**Введение**

Python – популярный язык программирования, используемый как для разработки самостоятельных программ, так и для создания прикладных сценариев в самых разных областях применения. Это мощный, переносимый, простой в использовании и свободно распространяемый язык. Программисты, работающие в самых разных областях, считают, что ориентация Python наэффективность разработки и высокое качество программного обеспечения дает ему стратегическое преимущество как в маленьких, так и в крупных проектах. Количеством числа пользователей Python является число, близкое 1 миллиону человек во всем мире (на 2011). Эта оценка основана на различных статистических показателях, таких как количество загрузок и результаты опросов разработчиков. Дать более точную оценку достаточно сложно, потому что Python является открытым программным обеспечением – для его использования не требуется проходить лицензирование. Более того, Python по умолчанию включается в состав дистрибутивов Linux, поставляется вместе с компьютерами Macintosh и некоторыми другими программными и аппаратными продуктами, что существенно затрудняет оценку числа пользователей. Вообще же количество пользователей Python значительно больше и вокруг него сплотилось очень активное сообщество разработчиков. Благодаря тому, что Python появился более 23 лет тому назад и получил широкое распространение, он отличается высокой стабильностью и надежностью. Python используется не только отдельными пользователями, он также применяется компаниями для создания продуктов, приносящих настоящую прибыль.

По оценке TIOBE Index1, на сентябрь 2015 Python занимает пятое место по популярности, обогнав такие языки, как PHP, JavaScript, Perl, Ruby, Delphi, Pascal, Swift и др. Такие известные компании, как Google и Intel, Cisco и Hewlett-Packard, используют язык Python, выбрав его за гибкость, простоту использования и обеспечиваемую им высокую скорость разработки. Он позволяет создавать эффективные и надежные проекты, которые легко интегрируются с программами и инструментами, написанными на других языках.

Универсальная природа языка обеспечивает возможность его применения в самых разных областях. Фактически с определенной долей уверенности .Итак, Python – один из наиболее популярных и эффективных языков программирования высокого уровня на данный момент, используемый множеством крупных компаний для решения задач самого разного рода. Поэтому целью данной работы стало рассмотреть язык программирования Python в общих чертах, а также подробно изучить способы его применения на практике

**История**

Разработка языка Python была начата в конце 1980-х годов сотрудником голландского института CWI Гвидо ван Россумом. Для распределённой ОС Amoeba требовался расширяемый скриптовый язык, и Гвидо начал писать Python на досуге, позаимствовав некоторые наработки для языка ABC (Гвидо участвовал в разработке этого языка, ориентированного на обучение программированию). В феврале 1991 года Гвидо опубликовал исходный текст в ньюсгруппе alt.sources. С самого начала Python проектировался как объектно-ориентированный язык. Название языка произошло вовсе не от вида пресмыкающихся. Автор назвал язык в честь популярного британского комедийного телешоу 1970х «Летающий цирк Монти Пайтона». Впрочем, всё равно название языка чаще ассоциируют именно со змеёй, нежели с передачей — пиктограммы файлов в KDE3 или в Microsoft Windows (Рисунок 4) и даже эмблема (Рисунок 3) на сайте python.org (до выхода версии 2.5) изображают змеиные головы. Наличие дружелюбного, отзывчивого сообщества пользователей считается наряду с дизайнерской интуицией Гвидо одним из факторов успеха Python. Развитие языка происходит согласно чётко регламентированному процессусоздания, обсуждения, отбора и реализации документов PEP (англ. Python Enhancement Proposal) — предложений по развитию Python. 3 декабря 2008 года, после длительного тестирования, вышла первая версия Python 3000 (или Python 3.0, также используется сокращение Py3k). В Python 3000 устранены многие недостатки архитектуры с максимально возможным (но не полным) сохранением совместимости со старыми версиями Python. На сегодня поддерживаются обе ветви развития (Python 3.x и 2.x). Появившись сравнительно поздно, Python создавался под влиянием множества языков программирования:

Miranda — списочные выражения;

Java — модули logging, unittest, threading (часть возможностей оригинального модуля не реализована), xml.sax стандартной библиотеки, совместное использование finally и except при обработке исключений, использование @ для декораторов;

Icon — генераторы.

Python поддерживает несколько парадигм программирования, втом числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление .

2.**3 Средства оптимизации скорости выполнения**

Также существуют и другие реализации Python, включая динамический компилятор Psyco и транслятор Shedskin C++, которые пытаются оптимизировать основную модель выполнения.

Система Psyco – это не совсем другая реализация языка Python, а скорее компонент, расширяющий модель выполнения байт-кода, что позволяет программам выполняться быстрее.

Shedskin – это еще одна система, которая реализует нетрадиционный подход к выполнению программ на языке Python. Она преобразует исходный код на языке Python в исходный код на языке C++, который затем может быть скомпилирован в машинный код. Кроме того, эта система реализует платформонезависимый подход к выполнению программного кода Python.

**2.4 Типы и структуры данных**

В Python все значения являются объектами, в том числе функции, методы, модули, классы. Каждый объект имеет свой тип и список имен, ссылающихся на данный объект. Когда какому-либо имени присваивается новый объект (который может быть другого типа), имя удаляется из списка старого объекта и добавляется в список нового объекта. Таким образом, переменные могут как бы менять свой тип, но тип самих объектов при этом не изменяется. Данный прием называется строгой динамической типизацией. В Python имеются встроенные типы: булевый, строка, Unicode-строка, целое число произвольной точности, число с плавающей запятой, комплексное число и некоторые другие. Из коллекций в Python встроены: список, кортеж (неизменяемый список), словарь, множество и другие.

Добавить новый тип можно либо написав класс (class), либо определив новый тип в модуле расширения (например, написанном на языке C). Система классов поддерживает наследование (одиночное и множественное) и метапрограммирование. Возможно наследование от большинства встроенных типов и типов расширений. Все объекты делятся на ссылочные и атомарные. К атомарнымотносятся int, long (в версии 3 любое число int, так как в версии 3 нет ограничения на размер), complex и некоторые другие. При присваивании атомарных объектов копируется их значение, в то время как для ссылочных копируется только указатель на объект, таким образом, обе переменные после присваивания используют одно и то же значение.

**2.5 Портируемость**

Python портирован и работает почти на всех известных платформах — от КПК до мейнфреймов. Существуют порты под Microsoft Windows, практически все варианты UNIX (включая FreeBSD и Linux), Plan 9, Mac OS и Mac OS X, iPhone OS 2.0 и выше, Palm OS, OS/2, Amiga, HaikuOS, AS/400 и даже OS/390, Windows Mobile, Symbian и Android.

**2.6 Virtualenv и PEP**

Virtualenv – это инструмент, позволяющий создавать изолированные виртуальные окружения. Это позволяет разрабатывать сразу на нескольких версиях Python и помещать каждой приложение в отдельное окружение, что позволяет избежать влияния приложений друг на друга и на системное окружение. Данный инструмент очень широко распространен и рекомендуется для разработки официальными документациями и авторами многих проектов. Python Enhancement Proposal («PEP») — это документ со стандартизированным дизайном, предоставляющий общую информацию о языке Python, включая новые предложения, описания и разъяснения возможностей языка. PEP предлагаются как основной источник для предложения новых возможностей.

**2.8 Основные преимущества и недостатки**

Объектно-ориентированный

Свободный (бесплатный)

Переносимый

Мощный (с точки зрения функциональных возможностей)

Соединяемый (можно использовать в других языковых средах)

Удобен и прост в изучении

**2.8.1 Недостатки**

Основным недостатком Python, как и у большинства языков программирования высокого уровня, является скорость выполнения программ, которая не всегда такая же высокая как у языков программирования низкого уровня (таких как С и С++). Однако, по мнению Марка Лутца, ведущего специалиста в области обучения языку программирования Python и автора самых ранних и наиболее популярных публикаций, данная проблема ярко выражена лишь для программ, выполняющих задачи определенного рода: «Будет ли вас когда-нибудь волновать разница в скорости выполнения программ, зависит от того, какого рода программы вы пишете. Python многократно подвергался оптимизации и в отдельных прикладных областях программный код на этом языке отличается достаточно высокой скоростью выполнения. Кроме того, когда в сценарии Python делается что-нибудь «значительное», например, обрабатывается файл или конструируется графический интерфейс, ваша программа фактически выполняется со скоростью, которую способен дать язык C, потому что такого рода задачи решаются компилированным с языка С программным кодом, лежащим в недрах интерпретатора Python. Гораздо важнее, что преимущество в скорости разработки порой важнее потери скорости выполнения, особенно если учесть быстродействие современных компьютеров. К другим недостаткам также можно отнести невозможность модификации встроенных классов и глобальную блокировку интерпритатора.

