ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ЦИФРОВЫХ ОБРАЗОВАТЕЛЬНЫХ ТЕХНОЛОГИЙ

Кафедра «Информационные технологии»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к проведению лабораторных занятий
по дисциплине

«Межплатформенное программирование»

Авторы

Васильев П.В.,
Подколзина Л.А.

Ростов-на-Дону, 2023

Аннотация

Методические указания предназначены для студентов всех форм обучения направлений подготовки бакалавров 09.03.02 «Информационные системы и технологии».

Авторы

Старший преподаватель кафедры
«Информационные технологии» Васильев П.В.

Программист кафедры
«Информационные технологии» Подколзина Л.А.

Оглавление

ЛАБОРАТОРНАЯ РАБОТА №1: настройка среды и создание GIT репозитория

Данная лабораторная работа включает в себя:

- установку интерпретатора Python,
- подготовку и настройку среды для разработки на Python;
- добавление проекта в систему контроля версий;
- создание репозитория и выгрузку проекта на GitHub.com.

Теоретическая часть: краткий справочник

Интерпретатор Python [1,2] используется для обработки инструкций исходного кода сценариев на языке Python, последующей компиляции его в байт-код и выполнения на виртуальной машине. Любой интерпретатор представляет собой слой программной логики между программным кодом и аппаратной реализацией.

IDE (*Integrated development environment*) - *Интегрированная среда разработки* - представляет собой инструмент (комплекс программных средств), применяемый при разработке программного обеспечения. Служит для облегчения написания программного кода. Минимальный набор включает в себя текстовый редактор, компилятор и / или интерпретатор, средства для автоматизации сборки и отладчик. Также зачастую поддерживает работу с вспомогательными системами. Как правило, IDE загружает весь проект целиком, поэтому может предоставлять автодополнение по функциям всего проекта, удобную навигацию по его файлам и т.п. зачастую для правки кода используются легкие редакторы - открывают файл для редактирования практически мгновенно, более гибкие, однако, менее функциональны и для полноценной работы часто требуют установки дополнительных плагинов.

Для успешной работы над проектом как одного программиста, так и группы, необходимо использовать систему контроля версий. Она позволяет возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое.

В нашей работе мы будем использовать мощную распре-

деленную систему контроля версий - ***Git***. Понимание всех возможностей **git** открывает для разработчика новые горизонты в управлении исходным кодом. Ядро Git представляет собой набор утилит командной строки с параметрами. Все настройки хранятся в текстовых файлах конфигурации. Такая реализация делает Git легко портируемым на любую платформу и даёт возможностьлегко интегрировать Git в другие системы (в частности, создавать графические git-клиенты с любым желаемым интерфейсом). Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

1.1 Установка интерпретатора Python и среды разработки PyCharm (Для Windows)

Для установки интерпретатора языка Python необходимо скачать (https://www.python.org/downloads/) с официального сай-та установщик версии 3.5.2 (на данный момент самая последняя версия). В процессе установки выберите соответствующий пункт для того, чтобы путь к интерпретатору был добавлен в переменную среды PATH.

Для установки среды разработки PyCharm необходимо скачать последнюю версию с официального сайта (https://www.jetbrains.com/pycharm/download/) для своей операционной системы. Редакция Community является наиболее приемлемым вариантом.

1.2 Установка интерпретатора Python и среды разработки PyCharm (Для Linux)

В **Ubuntu** установлены по умолчанию обе актуальные версии интерпретаторов **Python**.

Для установки среды разработки ее необходимо скачать по адресу https://www.jetbrains.com/pycharm/download/ и выполнить следующие команды:

cd install - переход в папку с программой (папка в которую будет установлена программа)

ls - просмотреть содержимое папки;

tar xfz pycharm-community-2016.2.3.tar.gz - распаковка файла установки;

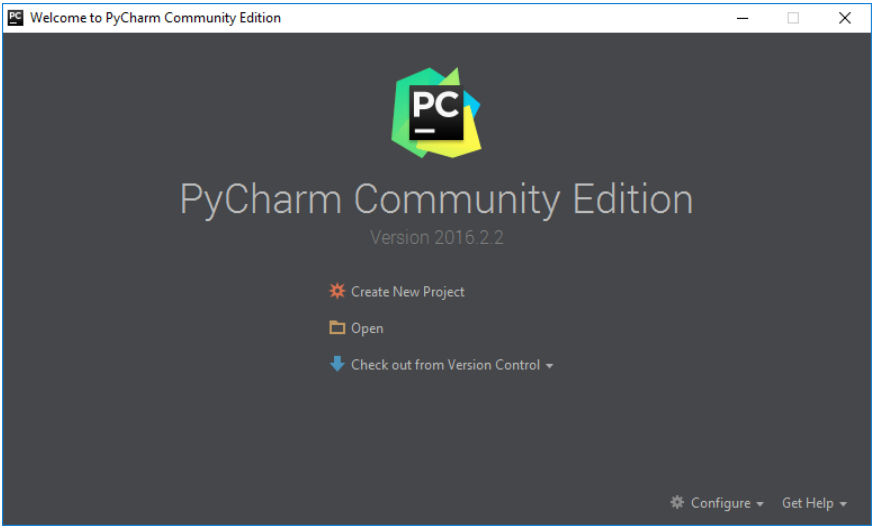
cd pycharm-2016.2.3/bin - переход в папку с файлом запуска;

sh pycharm.sh - запуск приложения;

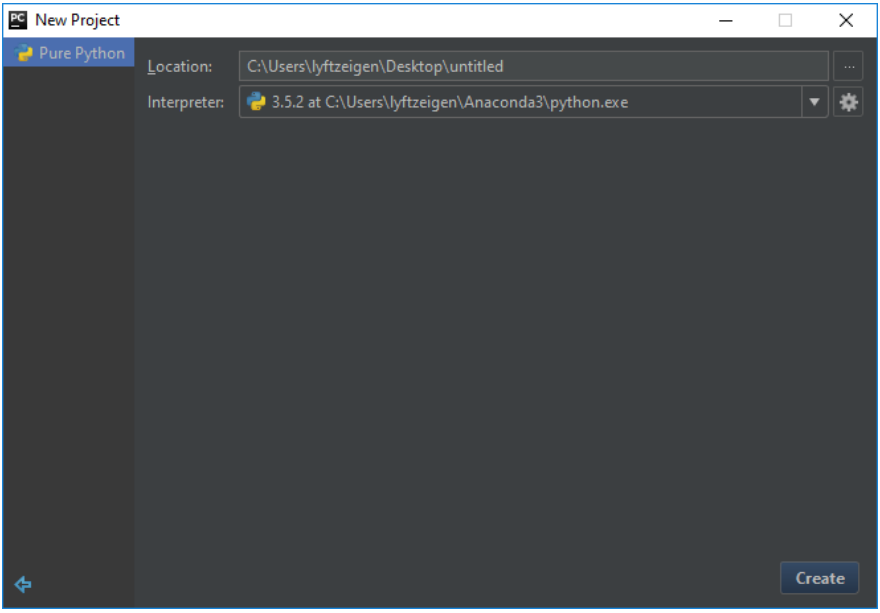
После первого запуска, среда создаст значок запуска в меню **dash**.

1.3 Создание проекта и его запуск

Для того, чтобы создать проект необходимо запустить среду **PyCharm** и выбрать пункт **Create New Project**.



В поле **Location** нужно указать путь к новому проекту, в поле **Interpreter** выбрать установленный интерпретатор языка Python и нажать **Create**.

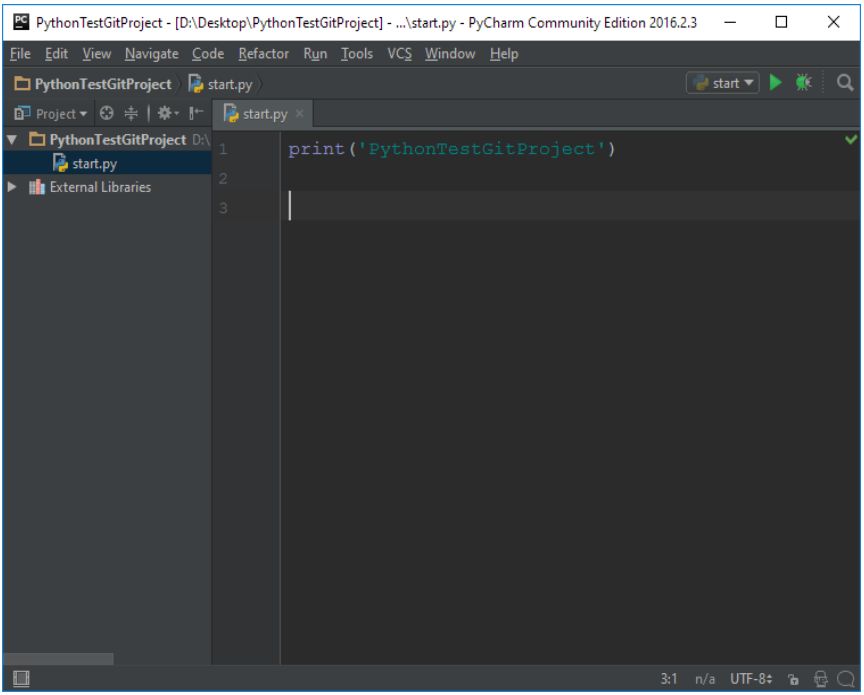


1.4 Добавление проекта в систему контроля версий

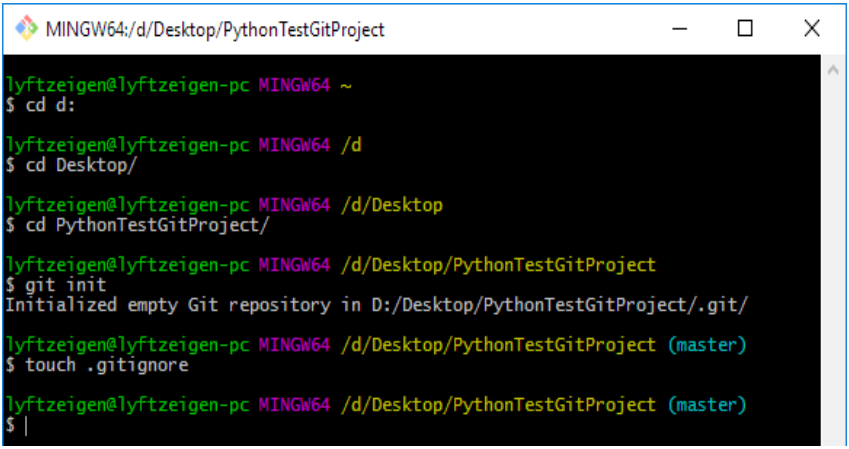
Для использования **git** его необходимо установить https://githowto.com/ru. Также необходимо создать аккаунт, например, **GitHub** или **Bitbucket**.

В **учебных целях** создадим проект **PyCharm** и добавим его в систему контроля версий **git**. Это можно выполнить с помо- щью самой **IDE** или командной строки **GIT Bash** (**Windows**).

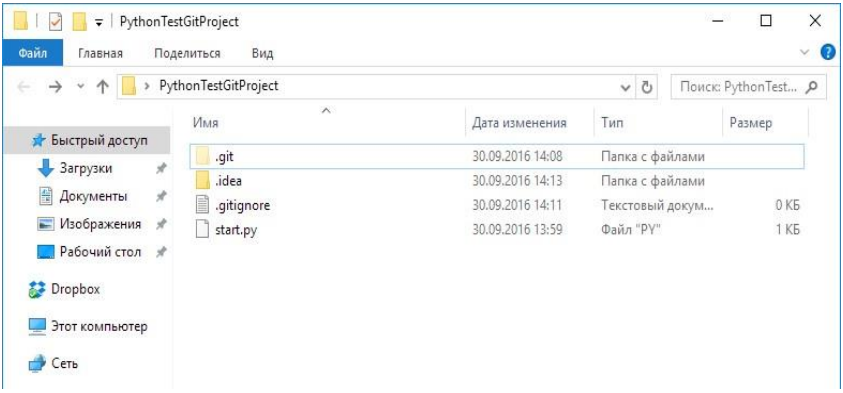
Создадим проект и назовет его **PythonTestGitProject**. Добавим в него файл **start.py** и выведем немного информации на экран.



