1. Цель работы:

Ознакомление с основными элементами определения, представления, проектирования и моделирования программных систем с помощью языка UML.

1. Задание для практических (лабораторных) работ

Задание 1. Изучение объектного моделирования Информационных систем при помощи языка графического описания UML.

UML обозначает Unified Modeling Language. Это язык моделирования общего назначения, позволяющий стандартизировать способ визуализации архитектуры программных систем. Он был разработан Грэди Бучом, Иваром Джекобсоном и Джеймсом Рамбо в Rational Software в 1994–1995 годах. Позже в 1997 году он был принят в качестве отраслевого стандарта. UML — это способ выразить дизайн программных компонентов с помощью широко распространенных графических обозначений. Часто полезно иметь графическую модель, прежде чем начинать кодировать саму модель, используя любые текстовые языки программирования. Позже модель также может быть использована для документирования.

Полезность UML можно описать с помощью следующего сценария из реальной жизни: когда мы работаем с новыми разработчиками, мы не хотим, чтобы они читали каждую строку кода и угадали, что это такое; мы хотим дать им обзор всей системы. Точно так же, когда мы хотим, чтобы они работали над функцией, мы не хотим, чтобы они просто были проинформированы в устной форме; мы хотим, чтобы у них был план требуемой функции, следовательно, использование диаграмм UML. Следование упомянутому выше подходу может облегчить множество головных болей и недоразумений в отношении систем. Часто в этом процессе также обнаруживаются недостатки дизайна. Для обсуждения дизайна с членами команды, UML моделирование начинается с эскиза на доске с минимальным количеством деталей. А когда решение окончательно оформлено, соответствующий эскиз разворачивается в более сложный инструмент, где конечный результат служит прототипом для работы программиста.

Диаграмма последовательности (sequence diagrams)

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций.

На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная

линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.

Каждое сообщение представляется в виде стрелки между линиями жизни двух объектов. Сообщения появляются в том порядке, как они показаны на странице сверху вниз. Каждое сообщение помечается как минимум именем сообщения. При желании можно добавить также аргументы и некоторую управляющую информацию. Можно показать самоделегирование (self-delegation) – сообщение, которое объект посылает самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни.

PlantUml едва ли не лучшее решение для рендеринга UML синтаксиса. Его использование упрощает жизнь давая возможность создавать диаграммы UML в виде простого текста.

PlantUML был интегрирован в виде плагина интегрирован в несколько высокоуровневых инструментов, поэтому **визуализировать создаваемый код можно несколькими способами**.

