**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.

Запись о работе, выполненной на практике.

|  |  |  |
| --- | --- | --- |
| Месяц и число | Краткое содержание выполненной работы | Подпись руководителя |
| 20.05.2019 | Инструктаж по технике безопасности |  |
| 21.05.2019 | Заправка картриджей |  |
| 22.05.2019 | Анализ структуры предприятие и структуры под-я предприятие |  |
| 23.05.2019 | Изучение функциональных обязанностей программиста |  |
| 24.05.2019 | Выходный день |  |
| 27.05.2019 | HTML и CSS |  |
| 28.05.2019 | Создания сайта с помощю HTML и CSS |  |
| 29.052019 | Переустановка Windows |  |
| 30.05.2019 | Подготовка рабочего место и ознакомление с программных обеспечением |  |
| 01.06.2019 | Microsoft-office |  |
| 03.06.2019 | Сайт w3schools.com |  |
| 04.06.2019 | Изучение Web программ |  |
| 05.06.2019 | Ознакомление язык программированием PHP |  |
| 06.06.2019 | Ознакомление с касперским антивирусом |  |
| 07.06.2019 | Ознакомление с виртуальных машин virtualbox |  |

**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. Добавить новый тип можно либо написав класс (class), либо определив новый тип в модуле расширения (например, написанном на языке C). Возможно наследование от большинства встроенных типов и типов расширений.

**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.

**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. Python Enhancement Proposal («PEP») — это документ со стандартизированным дизайном, предоставляющий общую информацию о языке Python, включая новые предложения, описания и разъяснения возможностей языка.

**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. «значительное», например, обрабатывается файл или

**Сравнение с другими языка**

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.

**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.

**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

**Веб-разработка**

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.

**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.

**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 реализуют в самом Flask.

**Web Приложение**

Установить Flask легко и просто. Cmd пропишим pip install flask

Когда вы закончите установку Flask, создайте папку FlaskApp. Перейдите в папку FlaskApp и создайте файл с именем app.py. Импортируйте модуль flask и создайте приложение с помощью Flask, как показано ниже:

from flask import Flask

app = Flask(\_\_name\_\_)

Теперь определим основной путь / и соответствующий ему обработчик запросов:

@app.route("/")

def main():

return "Welcome!"

Затем проверьте, является ли исполняемый файл главной программой и запустите приложение:

if \_\_name\_\_ == "\_\_main\_\_":

app.run()

Сохраните изменения и выполните app.py:

python app.py

Укажите браузеру на http://localhost:5000/ и у вас должно появиться приветственное сообщение. Advertisement

Создание домашней страницы

Во-первых, при запуске приложения мы должны показать домашнюю страницу с последними элементами списка дел, добавленными пользователями. Итак, добавим нашу домашнюю страницу в папку приложения.

Flask ищет файлы шаблонов внутри папки templates. Перейдите в папку PythonApp и создайте папку под названием templates. Внутри templates создайте файл index.html.

<!DOCTYPE html>

<html lang="en">

<head>

<title>Python Flask Bucket List App</title>

<link href="http://getbootstrap.com/dist/css/bootstrap.min.css" rel="stylesheet">

<link href="http://getbootstrap.com/examples/jumbotron-narrow/jumbotron-narrow.css" rel="stylesheet">

</head>

<body>

<div class="container">

<div class="header">

<nav>

<ul class="nav nav-pills pull-right">

<li role="presentation" class="active"><a href="#">Home</a>

</li>

<li role="presentation"><a href="#">Sign In</a>

</li>

<li role="presentation"><a href="showSignUp">Sign Up</a>

</li>

</ul>

</nav>

<h3 class="text-muted">Python Flask App</h3>

</div>

<div class="jumbotron">

<h1>Bucket List App</h1>

<p class="lead"></p>

<p><a class="btn btn-lg btn-success" href="showSignUp" role="button">Sign up today</a></p>