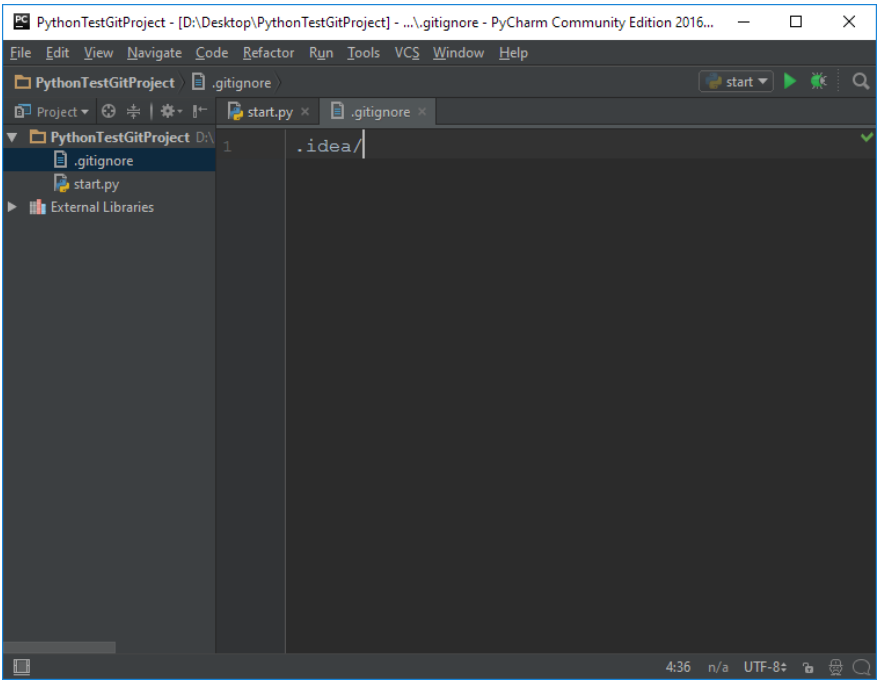
Откроем **GIT Bash** (для системы **Windows**) и перейдем в директорию проекта командой **cd.** Добавим файл **.gitignore** (командой **touch .gitignore**) в котором укажем файлы и папки, которые система **git** будет игнорировать. Затем добавим проект в систему контроля версий **git**, для этого выполним команду **git init** для инициализации системы.



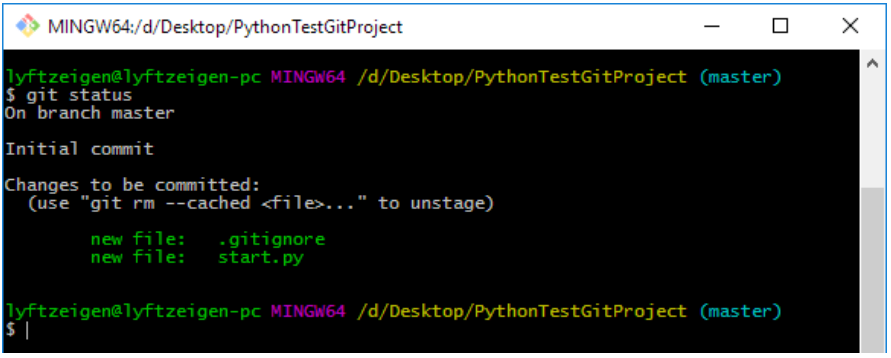
В папке проекта директория **.idea** предназначена для хранения настроек проекта **PyCharm**. Её необходимо исключить из системы контроля версий.



Для этого в файл **.gitignore** добавим строку **.idea/**.

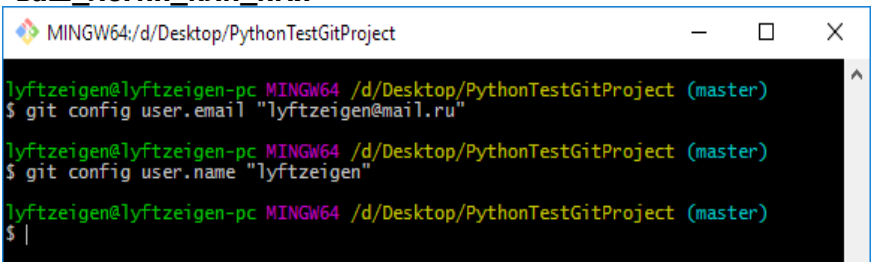


Добавим в систему контроля версий все файлы проекта командой «**git add .**» и выполним команду **git status** для получения информации о добавленных файлах.

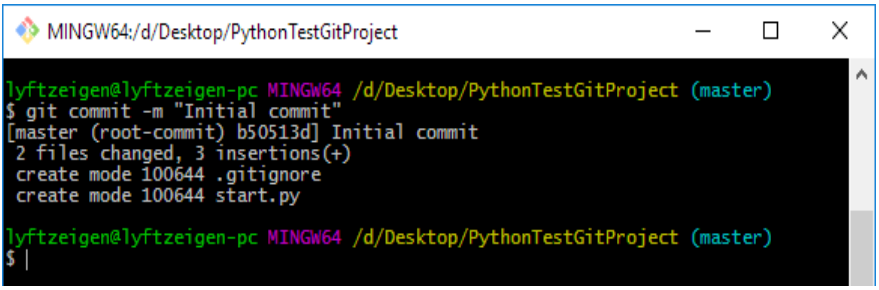


Добавим необходимую информацию о пользователе в систему **git** только для этого проекта командами **git config user.email "ваша_почта"** и **git config user.name**

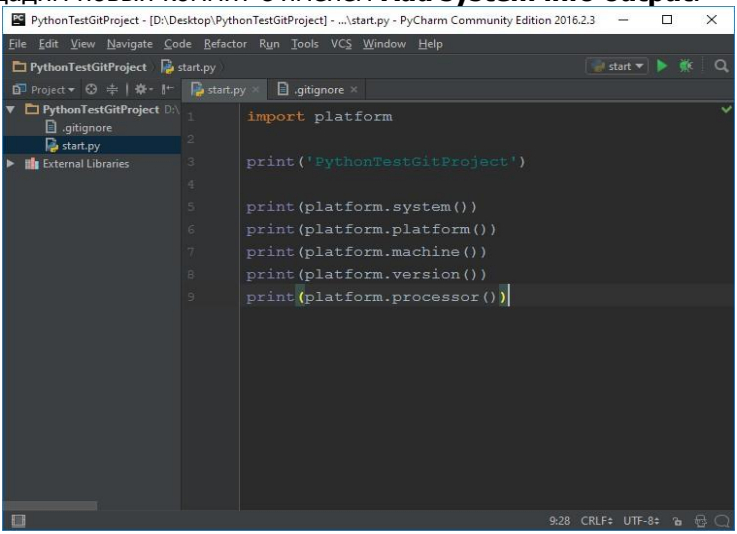
“ваш_логин_или_имя”



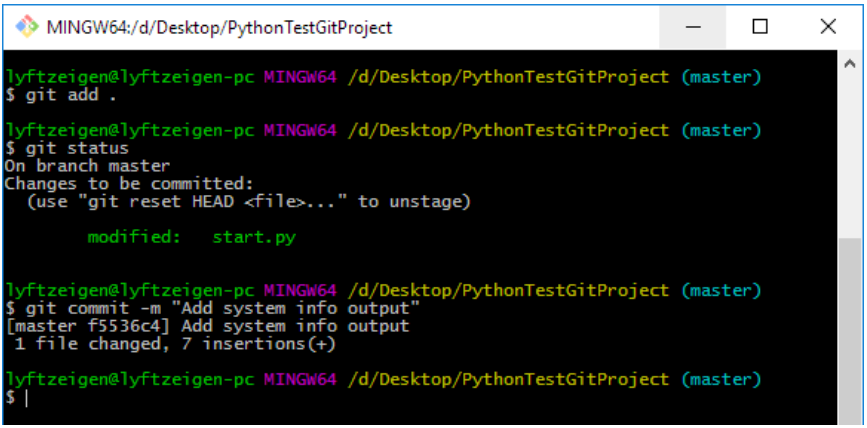
Теперь зафиксируем изменения командой **git commit** и назовем этот коммит как **Initial commit**.



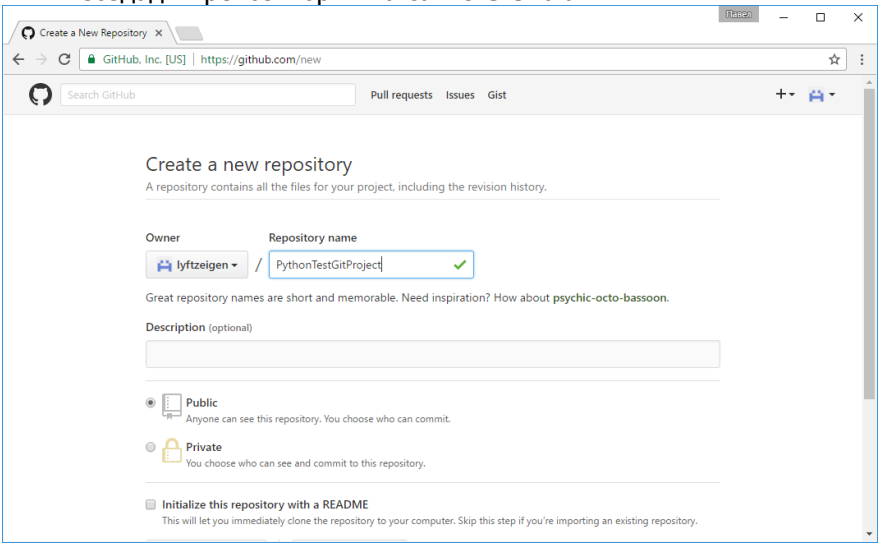
Добавим в файл **start.py** вывод информации о системе и создадим новый коммит с именем **Add system info output**.



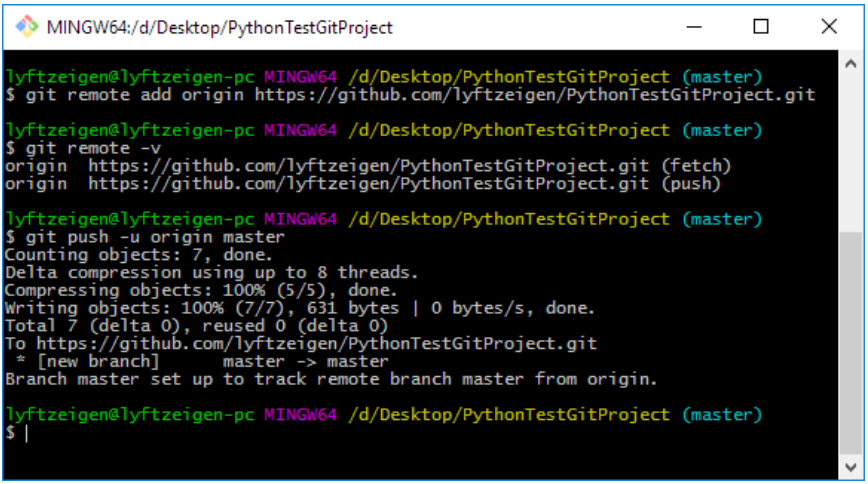
Аналогично выполним команды **git add .** и **git commit**.



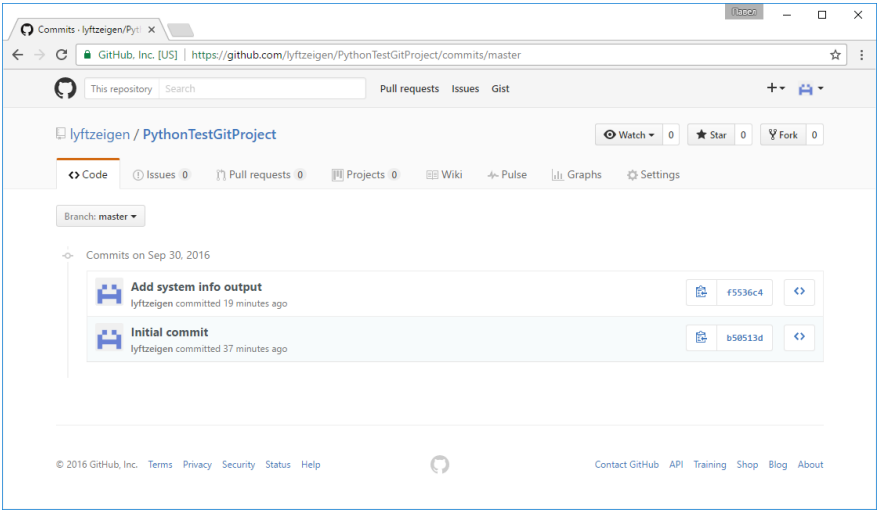
Создадим репозиторий на сайте **GitHub**.



Теперь добавим удаленный репозиторий командой **git remote add origin https://github.com/имя_пользователя/имя_репозитория.git** и выгрузим созданный проект на **GitHub** командой **git push -u origin master**.



Зайдем на GitHub и убедимся, что репозиторий содержит все созданные нами коммиты.



Каждую лабораторную работу из этого пособия необходимо добавлять в систему контроля версий и отдельный репозиторий.