1. С ПОМОЩЬЮ СЕРВИСОВ PLANTUML

https://plantuml-documentation.readthedocs.io/en/latest/index.html

http://www.plantuml.com/plantuml/uml

https://www.planttext.com/

https://plantuml-editor.kkeisuke.com/

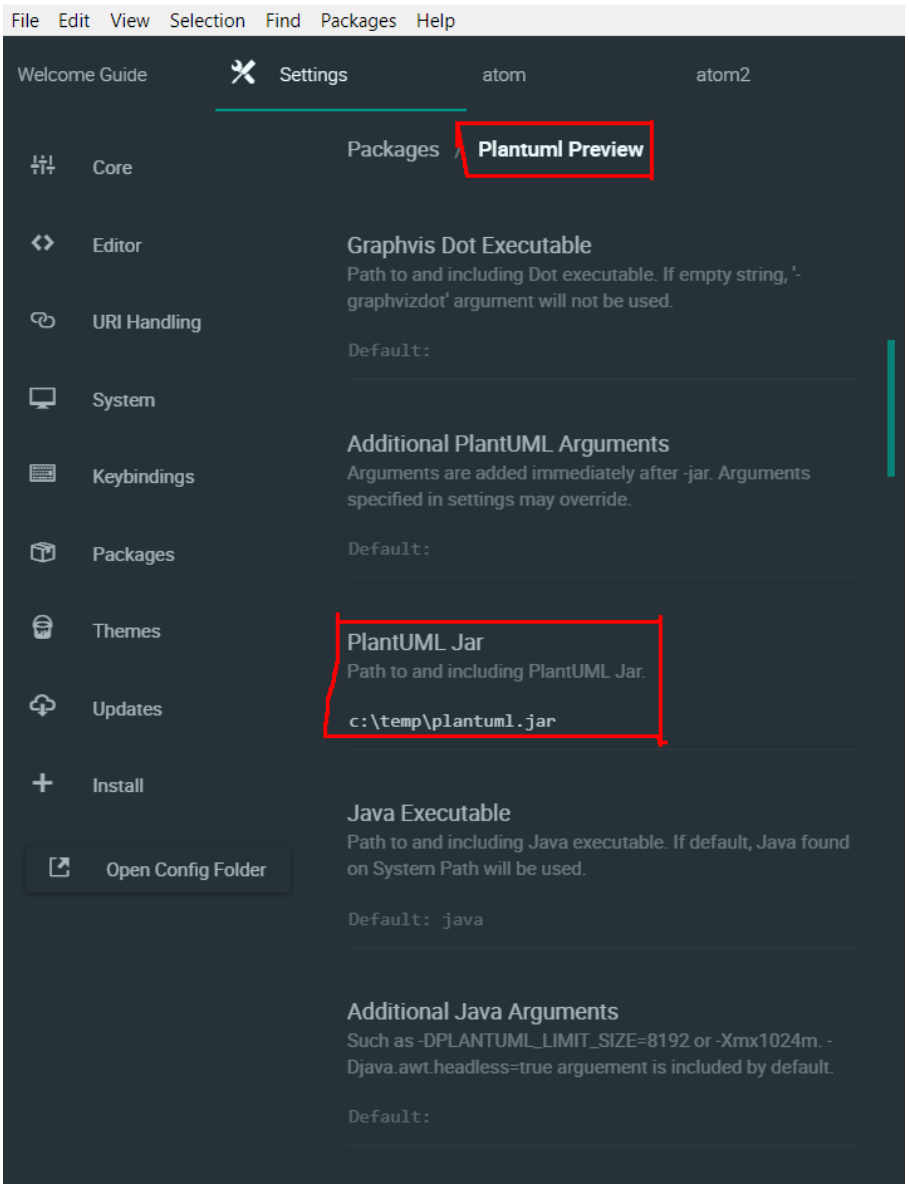
2. С ПОМОЩЬЮ РЕКОНФИГУРИРУЕМОГО РЕДАКТОРА ATOM

   https://atom.io/packages/plantuml

   Настройка среды для Atom

- Graphviz
- Java
- PlantUml
- language-plantuml плагин Atom
- plantuml-preview плагин Atom
- Настроить плагины

Необходимо положить **plantuml.jar** в папку находящуюся в PATH либо прописать путь в настройках:

в 

Graphviz должен установиться и сконфигурироваться по умолчанию. Руководство по сервису PlantUML, частично русифицированное, находится по адресу.

http://plantuml.com/ru/guide

Необходимо воспроизвести **диаграммы классов/блоксхем и последовательности** максимально близко к примерам из вариантов, расположенных здесь (у каждого свой в приложенном файле XLSX). Варианты нераскрашенные или раскрашенные по умолчанию необходимо раскрасить по собственному усмотрению.

В отчете представляется по 2 **картинки из варианта задания**, **uml-код** и **результат рендеринга**.

Результирующая картинка должна иметь название, взятое из названия файла.

Файл отчета должен быть озаглавлен ФИО_вариант.doc

ЛАБОРАТОРНАЯ РАБОТА №2 "УСТАНОВКА, НАСТРОЙКА ВИРТУАЛЬНОЙ СРЕДЫ РАЗРАБОТКИ ANACONDA JUPYTER"

Цели лабораторной работы:

1. Изучить установку, настройку и основные методы работы в интегрированной среде рабочей разработки программного обеспечения Anaconda + IPython Jupyter.
2. Получить практические навыки обработки полученных данных в среде IPython Jupyter с использованием языка программирования Python.