**2.9 Сравнение с другими языками и синтаксис**

Python обладает очень удобным и понятным синтаксисом по сравнению с другими языками программирования, и кроме того самым минималистичным. Это можно увидеть на примере сравненияминимальных программ, выводящих “Hello, World!” на экран, написанных на разных языках.

Сравнение синтаксиса разных языков программирования

**2.9.2 Скорость выполнения**

Сравнение скорости выполнения программ на разных языках программирования очень относительное, т.к. производительность сильно отличается для различных алгоритмов, платформ, использованных средств языка и т.д. Однако, проведя некоторые тесты, можно дать условную оценку производительности относительно других языков. языками

**3 Применение**

Python можно применять в самых разных областях. Для этого существуют специальное программное обеспечение, как правило представляющее собой пакет из нескольких модулей Python. Но очень часто для решения простых задач достаточно и стандартных модулей, поставляемых с интерпритатором Python. Все же остальные пакеты можно легко найти в PyPI9. PyPI на данный момент включает в себя более 66 тысяч пакетов. Список самого популярного программного обеспечения для разработки на Python всегда можно найти на официальном сайте python.org на странице Applicationsfor Python10. Рассмотрим наиболее востребованные области применения языка и самые популярные средства для достижения результата.

**3.1 Системное программирование**

Встроенные в Python интерфейсы доступа к службам операционных систем делают его идеальным инструментом для создания переносимых программ и утилит системного администрирования (иногда они называются инструментами командной оболочки). Программы на языке Python могут отыскивать файлы и каталоги, запускать другие программы, производить параллельные вычисления с использованием нескольких процессов и потоков и делать многое другое. Стандартная библиотека Python полностью отвечает требованиям стандартов POSIX11 и поддерживает все типичные инструменты операционных систем: переменные окружения, файлы, сокеты, каналы, процессы, многопоточную модель выполнения, поиск по шаблону с использованием регулярных выражений, аргументы командной строки, стандартные интерфейсы доступа к потокам данных, запуск команд оболочки, дополнение имен файлов и многое другое. Кроме того, системные интерфейсы в языке Python созданы переносимыми, например, сценарий копирования дерева каталогов не требует внесения изменений, в какой бы операционной системе он ни использовался. Система Stackless Python, используемая компанией EVE Online, также предлагает улучшенные решения, применяемые для параллельной обработки данных.

**3.2 Создание программ с графическим интерфейсом**

В состав Python входит стандартный объектно-ориентированный интерфейс к Tk GUI API, который называется tkinter (в Python 2.6 он называется Tkinter), позволяющий программам на языке Python реализовать переносимый графический интерфейс с внешним видом, присущим операционной системе. Графические интерфейсы на базе Python/tkinter без изменений могут использоваться в MS Windows, X Window (в операционных системах UNIX и Linux) и Mac OS(как в классической версии, так и в OS X). Свободно распространяемый пакет расширения PMW содержит дополнительные визуальные компоненты для набора tkinter. Кроме того, существует прикладной интерфейс wxPython GUI API, основанный на библиотеке C++, который предлагает альтернативный набор инструментальных средств построения переносимых графических интерфейсов на языке Python. Инструменты высокого уровня, такие как PythonCard и Dabo, построены на основе таких API, как wxPython и tkinter. При выборе соответствующей библиотеки вы также сможете использовать другие инструменты создания графического интерфейса,

такие как Qt (с помощью PyQt), GTK (с помощью PyGtk), MFC (с помощью PyWin32), .NET (с помощью IronPython), Swing (с помощью Jython – реализации языка Python на Java, или JPype). Также существует фреймворк Kivy – библиотека Python с открытым

исходным кодом, предназначенная для быстрой разработки приложений, использующих современные пользовательские интерфейсы, такие как мультисенсорные приложения. Kivy предназначен для Windows, Linux, Android и iOS.

