**Курс: Web-разработка**

**Дисциплина: Создание web-приложений с использованием фреймворка Django**

**Занятие №58-60: Первый проект: Банк рецептов**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**Примеры и идеи проектов**

* [**Выбор проектной платформы**](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#choose)
  + [Веб](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#web)
  + [Десктоп](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#desktop)
  + [Командная строка](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#command-line)
* [**Идеи веб-приложений**](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#web-ideas)
  + [Агрегатор контента](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#aggregator)
  + [Regex Query Tool](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#regex)
  + [URL Shortener](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#shortener)
  + [Post-It Note](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#note)
  + [Приложение-викторина](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#quiz)
* [**Идеи десктопных приложений**](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#desktop-ideas)
  + [MP3-плеер](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#mp3)
  + [Alarm Tool](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#alarm)
  + [Файловый менеджер](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#file-manager)
  + [Expense Tracker](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#expense)
* [**Идеи приложений для командной строки**](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#command-line-ideas)
  + [Контактная книга](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#contact)
  + [Site Connectivity Checker](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#connectivity)
  + [Bulk File Rename Tool](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#rename)
  + [Генератор древа каталогов](https://tproger.ru/translations/13-python-projects-for-intermediate-developers/#generator)

**Выбор проектной платформы**

Вам нужно создать такое ПО, которое бы облегчало людям, не имеющим определённых технических знаний, работу на платформе. Веб, десктоп и командная строка — три основные платформы, для которых вы можете создать свои проекты.

**Веб**

Веб-приложения — это приложения, которые работают в интернете, к ним можно получить доступ на любом устройстве без загрузки приложения на само устройство, если есть доступ к интернету. Веб-приложение состоит из бэкенда (серверной части) и фронтенда (пользовательского интерфейса). Для Python-разработчиков основное поле деятельности — бэкенд. Однако визуальная составляющая тоже важна, поэтому вам понадобятся некоторые знания HTML, CSS и, возможно, JavaScript. Чтобы создать простой интерфейс, основы будет достаточно.

Другой вариант — использовать Python как для фронтенда, так и для бэкенда с помощью библиотеки [anvil](https://anvil.works/learn). Вы можете создавать веб-приложения с помощью Python через такие веб-фреймворки как [Django](https://www.djangoproject.com/) и [Flask](https://palletsprojects.com/p/flask/). Список фреймворков для создания веб-приложений на Python очень длинный, но Django и Flask остаются самыми популярными из них.

**Десктоп**

Существуют специальные библиотеки для создания десктопных приложений. Одна из них — [PySimpleGUI](https://pysimplegui.readthedocs.io/tutorial/" \t "_blank), она хорошо подходит для разработчика среднего уровня. [PyQt5](https://pypi.org/project/PyQt5/) более мощная, но может оказаться слишком сложной в обучении.

Программное обеспечение, которое вы разрабатываете, может работать на любой операционной системе, будь то Windows, Linux или Mac. Всё, что нужно сделать после создания проекта, — это скомпилировать его в исполняемый файл для выбранной операционной системы.

**Командная строка**

Приложения командной строки — это те приложения, которые работают в окне консоли. В Windows в роли консоли выступает сама командная строка, а в Linux и Mac это терминал.

При работе с привычными вам приложениями вы щёлкаете по разным иконкам, набираете текст и т. п., но при работе с командой строкой вы вводите специальные команды, так что вам нужно владеть некоторыми техническими знаниями.

Приложения для командной строки могут быть не такими красивыми или простыми в использовании, как десктопные или веб-приложения, но это не делает их менее мощными. Можно улучшать внешний вид таких приложений, применяя цвета к тексту. Есть библиотеки, которые помогут вам с этим: [Colorama](https://pypi.org/project/colorama/" \t "_blank) и [Colored](https://pypi.org/project/colored/" \t "_blank). Фреймворки [Docopt, Argarse и Click](https://realpython.com/comparing-python-command-line-parsing-libraries-argparse-docopt-click/" \t "_blank) упростят вам процесс создания приложений.