ВИРТУАЛЬНАЯ СРЕДА РАЗРАБОТКИ ANACONDA+JUPYTER

Anaconda, основанная Трэвисом Олифантом, автором библиотеки NumPy, стала неотъемлемым инструментом в области работы с данными, имеющая в своем арсенале большое количество библиотек и плагинов, которые охватывают большинство аналитических случаев. Поскольку Python является интерпретированным языком, с поддержкой REPL, вы можете тестировать фрагменты кода из командной строки, работать с источниками данных перед запуском более сложных скриптов.

Anaconda представляет собой сборку, предназначенную для разработчиков, которые используют Python для анализа данных. Она включает в себя GUI, множество научно ориентированных рабочих сред и инструменты для упрощения процесса обработки данных. Его также можно использовать в качестве общей замены стандартного дистрибутива Python, если вас для вас не существенны различия между ними**.**

Таким образом **Conda**—это менеджер пакетов (как это принято в *nix архитектуре) с открытым кодом и система управления средой, которая работает

на Windows, macOS и Linux.

Самым существенным!! **плюсом** настроенной среды **Conda** является ее возможность нативного использования в высокоуровневых средах разработки **VS Code, JetBrains** итп.

- **Anaconda** — более 150 предустановленных пакетов (около 3 Гб) + более 250 пакетов, готовых к установке командой conda install package_name. Также включает в себя Интегрированную среду разработки Spider 2.

- **Miniconda**— более 400 пакетов, готовых к установке командой conda install package_name

и Анаконда и Миниконда включают:

- conda
- интерпретатор питона
- pip

Сценарии установки
https://docs.conda.io/en/latest/miniconda.html
и развертывания
https://devpractice.ru/python-lesson-6-work-in-jupyter-notebook/

Таким образом, после установки данных компонентов мы имеем практически полный инструментарий для разработки и анализа данных.

Для успешного выполнения лабораторной работы предусмотрен следующий порядок действий.

1. Установить на свой персональный компьютер бесплатную версию Miniconda с текущей версией **Python 3.10**.
2. В появившемся в меню **Пуск** разделе **Anaconda** запустить на исполнение оснастку PowerShell или CommandLine (здесь и далее только установленных с Anaconda).
3. Обновить версию Python до последней (latest python 3.10) скопировав и запустив команду **conda install -c anaconda python = 3.10**.

```
base) PS C:\Users\user> conda install -c anaconda python=3.9.1
ollecting package metadata (current_repodata.json): done
olving environment: failed with initial frozen solve. Retrying with flexible s
ve.
ollecting package metadata (repodata.json): done
olving environment: done

# Package Plan ##

 environment location: C:\Users\user\miniconda3

 added / updated specs:
   - python=3.9.1


he following packages will be downloaded:

   package                    |            build
   ---------------------------|-----------------
   brotlipy-0.7.0             |py39h2bbff1b_1003         411 KB
   ca-certificates-2020.10.14 |                0         159 KB  anaconda
   certifi-2020.12.5          |   py39haa95532_0         141 KB
   cffi-1.14.5                |   py39hcd4344a_0         223 KB
   chardet-3.0.4              |py39haa95532_1003         192 KB
   conda-4.9.2                |   py39haa95532_0         2.9 MB
```

4. Создать новое виртуальное окружение **conda create -n <newenv>** Вместо **<newenv>** подставьте **обязательно** ваше ФИ (алиас) плюс p310**.** Например, **Ivanov_I_p310.** Активизируйте окружение **conda activate Ivanov_I_p310**

```
(base) PS C:\Users\dom> conda create -n Mamedov_Y_p39
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\dom\miniconda3\envs\Mamedov_Y_p39


Proceed ([y]/n)?
```