**3.3 Интеграция компонентов**

Возможность Python расширяться и встраиваться в системы на языке C и C++ делает его удобным и гибким языком для описания поведения других систем и компонентов. Например, интеграция с библиотекой на языке C позволяет Python проверять наличие и запускать библиотечные компоненты, а встраивание Python в программные продукты позволяет производить настройку программных продуктов без необходимости

пересобирать эти продукты или поставлять их с исходными текстами. Такие инструменты, как Swing и SIP, автоматически генерирующие программный код, могут автоматизировать действия по связыванию скомпилированных компонентов в Python для последующего их использования в сценариях, а система Cython позволяет программистам смешивать программный код на Python и C. Такиеогромные платформы на Python, как поддержка COM в MS Windows, Jython – реализация на языке Java, IronPython – реализация на базе .NET и разнообразные реализации CORBA, предоставляют альтернативные способы

организации взаимодействий с программными компонентами. Например, в операционной системе Windows сценарии на языке Python могут использовать платформы управления такими приложениями, как MS Word и Excel.

**3.5 Быстрое создание прототипов**

В программах на языке Python компоненты, написанные на Python и на C, выглядят одинаково.Благодаря этому можно сначала создавать прототипы систем на языке Python, а затем переносить выбранные компоненты на компилирующие языки, такие как C и C++. В отличие от ряда других инструментов разработки прототипов, язык Python не требует, чтобы система была полностью переписана, как только прототип будет отлажен. Части системы, которые не требуют такой эффективности выполнения, какую обеспечивает C++, можно оставить на языке Python, что существенно упростит сопровождение и использование такой системы.

3.6 Математические и научные вычисления Для Python существует множество расширений для математических и научных вычислений.

Расширение NumPy для математических вычислений включает такие мощные элементы, как объекты массивов, интерфейсы к стандартным математическим библиотекам, и многое другое. Расширение NumPy – за счет интеграции с математическими библиотеками, написанными на компилирующих языках программирования – превращает Python в сложный, но удобный инструмент программирования математических вычислений, который зачастую может заменить существующий программный код, написанный на традиционных компилирующих языках, таких как FORTRAN и C++. Дополнительные инструменты математических вычислений для Python поддерживают возможность создания анимационных эффектов и трехмерных объектов, позволяют организовать параллельные вычисления и так далее. Например, популярные расширения SciPy и ScientificPython предоставляют дополнительные библиотеки для научных вычислений и используют возможности расширения NumPy. Для SciPy существует дополнение IPtyhon − интерактивная оболочка, которая предоставляет расширенную интроспекцию, дополнительный командный синтаксис, подсветку кода и автоматическое дополнение. Пакет Pandas делает Python мощным инструментом для анализа данных..

**3.7 Веб-разработка**

По данным Wappalyzer на Python написано около 1% всех сайтов (Рисунок 6). Так, Python делит 3-е место с Ruby и Perl. Один из самых популярных социальных новостных сайтов Reddit написан на Python. Кроме того, на Python написан известный BitBucket, веб-сервис для хостинга проектов и их совместной разработки, основанный на системе контроля версий Mercurial и Git.

Если для разработчиков на C# или, например, Ruby выбор фреймворка очевиден (ASP.NET и Ruby on Rails соответственно), то для разработчиков на Python существует огромный выбор средств разработки под веб. Для веб-разработки разработки на Python существуют множество фрейморков (Django, Pylons и его приемник Pyramid, TurboGears, Zope, Twisted, CherryPy и т.д.), микрофрейморков (Flask, Bottle и др.) и CMS12 (Django-CMS, Plone, Mezzanine, Quokka и др.)13. Многие фреймворки и CMS для удобства разработки имеют встроенный локальный веб-сервер.

Однако, для разработки веб-приложений на Python совсем не обязательно использовать готовые решения. В поставку стандартного интерпритатора уже входит множество модулей, необходимых для разработки в веб.

**3.7.1 WSGI**

WSGI (англ. Web Server Gateway Interface, обычно произносится сообществом как «висги» или «виски») — стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером, например, Apache. Как уже упоминалось, в Питоне существует большое количество веб-фреймворков, тулкитов и библиотек. Все они по-своему устанавливаются и настраиваются, и часто возникает проблема их взаимодействия между собой. По этой причине был разработан WSGI (Web Server Gateway Interface) - стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером.

