициатив, направленных на рост бизнеса. Его задачи включают анализ новых рыночных возможностей, разработку стратегий для увеличения доли рынка и оптимизацию бизнес-процессов.

2. Отдел тестирования и контроля качества:

- Этот отдел отвечает за проверку качества программного и аппаратного обеспечения. Специалисты проводят тестирование на различных этапах разработки, выявляя и устраняя возможные дефекты для обеспечения высокого качества конечного продукта.

3. Инженерные службы

- Руководитель направления системной интеграции: Отвечает за управление проектами по интеграции различных IT-систем и технологий в единую инфраструктуру, координируя работу команд и обеспечивая выполнение проектов в срок.

- Отдел системной интеграции: Занимается интеграцией решений в инфраструктуру клиентов, обеспечивая их совместимость и функциональность.

- Отдел технической поддержки: Обеспечивает поддержку клиентов, решая технические проблемы и отвечая на запросы пользователей, что является важным аспектом для поддержания высокого уровня удовлетворенности клиентов.

4. Отдел разработки ПО

- Руководитель отдела разработки: Управляет процессами создания программного обеспечения, включая разработку алгоритмов.

1.2 Внутренний распорядок работы предприятия, охрана труда на предприятии (организации).

В компании "Малленом Системс" действует график работы 5/2, с 09:00 до 18:00. В штате компании есть отдельный специалист по охране труда, который проводит вводные инструктажи при приеме на работу и практике, а также занимается выдачей пропусков для пусконаладочных работ инженеров. В компании 20.09.2018 г. была проведена специальная оценка условий труда, согласно которой рабочие места, на территории которых установлены вредные производственные факторы, отсутствуют.

1.3 Должностные инструкции ИТ-специалистов предприятия

1. Общие положения

1.1. Настоящая должностная инструкция определяет должностные обязанности, права и ответственность Техника Общества с ограниченной ответственностью «Малленом Системс» (далее – Техник, Общество).

1.2. Техник относится к категории специалистов.

1.3. Техник принимается на работу и увольняется приказом генерального директора или уполномоченным им лицом.

1.4. На должность Техника назначается лицо, без предъявления требований к образованию и опыту работы.

1.5. Техник подчиняется непосредственно руководителю структурного подразделения, ведущему программисту и/или руководителю направления/проекта, в котором работает в настоящее время.

1.6. Техник должен знать:

– методы автоматической и автоматизированной проверки работоспособности программного обеспечения;

– основные виды диагностических данных и способы их представления; – языки, утилиты и среды программирования, и средства пакетного выполнения процедур;

– типовые метрики программного обеспечения;

– основные методы измерения и оценки характеристик программного обеспечения; – методы создания и документирования контрольных примеров и тестовых наборов данных;

– правила, алгоритмы и технологии создания тестовых наборов данных;

– требования к структуре и форматам хранения тестовых наборов данных;

– методы и средства проверки работоспособности программного обеспечения;

– среду проверки работоспособности и отладки программного обеспечения;

– внутренние нормативные документы, регламентирующие порядок документирования результатов проверки работоспособности программного обеспечения;

– методы и средства рефакторинга и оптимизации программного кода;

– языки программирования и среды разработки;

– внутренние нормативные документы, регламентирующие требования к программному коду, порядок отражения изменений в системе контроля версий; – внутренние нормативные документы, регламентирующие порядок отражения результатов рефакторинга и оптимизации в коллективной базе знаний;

– методы и приемы отладки программного кода;

– типовые ошибки, возникающие при разработке программного обеспечения, и методы их диагностики и исправления;

– локально-нормативные акты Общества, касающиеся выполнения его должностных обязанностей;

– требования охраны труда и правила пожарной безопасности.