5. Установить виртуальную среду разработки Jupyter: **conda install -c conda-forge jupyterlab**

```
  requests           conda-forge/noarch::requests-2.25.1-pyhd3deb0d_0
  send2trash         conda-forge/noarch::send2trash-1.5.0-py_0
  setuptools         conda-forge/win-64::setuptools-49.6.0-py39hcbf5309_3
  six                conda-forge/noarch::six-1.15.0-pyh9f0ad1d_0
  sniffio            conda-forge/win-64::sniffio-1.2.0-py39hcbf5309_1
  sqlite             conda-forge/win-64::sqlite-3.34.0-h8ffe710_0
  terminado          conda-forge/win-64::terminado-0.9.2-py39hcbf5309_0
  testpath           conda-forge/noarch::testpath-0.4.4-py_0
  tornado            conda-forge/win-64::tornado-6.1-py39hb82d6ee_1
  traitlets          conda-forge/noarch::traitlets-5.0.5-py_0
  tzdata             conda-forge/noarch::tzdata-2021a-he74cb21_0
  urllib3            conda-forge/noarch::urllib3-1.26.3-pyhd8ed1ab_0
  vc                 conda-forge/win-64::vc-14.2-hb210afc_4
  vs2015_runtime     conda-forge/win-64::vs2015_runtime-14.28.29325-h5e1d092_4
  wcwidth            conda-forge/noarch::wcwidth-0.2.5-pyh9f0ad1d_2
  webencodings       conda-forge/noarch::webencodings-0.5.1-py_1
  wheel              conda-forge/noarch::wheel-0.36.2-pyhd3deb0d_0
  win_inet_pton      conda-forge/win-64::win_inet_pton-1.1.0-py39hcbf5309_2
  wincertstore       conda-forge/win-64::wincertstore-0.2-py39hcbf5309_1006
  winpty             conda-forge/win-64::winpty-0.4.3-4
  zeromq             conda-forge/win-64::zeromq-4.3.4-h0e60522_0
  zipp               conda-forge/noarch::zipp-3.4.1-pyhd8ed1ab_0


Proceed ([y]/n)? y_
```

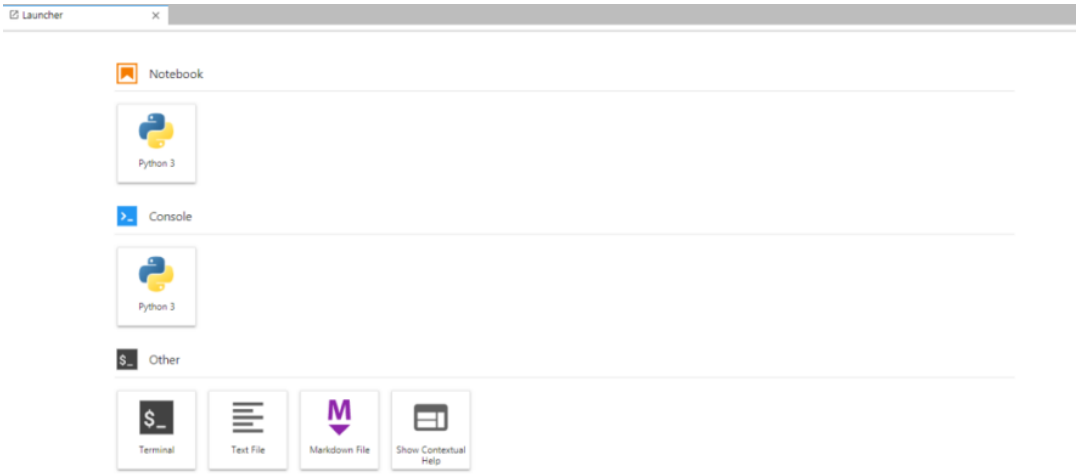
6. Обновить всю инсталляцию Conda с помощью **conda update conda** или **conda install --channel "anaconda" package** или **conda update --all.**
7. Установить пакетный менеджер **python PIP и GIT**: **conda install -c anaconda pip git**
8. Установите в активное окружение пакеты с необходимыми для работы