Вопросы

1. Расскажите об альтернативных реализациях Python (Jython и IronPython).
2. Объясните, как работают средства оптимизации скорости выполнения
3. Дайте определение виртуальной машине Python (PVM)
4. Расскажите о процессе динамической компиляции и байт-коде
5. Проведите обзор систем контроля версий. Выделите преимущества и недостатки каждой из них
6. Объясните работу команд git (add, commit и clone)

Задания

Закрепите полученные знания, добавив в систему контроля версий и выложив на GitHub свой первый проект.

Лабораторная работа №2: OpenWeatherMap API

Данная лабораторная работа включает в себя изучение клиент-серверного взаимодействия на примере OpenWeatherMap API.

Теоретическая часть: краткий справочник

API - набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

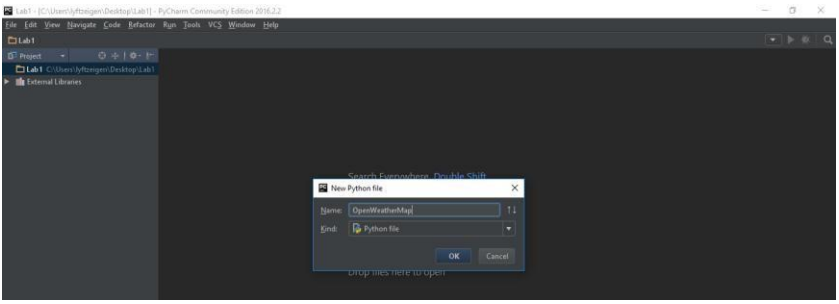
OpenWeatherMap предоставляет бесплатный API ко всем данным о погоде, к их истории, прогнозам и всему многообразию погодных карт. API представлено в двух видах — JSON для получения данных и Tile / WMS для построения картографии.

Устанавливаемые библиотеки:

PyOWM это библиотека Python, обертка для клиентской части OpenWeatherMap [3] (OWM) веб API. Позволяет легко и быстро поучить погодные данные с OWM в Python приложении с помощью простой объектной модели в дружелюбном формате. При выполнении данной лабораторной работы установки дополнительных библиотек не требуется кроме стандартных модулей. Документация https://pyowm.readthedocs.io/en/latest и исходный код библиотеки https://github.com/csparpa/pyowm .

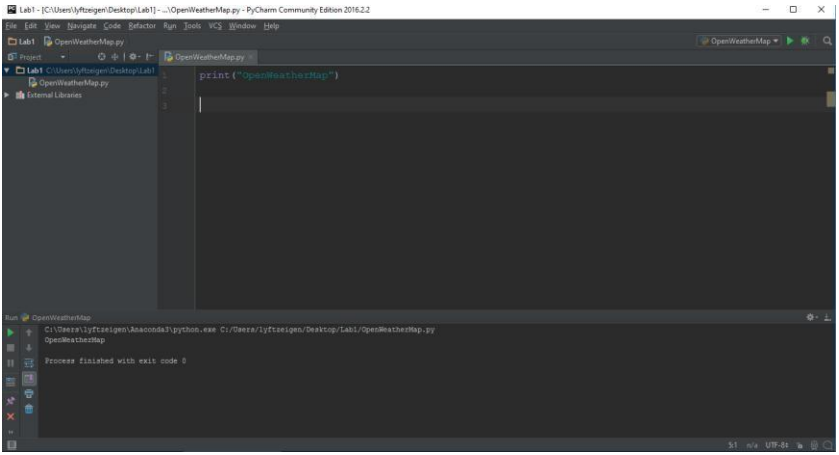
2.1 Создание проекта

Создадим проект и назовем его **Lab1.** Чтобы добавить новый файл, в котором будет находится программный код выполним **File -> New -> Python File** и назовем его **OpenWeatherMap**.



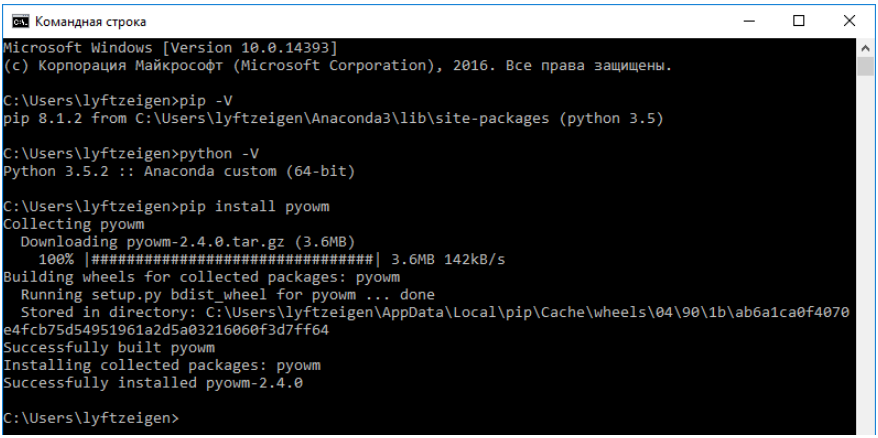
Для того, чтобы запустить файл с кодом выполним **Run -> Run... -> OpenWeatherMap**. Таким образом проект сконфигурируется так, что файл **OpenWeatherMap** будет запускаться по умолчанию (**Shift+F10**).

Выведем название нашей программы с помощью функции **print** и запустим проект. Если все было сделано правильно, по увидим примерно следующее.



2.2 Установка необходимых библиотек Python

Для того чтобы установить библиотеку Python нужно выполнить команду **pip install <название библиотеки>**. Установим **PyOWM** - специализированную библиотеку для работы на Python с OpenWeatherMap.



OpenWeatherMap API требует API ключ как для бесплатного использования, так и для платной подписки, чтобы настроить взаимодействие с сервисом. PyOWM не будет работать если такого ключа нет. Вы можете зарегистрироваться бесплатно и получить API ключ на сайте OWM (https://home.openweathermap.org/users/sign_up). Для того, чтобы API ключ начал работать должно пройти некоторое время после его генерации. В противном случае может возникнуть ошибка 401. Также, стоит отметить, что запросы к сервису на бесплатной основе могут выполняться с задержкой. После регистрации на сайте и получения API ключа нужно вызвать соответствующую функцию для получения погодных данных.

Подключим библиотеку к проекту командой **import pyowm.**

2.3 Получение метеоданных

Теперь попробуем получить погодные данные в **Ростове-на-Дону**.

```
owm = pyowm.OWM('                    fabee52fff0b767c')
```

Функция **OWM** из модуля **pyowm** служит точкой входа в библиотеку и инициализирует объект типа **pyowm.webapi25.owm25.OWM25**. Этот объект содержит функции для получения различных прогнозов, технической информации метеостанций, получения метеоданных по координатам

или городу. При вызове этой функции в качестве параметра передается секретный API ключ зарегистрированного пользователя. Таким образом в переменной **owm** будет хранится объект, через который можно получить доступ к метеоданным.

Теперь получим погоду в городе **Ростове-на-Дону** с помощь функции **weather_at_place**.

```
observation = owm.weather_at_place('Rostov-on-Don,ru')
```

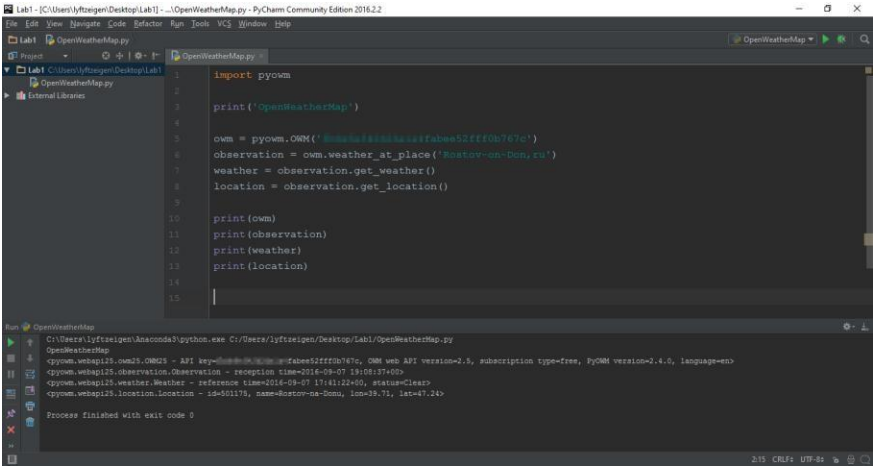
Эта функция принимает в качестве входного параметра название местоположения. В данном случае это **'Rostov-on-Don,ru'**. Таким образом в переменной **observation** будет хранится объект типа **pyowm.webapi25.observation.Observation**.

Этот тип данных представляет погоду, которая была получена в определенной точке на земле. Местоположение представлено инкапсулированным (встроенным) объектом типа **Location - pyowm.webapi25.location.Location**, в то время как полученные метеоданные хранятся в объекте типа **Weather - pyowm.webapi25.weather.Weather**.

Таким образом, чтобы получить метеоданные и координаты нужно вызвать функцию **get_weather** и **get_location**, соответственно. В переменные **weather** и **location** сохраним поученные данные.

```
weather = observation.get_weather()
location = observation.get_location()
```

Выведем информацию о всех этих объектах на экран с помощью функции **print**. Таким образом можно определить тип каждого объекта и его базовую информацию.



Теперь выведем на экран более детальную информацию о метеоданных и местоположении. Описание каждой функции можно посмотреть в документации в соответствующих разделах: **pyowm.webapi25.location** и **pyowm.webapi25.weather**.

```
print('Страна: ' + location.get_country())
print('Город: ' + location.get_name())
print('Долгота: ' + str(location.get_lon()))
print('Широта: ' + str(location.get_lat()))
print('Облачность: ' + str(weather.get_clouds()))
print('Статус: ' + str(weather.get_detailed_status()))
print('Давление: ' + str(weather.get_pressure()))
print('Дождь: ' + str(weather.get_rain()))
print('Снег: ' + str(weather.get_snow()))
print('Время: ' + str(weather.get_reference_time('iso')))
print('Статус: ' + str(weather.get_status()))
print('Восход: ' + str(weather.get_sunrise_time('iso')))
print('Закат: ' + str(weather.get_sunset_time('iso')))
print('Температура: ' + str(weather.get_temperature('celsius')))
print('Видимость: ' + str(weather.get_visibility_distance()))
print('Изображение: ' + weather.get_weather_icon_name())
print('Ветер: ' + str(weather.get_wind()))
```

И получим следующий результат:

```
C:\Users\lyftzeigen\Anaconda3\python.exe C:/Users/lyftzeigen/Desktop/Lab1/OpenWeatherMap.py
OpenWeatherMap
Страна: RU
Город: Rostov-na-Donu
Долгота: 39.71
Широта: 47.24
Облачность: 0
Статус: clear sky
Давление: {'sea_level': None, 'press': 1018}
Дождь: {}
Снег: {}
Время: 2016-09-07 19:00:00+00
Статус: Clear
Восход: 2016-09-07 02:49:26+00
Закат: 2016-09-07 15:47:15+00
Температура: {'temp': 20.0, 'temp_max': 20.0, 'temp_kf': None, 'temp_min': 20.0}
Видимость: None
Изображение: 01n
Ветер: {'speed': 3, 'deg': 360}

Process finished with exit code 0
```

В данном примере рассмотрены лишь базовые функции, необходимые для составления простого прогноза. Стоит отметить, что некоторые функции возвращают пустой результат или **None** в связи с отсутствием прогноза или недоступностью данных.

Ниже показан пример выводящий информацию о погоде в г. Ростове-на-Дону в дружелюбной форме.

```
# Подключаем библиотеку pyowm
import pyowm

# Выводим на экран название программы
print('OpenWeatherMap')

# Инициализируем библиотеку API ключом
owm = pyowm.OWM('[illegible]fabee52fff0b767c')

# Получаем данные из Ростова-на-Дону
observation = owm.weather_at_place('Rostov-on-Don,ru')

# Получаем погодные данные
weather = observation.get_weather()

# Получаем данные местоположения
location = observation.get_location()

# Создаем словарь для перевода названия городов
translate = {'Rostov-na-Donu': 'Ростов-на-Дону', 'Moscow': 'Москва'}
```

```
# Создадим функцию которая определяет облачность
def WhatIsCloudness():
    if 0 <= weather.get_clouds() <= 10:
        return 'ясная'

    if 10 < weather.get_clouds() <= 30:
        return 'немного облачная'

    if 30 < weather.get_clouds() <= 70:
        return 'пасмурная'

    if 70 < weather.get_clouds() <= 100:
        return 'мрачная'

# Выводим на экран данные в дружелюбном формате
print('Погода в городе ' + translate[location.get_name()] + ' ' +
      WhatIsCloudness() + ', температура ' +
      str(weather.get_temperature('celsius')['temp']) + ' градусов цельсия ' +
      'скорость ветра ' + str(weather.get_wind()['speed']) + ' м/с.')
```

В результате выполнения данного кода получим:

```
C:\Users\lyftzeigen\Anaconda3\python.exe D:/Desktop/Lab1/OWMExample.py
OpenWeatherMap
Погода в городе Ростов-на-Дону ясная, температура 20.74 градусов цельсия скорость ветра 3 м/с.

Process finished with exit code 0
```

Вопросы

1. Дайте определение и приведите основные особенности Web Map Service (WMS)
2. Опишите формат обмена данными JSON.
3. Расскажите о методах создания больших изображений, используемых в картографии (например, tile)
4. Объясните принцип работы OpenWeatherMap API. Опишите общую архитектуру системы
5. Данные каких метеослужб используются при работе с OpenWeatherMap API
6. Какие виды данных можно получать, используя OpenWeatherMap API
7. Приведите спектр применения данного API, укажите основные преимущества и недостатки

Задания

1. Создать функцию, которая генерирует погодный прогноз на основе данных, полученных с OpenWeatherMap.