1.4. Техник должен знать и уметь:

– писать программный код процедур проверки работоспособности программного обеспечения на выбранном языке программирования под руководством наставника;

– использовать выбранную среду программирования для разработки процедур проверки работоспособности программного обеспечения на выбранном языке программирования;

– применять методы и средства проверки работоспособности программного обеспечения;

– анализировать значения полученных характеристик программного обеспечения; – документировать результаты проверки работоспособности программного обеспечения;

– применять методы, средства для рефакторинга и оптимизации;

– публиковать результаты рефакторинга и оптимизации в коллективной базе знаний в виде лучших практик;

– использовать систему контроля версий для регистрации произведенных изменений;

– применять методы и приемы отладки дефектного программного кода;

– интерпретировать сообщения об ошибках, предупреждения, записи технологических журналов, возникающих при выполнении дефектного кода.

2. Должностные обязанности

Техник выполняет следующие должностные обязанности:

2.1 Выполняет работу по проведению необходимых технических расчетов;

2.2 Осуществляет наладку, настройку, регулировку и опытную проверку оборудования и систем, следит за его исправным состоянием;

2.3 Принимает участие в проведение экспериментов и испытаний;

2.4 Принимает участие в разработке программ, инструкций и другой технической документации, в изготовлении макетов, а также в испытаниях и экспериментальных работах;

2.5 Выполняет работу по сбору, обработке и накоплению исходных материалов, данных статистической отчетности, научно-технической информации;

2.6 Составляет описания проводимых работ, необходимые спецификации, диаграммы, таблицы, графики и другую техническую документацию;

2.7 Выполняет работу по оформлению плановой и отчетной документации, вносит необходимые изменения и исправления в техническую документацию в соответствии с решениями, принятыми при рассмотрении и обсуждении выполняемой работы; 2.8 Систематизирует, обрабатывает и подготавливает данные для составления отчетов о работе;

2.9 Принимает необходимые меры по использованию в работе современных технических средств.

3. Права

Техник имеет право:

3.1. Участвовать в обсуждении проектов решений, в совещаниях по их подготовке и выполнению.

3.2. Запрашивать у непосредственного руководителя разъяснения и уточнения по данным поручениям, выданным заданиям.

3.3. Запрашивать по поручению непосредственного руководителя и получать от других работников организации необходимую информацию, документы, необходимые для исполнения поручения.

3.4. Знакомиться с проектами решений руководства, касающихся выполняемой им функции, с документами, определяющими его права и обязанности по занимаемой должности, критерии оценки качества исполнения своих трудовых функций.

3.5. Вносить на рассмотрение своего непосредственного руководителя предложения по организации труда в рамках своих трудовых функций.

3.6. Участвовать в обсуждении вопросов, касающихся исполняемых должностных обязанностей.

4. Обязанности и ответственность Техник обязан:

4.1. Соблюдать локально-нормативные акты Общества.

4.2. Не разглашать информацию и сведения, являющиеся коммерческой тайной.

4.3. Использовать только принятые в Обществе программные инструменты и технологию разработки программного обеспечения.

4.4. Соблюдать трудовую и производственную дисциплину, правила и нормы охраны труда, требования производственной санитарии и гигиены, требования противопожарной безопасности.

Ведущий программист привлекается к ответственности:

4.5. За ненадлежащее исполнение или неисполнение своих должностных обязанностей, предусмотренных настоящей должностной инструкцией, в порядке, установленном действующим трудовым законодательством Российской Федерации;

4.6. За правонарушения и преступления, совершенные в процессе своей деятельности, в порядке, установленном действующим административным, уголовным и гражданским законодательством Российской Федерации;

4.7. За причинение ущерба организации в порядке, установленном действующим трудовым законодательством Российской Федерации.

3.Выполняемые задания.

Техническое задание на разработку модулей.

Заказчик:

«Малленом Системс»

Выполняющий:

Студент Группы ИС-23 Куликов Сергей Артёмович

Введение.