фреймворками **conda install -c conda-forge pandas scipy scikit-learn pandas matplotlib numpy.** Проверьте состояние инсталляции **conda info.** Скриншот этого вывода нужно приложить к отчету**.** Информацию о пакетах можно узнавать с помощью команды, например **pip show pandas**

```
  pyqt                conda-forge/win-64::pyqt-5.12.3-py39hcbf5309_7
  pyqt-impl           conda-forge/win-64::pyqt-impl-5.12.3-py39h415ef7b_7
  pyqt5-sip           conda-forge/win-64::pyqt5-sip-4.19.18-py39h415ef7b_7
  pyqtchart           conda-forge/win-64::pyqtchart-5.12-py39h415ef7b_7
  pyqtwebengine       conda-forge/win-64::pyqtwebengine-5.12.1-py39h415ef7b_7
  pytz                conda-forge/noarch::pytz-2021.1-pyhd8ed1ab_0
  qt                  conda-forge/win-64::qt-5.12.9-hb2cf2c5_0
  scikit-learn        conda-forge/win-64::scikit-learn-0.24.1-py39he931e04_0
  scipy               conda-forge/win-64::scipy-1.6.0-py39hc0c34ad_0
  threadpoolctl       conda-forge/noarch::threadpoolctl-2.1.0-pyh5ca1d4c_0
  tk                  conda-forge/win-64::tk-8.6.10-h8ffe710_1
  xz                  conda-forge/win-64::xz-5.2.5-h62dcd97_1
  zstd                conda-forge/win-64::zstd-1.4.9-h6255e5f_0

The following packages will be UPDATED:

  ca-certificates     anaconda::ca-certificates-2020.10.14-0 --> conda-forge::ca-
certificates-2020.12.5-h5b45459_0
  certifi             pkgs/main::certifi-2020.12.5-py39haa9~ --> conda-forge::cer
tifi-2020.12.5-py39hcbf5309_1

The following packages will be SUPERSEDED by a higher-priority channel:

  openssl              pkgs/main::openssl-1.1.1j-h2bbff1b_0 --> conda-forge::ope
nssl-1.1.1j-h8ffe710_0


Proceed ([y]/n)?
```

```
      user config file : C:\Users\User\.condarc
 populated config files :
          conda version : 4.9.2
    conda-build version : not installed
         python version : 3.8.5.final.0
       virtual packages : __win=0=0
                          __archspec=1=x86_64
       base environment : C:\Users\User\miniconda3  (writable)
           channel URLs : https://repo.anaconda.com/pkgs/main/win-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/r/win-64
                          https://repo.anaconda.com/pkgs/r/noarch
                          https://repo.anaconda.com/pkgs/msys2/win-64
                          https://repo.anaconda.com/pkgs/msys2/noarch
          package cache : C:\Users\User\miniconda3\pkgs
                          C:\Users\User\.conda\pkgs
                          C:\Users\User\AppData\Local\conda\conda\pkgs
       envs directories : C:\Users\User\miniconda3\envs
                          C:\Users\User\.conda\envs
                          C:\Users\User\AppData\Local\conda\conda\envs
                platform : win-64
            user-agent : conda/4.9.2 requests/2.24.0 CPython/3.8.5 Windows/7 Wi
ndows/6.1.7601
          administrator : False
             netrc file : None
           offline mode : False
```

9. Запустите виртуальную среду разработки командой **jupyter lab --NotebookApp.notebook_dir="C:/Users/<ваш рабочий каталог>".** В окне вашего браузера откроется новая вкладка. Скриншот этой вкладки нужно приложить к отчету.



ЛАБОРАТОРНАЯ РАБОТА №3 " ОСНОВЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON "

Результатом выполнения, оформления и защиты лабораторной работы является формирование следующих компетенций:

1. ПК-2: умение разрабатывать новые методы и средства проектирования информационных систем;
2. ПК-3: умение разрабатывать новые технологии проектирования информационных систем;
3. ПК-10: умение осуществлять моделирование процессов и объектов на базе стандартных пакетов автоматизированного проектирования и исследований.