Пример 1:

Погода в городе Москва (Россия) на сегодня в 10:15 солнеч- ная, облачность составляет 5%, давление 760 мм рт. ст., темпе- ратура 20 градусов Цельсия, ночью 8 днем 25 градусов Цельсия, ветер северо-западный, 5 м/с.

Пример 2:

Погода в городе Нижний Новгород (Россия) на сегодня в 17:10 пасмурная, облачность составляет 95%, давление 820 мм рт. ст., температура 6 градусов Цельсия, ночью -3 днем 15 градусов Цельсия, ветер северо-западный, 1 м/с, небольшой дождь.

2. Создать функцию, выводящую историческую справку по интересующему городу. В случае отсутствия такового, реализовать возможность изменения поискового запроса.

Дополнительное задание

Вывести карту облаков (облачность и фронты) в Централь- ной части России.

ЛАБОРАТОРНАЯ РАБОТА №3: формат JSON

Данная лабораторная работа включает в себя:

– Разработку приложения для визуализации геолокации фотографий своих друзей в социальной сети vk.com.

Теоретическая часть: краткий справочник

JSON - текстовый формат обмена данными, основанный на JavaScript [4]. За счёт своей лаконичности по сравнению с XML, формат JSON может быть более подходящим для сериализации сложных структур. Если говорить о веб-приложениях, в таком ключе он уместен в задачах обмена данными как между браузером и сервером (AJAX), так и между серверами (программные HTTP-сопряжения). Многие сервисы предоставляют свои услуги через JSON интерфейс, к примеру Flickr.com предоставляет API, с помощью которого можно работать с их данными. Наиболее популярными форматами обмена данными в сети интернет являются XML и JSON (расширения .xml и .json соответственно). Каждый имеет свои недостатки и преимущества.

Преимущества JSON:

• Простота использования;

• Упрощенный синтаксис обеспечивающий легкость прочтения кода.

Преимущества XML:

• XML поддерживается почти всеми языками программирования;

• XML расширяем.

В большинстве случаев, вы будете думать о JSON как альтернативе XML формату – по крайней мере в рамках веб приложений. Концепция AJAX, в оригинале использует XML для обмена данными между сервером и браузером, но в последние годы JSON стал более популярным для передачи AJAX данных.

Хотя XML это испытанная и хорошо тестированная технология, используемая множеством приложений, преимущества JSON формата в том, что он более компактен и прост в написании и чтении.

3.1 Структуры JSON

Массив - список упорядоченных значений. Массив в JSON обозначается квадратными скобками []. Массив в JSON может содержать неограниченное количество значений. Синтаксис:

[значение1, значение2, значениеN]

Объект может содержать неограниченное количество пар имя/значения. Каждое из значений само может являться объектом или массивом. Такие объекты или массивы называются вложенными. Синтаксис:

{имя1:значение1, имя2:значение2, имяN:значениеN}

```
{имя1:значение1,
 имя2:
   {имя2_1:значение2_1,имя2_2:значение2_2}
}
```

Литералы. Объекты и массивы в JSON являются конструкциями, а литералы непосредственно данными, которые группируются этими конструкциями. Литералы JSON:

Строка;
Число;
Логическое значение;
Значение null.
Синтаксис:

{имя1:"строка", имя2:13, имя3:true, имя4:false, имя5:null}

3.2 Основные правила создания JSON строк

JSON строка содержит как массив значений, так и объект (ассоциативный массив с парами имя/значение).

Массив должен быть обвернут в квадратные скобки, [и], может содержать список значений, которые отделяются через кому.

Объекты обворачиваются с помощью фигурных дужек, { и }, также содержат разделенные комой пары имя/значение.

Пары имя/значение состоят из имя поля (в двойных кавычках), после чего следует двоиточие (:) , после значение данного поля.

Значения в массиве или объекте могут быть:

- Числовые (целые или дробные с точкой)
- Строковые (обвернутые в двойные кавычки)
- Логические (true или false)
- Другие массивы (обвернутые в квадратные скобки [и])
- Другие объекты (обвернутые в фигурные дужки { и })
- Нулевое значение (null)

3.3 Пример сохранения данных в JSON

```
{
  "orderID": 43512,
  "UserName": "Kim Alimov",
  "shopperEmail": "kimal@google.com",
  "contents": [
    {
      "productID": 323-090-653,
      "productName": "Phone highscreen power five",
      "quantity": 1
    },
    {
      "productID": 560-876-082,
      "productName": "Battery LR6-AA-8",
      "quantity": 3
    }
  ],
  "orderCompleted": true
}
```

Код выше описывает в JSON-формате данные о покупке некоего Кима Алимова в интернет-магазине. Рассмотрим подробнее представленную информацию.

В начале и конце мы использует фигурные дужки { и }, которые дают понять, что это объект.

Внутри объекта мы имеем несколько пар имя/значение:

"orderID": 43512 – поле с именем orderId и значение 43512

"shopperName": " Kim Alimov " – поле с именем shopperName и значение Kim Alimov

"shopperEmail": " kimal@google.com " – подобно, как и в предыдущем поле, здесь храниться email покупателя.

"contents": [...] – поле с именем content, значение которо-

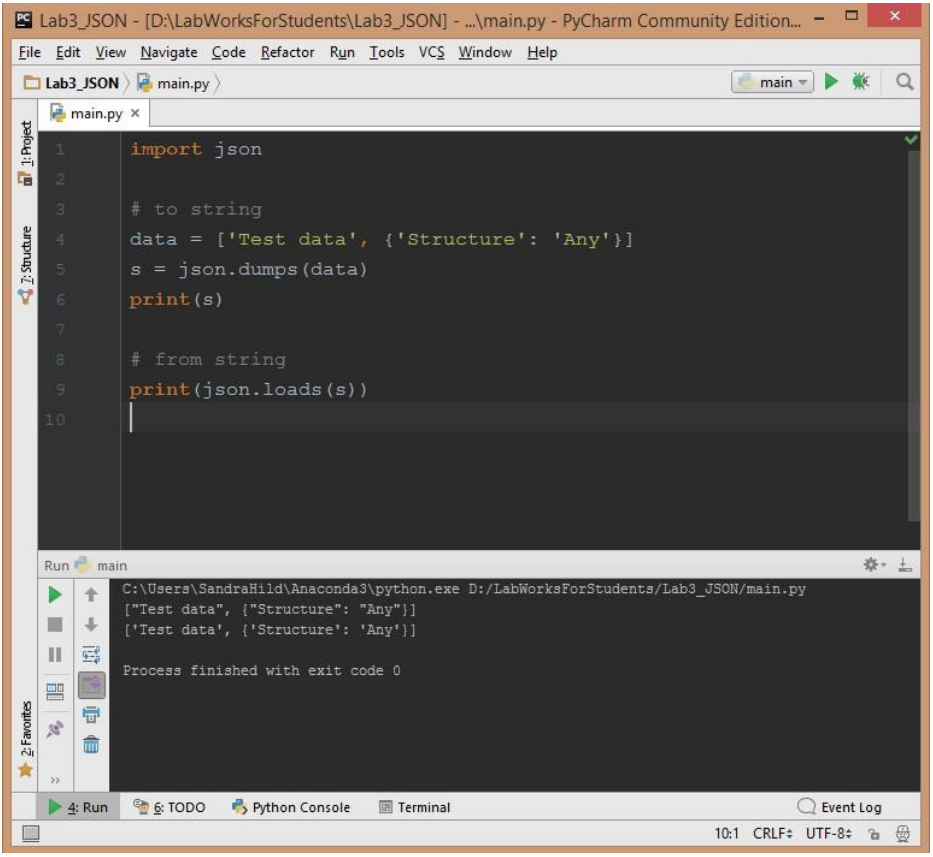
го массив.

"orderCompleted": true – поле с именем orderCompleted, значение которого true

Внутри массива contents, мы имеем два объекта, которые отображают содержимое корзины. Каждый объект продукта имеет три свойства: productID, productName, quantity.

3.4 Использование JSON-формата с Python

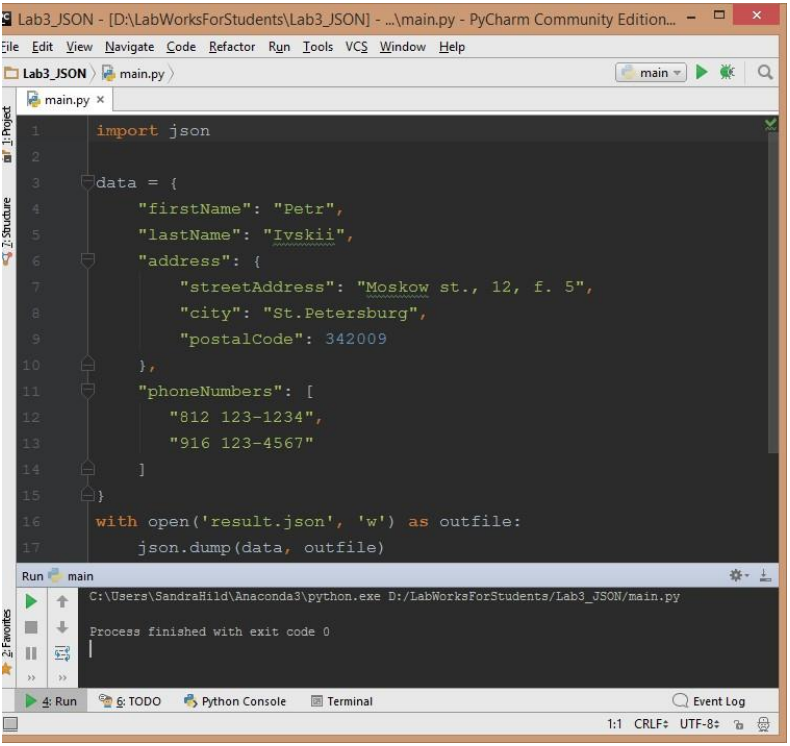
Для работы в Python необходимо подключить модуль JSON. Создадим JSON-строку из python-переменной и обратно.



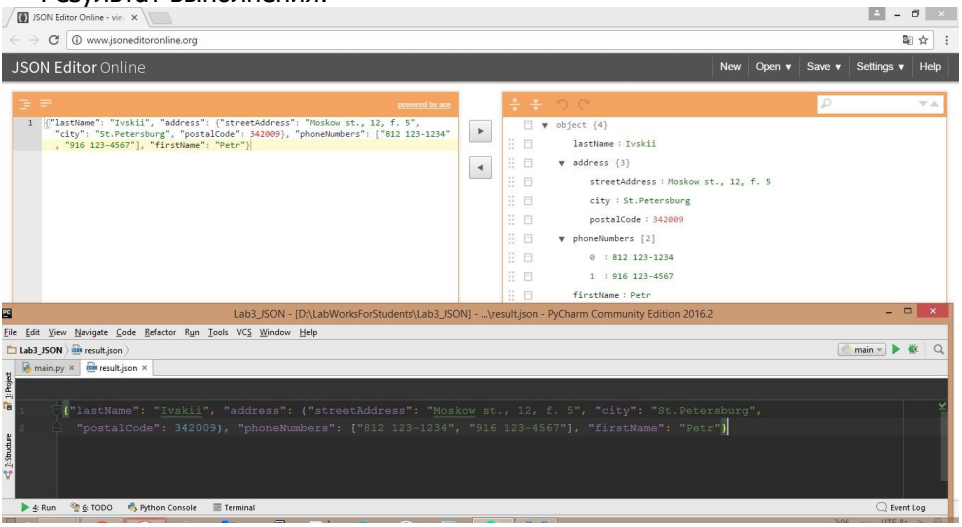
Теперь средствами python создадим JSON-файл следующего

содержания:

```
{
   "firstName": "Petr",
   "lastName": "Ivskii",
   "address": {
      "streetAddress": "Moskow st., 12, f. 5",
      "city": "St.Petersburg",
      "postalCode": 342009
   },
   "phoneNumbers": [
     "812 123-1234",
     "916 123-4567"
   ]
}
```



Результат выполнения:



Вопросы

1. Приведите пример отличия JSON и XML. Каковы их главные преимущества и недостатки
2. Назовите основные типы структур в JSON
3. Перечислите литералы JSON
4. Расскажите правила создания JSON-строк
5. Какие значения могут присутствовать в массиве или объекте JSON
6. Опишите принцип использования форматтеров на основе работы с сервисом http://codebeautify.org/jsonviewer

Задание

Создайте JSON-представление объекта, описывающего старосту Вашей группы. В объекте должны присутство- вать строковые поля имени, фамилии, отчества; объект, описывающий адрес; массив, содержащий список телефонов. Сохраните данные в формате JSON. Прочитайте созданные данные и измените их: добавьте информацию о своей группе (ФИО студента, адрес, телефон, контактные данные - страница vk.com и пр.). К каждому одногруппнику добавьте поле «Примечание» и заполните его информацией.