В том числе, WSGI определяет middleware-компоненты, которые предоставляют интерфейсы как приложению, так и серверу. То есть, для сервера middleware является приложением, а для приложения — сервером. Это позволяет составлять цепочки WSGI-совместимых middlewares. Таким образом (в теории), подбирая нужные middleware-компоненты, можно составлять собственные фреймворки. Эта концепция наиболее широко проявила себя в Pylons. MVC – наиболее часто используемый в веб-разработке паттерн. Это коснулось и Python. Многие Python-фреймворки построены на MVC.

**MVC** (Model-view-controller) представляет собой шаблон проектирования, при котором модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента (Рисунок 7), что позволяет модифицировать отдельные компоненты с минимальным воздействием на остальные. Концепция MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

Модель (англ. Model). Модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать.Представление, вид (англ. View). Отвечает за отображение информации (визуализацию).

**3.7.3 Django**

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation.

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других. Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself) Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений, а не выводятся автоматически из структуры моделей контроллеров. Для работы с базой данных Django использует собственный ORM14, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных. Имеет встроенный интерфейс администратора и встроенный локальный веб-сервер для отладки.

**3.7.4 Flask**

Среди микрофреймворков лидирующее положение занимает Flask. “Микро” в слове “микрофреймворк” означает, что Flask стремится придерживаться простого, но расширяемого ядра. По умолчанию, Flask не включает уровень абстракции баз данных, валидации форм или каких-то иных, для чего уже существуютразличные занимающиеся этим библиотеки. Вместо этого, Flask поддерживает расширения для добавления подобной функциональности в ваше приложение, таким образом, как если бы это было реализовано в самом Flask. Многочисленные расширения обеспечивают интеграцию с базами данных, валидацию форм, обработку загрузок на сервер, различные открытые технологии аутентификации и так далее.

3**.7.5 Django-CMS**

Django-CMS – свободная система управления контентом, основанная на Django.

Среди возможностей, предоставляемых CMS можно выделить следующие:

Визуальное редактирование

Многоразовые плагины

Гибкая архитектура

Поисковая оптимизация

Управление правами

Контроль версий

Перевод на множество языков (около 40 на 2012 г.)

Создание нескольких сайтов на одном интерфейсе администратора

**3.8 Обучение**

Python является отличным языком для обучения программированию, как на начальном, так и на продвинутом уровне. К тому же, Python относится к тому небольшому числу языков, которые одновременно просты для изучения и мощны, а также востребованы на практике.

Для обучения языку существует огромное количество учебной литературы и интерактивных курсов в сети, кроме того Python преподается во многих американских школах и высших учебных заведениях в качестве первого языка программирования, т.к. Python – язык с низким уровнем вхождения (минимальным уровнем знаний, необходимых для успешного начала изучения).

Так, Майкл Доусон в своей книге-бестселлере «Программируем на Python» пишет: Если в программировании вы новичок, то ваш выбор можно только одобрить. Python - лучший вариант для начинающих. У него ясный и простой синтаксис, который позволит вам очень скоро начать писать полезные программы. Кроме того, у Python имеется интерактивный режим с возможностью тестирования свежих идей буквально на лету. Если у вас уже есть опыт программирования, вы тоже на верномпути. Python располагает всей той мощью и гибкостью, которую можно ожидать от современного объектно-ориентированного языка. Но при несомненной мощи Python, программы на нем пишутся удивительно быстро. В сущности, путь от идеи к компьютерной реализации сокращен настолько, что Python даже называют языком «программирования со скоростью мысли

3.9 Игры, изображения, искусственный интеллект, XML роботы и другое язык программирования Python можно использовать для решения более широкого круга задач, чем может быть упомянуто здесь. Например: Создавать игровые программы и анимационные ролики с помощью системы pygame.Писать модификации для популярных игр, поддерживающих Python. Обмениваться данными с другими компьютерами через последовательный порт с помощью расширения PySerial. Обрабатывать изображения с помощью расширений PIL, PyOpenGL, Blender, Maya и других. Управлять роботом с помощью инструмента PyRo.

Производить разбор XML-документов с помощью пакета xml, модуля xmlrpclib и расширений сторонних разработчиков.