</div>

<div class="row marketing">

<div class="col-lg-6">

<h4>Bucket List</h4>

<p>Donec id elit non mi porta gravida at eget metus. Maecenas faucibus mollis interdum.</p>

<h4>Bucket List</h4>

<p>Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Cras mattis consectetur purus sit amet fermentum.</p>

<h4>Bucket List</h4>

<p>Maecenas sed diam eget risus varius blandit sit amet non magna.</p>

</div>

<div class="col-lg-6">

<h4>Bucket List</h4>

<p>Donec id elit non mi porta gravida at eget metus. Maecenas faucibus mollis interdum.</p>

<h4>Bucket List</h4>

<p>Morbi leo risus, porta ac consectetur ac, vestibulum at eros. Cras mattis consectetur purus sit amet fermentum.</p>

<h4>Bucket List</h4>

<p>Maecenas sed diam eget risus varius blandit sit amet non magna.</p>

</div>

</div>

<footer class="footer">

<p>&copy; Company 2015</p>

</footer>

</div>

</body>

</html>

Откройте app.py и импортируйте render\_template, который мы будем использовать для рендеринга файлов шаблонов.

from flask import Flask, render\_template

Измените основной метод, чтобы вернуть созданный файл шаблона.

def main():

return render\_template('index.html')

Сохраните изменения и перезапустите сервер.

Создание страницы регистрации

**Шаг 1. Настройка базы данных**

Мы будем использовать MySQL в качестве сервера. Войдите в MySQL из командной строки. Сначала создайте базу данных BucketList. Из командной строки:

mysql -u <username> -p

Введите требуемый пароль и при входе в систему выполните следующую команду для создания базы данных:

CREATE DATABASE BucketList;

Как только база данных будет создана, создайте таблицу tbl\_user, как показано ниже:

CREATE TABLE `BucketList`.`tbl\_user` (

`user\_id` BIGINT NULL AUTO\_INCREMENT,

`user\_name` VARCHAR(45) NULL,

`user\_username` VARCHAR(45) NULL,

`user\_password` VARCHAR(45) NULL,

PRIMARY KEY (`user\_id`));

Мы будем использовать Stored procedures в приложении Python для взаимодействия с базой данных MySQL. Поскольку таблица tbl\_user была создана, создайте процедуру сохранения под названием sp\_createUser, чтобы зарегистрировать пользователя.

При создании этой процедуры в таблице tbl\_user сначала нужно проверить, не существует ли пользователь с тем же именем username. Если существует, нам нужно выдать ошибку, иначе мы создадим пользователя в таблице user. Вот как должна выглядеть процедура sp\_createUser :

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_createUser`(

IN p\_name VARCHAR(20),

IN p\_username VARCHAR(20),

IN p\_password VARCHAR(20)

)

BEGIN

if ( select exists (select 1 from tbl\_user where user\_username = p\_username) ) THEN

select 'Username Exists !!';

ELSE

insert into tbl\_user

(

user\_name,

user\_username,

user\_password

)

values

(

p\_name,

p\_username,

p\_password

);

END IF;

END$$

DELIMITER ;

**Шаг 2. Создание интерфейса регистрации**

Перейдите в каталог PythonApp/templates и создайте файл HTML с именем signup.html. Добавьте следующий код HTML в signup.html:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Python Flask Bucket List App</title>

<link href="http://getbootstrap.com/dist/css/bootstrap.min.css" rel="stylesheet">

<link href="http://getbootstrap.com/examples/jumbotron-narrow/jumbotron-narrow.css" rel="stylesheet">

<link href="../static/signup.css" rel="stylesheet">

</head>

<body>

<div class="container">

<div class="header">

<nav>

<ul class="nav nav-pills pull-right">

<li role="presentation" ><a href="main">Home</a></li>

<li role="presentation"><a href="#">Sign In</a></li>

<li role="presentation" class="active"><a href="#">Sign Up</a></li>

</ul>

</nav>

<h3 class="text-muted">Python Flask App</h3>

</div>

<div class="jumbotron">

<h1>Bucket List App</h1>

<form class="form-signin">

<label for="inputName" class="sr-only">Name</label>

<input type="name" name="inputName" id="inputName" class="form-control" placeholder="Name" required autofocus>