В современном мире обработка изображений является важной частью множества процессов, начиная от создания графического контента и заканчивая автоматизированными системами обработки данных. С учетом растущих потребностей пользователей в эффективных инструментах для работы с изображениями, данное техническое задание описывает разработку двух ключевых модулей. Первый модуль будет сосредоточен на обработке изображений, включая функции изменения их размера и поворота. Второй модуль будет отвечать за взаимодействие с пользователем и организацию данных, что обеспечит интуитивно понятный интерфейс и удобное хранение информации о выполненных операциях.

Основания для разработки.

В последние годы наблюдается значительный рост интереса к графическому дизайну и визуальному контенту. Это связано с увеличением числа пользователей социальных сетей, блогеров и малых предприятий, которые стремятся улучшить свои визуальные материалы. Однако многие существующие решения для обработки изображений требуют от пользователей специальных навыков или сложных манипуляций, что ограничивает их доступность. Данное ТЗ направлено на создание программного продукта, который будет простым в использовании, но при этом мощным и функциональным.

Назначение разработки.

Основная цель разработки заключается в создании программного изделия, которое позволит пользователям легко изменять размер и поворачивать изображения без необходимости использования сложных графических редакторов. Программный продукт должен быть доступен как для опытных пользователей, так и для новичков, что сделает его универсальным инструментом для всех категорий пользователей. Кроме того, система должна обеспечивать надежное хранение информации о выполненных операциях, что позволит пользователям отслеживать изменения и использовать ранее обработанные изображения.

Требования к программе или программному изделию.

а) Модуль обработки и работы с изображениями

1. Изменение размера изображения.

– Пользователь должен иметь возможность вводить новые значения ширины и высоты изображения в пикселях. При этом система должна обеспечивать пропорциональное изменение размера или же возможность выбора режима (с сохранением пропорций или без).

– Поддержка различных форматов изображений (JPEG, PNG, BMP) должна быть реализована с учетом особенностей каждого формата, таких как степень сжатия и качество.

– Измененное изображение должно сохраняться в том же формате или в формате, выбранном пользователем, с учетом возможных потерь качества.

2. Поворот изображения.

– Пользователь должен иметь возможность задавать угол поворота изображения в градусах. Программа должна поддерживать стандартные углы (0°, 90°, 180°, 270°) для быстрого доступа, а также возможность ввода произвольного угла.

– При выполнении операции поворота необходимо учитывать возможные изменения размеров изображения и сохранение его целостности.

б) Модуль взаимодействия с пользователем и формирования/хранения данных

- Интерфейс пользователя должен быть интуитивно понятным и доступным. Он должен включать в себя поля для ввода пути к исходному изображению, настройки параметров изменения размера и поворота, а также кнопки для выполнения операций.

- Все данные о выполненных операциях должны сохраняться в локальной базе данных или файле журнала, что позволит пользователю просматривать историю изменений и возвращаться к предыдущим версиям изображений при необходимости.

- Вывод результатов должен осуществляться через интерфейс с возможностью предварительного просмотра измененного изображения перед его окончательным сохранением.

Требования к программной документации.

- Документация должна включать подробное описание архитектуры системы, включая схемы взаимодействия между модулями.

- API модулей должен быть документирован с указанием всех доступных функций, параметров и примеров использования.
- Пользовательская инструкция должна содержать пошаговые руководства по работе с программным изделием, а также советы по устранению возможных проблем.

- Техническая документация должна включать комментарии в коде, описания алгоритмов обработки изображений и рекомендации по дальнейшему развитию системы.

Технико-экономические показатели.

Оценка затрат на разработку должна учитывать трудозатраты команды разработчиков, а также расходы на тестирование и внедрение продукта.

Ожидаемая экономическая эффективность будет определяться путем анализа потенциального рынка использования данного программного решения и его сравнением с затратами на его разработку.

Сроки окупаемости проекта могут быть рассчитаны на основе предполагаемого числа пользователей и стоимости лицензий или подписок на продукт.

Стадии и этапы разработки.