**Заключение**

Итак, Python –популярный язык программирования, созданный Гвидо ван Россумом, в первую очередь объектно-ориентированный со строгой динамической типизацией. Существует огромное количество реализаций Python, язык портирован на большинство популярных платформ. Упор в языке сделан на простоту и очевидность, на ускорение разработки и на читабельность кода. И это всего несколько из множества плюсов. Основным же минусом является низкая производительность в некоторых реализациях, хотя для некоторых задач этим можно пренебречь.

Читабельность достигается за счет того, что из языка убрано множество лишних знаков, например, скобки для выделения блоков кода или точка с запятой, показывающая завершение команды. Вместо этого используются отступы в виде пробелов или табуляции. Кроме того, существуют правила, которых придерживаются большинство разработчиков на Python, эти правила описаны в Дзене Питона и в PEP.

Спектр применения языка очень широк. Это и системное программирование, и создание графических интерфейсов, и научные вычисления, и веб-разработка и многое другое. Для реализации самого разного рода задач существует официальный индекс пакетов. На данный момент в нём более 66 тысяч пакетов, которые позволяют решить практически любую задачу без особых усилий, в кратчайшие сроки, в минимальное количество строк кода, и что немаловажно, качественно. Python используется большинством крупных компаний мира, например, такими как Google, Intel, Cisco, Hewlett-Packard и Яндекс. По заявлению компании Google, в которой долгое время работал сам Гвидо ван Россум, Python является неотъемлемой частью Google. Знание Python является обязательным при устройстве на работу в Google. На Python написаны такие популярные сайты как Reddit и Bitbucket. Активно Python используется в программах для 3D моделирования и в играх, к примеру, для написания модификаций. Несмотря нато, что Python – язык программирования высокого уровня с относительно низкой производительностью, на нем написано множество игр. В основном, это 2D игры под мобильные устройства, но есть и крупномасштабные проекты, например, EVE Online.

Определенно, Python является перспективным языком, особенно в плане обучения. Так как он прост в изучении основ и имеет низкий уровень вхождения, но при этом учит хорошему стилю программирования и делает понятным объектно-ориентированный подход. И кроме того, в отличие от большинства языков для обучения, таких как Pascal или Visual Basic, Python после изучения можно успешно применять на практике

**Приложение**

Итак, приступим к созданию приложения. Это делается следующим образом: python manage.py startapp riddles. как только приложение создано, давайте напишем простой вид, по правилам Django все виды должны храниться в файле views.py.

*riddles/views.py*

from django.http import HttpResponse

def index(request):

return HttpResponse("Hello, World!")

Теперь, чтобы привязать наш вид к URL, создадим файл urls.py.

*riddles/urls.py*

from django.conf.urls import url

from . import views

app\_name = 'riddles'

urlpatterns = [

url(r'^$', views.index, name='index'),

]

В urls.py мы должны написать следующее:

*django\_example/urls.py*

from django.conf.urls import include, url

from django.contrib import admin

urlpatterns = [

url(r'^riddles/', include('riddles.urls')),

url(r'^admin/', admin.site.urls),

]

Теперь, если мы запустим наше приложение http://127.0.0.1:8000/riddles/, мы увидим «Hello, World!».

Теперь создадим нашу модель. Для начала создадим Riddle и Option. В Riddle

*riddles/models.py*

from django.db import models

class Riddle(models.Model):

riddle\_text = models.CharField(max\_length=255)

pub\_date = models.DateTimeField('date published'

class Option(models.Model):

riddle = models.ForeignKey(Riddle, on\_delete=models.CASCADE)

text = models.CharField(max\_length=255)

correct = models.BooleanField(default=False)

Теперь нам нужно привязать наше приложение к нашему проекту, делается это следующим образом: *django\_example/settings.py* После этого нужно сделать миграцию: python manage.py makemigrations riddles. Теперь дадим админу возможность изменять наши модели. Делается это так: *riddles/admin.py*

from django.contrib import admin

from .models import Option, Riddle

Начнем с шаблонов. Создадим папку templates внутри папки riddle, а в ней создадим index.html.