<label for="inputEmail" class="sr-only">Email address</label>

<input type="email" name="inputEmail" id="inputEmail" class="form-control" placeholder="Email address" required autofocus>

<label for="inputPassword" class="sr-only">Password</label>

<input type="password" name="inputPassword" id="inputPassword" class="form-control" placeholder="Password" required>

<button id="btnSignUp" class="btn btn-lg btn-primary btn-block" type="button">Sign up</button>

</form>

</div>

<footer class="footer">

<p>&copy; Company 2015</p>

</footer>

</div>

</body>

</html>

Добавим такой CSS как signup.css в постоянную папку внутри PythonApp.

body { padding-top: 40px; padding-bottom: 40px;}

.form-signin { max-width: 330px; padding: 15px; margin: 0 auto;}

.form-signin .form-signin-heading,

.form-signin .checkbox { margin-bottom: 10px;}

.form-signin .checkbox {font-weight: normal;}

.form-signin .form-control { position: relative;height: auto;

-webkit-box-sizing: border-box;

-moz-box-sizing: border-box;

box-sizing: border-box;

padding: 10px;

font-size: 16px;}

.form-signin .form-control:focus { z-index: 2;}

.form-signin input[type="email"] {margin-bottom: -1px; border-bottom-right-radius: 0;

border-bottom-left-radius: 0;}

.form-signin input[type="password"] {margin-bottom: 10px; border-top-left-radius: 0;

border-top-right-radius: 0;}

В app.py добавьте метод showSignUp для отображения страницы регистрации после поступления запроса в /showSignUp:

@app.route('/showSignUp')

def showSignUp():

return render\_template('signup.html')

**Шаг 3. Внедрение метода регистрации**

Затем нам нужен server-side метод для UI взаимодействия с базой данных MySQL. Перейдите в PythonApp и откройте app.py. Создайте новый метод signUp, а также добавьте /signUp. Так это выглядит:

@app.route('/signUp')

def signUp():

# create user code will be here !!

Мы будем использовать jQuery AJAX для публикации данных регистрации в методе signUp, поэтому укажем метод в определении маршрута.

@app.route('/signUp',methods=['POST'])

def signUp():

# create user code will be here !!

Чтобы прочитать опубликованные значения, нам нужно импортировать request из Flask.

from flask import Flask, render\_template, request

С помощью request мы прочитаем значения, как показано ниже:

@app.route('/signUp',methods=['POST'])

def signUp():

# read the posted values from the UI

\_name = request.form['inputName']

\_email = request.form['inputEmail']

\_password = request.form['inputPassword']

Как только значения прочитаны, мы проверяем, верны ли они, а пока давайте вернём простое сообщение:

@app.route('/signUp',methods=['POST'])

def signUp():

# read the posted values from the UI

\_name = request.form['inputName']

\_email = request.form['inputEmail']

\_password = request.form['inputPassword']

# validate the received values

if \_name and \_email and \_password:

return json.dumps({'html':'<span>All fields good !!</span>'})

else:

return json.dumps({'html':'<span>Enter the required fields</span>'})

**Заключение**

За время прохождения учебной практики в ФИИТ КНУ я получил огромный опыт работы в коллективе, я смог на практики увидеть, использовать все полученные мною знания за 2 года моей учебы. Я укрепил свои знания в язык программирование python,php,MySQL ,Flesk Теперь я знаю, как создается сайты на языке python с исползованием фреймворка flask и django Во время прохождения практики я узнал многие пункты по информацтонных безопасности, которых я не знал. А также многие недостатки языка python и примущества. Эта практика научил меня ознакомить базы-данными и создание различный web сайты на разным языке html,php,Django,javascript,java Я стал ответственным, терпеливым и так думаю хорошим специалистом

**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