1. Анализ требований — на данном этапе команда разработчиков будет собирать и уточнять требования к функциональности модулей через взаимодействие с потенциальными пользователями и заинтересованными сторонами.

2. Проектирование — создание архитектуры системы с учетом всех требований, разработка пользовательского интерфейса и взаимодействия между модулями.

3. Реализация — этап кодирования, на котором разработчики будут создавать функционал модулей согласно утвержденной архитектуре.

4. Тестирование — проверка работоспособности модулей на соответствие функциональным требованиям, выявление и исправление ошибок.

5. Внедрение — интеграция модулей в единую систему, подготовка к эксплуатации и обучение пользователей.

6. Поддержка — обеспечение технической поддержки пользователей, регулярные обновления системы на основе обратной связи.

Порядок контроля и приемки.

- Контроль качества разработки будет осуществляться на каждом этапе через регулярные проверки соответствия функциональным требованиям.

- Приемка модулей будет проводиться по критериям: корректность работы (все функции должны работать без ошибок), производительность (время обработки изображений не должно превышать заданные параметры), удобство использования (интерфейс должен быть интуитивно понятным).

- Результаты тестирования будут документироваться для обеспечения прозрачности процесса приемки.

Срок выполнения

Срок производной практики

Заключение.

Данное техническое задание описывает разработку двух модулей для обработки изображений, включая функции изменения размера и поворота. Эти инструменты обеспечат удобство и эффективность для пользователей различного уровня.

Созданный продукт будет удовлетворять актуальным требованиям рынка, позволяя пользователям легко выполнять необходимые операции с изображениями. Успешная реализация проекта повысит качество визуального контента и расширит творческие возможности пользователей. В будущем возможны обновления и расширения функциональности на основе отзывов пользователей, что обеспечит актуальность системы.

1.PEP 8: Основные рекомендации по стилю кода Python

PEP 8 — это стиль кодирования для языка Python, который был разработан для улучшения читаемости кода и его совместимости между разными проектами. Следование этим рекомендациям помогает разработчикам легче понимать и поддерживать код, написанный другими.

1. Именование

- Переменные и функции: Используйте стиль snake_case. Например, def calculate_area():.

- Классы: Для имен классов применяйте стиль CamelCase. Например, class Circle:.

- Константы: Для констант используйте UPPER_CASE. Например, MAX_SIZE = 100.

Эти соглашения помогают сразу понять, что представляет собой тот или иной элемент кода.

2. Отступы

- Используйте 4 пробела для отступов. Это стандартный отступ в Python и помогает поддерживать единообразие в коде.

- Избегайте использования табуляции, так как это может привести к путанице при работе с разными текстовыми редакторами.

3. Максимальная длина строки

- Рекомендуется ограничивать длину строк до 79 символов. Это облегчает чтение кода на экранах с меньшим разрешением и при просмотре в редакторах с разделением на окна.

4. Пробелы

- Используйте пробелы вокруг операторов (например, a + b вместо a+b).

- Ставьте пробелы после запятых, но не ставьте их перед запятыми, точками с запятой и двоеточиями.

Эти правила делают код более читаемым и понятным.

5. Импортирование модулей

- Импорты должны располагаться в начале файла, и их следует группировать по типу:

- Сначала стандартные библиотеки.

- Затем сторонние библиотеки.

- В конце — ваши собственные модули.

Каждый импорт должен быть на отдельной строке. Это позволяет быстро увидеть зависимости вашего кода.

6. Докстринги

- Для документирования функций и классов используйте тройные кавычки.

2. Инспектирование модуля 1: (Модуль 1) Изменение размера изображения 2) Поворот изображения) по стандарту PEP 8:

– Импорт: Лучше всего размещать импорты в начале файла, а стандартные библиотеки обычно идут перед сторонними.

– Докстринги: Следует добавлять пустую строку перед и после каждого метода с документацией.

– Именование переменных: Все имена переменных и методов уже соответствуют стилю PEP 8.