**Идеи веб-приложений**

**Агрегатор контента**

Контент — самая важная часть всех ресурсов. Он существует в сети везде, от блогов до социальных сетей. Чтобы оставаться в курсе интересующей вас темы, часто приходится мониторить множество ресурсов. Один из способов облегчить себе и другим эту задачу — сделать агрегатор контента. Агрегатор извлекает информацию из разных сайтов и собирает её в одном месте. Люди могут видеть сообщения и новости, которые их интересуют, особо не копаясь в интернете.

Примеры контент-агрегаторов:

* [Alltop](https://alltop.com/)
* [Hvper](https://www.hvper.com/)

**Технические детали**

Во-первых, нужно знать, с каких сайтов вы хотите получать контент. Затем вы можете использовать различные библиотеки, например, [Requests](https://realpython.com/python-requests/) для отправки HTTP-запросов, а также [BeautifulSoup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/" \t "_blank) для парсинга и сбора нужной информации.

Хорошо реализовать агрегацию контента в качестве фонового процесса. Библиотеки, вроде [Celery](http://www.celeryproject.org/" \t "_blank) или [APScheduler](https://apscheduler.readthedocs.io/en/latest/" \t "_blank) могут помочь с этим. В частности, APScheduler отлично подходит для небольших фоновых процессов.

После парсинга содержимого с разных сайтов вам нужно его где-то сохранить. Следовательно, вам надо будет использовать базу данных.

**Дополнительно**

Можно добавить ещё одну фишку: пользователь может подписаться на определённые сайты из тех, с которых вы собираете информацию, и в конце дня агрегатор отправит статьи за этот день на адрес электронной почты пользователя.

**Regex Query Tool**

Иногда нужно найти в тексте определённую информацию, которую непросто найти с помощью обычного инструмента поиска в текстовых редакторах. Поэтому можно сделать другой инструмент, основанный на регулярных выражениях — Regex Query Tool.

С помощью регулярных выражений можно задавать более общие условия, например, можно искать любые даты, записанные в определённом виде. Когда регулярное выражение сопоставляет шаблоны c текстом, оно подсвечивает совпавшие части, таким образом сообщая о них пользователю. Так, ваше приложение проверит наличие в тексте строк, переданных пользователем. С помощью Regex Query Tool пользователи смогут быстро найти то, что им нужно, намного быстрее, чем в текстовом редакторе.

Примеры Regex Query Tools:

* [FreeFormatter](https://www.freeformatter.com/regex-tester.html)
* [RegexTester](https://www.regextester.com/)

**Технические детали**

Вам нужно сделать так, чтобы приложение давало положительный или отрицательный ответ, такой как «строка найдена» и «строка не найдена», обозначая положительный ответ зелёным цветом, а отрицательный — красным.

Вам не нужно реализовывать алгоритм с нуля. Вы можете использовать re — стандартную библиотеку Python. Библиотека будет возвращать совпавшие строки или None, когда строка запроса не совпадает ни с чем.

**Дополнительно**

Вы также можете добавить ещё одну функцию — приложение сначала проверит наличие шаблонов, переданных самим пользователем, а после предложит альтернативные варианты запросов, похожие по смыслу, чтобы помочь пользователю, если он неправильно сформулировал запрос.

**URL Shortener**

Когда люди делятся ссылками или размещают URL-адреса в постах в социальных сетях, длинный адрес может оказаться очень неудобным и некрасивым. Облегчить задачу поможет URL Shortener. Он уменьшает количество символов в URL-адресе, облегчая их чтение и запоминание. К примеру, xyz.com/wwryb78&svnhkn%sghq?sfiyh может быть сокращено до xyz.com/piojwr.

Примеры таких приложений:

* [Bitly](https://bitly.com/)
* [MeShort](http://www.msht.us/)

**Технические детали**

В приложении пользователи будут вводить исходный URL-адрес и в результате получать новый, сокращённый. Чтобы создать символы для укороченной версии адреса, можно использовать комбинацию, сгенерированную при помощи модулей random и string.

Поскольку есть вероятность, что пользователи будут посещать сокращенные URL-адреса через несколько дней, месяцев или даже лет, вам необходимо сохранить исходные и сокращенные адреса в базе данных. При поступлении запроса приложение проверяет, существует ли URL-адрес, и перенаправляет его на оригинал, в противном случае на страницу 404.