ЛАБОРАТОРНАЯ РАБОТА №4: клиент-серверное взаимодействие на примере VK API и Google Maps

Данная лабораторная работа включает в себя разработку приложения для визуализации геолокации фотографий своих друзей в социальной сети vk.com.

Теоретическая часть: краткий справочник

Устанавливаемые библиотеки:
vk 2.0.2
Это обертка API vk.com крупнейшей российской социальной сети. Цель данного проекта - это поддержка всех API методов (текущих и будущих) которые могут быть доступны на сервере.

Необходимая информация:

- описание методов https://vk.com/dev/methods,
- документация vk.readthedocs.io/en/latest/,
- исходный код библиотеки github.com/dimka665/vk,
- ссылка на Python.org https://pypi.python.org/pypi/vk.

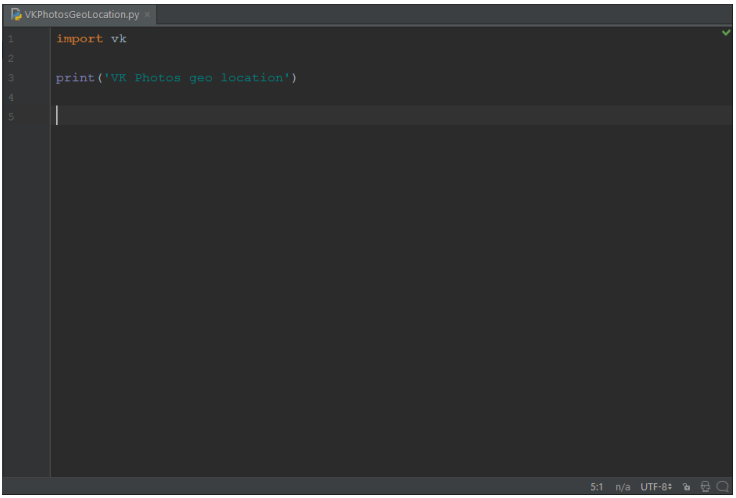
Встречающиеся определения:
Access_token — ключ доступа, своего рода «подпись» пользователя в Вашем приложении. Он сообщает серверу, от имени какого пользователя осуществляются запросы, и какие права доступа он выдал Вашему приложению.

4.1 Создание проекта

Создадим проект и назовем его **Lab4.** Чтобы добавить новый файл, в котором будет находится программный код выполним **File -> New -> Python File** и назовем его **VKPhotosGeoLocation**.

Для того чтобы запустить файл с кодом выполним **Run -> Run... -> VKPhotosGeoLocation**. Таким образом проект сконфигурируется так, что файл **VKPhotosGeoLocation** будет запускаться по умолчанию (**Shift+F10**).

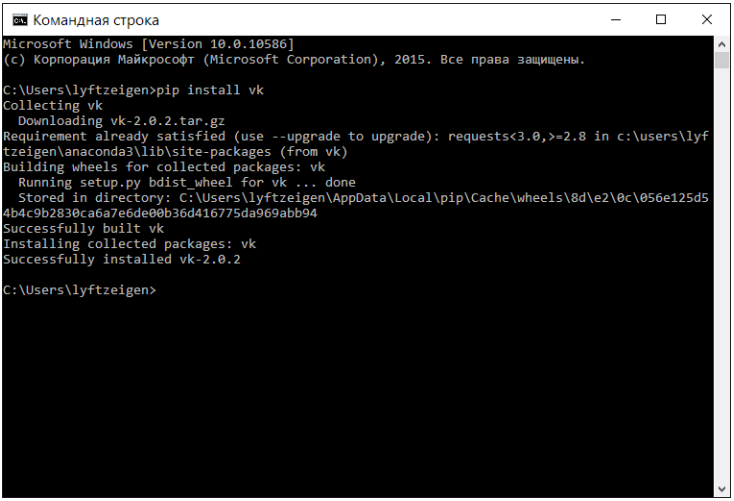
Подключим библиотеку командой **import** и выведем название нашей программы с помощью функции **print** и запустим проект. Если все выполнено правильно, то увидим примерно следующее



4.2 Установка библиотеки VK

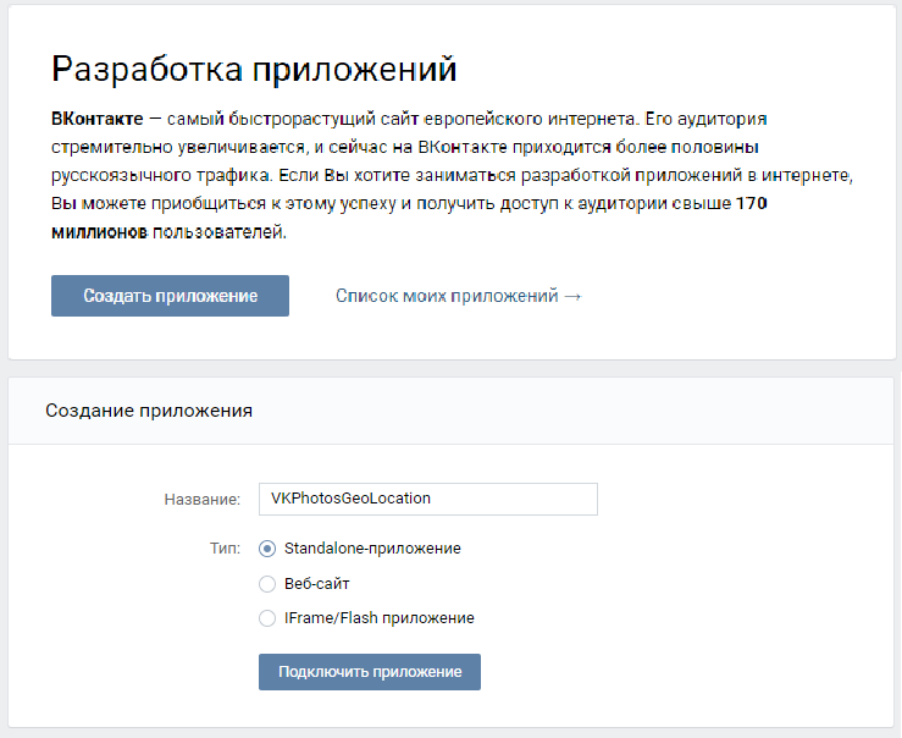
Для того чтобы установить библиотеку Python нужно выполнить команду:

pip install vk



4.3 Подготовка к разработке

Чтобы начать использовать vk.com API нужно зарегистрировать. Более детальную информацию можно получить по ссылке https://vk.com/dev/main.



1. Зайдите в соц. сеть.
2. Перейдите по ссылке https://vk.com/dev/standalone и создайте новое приложение. Выберите имя и тип **standalone**.
3. Запомните **id** приложения.
4. Используйте **id** приложения, список требуемых разрешений и реквизиты авторизации пользователя, чтобы получить **access token** (https://vk.com/dev/implicit_flow_user). Получить ключ доступа пользователя можно одним из этих способов:
5. Implicit flow. Для работы с API от имени пользователя в Javascript-приложениях и Standalone-клиентах (десктопных или мобильных).

6. Authorization code flow. Для работы с API от имени пользователя с серверной стороны Вашего сайта

1. Используйте этот access token чтобы вызывать необходимые методы. Весь список находится по ссылке https://vk.com/dev/methods.

Стоит отметить, что некоторые методы не требуют **access token**. Также можно не использовать **access token** для инициализации сессии и воспользоваться функцией **AuthSession**.

4.4 Получение списка друзей

Для получения списка друзей необходимо вызвать метод **friends.get** через проинициализированный объект **API**. Чтобы вывести на экран количество друзей нужно вызвать функцию **len**. Теперь выведем на экран количество друзей и их идентификаторы используя функцию **print**.

В результате должен получится примерно следующий код:

VKPhotosGeoLocation.py

```
import vk

print('VK Photos geo location')

# Авторизуем сессию с помощью access token
session = vk.Session('521ebcfb...')

# или с помощью id приложения и данных авторизации пользователя
# session = vk.AuthSession('app id', 'user login', 'user password')

# Создаем объект API
api = vk.API(session)

# Запрашиваем список всех друзей
friends = api.friends.get()

print(len(friends))
print(friends)
```

20:1 n/a UTF-8

4.5 Получение данных друзей, их альбомов и фотографий

Получим фамилию и имя всех друзей с помощью метода users.get и выведем этот список на экран с помощью цикла **for**.

```
# Получаем список всех друзей
friends = api.friends.get()

# Получаем информацию о всех друзьях
friends_info = api.users.get(user_ids=friends)

# Выведем список друзей в удобном виде
for friend in friends_info:
    print('ID: %s Имя: %s %s' % (friend['uid'], friend['last_name'], friend['first_name']))
```

Теперь получим все альбомы каждого пользователя и все фотографии из каждого альбома, а точнее их геоданные, если такие имеются. Для этого пройдемся по всем друзьям с помощью цикла **for** и вызовем метод **photos.getAlbums**. Затем из каждого альбома получим все его фотографии методом **photos.get**. Некоторые фото могут быть защищены настройками приватности, поэтому, чтобы избежать возникновения ошибок задействуем обработчик ошибок **try**. Для каждой полученной фотографии попробуем найти геоданные, которые могут находится в полях **lat** и **long**.

Необходимо после каждого запроса немного подождать, чтобы не возникла ошибка слишком частого обращения к серверу. Для этого используем стандартную функцию Python **sleep** из модуля **time**, который подключим командой import **time**.

В результате получаем следующий код:

```
# Здесь будут храниться геоданные
geolocation = []

# Получим геоданные всех фотографий каждого друга
# Цикл перебирающий всех друзей
for id in friends:
    print('Получаем данные пользователя: %s' % id)
    # Получаем все альбомы пользователя, кроме служебных
    albums = api.photos.getAlbums(owner_id=id)
    print('\t...альбомов %s...' % len(albums))
    # Цикл перебирающий все альбомы пользователя
    for album in albums:
        # Обрабатываем исключение для приватных альбомов/фото
        try:
            # Получаем все фотографии из альбома
            photos = api.photos.get(owner_id=id, album_id=album['aid'])
            print('\t\t...обрабатываем фотографии альбома...')
            # Цикл перебирающий все фото в альбоме
            for photo in photos:
                # Если в фото имеются геоданные, то добавим их в список geolocation
                if 'lat' in photo and 'long' in photo:
                    geolocation.append((photo['lat'], photo['long']))
            print('\t\t...найдено %s фото...' % len(photos))
        except:
            pass
        # Задержка между запросами photos.get
        time.sleep(0.5)
    # Задержка между запросами photos.getAlbums
    time.sleep(0.5)
```

4.6 Визуализация данных с применением GoogleMaps

Приступим к визуализации данных с помощью **GoogleMaps**. Этот способ гарантирует высокую интерактивность и простоту реализации. Создадим в папке проекта **html** файл **map.html** (http://pastebin.com/hD34HBUV) в котором будет нахо- дится разметка с картой **GoogleMaps** во весь экран. С помощью **Python** сгенерируем **JavaScript** код, добавляющий маркеры по- зиций на карту. Затем откроем получившийся файл в браузере.

Для работы с картами необходимо получить API ключ, что можно сделать согласно официальной документации https://developers.google.com/maps/documentation/javascript/get-api-key?hl=ru. Далее, добавьте его в **html** файл, для корректной работы с картами. Добавим **placeholder** - метку, в **JavaScript** коде, ориентируясь по которой будем добавлять маркеры с дан- ными.

Теперь напишем программу, которая генерирует **JavaScript** код добавления маркеров и вставляет этот ков вместо placeholder в html файл.

