МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные технологии»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «МЕЖПЛАТФОРМЕННОЕ ПРОГРАММИРОВАНИЕ»

Ростов-на-Дону
ДГТУ
2018

УДК 004.9

Составитель П.В. Васильев

Методические указания для выполнения лабораторных работ по дисциплине «Межплатформенное программирование». – Ростов-на-Дону: Донской гос. техн. ун-т, 2018. – 10 с.

Рассмотрены вопросы базовых понятий, подходов и методов использования языка программирования в рамках межплатформенного программирования, выполнения лабораторных работ.

Методические указания соответствует программе дисциплины и общеобразовательным стандартам, снабжено тестовыми заданиями и заданиями для самостоятельной работы.

Содержат сведения о структуре дисциплины, ее содержании, а также рекомендации по изучению дисциплины.

Предназначены для обучающихся направления 09.03.02 Информационные системы очной формы обучения.

УДК 004.9

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

Ответственный за выпуск зав. кафедрой «Информационные технологии»,
д-р техн. наук, профессор Б.В. Соболь

В печать 25.12. 2018г.
Формат 60×84/16. Объем 0,6 усл. п. л.
Тираж 50 экз. Заказ № 1868.

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина,1

Лабораторная работа №1
НАСТРОЙКА СРЕДЫ И СОЗДАНИЕ GIT РЕПОЗИТОРИЯ

Цель работы: освоить настройку среды и создать Git репозиторий.
Форма отчета: продемонстрировать результаты работы преподавателю.

Методические указания:
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 распределенную 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 возможность легко 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 ботки PyCharm (Для Windows).

Для установки интерпретатора языка Python необходимо скачать (https://www.python.org/downloads/) с официального сайта 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 разра- ботки 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 помощью 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 us-

er.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.g it 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

Цель работы: освоить работу с openweathermap api

Форма отчета: продемонстрировать результаты работы преподавателю.

Методические указания:

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 .

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.

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.

3. Получение метеоданных

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

```
owm = pyowm.OWM('[redacted]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-onDon,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.

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.

Задание

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

Пример 1:

Погода в городе Москва (Россия) на сегодня в 10:15 солнечная, облачность составляет 5%, давление 760 мм рт. ст., температура 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.

Контрольные 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.
5. Опишите общую архитектуру системы
6. Данные каких метеослужб используются при работе с OpenWeatherMap API
7. Какие виды данных можно получать, используя
8. OpenWeatherMap API
9. Приведите спектр применения данного API, укажите основные преимущества и недостатки

Лабораторная работа №3
ФОРМАТ JSON

Цель работы: освоить работу с формат JSON

Форма отчета: продемонстрировать результаты работы преподавателю.

Методические указания:

Теоретическая часть:

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.

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}

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. 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.

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"
```

```
    ]
}
```

Задание

Создайте JSON-представление объекта, описывающего старосту Вашей группы. В объекте должны присутствовать 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.

Контрольные 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

ЛИТЕРАТУРА

1. pep8 [Электронный ресурс]: Использование интерпретатора Python / Режим доступа: http://pep8.ru/doc/tutorial2.6/3.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]:
5. Формат JSON, метод toJSON / Режим доступа: https://learn.javascript.ru/json, свободный. - Загл. с экрана.