– Обработка исключений: Убедитесь, что ошибка будет информативной и пользователь сможет понять, что именно произошло.

– Расстояния: Убедитесь, что между методами есть одна пустая строка.

2.1 Инспектирование модуля 2: (Модуль 1. Пользователь задает путь к изображению и новый размер изображения, модуль передает данные модулю работы с изображением и выдает результат.)

1. Импорт библиотек: Импортируем необходимые библиотеки PIL и argparse.

2. Класс ImageProcessor:

- Конструктор принимает путь к изображению и открывает его.

- Метод resize_image изменяет размер изображения. Если ширина или высота равны -1, сохраняется пропорция.

- Метод save_image сохраняет измененное изображение по указанному пути.

3. Функция main:

- Использует argparse для обработки аргументов командной строки.

- Создает экземпляр ImageProcessor, изменяет размер изображения и сохраняет его.

4. Запуск программы: Проверка на то, что скрипт запущен как основная программа.

Как использовать:

Сохраните этот код в файл, например, image_processor.py, и запустите его из командной строки.

3. Метод Интеграции Модуля API:

1. Интеграция с помощью API (Application Programming Interface)

- RESTful API: Этот подход использует HTTP-запросы для связи между клиентом и сервером и наиболее часто применяется в веб-приложениях.
- SOAP (Simple Object Access Protocol): Это протокол, предназначенный для обмена структурированными сообщениями на основе XML, который подходит для более сложных интеграционных задач.

- GraphQL: Это альтернатива REST, позволяющая запрашивать только необходимые данные, что делает взаимодействие более эффективным.

Рассмотрим на примере кода 1: (Модуль 1. Изменение размера изображения 2. Поворот изображения):

Приложение: 3

Интеграции модулей методом API:(1. 1. Пользователь задает путь к изображению и новый размер изображения, модуль передает данные модулю работы с изображением и выдает результат)
Приложение: 4

4.В процессе отладки были проверены два модуля:

1. Проверка корректности ввода:

- Пользователь вводит правильные пути к изображениям.

- Проверка наличия файлов перед их обработкой.

2. Обработка ошибок:

- Функции корректно обрабатывают исключения. Например, если файл изображения отсутствует, программа уведомляет пользователя об этом.

3. Тестирование функций:

- Для первого модуля (обработка и работа с изображениями) были протестированы как существующий, так и несуществующий путь.

- Для второго модуля (взаимодействие с пользователем и формирование/хранение данных) была проверена работа с двумя существующими изображениями и одним несуществующим.

В ходе отладки ошибок не обнаружено.

5)

аименовани е теста	естовые данные	жидаемый результат	актический результат	езультат тестировани я	омментарий
ест 1	ирина 800 ысота 600 оворот на 90 градусов	зменение размера изображени я и поворот изображени я	зображение unput сохранено и изменено на output зображение повёрнуто на 90 градусов	од работает без ошибок риём файлов с расширение м любым	нструмент очень удобен, но не универсален
ест 2	ользователь задает путь к изображени ю и новый размер изображени я, модуль передает данные модулю работы с изображени	одуль определяет путь к изображени ю и новый размер, затем идёт передача данных к модулю работы.	зображение найдено, новый размер указан, анные переданны	од работает без ошибок	нструментп олезен и практичен.

	ем и выдает результат.				

Создание Репозитория

Github:

a. Отчет (Report)

b. Задания (Src)

c. Документы (Docs)

Заключение.

Во время производной практики я узнал и научился новому, а именно: Разработке модулей и работы с Github.

В дальнейшем этот опыт мне поможет при работе в реальном времени.