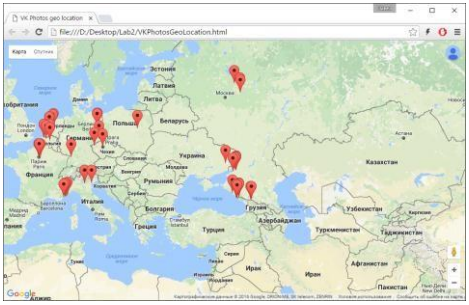
```
# Здесь будет хранится сгенерированый JavaScript код
js_code = ""

# Проходим по всем геоданным и генерируем JS команду добавления маркера
for loc in geolocation:
    js_code += 'new google.maps.Marker({ position: {lat: %s, lng: %s}, map: map });\n' % (loc[0], loc[1])

# Считываем из файла-шаблона html данные
html = open('map.html').read()
# Заменяем placeholder на сгенерированый код
html = html.replace('/* PLACEHOLDER */', js_code)

# Записываем данные в новый файл
f = open('VKPhotosGeoLocation.html', 'w')
f.write(html)
f.close()
```

Откройте VKPhotosGeoLocation.html в браузере и увидеть примерно **такую картинку (все зависит от ваших друзей)**



Вопросы

1. Какими способами можно получить ключ доступа пользователя для приложений vk
2. Опишите основные особенности протокола QAuth
3. Перечислите основные объекты, используемые vk API
4. Перечислите объекты, входящие в медиаконтент и вложения
5. Перечислите вспомогательные объекты и наборы значений
6. Перечислите права доступа для токена пользователя и токена сообщества
7. Опишите процесс получения ключей доступа для: пользователя, сообщества, сервиса

Задания

Проведите визуализацию геоданных фотографий друзей для своей учетной записи VK

ЛАБОРАТОРНАЯ РАБОТА №5: разработка бота с дальнейшим внедрением его на VPS сервер

Теоретическая часть: краткий справочник

VPS / VDS - виртуальный выделенный сервер, запущенный на базе физического сервера.

SSH — это специальный сетевой протокол, позволяющий получать удаленный доступ к компьютеру с большой степенью безопасности соединении.

5.1 Разработка VK-бота

Создадим проект **PyCharm** и назовем его Lab5. Файл с исходным кодом назовем VKBot.py.

Алгоритм работы программы будет заключаться в следующем:

1. Инициализация сессии VK API.
2. Получение последних входящих сообщений.
3. Выбор непрочитанных сообщений-команд.
4. Ответ на каждое сообщение-команду.
5. Установка статуса сообщений-команд как «Прочитанных»
6. Выполнение пункта 2.

При получении access-токена (https://vk.com/dev/implicit_flow_user) и дальнейшей работе с сообщениями необходимо установить соответствующие разрешения (https://vk.com/dev/permissions).

Результат реализации данного алгоритма:

```
import vk
import time
import datetime

print('VKBot')

# Авторизуем сессию с помощью access токена
session = vk.Session('...eea9fbd03deae469e9ee1...')

# Создаем объект API
api = vk.API(session)
```

```
while(True):
    # Получим 20 последних входящих сообщений
    messages = api.messages.get()

    # Создадим список поддерживаемых комманд
    commands = ['help', 'weather']

    # Найдем среди них непрочитанные сообщения с поддерживаемыми командами
    # таким образом получим список в формате [(id_пользователя, id_сообщения, команда), ...]
    messages = [(m['uid'], m['mid'], m['body'])
                for m in messages[1:] if m['body'] in commands and m['read_state'] == 0]

    # Отвечаем на полученные команды
    for m in messages:
        user_id = m[0]
        message_id = m[1]
        comand = m[2]

        # Сформируем строку с датой и временем сервера
        date_time_string = datetime.datetime.now().strftime('[%Y-%m-%d %H:%M:%S]')

        if comand == 'help':
            api.messages.send(user_id=user_id,
                              message=date_time_string + '\n>VKBot v0.1\n>Разработал: lyftzeigen')

        if comand == 'weather':
            api.messages.send(user_id=user_id,
                              message=date_time_string + '\n>Погода отличная!')

    # Формируем список id всех сообщений с командами через запятую
    ids = ', '.join([str(m[1]) for m in messages])

    # Помечаем полученные сообщения как прочитанные
    if ids:
        api.messages.markAsRead(message_ids=ids)

    # Проверяем сообщения каждые 3 секунды
    time.sleep(3)
```

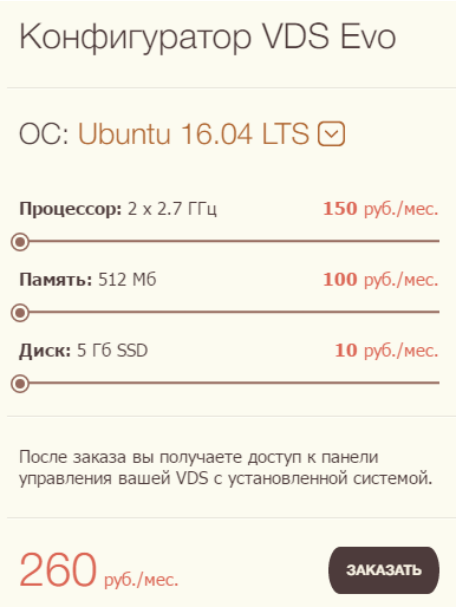
Таким образом, разработан простой VK-бот, способный отвечать на сообщения пользователей.

5.2 Развертывание VPS/VDS сервера

В учебных целях воспользуемся бесплатным (на момент написания данного руководства) сервисом от **Amazon** https://aws.amazon.com/ru/. Сервис позволяет получить бесплатный доступ к **VPS** серверу на срок в один год. Если возникнут трудности с регистрацией, то в сети можно найти руководство.

В данном пособии рассмотрено взаимодействие с **платным VDS** хостингом http://timeweb.com/ru/ со следующей (самой про-

стой) конфигурацией сервера:



После регистрации и оплаты хостинга, **в нашем случае**, получим доступ к панели управления сервером и доступ к **SSH**.

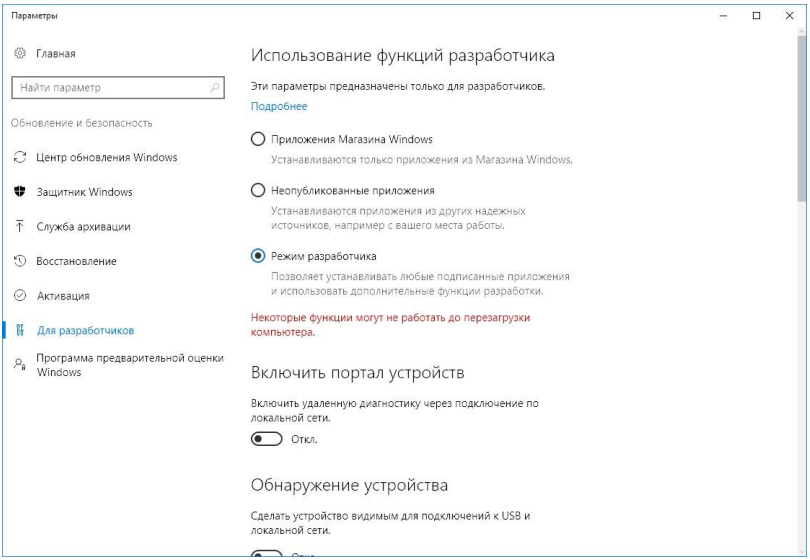
Для подключения по SSH к серверу воспользуемся программой PuTTY для Windows или терминалом в Ubuntu.

---

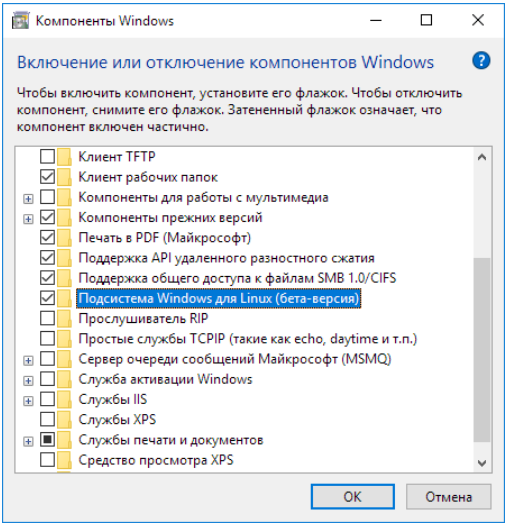
*Также стоит отметить, что с обновлением **Windows 10 Anniversary Update** стала доступна возможность использовать подсистему **Linux** на базе **Ubuntu 14.04.4 LTS** (на момент создания данного руководства).*

5.3 Включение подсистемы Windows для Linux (Windows 10 Anniversary Update)

Для активации данной подсистемы нужно включить «**Режим разработчика**»



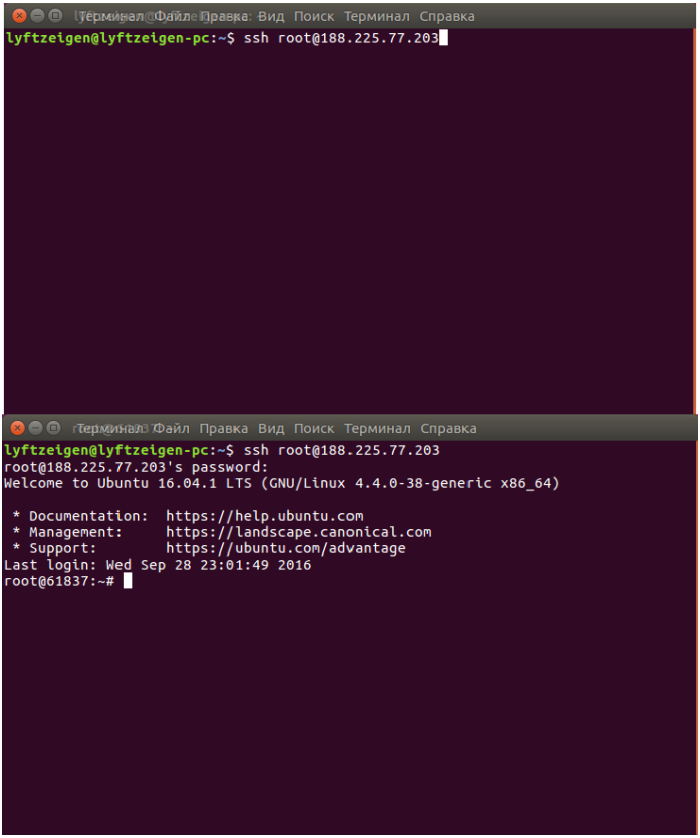
И включить компонент подсистемы Windows для Linux:



Затем выполните команду bash, после чего произойдет получение и установка необходимых компонентов с дальнейшей их конфигурацией.

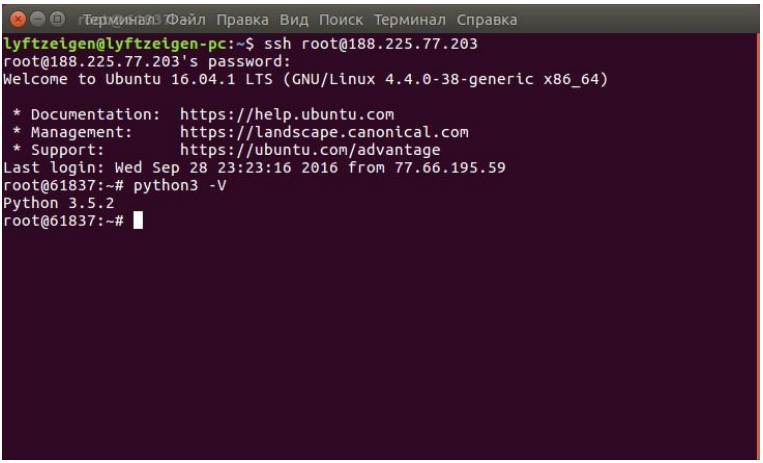
5.4 Подключение к VPS/VDS серверу по SSH и его настройка

Подключимся, например, с помощью терминала Ubuntu командой ssh. Вводим логин пользователя и адрес сервера, а затем и пароль:

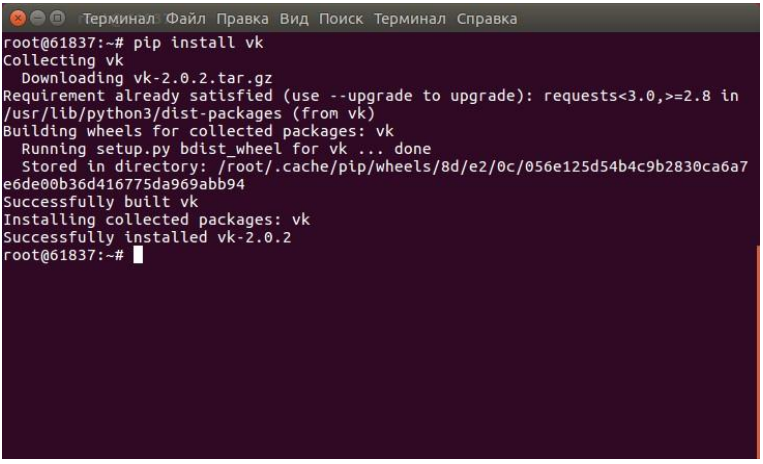


После того, как подключение прошло успешно, нужно настроить среду на виртуальном сервере согласно поставленным задачам.