**Дополнительно**

Можно добавить функцию для настройки генерации URL-адреса. Без сомнения, xyz.com/mysiteURL лучше, чем случайно сгенерированный xyz.com/piojwrURL.

**Post-It Note**

С тем количеством информации, которая проходит через нас ежедневно, неудивительно, что многое забывается. Здесь и помогают стикеры и заметки. Идея веб-проекта Post-It Note — позволить делать себе короткие напоминалки, которые можно будет посмотреть из любого браузера.

С Post-It Note люди могут делать записи в любое время и в любом месте, не опасаясь забыть или потерять что-либо как в случае с обычными стикерами.

Примеры таких приложений:

* [Pinup](https://pinup.com/)
* [Note.ly](https://note.ly/)

**Технические детали**

Основная цель этого проекта — предоставить людям место, где они смогут записывать свои мысли. Это означает, что у каждого пользователя будут свои заметки, поэтому в приложении должны быть учётные записи. Здесь важно, чтобы заметки каждого пользователя оставались приватными.

В Django встроена система аутентификации пользователя, так что в случае с ним всё просто. Вы можете использовать другие фреймворки, такие как Bottle или Flask, но вам придется самостоятельно внедрять систему аутентификации пользователей.

В какой-то момент пользователям может потребоваться разделить свои заметки по разным категориям. Реализация функции, позволяющей пользователям классифицировать свои заметки, сделает приложение более полезным. Например, вам могут понадобиться заметки об алгоритмах и структурах данных, поэтому приложение должно предоставить пользователю возможность разделить заметки по этим категориям.

Вам нужно будет хранить информацию и заметки каждого пользователя, поэтому база данных станет неотъемлемой частью этого проекта. Если вы хотите использовать базу данных MySQL, то вам понадобится модуль MySQLdb или psycopg2-модуль для базы данных PostgreSQL. Есть и другие модули, которые вы можете использовать, — всё зависит от базы данных, которую вы выбрали.

**Дополнительно**

Так как люди могут забывать о своих идеях, они так же могут забыть, что они где-то сделали заметку. Вы можете добавить функцию, которая позволит пользователям устанавливать время для уведомлений.

**Приложение Викторина**

В мире много всего, чему можно научиться, и тесты помогают проверить знания и выяснить, что вы ещё не до конца понимаете. Это приложение — удобный инструмент, чтобы создавать тесты и викторины.

Специальные пользователи, которых вы сделаете администраторами, смогут создавать тесты, а обычные пользователи будут отвечать на вопросы и проверять свои знания на определённые темы.

Примеры приложений викторины:

* [myQuiz](https://myquiz.org/)
* [Kahoot](https://kahoot.com/)

**Технические детали**

Одни пользователи создают викторины и тесты, а другие проходят их. Затем приложение отображает итоговую оценку и правильные ответы. Если вы хотите, чтобы пользователи имели возможность записывать свои оценки, вам придется создать учётные записи.

Кроме того, вам нужно создать базу данных для этого проекта. Она будет хранить вопросы, возможные ответы, правильные ответы и оценки каждого пользователя.

**Дополнительно**

Создателям тестов будет удобно, если викторины будут конвертироваться из простого текстового файла определённого формата. Также можно разрешить пользователям добавлять таймеры в тесты. Таким образом создатели теста смогут определить, сколько секунд или минут пользователь должен потратить на каждый вопрос в тесте. Ещё было бы замечательно иметь функцию обмена тестами, чтобы пользователи могли делиться интересными тестами со своими друзьями.

**Идеи десктопных приложений**

**MP3-плеер**

MP3-плеер — это устройство для воспроизведения MP3-файлов и других цифровых аудиофайлов. GUI MP3-плеера обычно подражает физическому MP3-плееру.

Примеры MP3-плееров:

* [MusicBee](https://getmusicbee.com/downloads/)
* [Foobar2000](https://www.foobar2000.org/)

**Технические детали**

Чтобы быть привлекательным для пользователей, приложение должно иметь простой и красивый интерфейс. Плеер должен отображать информацию о воспроизводимом файле — имя, длительность и воспроизведённую и не воспроизведённую часть файла в минутах и ​​секундах.