<h1>Available Riddles</h1>

{% if message %}

<p><strong>{{ message }}</strong></p>

{% endif %}

{% if latest\_riddles %}

<ul>

{% for riddle in latest\_riddles %}

<li>

<a href="/riddles/{{ riddle.id }}/">

{{ riddle.riddle\_text }}

</a>

</li>

{% endfor %}

</ul>

{% else %}

<p>No riddles are available right now.</p>

{% endif %}

Теперь создадим макет для ответов:

*riddles/templates/answer.html*

<h1>{{ riddle.riddle\_text }}</h1>

{% if error\_message %}

<p> <strong>{{ error\_message }}</strong></p>

{% endif %}

<form action="answer' riddle.id %}" method="post">

{% csrf\_token %}

{% for option in riddle.option\_set.all %}

<input type="radio" name="option" id="option{{ forloop.counter }}" value="{{ option.id }}" />

<label for="option{{ forloop.counter }}">{{ option.text }}</label><br>

{% endfor %}

<input type="submit" value="Answer" />

</form>

Теперь добавим наши функции в urls.py:

*riddles/urls.py*

from django.conf.urls import url

from . import views

app\_name = 'riddles'

urlpatterns = [

url(r'^$', views.index, name='index'),

url(r'^(?P[0-9]+)/$', views.detail, name='detail'),

url(r'^(?P[0-9]+)/answer/$', views.answer, name='answer')

]

Добавим немного стилей для начала создадим директорию static, а в ней создадим файл main.css.

*riddles/static/main.css*

body{ margin:40px auto; max-width:650px; line-height:1.6; font-size:18px; color:#444; padding:0 10px;

}

h1,h2,h3{

line-height:1.2;

text-align: center;

}

a { color: blue;}

form { margin: 0 auto; padding: 1em; border: 1px solid #CCC; border-radius: 1em;}

form div + div { margin-top: 1em;}

label {

display: inline-block; text-align: center; width: 40%;}

input {

font: 1em sans-serif; -moz-box-sizing: border-box; box-sizing: border-box; border: 1px solid #999;

width: 50%; }

input:focus {

border-color: #000;}

p, div.button {

text-align: center;

}

p.error-message {color: lightcoral;}

Немного изменим наши шаблоны:

*riddles/templates/index.html*

{% load staticfiles %}

<link rel="stylesheet" type="text/css" href="{% static 'main.css' %}" />

<h1>Available Riddles</h1>

{% if message %}

<p>

<strong>{{ message }}</strong>

</p>

{% endif %}

{% if latest\_riddles %}

<ul>

{% for riddle in latest\_riddles %}

<li>

<a href="/riddles/{{ riddle.id }}/">

{{ riddle.riddle\_text }}

</a>

</li>

{% endfor %}

</ul>

{% else %}

<p>No riddles are available right now.</p>

{% endif %}

*riddles/templates/answer.html*

{% load staticfiles %}

<link rel="stylesheet" type="text/css" href="{% static 'main.css' %}" />

<h1>

{{ riddle.riddle\_text }}

</h1>

{% if error\_message %}

<p>

<strong>{{ error\_message }}</strong>

</p>

{% endif %}

<form action="answer' riddle.id %}" method="post">

{% csrf\_token %}

{% for option in riddle.option\_set.all %}

<input type="radio" name="option" id="option{{ forloop.counter }}" value="{{ option.id }}" />

<label for="option{{ forloop.counter }}">{{ option.text }}</label><br>

{% endfor %}

<input type="submit" value="Answer" />

</form>

**Список источников**

1.Rossum G.V. Foreword for "Programming Python" (1st ed.) // Python.org. 1996. URL: https://www.python.org/doc/essays/foreword/ (дата обращения: 15.сентябрь.2015).

2. Лутц М. Изучаем Python. Санкт-Петербург: Символ-Плюс, 2010.

3. Доусон М. Программируем на Python. СПб: Питер, 2012.

4. Лутц М. Программируем на Python. Vol 1, 2. Санкт-Петербург: Символ-Плюс, 2011.

5. Content Management Systems [Электронный ресурс] // Python Wiki: [сайт]. URL: https://wiki.python.org/moin/ContentManagementSystems

6. [Электронный ресурс] // Kivi.org: [сайт]. URL: http://kivy.org/

7. Wappalyzer: [сайт]. URL: https://wappalyzer.com/

8. Python [Электронный ресурс] // Wikipedia: [сайт]. [2015]. URL: http://ru.wikipedia.org