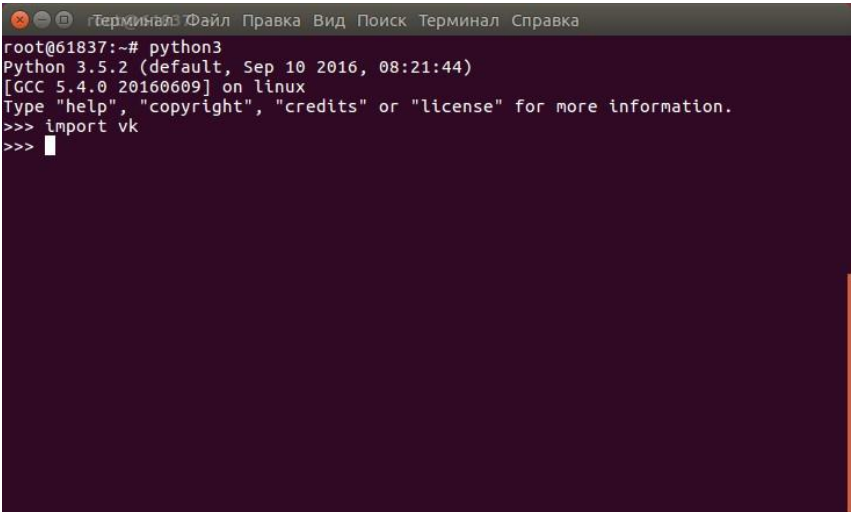
Проверим версию Python и установим утилиту pip, если это необходимо:



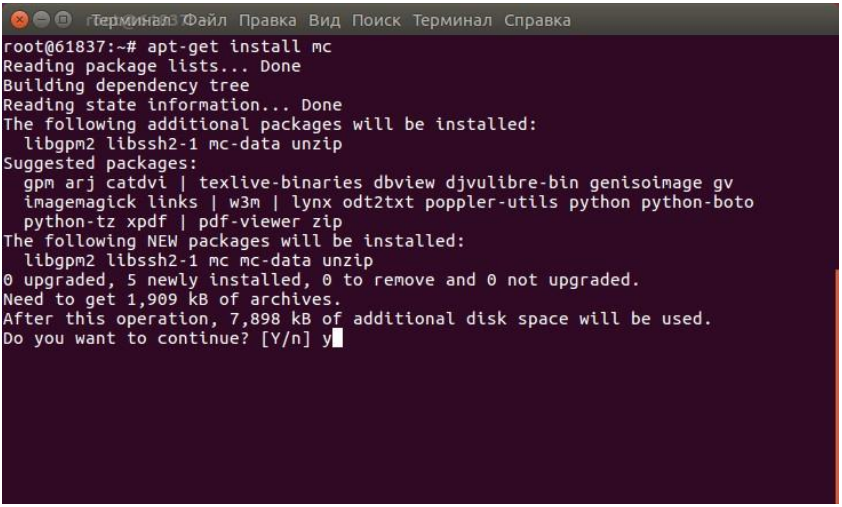
Установим модуль для работы с VK API:

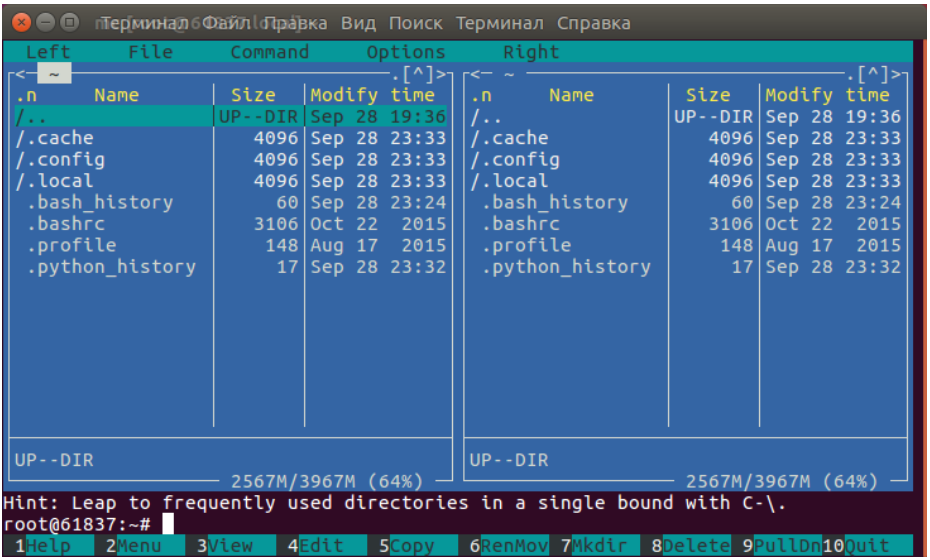


Проверим работу данного модуля:

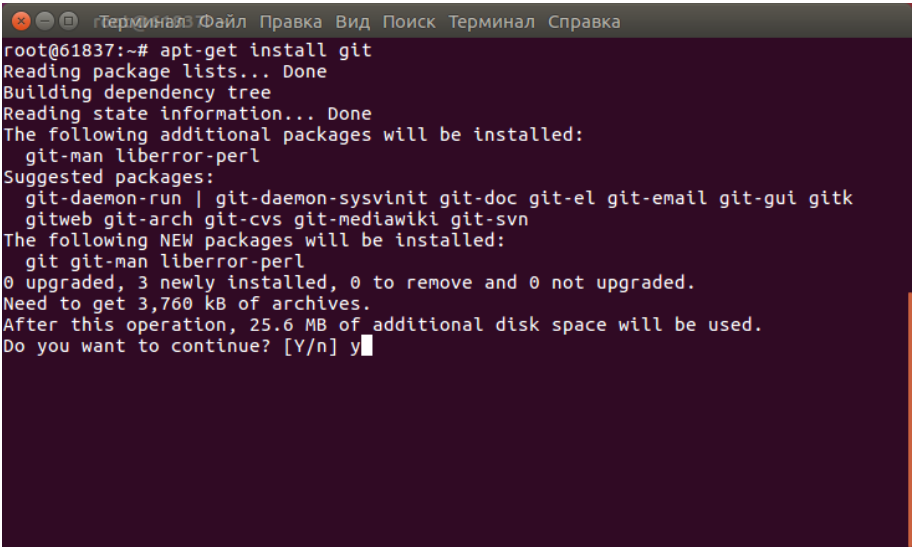


Если возникают сложности при работе с файловой системой, то можно установить файловый менеджер Midnight Commander командой apt-get install mc, а затем запустить его командой mc.



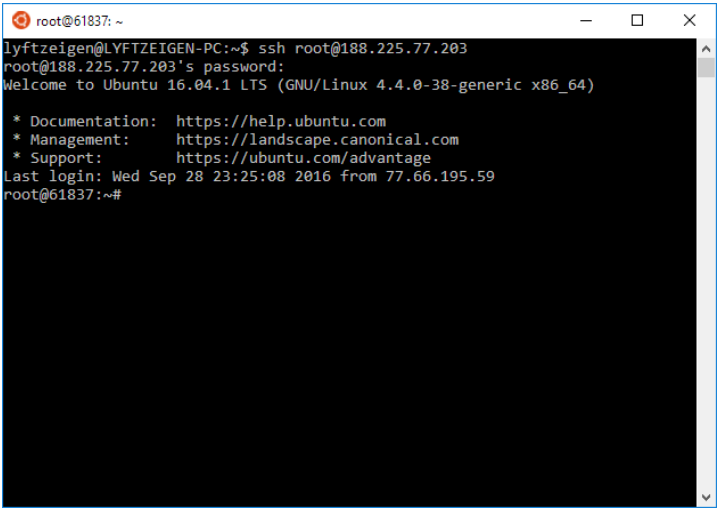


Также для дальнейшей работы понадобится программа git. Ее следует установить командой apt-get install git.

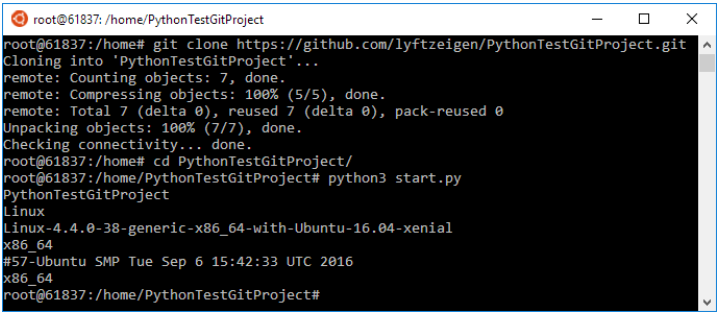


5.5 Загрузка проекта на VPS/VDS сервер из репозитория и его запуск

Подключимся по **SSH** к удаленному серверу, используя ранее полученный логин и пароль. В этот раз я буду использовать для подключения командную строку подсистемы Windows для Linux «**Bash на Ubuntu на Windows**».



Загрузим проект из репозитория на удаленный сервер командой git clone в папку /home и запустим его командой python3.



Вопросы

1. Расскажите о принципе и особенностях работы виртуального выделенного сервера
2. Сравните VDS/VPS с виртуальным(shared) хостингом
3. Сравните с физическим выделенным сервером
4. Виды виртуализации сервера
5. Расскажите о реализации SSH в виде двух приложений SSH-сервера и SSH-клиента.

Задание

1. Создать программу VK-бота, который будет отвечать на сообщения пользователей и выдавать информацию о погоде в его городе.
2. Загрузить созданный проект в репозиторий GitHub или любой другой.
3. Получить доступ к VPS/VDS серверу и настроить его.
4. Загрузить проект из репозитория на удаленный сервер и запустить его.
5. Опробовать разработанного VK-бота.

ЛАБОРАТОРНАЯ РАБОТА №6: СОЗДАНИЕ TELEGRAM-БОТА

В данной лабораторной работе вы разработаете приложение-бот с применением Telegram API. Данный бот осуществляет поиск изображений и вывод полученных результатов в диалог с пользователем.

Теоретическая часть: краткий справочник

Telegram Bot API - представляет из себя HTTP-интерфейс для работы с ботами в Telegram. Для успешной разработки вам потребуется авторизовать бот. Каждому боту при создании присваивается уникальный токен вида 123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11. В документации для простоты используется название <token>.

Библиотека PIL (Python Imaging Library) - библиотека языка Python (версии 2), предназначенная для работы с растровой графикой.

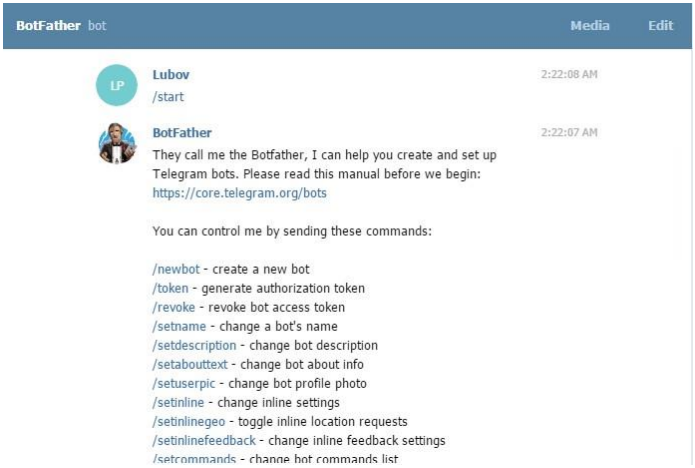
Beautiful Soup - это парсер для синтаксического разбора файлов HTML/XML, написанный на языке программирования Python, который может преобразовать даже неправильную разметку в дерево синтаксического разбора. Он поддерживает простые и естественные способы навигации, поиска и модификации дерева синтаксического разбора.