В Python есть библиотеки — [pygame](https://pypi.org/project/pygame/" \t "_blank), [pymedia](https://pypi.org/project/PyMedia/" \t "_blank) и [simpleaudio](https://pypi.org/project/simpleaudio/" \t "_blank), позволяющие работать с мультимедийными файлами в пару строк кода. Они могут обрабатывать файлы и других типов, а не только файлы MP3.

**Дополнительно**

Вы можете реализовать функцию, которая позволяет пользователям создавать плейлисты. Для этого вам понадобится база данных для хранения информации о созданных плейлистах. Модуль sqlite3 позволяет использовать базу данных SQLite, она является лучшим вариантом в этом случае, поскольку основана на файлах и её проще настроить, чем другие базы данных SQL.

Вы можете добавить функцию повторения воспроизводимых в данный момент аудио или даже перемешивать список файлов для воспроизведения. Также возможно реализовать функцию, которая позволяет пользователям увеличивать и уменьшать скорость воспроизведения аудиофайла.

**Alarm tool**

Будильник, он и есть будильник. Подаёт звуковой сигнал в определённое время.

Примеры таких проектов:

* [FreeAlarmClock](http://freealarmclocksoftware.com/)
* [TimerForMac](https://www.apimac.com/mac/timer/)

**Технические детали**

Основная задача данного приложения — активировать звуковые сигналы в определённое время. Таким образом, информация о времени и звуковой сигнал, который должен быть воспроизведён, — основные части Alarm Tool. Приложение должно постоянно проверять установленное время будильника. Когда оно наступает, запускается функция воспроизведения сигнала. Для воспроизведения звука подойдёт библиотека pygame.

Пользователи должны иметь возможность редактировать и удалять напоминания. Приложение должно отображать все напоминания, даже неактивные. В базе данных вы будете хранить такие вещи, как дата и время будильника и путь до воспроизводимого звукового сигнала.

**Дополнительно**

В качестве дополнительной функции вы можете разрешить пользователям устанавливать повторяющиеся напоминания. Они смогут устанавливать сигналы будильника, которые будут звучать в определенное время в определённые дни недели. Например, будильник может быть установлен на 14:00 каждый понедельник.

Вы также можете добавить функцию откладывания, чтобы ваши пользователи могли не только отключать его.

**Файловый менеджер**

Как правило, на персональном компьютере хранится множество файлов. Если бы все эти файлы были размещены в одном каталоге, было бы трудно ориентироваться и находить файлы или подкаталоги. Поэтому существует необходимость организовывать файлы и для этого существуют файловые менеджеры. Они позволяют пользователям управлять файлами и каталогами через интерфейс.

С помощью файлового менеджера пользователи могут правильно организовывать свои файлы и каталоги, получать к ним доступ и управлять ими. Также файловый менеджер позволяет пользователям копировать, перемещать и переименовывать файлы или каталоги.

Примеры файловых менеджеров:

* [FreeCommander](https://freecommander.com/en/summary/)
* [Explorer++](https://explorerplusplus.com/)

**Технические детали**

Основная задача файлового менеджера — предоставить пользователям интерфейс для управления своими файлами. Он должен быть простым и удобным в использовании.

Вы можете использовать PySimpleGUI для создания пользовательского интерфейса без особой сложности. Библиотеки sys, os и shutil будут весьма полезны для этого проекта, поскольку в них присутствуют функции для выполнения действий с файлами в фоновом режиме, пока пользователь щёлкает мышью. Отображение файлов в виде сетки или спискапопулярнее всего, поэтому можно реализовать в приложении оба варианта. Это даст пользователю возможность выбрать, какой вариант просмотра ему подходит.

**Дополнительно**

Чтобы сделать файловый менеджер немного более продвинутым, вы можете реализовать функцию поиска. Таким образом, пользователи могут искать файлы и каталоги, не пытаясь найти их вручную.

Также можно реализовать функцию сортировки. Это позволит пользователям сортировать файлы по времени создания, названию или размеру.