Источники

Сайт Компании по практике https://www.mallenom.ru

Работы с Pillow https://python-scripts.com/pillow

ГОСТ 7.80-2000 «Библиографическая запись. Заголовок. Общие требования и правила составления» [Электронный ресурс]/ Электронный правовой и нормативно-технической документации- режим доступа: http://docs.cntd.ru/document/gost-7-80-2000

ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе. Структура

и правила оформления» [Электронный ресурс]/ Электронный правовой и нормативно-технической документации- режим доступа: http://docs.cntd.ru/document/gost-7-32-2001-sibid

ГОСТ 7.1-2003 № 332-ст «Система стандартов по информации,

библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления» [Электронный ресурс]/ Электронный правовой и нормативно-технической документации- режим доступа: http://docs.cntd.ru/document/gost-7-1-2003-sibid

ГОСТ Р 7.0.5-2008 «Система стандартов по информации,

библиотечному и издательскому делу. Библиографическая ссылка. Общие требования и правила составления» [Электронный ресурс]/ Библиотека ГОСТов стандартов и нормативов- режим доступа: http://www.standartov.ru/norma_doc/53/53649/index.htm

ГОСТ Р 7.0.11-2011 «Система стандартов по информации,

библиотечному и издательскому делу. Диссертация и автореферат диссертации. Структура и правила оформления» [Электронный ресурс]/ Электронный правовой и нормативно-технической документации- режим доступа: http://docs.cntd.ru/document/gost-r-7-0-11-2011

Модули в Python https://docs.python.org/3/tutorial/modules.html

Модуль os https://docs.python.org/3/library/os.htm

Приложения

1)Модуль обработки и работы с изображениям

```python
from PIL import Image

class ImageProcessor:
    def __init__(self, image_path):
        try:
            self.image = Image.open(image_path)
        except FileNotFoundError:
            raise FileNotFoundError(f"Image not found at path: {image_path}")
        except Exception as e:
            raise Exception(f"Error opening image: {e}")


    def resize_image(self, width, height):
        """Изменяет размер изображения. Сохраняет пропорции, если один из параметров равен -1."""
        if width == -1:
            width = int(self.image.width * (height / self.image.height))
        elif height == -1:
            height = int(self.image.height * (width / self.image.width))
        self.image = self.image.resize((width, height))


    def rotate_image(self, degrees):
        """Поворачивает изображение на заданный угол."""
        self.image = self.image.rotate(degrees)


    def save_image(self, output_path):
        """Сохраняет обработанное изображение."""
        try:
            self.image.save(output_path)
            print(f"Image saved to {output_path}")
        except Exception as e:
            raise Exception(f"Error saving image: {e}")


# Пример использования:

try:
    processor = ImageProcessor("input.jpg") # Замените "input.jpg" на путь к вашему изображению

    processor.resize_image(400, -1) # Изменяем ширину на 400 пикселей, высота масштабируется пропорционально
    processor.rotate_image(90) # Поворачиваем на 90 градусов

    processor.save_image("output.jpg") # Сохраняем обработанное изображение

except FileNotFoundError as e:
    print(f"Ошибка: {e}")
except Exception as e:
    print(f"Ошибка: {e}")
```

2)Модуль взаимодействия с пользователем и формирование и хранений данных

```python
from PIL import Image
import argparse

class ImageProcessor:
    # ... (код класса ImageProcessor из предыдущего примера) ...


def main():
    parser = argparse.ArgumentParser(description="Resize and rotate images.")
    parser.add_argument("input_path", help="Path to the input image")
    parser.add_argument("width", type=int, help="New width of the image (-1 to keep aspect ratio)")
    parser.add_argument("height", type=int, help="New height of the image (-1 to keep aspect ratio)")
    parser.add_argument("output_path", help="Path to save the output image")
    args = parser.parse_args()

    try:
        processor = ImageProcessor(
        processor.resize_image(arg
        processor.save_image(args.output_path)

    except FileNotFoundError as e:
        print(f"Error: {e}")
    except Exception as e:
        print(f"An error occurred: {e}")


if __name__ == "__main__":
    main()
```