Цели лабораторной работы:

1. Изучить основные методы использования языка программирования Python.
2. Изучить основные типы и источники данных, а также методы получения данных.
3. Изучить методы первичной обработки данных с помощью регулярных выражений.

1. КРАТКИЕ СВЕДЕНИЯ ИЗ ТЕОРИИ

Во всём мире Python широко используется и как первый язык программирования, предназначенный для начального знакомства с информатикой, и как профессиональный инструмент, например, для научных исследований или прототипирования будущих программных проектов.

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра ***Python*** минималистичен, однако, в то же время **стандартная библиотека** включает большой объём полезных функций [1,2].

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное.

По своей природе *Python* имеет простой, удобочитаемый синтаксис и ясную модель программирования. Если же обучающийся только начинает свой путь в области разработки ПО, то *Python* станет идеальным «вводным» языком программирования. Благодаря своей лаконичности он позволит быстрее овладеть синтаксисом языка.

Любой язык программирования состоит из двух частей – словаря и синтаксиса. Язык *Python* организован точно так же, предоставляя синтаксис для формирования выражений, образующих исполняемые программы, и словарь – набор функциональности в виде стандартной библиотеки и подключаемых модулей [4,5].

Python является языком с **динамической** типизацией. Имеются следующие типы данных:

- целое числа – *Int*;
- вещественные числа – *Float*;
- строки – *String*;
- булевы значения – *Boolean*;

– Список – *List*;
– Кортеж – *Tuple*;
– Словарь – *Dict*;
– Множество – *Set*, и другие.

Все значения являются объектами, в том числе функции, методы, модули, классы.

Python имеет достаточно традиционный набор операторов:

– оператор присваивания = ;
– математические операторы +, -, /, *, **, %;
– блок условных операторов *if* и *else*;
– операторы цикла *while*, for и дополнительные к ним операторы *break*, *continue*;
– операторы сравнения ==, !=, >, <.
– операторы *and* и *or*;
– оператор ввода *input*;
– оператор вывода *print*.

В рамках данного учебно-методического пособия мы рассмотрим лишь базовые операторы, которые дадут нам основные знания и понятия языка *Python*, но уже которых достаточно для использования для применения в анализе данных.

В качестве первичного обучающего средства может использоваться визуализатор[6]:

Визуализатор языка программирования *Python*, позволяет работать непосредственно в браузере, проверять правильность решения на разных входных и выходные данных, что дает значительное преимущество на первоначальном этапе. На сайте визуализатора имеются примеры с подробным описанием некоторых задач, которые также помогут с решением первичных трудностей.

- sagemath.org (IPython 2)
- wakari.io (IPython)

- Trinket.io (Python 2)
- tutorialspoint.com (IPython, Python 2/3)
- repl.it (Python 2/3)
- ideone.com (Python 2/3)
- dbgr.cc (Python 2/3)
- pythontutor.com (Python 2/3)
- labs.codecademy.com (Python 2)
- pythonfiddle.com (Python 2)
- pythonanywhere.com (IPython 2)
- codeskulptor.org (Python 2)
- Onlinegdb.com

1 В визуализаторе (п1.2) языка программирования *Python* выполняем все нижеуказанные задания:

1.1 Ввод и вывод данных:

- Напишите программу, которая считывает три числа и выводит их сумму. Каждое число записано в отдельной строке.
- Напишите программу, которая считывает длины двух катетов в прямоугольном треугольнике и выводит его площадь. Каждое число записано в отдельной строке.

4.2. Условный оператор:

4.2.1. Даны два целых числа. Выведите значение наименьшего из них.

4.2.2. Даны три целых числа. Выведите значение наименьшего из них.

4.3. Математические вычисления:

4.3.1. Дано натуральное число. Выведите его последнюю цифру.

4.3.2. Дано положительное действительное число X. Выведите его дробную часть.

4.4. Циклические операторы:

4.4.1. Даны два целых числа A и B (при этом $A \leq B$). Выведите все числа от A до B включительно.