**Expense Tracker**

У каждого есть ежедневные расходы: люди тратят деньги на продукты, одежду и оплату счетов. Трекер поможет следить за своими расходами. Он будет анализировать расходы, следить за их ростом или снижением.

С помощью Expense Tracker пользователи могут устанавливать бюджет и отслеживать расходы, чтобы принимать более обдуманные финансовые решения.

Примеры реализаций таких трекеров:

* [Buddi](http://buddi.digitalcave.ca/)
* [GnuCash](https://www.gnucash.org/)

**Технические детали**

Нужно провести некоторый статистический анализ, чтобы предоставить пользователям правильную информацию о своих расходах и помочь им тратить меньше. Несмотря на то, что отслеживание расходов является ключевой функцией, хороший интерфейс также не помешает. Опять же, с этим поможет PySimpleGUI. Библиотеки PyData, такие как Pandas и Matplotlib помогут вам с техническими аспектами. [Pandas](https://pandas.pydata.org/pandas-docs/stable/" \t "_blank) может быть использована для анализа данных, а с помощью [Matplotlib](https://matplotlib.org/" \t "_blank) удобно строить графики, которые дадут пользователям наглядное представление о расходах.

Приложение будет получать данные от пользователей. Для их хранения можно использовать SQLite и Python-модуль Sqlite3.

**Дополнительно**

Чтобы пользователи могли извлечь выгоду из этого приложения, им нужно регулярно вводить свои расходы, но об этом легко забыть. Поэтому можно добавить функцию напоминания. Приложение будет отправлять уведомления в определённое время дня или недели, напоминая пользователям о том, что пора заглянуть в Expense Tracker.

**Идеи приложений для командной строки**

**Контактная книга**

Новые знакомые часто сообщают свои контакты для поддержания связи. Не потерять их поможет контактная книга. Обычно в ней хранится имя, адрес, номер телефона и адрес электронной почты. Вы можете создать ПО, которое люди будут использовать для сохранения и поиска контактных данных. Оно всегда будет доступно с их компьютера через командную строку.

Существуют похожие приложения, но большинство из них — мобильные, десктопные или веб-приложения:

* [Simple Contacts](https://play.google.com/store/apps/details?id=com.simplemobiletools.contacts&hl=en_US)
* [Pobuca Connect](https://connect.pobuca.com/)

**Технические детали**

Важно настроить команды, которые пользователи могут использовать для ввода контактных данных. Вы можете использовать фреймворки Argparse или Click. Они абстрагируют много сложных вещей, поэтому вам нужно сосредоточиться только на коде, который будет выполняться при вводе команд.

Должны существовать команды для удаления контактов, обновления контактной информации и вывода списка сохранённых контактов. Вы также можете добавить функцию отображения контактов в определённом порядке, например, по алфавиту или дате создания.

Для хранения отлично подойдёт база данных SQLite.

**Дополнительно**

Вы помните, что база данных SQLite хранится на компьютере пользователя? Что, если что-то случится, например, пользователь потеряет свои файлы? Это означает, что он также потеряет контактную информацию.

Вы можете подстраховаться от этого и сделать резервную копию базы данных на платформе онлайн-хранилища. Для этого приложение может загружать файлы базы данных в облако через определённые промежутки времени. Можно даже добавить команду, которая позволяет пользователям создавать резервные копии базы данных самостоятельно. Таким образом, даже если файл базы данных потерян, пользователь по-прежнему будет иметь доступ к контактам.

Следует помнить, что вам может потребоваться какая-либо форма идентификации, чтобы в книге контактов можно было узнать, какой файл базы данных принадлежит какому пользователю. Аутентификации пользователя является одним из способов решения этой проблемы.

**Site Connectivity Checker**

Иногда сайты могут быть недоступны по разным причинам, не зависящим от вас. Чтобы узнать, когда он заработает, можно пробовать открыть его раз в какое-то время, а можно использовать инструмент, который будет делать это за вас. Именно это и делает Site Connectivity Checker — посещает URL-адрес и возвращает его статус: активен или нет.