Telebot (pyTelegramBotAPI) - обертка над telegram API для языка Python

6.1 Предварительная подготовка

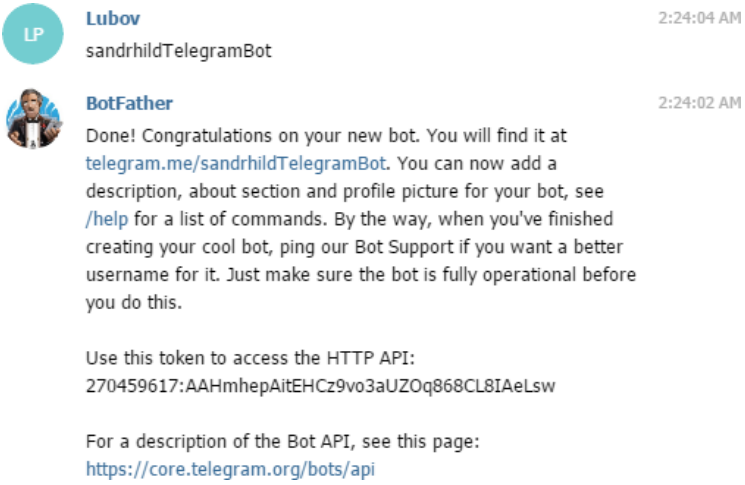
Чтобы наш бот функционировал, необходимо его создать. Для этого зайдите на сайт telegram.com и через поиск в вашем клиенте телеграмма добавьте мета-бот @BotFather командой /start.

Список команд мета-бота можно получить, написав в чате с ним команду /help. Для создания нового бота, выберете среди доступных команду /newbot



В следующем сообщении передайте название вашего бота (оно должно заканчиваться словом bot или _bot). В данном случае мы назвали его sandrhildTelegramBot. При необходимости повторите введенное название.

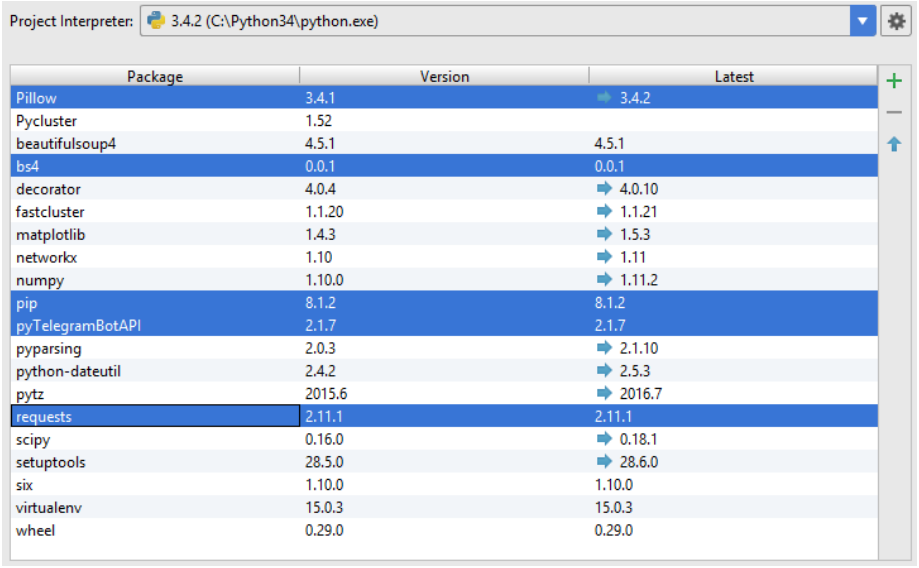
После успешного выполнения вам придет сообщение с API токеном, который помогает осуществлять соединение с ботом.



6.2 Создание проекта м подключение библиотек

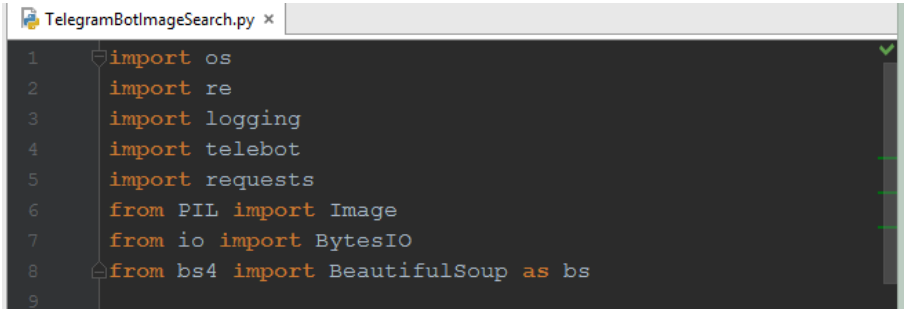
Создадим проект в PyCharm и назовем его Lab4, аналогично предыдущим заданиям. Файл с исходным кодом назовем TelegramBotImageSearch.py.

Для успешного выполнения работы вам потребуется установка следующих библиотек Settings-> Project Interpreter:



Package	Version	Latest
Pillow	3.4.1	3.4.2
Pycluster	1.52	
beautifulsoup4	4.5.1	4.5.1
bs4	0.0.1	0.0.1
decorator	4.0.4	4.0.10
fastcluster	1.1.20	1.1.21
matplotlib	1.4.3	1.5.3
networkx	1.10	1.11
numpy	1.10.0	1.11.2
pip	8.1.2	8.1.2
pyTelegramBotAPI	2.1.7	2.1.7
pyparsing	2.0.3	2.1.10
python-dateutil	2.4.2	2.5.3
pytz	2015.6	2016.7
requests	2.11.1	2.11.1
scipy	0.16.0	0.18.1
setuptools	28.5.0	28.6.0
six	1.10.0	1.10.0
virtualenv	15.0.3	15.0.3
wheel	0.29.0	0.29.0

Далее, перейдем к файлу TelegramBotImageSearch.py. Добавим импорт библиотек:



Создадим переменную bot, в которой будет храниться токен

для подключения к созданному ранее боту

```
bot = telebot.TeleBot('                                                    ')
```

6.3 Обработка поступающих сообщений

Теперь научим бота обрабатывать поступающее сообщение от пользователя с командой для запуска бота - /start. Для этого напишем обработчик, реагирующий на команду “start”:

```
@bot.message_handler(commands=['start'])
def SendInfo(message):
    bot.send_message(message.chat.id, 'Привет! я твой бот. Введи запрос для поиска изображения:')
```

Для получения справочной информации, выдаваемой пользователям по реализованным командам, напишем функцию, обрабатывающую команду “help”:

```
@bot.message_handler(commands=['help'])
def SendHelp(message):
    bot.send_message(message.chat.id, "Список доступных команд: /start, /about ")
```

Для осуществления поиска и вывода найденных результатов необходимо установить обработчик, реагирующий на все текстовые сообщения. В данной функции SendMessage пользователю выводится сообщение, оповещающее о запуске процесса поиска и управление передается в функцию SearchGoogleImages, с параметрами - сообщение для поиска и id-чата, из которого пришел запрос. В цикле for происходит вывод пользователю найденных изображений функцией SearchGoogleImages.

```
@bot.message_handler(content_types='text')
def SendMessage(message):
    bot.send_message(message.chat.id, "Ожидайте, мы ищем для Вас картиночки!")
    images = SearchGoogleImages(message.text, message.chat.id)
    for image in images:
        bot.send_photo(message.chat.id, open(image, 'rb'))
```

Основная работа по поиску изображений происходит в функции SearchGoogleImages. Сначала создадим папку по id-пользователя, в которой будем хранить найденные изображения

для конкретного пользователя.

```
def SearchGoogleImages(query, id):
    # Создаем папку по id-пользователя
    path = os.path.abspath(os.curdir)
    path = os.path.join(path, str(id))

    if not os.path.exists(path):
        os.makedirs(path)
```

Создадим запрос к поисковой системе. Для этого удалим лишние пробелы, заменим оставшиеся на символ «+», создадим строку поискового запроса.

```
35          # Запрос поисковой системы
36          query = query.split()
37          query = '+'.join(query)
38          query = 'https://www.google.ru/search?' \
39                  'q=' + query + \
40                  '&newwindow=1' \
41                  '&source=lnms' \
42                  '&tbm=isch'
43
```

Выполним сам запрос

```
    # Производим запрос
    req = requests.get(query, headers={
        'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) '
                      'Chrome/43.0.2357.134 Safari/537.36'})
    soup = bs(req.content, "html.parser")

    # Выбираем только img с data-src
    images = soup.find_all('img', {'data-src': re.compile('gstatic.com')})
```

Создадим массив imagePaths, в который будем добавлять пути найденных и сохраненных изображений в количестве 10 элементов. Названия будем писать как путь_к_изображению + номер + расширение_файла

```
    # Массив для хранения путей к найденным изображениям
    imagePaths = []

    # Загружаем первые 10 изображений и сохраняем их
    for number, tag in enumerate(images[:10]):
        data = requests.get(tag['data-src'])
        image = Image.open(BytesIO(data.content))
        imagePath = os.path.join(path, str(number) + '.' + image.format.lower())
        image.save(imagePath)
        imagePaths.append(imagePath)

    return imagePaths
```

Для успешной работы нашего кода и возможности отладки, добавим log-файл, в который записывается информация при запуске бота и в случае возникновения проблем с работой бота. Строка bot.pooling(none_stop=True) запускает Long Polling, а параметр none_stop=True указывает боту, что он должен стараться не прекращать работу при возникновении ошибок.

```
66  if __name__ == '__main__':
67      # Сконфигурируем Log-файл
68      logging.basicConfig(filename='botLog.log',
69                          format='%(filename)s[LINE:%(lineno)d]# '
70                                 '%(levelname)-8s [%(asctime)s]  '
71                                 '%(message)s',
72                          level=logging.DEBUG)
73
74      logging.info('Start the bot.')
75
76      # В случае возникновения ошибки в Log-файл
77      # будет добавлена инфорамция и перезапущен бот
78      try:
79          bot.polling(none_stop=True)
80      except Exception:
81          logging.critical('ERROR...')
82      finally:
83          bot.polling(none_stop=True)
```

Вопросы

1. Какие существуют обертки над Telegram API. Опишите преимущества и недостатки каждой
2. Для чего используется Beautiful Soup в данной работе
3. Каким образом должны осуществляться запросы к Telegram Bot API. В каком виде представляется ответ. В какой кодировке должны быть запросы.

4. Какие возможности для разработки представляет Telegram Bot API.
5. Возможна ли отправка уведомлений пользователям, инициированная ботом. Опишите принцип работы PushAll
6. Назовите доступные типы, используемые в Bot API
7. Опишите поведение requests.get() и назначение параметров
8. Каким образом можно получить обновления от бота

Задание

Выполните оптимизацию кода, ускорив процесс поиска. Повысьте качество выводимых пользователю изображений.

Разместите бот на сервере и проверьте возможность его одновременной работы с несколькими пользователями.

Реализуйте возможность добавления данного бота к любому диалогу.

Расширьте функционал: предусмотрите возможность ввода:

- запроса для поиска,
- количества выдаваемых результатов
- разрешения полученных изображений

Дополнительное задание

Создайте игру «Пятнашки» (калькулятор, напоминание или др.) с использованием TelegramBot API.

КОНТРОЛЬНЫЕ ВОПРОСЫ ПО КУРСУ

Перечислите основные шаги по настройке приложения для использования API

Объясните назначение ключа API

Опишите как чаще всего контролируется API

Расскажите о том, как избежать включения секретного ключа API в ваш репозиторий на Github (например, при сохранении его в коде)

Поясните для чего важно знать какую версию API вы используете

Дайте определение RESTful API и опишите преимущества его использования

Определите, что такое OAuth и для чего используется

Выскажите свою точку зрения, почему пользователю может выбрать вход на Ваш сайт через социальные сети вместо регистрации на Вашем ресурсе

Прокомментируйте этот процесс с точки зрения пользователя и разработчика приложения

Каков на Ваш взгляд, принцип действия IFTTT

Сформулируйте определение SDK и укажите почему он требуется при работе с API

ИТОГОВОЕ ЗАДАНИЕ

Самостоятельно выберите любой API или веб-продукт, используемый в повседневной жизни (пример: Google \ YouTube и пр.). Изучите его документацию и разработайте приложение, упрощающее повседневные задачи. Примеры:

- создайте бота, сохраняющего важную информацию;
- создайте скрипт, получающий последний комментарий под новым постом в группе пикабу vk и отправляйте его себе в любой из используемых мессенджеров
- и т.п.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. pep8 [Электронный ресурс]: Использование интерпретатора Python / Режим доступа : http://pep8.ru/doc/tutorial-2.6/3.html, свободный. - Загл. с экрана.

2. Python 3 для начинающих [Электронный ресурс]: Карта сайта / Режим доступа: https://pythonworld.ru/karta-sajta, свободный. - Загл. с экрана.

3. Current weather and forecasts in your city [Электронный ресурс] / Режим доступа: http://openweathermap.org/, свободный. - Загл. с экрана.

4. Современный учебник Javascript [Электронный ресурс]: Формат JSON, метод toJSON / Режим доступа: https://learn.javascript.ru/json, свободный. - Загл. с экрана.