(function) output_path: Any

3)

```python
from flask import Flask, request, jsonify
from PIL import Image
import os

app = Flask(__name__)

class ImageProcessor:
    def __init__(self, image_path):
        try:
            self.image = Image.open(image_path)
        except FileNotFoundError:
            raise FileNotFoundError(f"Image not found at path: {image_path}")
        except Exception as e:
            raise Exception(f"Error opening image: {e}")

    def resize_image(self, width, height):
        """Изменяет размер изображения. Сохраняет пропорции, если один из параметров равен -1."""
        if width == -1:
            width = int(self.image.width * (height / self.image.height))
        elif height == -1:
            height = int(self.image.height * (width / self.image.width))
        self.image = self.image.resize((width, height))

    def rotate_image(self, degrees):
        """Поворачивает изображение на заданный угол."""
        self.image = self.image.rotate(degrees)

    def save_image(self, output_path):
        """Сохраняет обработанное изображение."""
        try:
            self.image.save(output_path)
            return output_path
        except Exception as e:
            raise Exception(f"Error saving image: {e}")

@app.route('/resize', methods=['POST'])
def resize():
    data = request.json
    image_path = data.get('image_path')
    width = data.get('width', -1)
    height = data.get('height', -1)
    output_path = data.get('output_path', 'output_resized.jpg')

    try:
        processor = ImageProcessor(image_path)
        processor.resize_image(width, height)
        saved_path = processor.save_image(output_path)
        return jsonify({"message": "Image resized successfully", "output_path": saved_path}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 400

@app.route('/rotate', methods=['POST'])
def rotate():
    data = request.json
    image_path = data.get('image_path')
    degrees = data.get('degrees', 0)
    output_path = data.get('output_path', 'output_rotated.jpg')

    try:
        processor = ImageProcessor(image_path)
        processor.rotate_image(degrees)
        saved_path = processor.save_image(output_path)
        return jsonify({"message": "Image rotated successfully", "output_path": saved_path}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 400

if __name__ == '__main__':
    app.run(debug=True)
```

4)

```python
from flask import Flask, request, jsonify
from PIL import Image
import os

app = Flask(__name__)

class ImageProcessor:
    def __init__(self, image_path):
        try:
            self.image = Image.open(image_path)
        except FileNotFoundError:
            raise FileNotFoundError(f"Image not found at path: {image_path}")
        except Exception as e:
            raise Exception(f"Error opening image: {e}")

    def resize_image(self, width, height):
        """Изменяет размер изображения. Сохраняет пропорции, если один из параметров равен -1."""
        if width == -1:
            width = int(self.image.width * (height / self.image.height))
        elif height == -1:
            height = int(self.image.height * (width / self.image.width))
        self.image = self.image.resize((width, height))

    def rotate_image(self, degrees):
        """Поворачивает изображение на заданный угол."""
        self.image = self.image.rotate(degrees)

    def save_image(self, output_path):
        """Сохраняет обработанное изображение."""
        try:
            self.image.save(output_path)
            return output_path
        except Exception as e:
            raise Exception(f"Error saving image: {e}")

@app.route('/resize', methods=['POST'])
def resize():
    data = request.json
    image_path = data.get('image_path')
    width = data.get('width', -1)
    height = data.get('height', -1)
    output_path = data.get('output_path', 'output_resized.jpg')

    try:
        processor = ImageProcessor(image_path)
        processor.resize_image(width, height)
        saved_path = processor.save_image(output_path)
        return jsonify({"message": "Image resized successfully", "output_path": saved_path}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 400

@app.route('/rotate', methods=['POST'])
def rotate():
    data = request.json
    image_path = data.get('image_path')
    degrees = data.get('degrees', 0)
    output_path = data.get('output_path', 'output_rotated.jpg')

    try:
        processor = ImageProcessor(image_path)
        processor.rotate_image(degrees)
        saved_path = processor.save_image(output_path)
        return jsonify({"message": "Image rotated successfully", "output_path": saved_path}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 400

if __name__ == '__main__':
    app.run(debug=True)
```