Примеры таких приложений:

* [Ping](https://en.wikipedia.org/wiki/Ping_(networking_utility))
* [Site24x7](https://www.site24x7.com/check-website-availability.html)

**Технические детали**

Вам нужно выбрать, что вы будете использовать для своих подключений — [TCP](https://en.wikipedia.org/wiki/Transmission_Control_Protocol) или [ICMP](https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol). Модуль [Socket](https://docs.python.org/3/library/socket.html) понадобится непосредственно для самой проверки. Если вы с ним плохо знакомы, можете прочитать [Socket Programming on Python (Guide)](https://realpython.com/python-sockets/). Благодаря выбранному фреймворку, будь то Docopt, Click или Argparse, вы добавляете команды, чтобы пользователь мог взаимодействовать с вашим приложением. Он также должен иметь возможность запускать приложение, останавливать его и определять интервалы проверки.

Для хранения данных можно использовать уже знакомую вам SQLite.

**Дополнительно**

Постоянно проверять командную строку, чтобы узнать, заработал ли сайт, не очень-то удобно. Поэтому можно реализовать функцию уведомления. Это может быть просто какой-либо звуковой сигнал, который уведомит пользователя об изменении статуса сайта.

**Bulk File Rename Tool**

Иногда нужно назвать все файлы в каталоге в соответствии с определённым шаблоном. Например, вы можете присвоить всем файлам в каталоге имена с порядковыми номерами, которые увеличиваются в зависимости от количества файлов (File0001.jpg, File0002.jpg и т. д.). Выполнение этого вручную займёт уйму времени. Bulk File Rename Tool экономит пользователям много времени, ведь он избавляет их от необходимости выполнять скучную однотипную работу. С помощью Bulk File Rename Tool пользователи смогут переименовать тысячи файлов за пару секунд.

Примеры подобных инструментов:

* [Ren](https://en.wikipedia.org/wiki/Ren_(command))
* [Rename](https://www.tecmint.com/rename-multiple-files-in-linux/)

**Технические детали**

Приложение должно найти способ манипулировать целевыми файлами. В этом проекте будут полезны библиотеки os, sys и shutil. Пользователи смогут задавать шаблон, под который должны переименовываться все файлы, с этим поможет модуль regex.

Например, есть шаблон myfiles, и приложение должно переименовать все файлы в myfiles + XYZ, где XYZ — это число. Конечно же, пользователи должны иметь возможность выбрать каталог, в котором находятся файлы для переименования.

**Дополнительно**

Пользователям может потребоваться переименовать только определённое количество файлов. Реализуйте для этого функцию. Имейте ввиду, что для этого потребуется сортировка файлов по алфавиту, времени создания файла или размеру файла, в зависимости от требований пользователя.

**Генератор древа каталогов**

Каталоги похожи на родословные: каждый каталог имеет определённые отношения с другими каталогами. Никакие каталоги никогда не остаются самостоятельными, кроме пустого корневого каталога.

Когда вы работаете с файлами и каталогами, трудно увидеть взаимосвязь между ними, поскольку вы можете видеть только то, что существует в текущем уровне. С Генератором древа каталогов вы сможете увидеть связь между элементами. Это облегчает понимание их расположения.

Примеры таких генераторов:

* [Tree](https://en.wikipedia.org/wiki/Tree_(command))
* [Dirtreex](https://github.com/mauriziodimatteo/dirtreex)

**Технические детали**

Стандартная библиотека os будет ядром вашего приложения, она и будет демонстрировать связь между каталогами. Используйте библиотеки Docopt или Argparse для создания команд. Для придания вашему приложению более приятного вида можно использовать Colored. Надо решить, насколько глубоко будет заходить генератор. Например, если каталог имеет дочерние каталоги глубиной двенадцать уровней, стоит ли заходить дальше пятого уровня?

**Дополнительно**

Так как результаты сгенерированного древа каталогов будут в командной строке, вы можете пойти ещё дальше, сделав так, чтобы генератор создавал изображения из древа каталогов, т. е. он превратит текст в изображение. Для этого используется библиотека [Pillow](https://pillow.readthedocs.io/en/stable/" \t "_blank).