4.4.2. Дано несколько чисел. Вычислите их сумму. Сначала вводите количество чисел N, затем вводится ровно N целых чисел. Какое наименьшее число переменных нужно для решения этой задачи?

4.5. Строки:

4.5.1. Дана строка, состоящая из слов, разделенных пробелами.

Определите, сколько в ней слов через оператор count.

4.5.2. Дана строка. Сначала выведите третий символ этой строки. Во второй строке выведите предпоследний символ.

4.6. Списки:

4.6.1. Выведите все элементы списка с четными индексами (то есть *A*[0], *A*[2], *A*[4], ...).

ЛИТЕРАТУРА

1. *Майк МакГрат. Программирование на Python для начинающих*
2. *Бэрри П. - Изучаем программирование на Python (Мировой компьютерный бестселлер) – 2017*
3. *Зед Шоу Легкий способ выучить Python 3*
4. *Д. Златопольский. Основы программирования на языке Python*

1.1 Типы и источники открытых данных

С всё большим развитием сети Интернет постепенно растет и информационное поле, которое нас окружает каждый день. Практически любую информацию можно найти с использованием поискового сервиса, практически каждого человека можно найти в той или ной социальной сети.

Каждая информация имеет тот или иной внешний вид и определенный формат, с которым должен уметь работать обучающийся, который занимается анализом данных [8].

В рамках данного раздела мы рассмотрим основные типы данных, с которыми будем сталкиваться при дальнейшей работе с информацией.

1. *TXT* – компьютерный файл, содержащий текстовые данные. Текстовый файл содержит последовательность символов (в основном печатных знаков, принадлежащих тому или иному набору символов). Эти символы сгруппированы в строки, которые разделяются разделителями строк.

2. *CSV* (*Comma-Separated Values*) – текстовый формат, предназначенный для представления табличных данных, где каждая строка файла – это одна строка таблицы, а разделителем значений колонок является символ запятой или другой разделитель (\t или; или |).

3. *XLSX*(*XLS*) – формат файла электронных таблиц. Основной формат для *Microsoft Excel*.

4. *XML* (*eXtensible Markup Language*) – расширяемый язык разметки. Спецификация *XML* описывает *XML*-документы. *XML* разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете.

5. *JSON (JavaScript Object Notation*) – текстовый формат обмена данными, основанный на *JavaScript*.

6. *DB (Data Base File*) – файл базы данных (*SQLite, Microsoft Access, LibreOffice* и другие).

Именно в данных форматах можно обнаружить информацию на любом информационном портале. В качестве источником открытых данных мы будем использовать следующие источники информации:

1. Информация из порталов открытых данных. Например, Портал открытых данных Российской Федерации (*data.gov.ru*), или Портал открытых данных Правительства Москвы (*data.mos.ru*).

2. Реестры данных, например, Реестр открытых данных Минфина России (*minfin.ru/opendata*).

3. Веб-сайты с табличной структурой представления данных (например, *avito.ru* или *realty.e1.ru*).

4. Социальные сети как источники данных (*vk.ru, ok.ru*)

1.2 Методы API социальных сетей

API (программный интерфейс приложения – *Application Programming Interface*) набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной

системой для использования во внешних программных продуктах [9]. Используется программистами при написании всевозможных приложений.

API определяет функциональность, которую предоставляет программа, при этом *API* позволяет абстрагироваться от того, как именно эта функциональность реализована.

Практически все современные популярные веб-сервисы имеют свои *API*. В качестве примера, можно привести данные интерфейсы социальной сети ВКонтакте, *Facebook*, Одноклассники, *Viber* или *API Headhunter, API Google* и другие.

Для дальнейшей работы мы будем использовать программный интерфейс социальной сети ВКонтакте. Полностью ознакомиться с данным *API* можно на сайте разработчиков данной социальной сети *vk.com/dev/openapi* [9].

Open API – система для разработчиков сторонних сайтов, которая предоставляет возможность легко получать информацию о пользователях ВКонтакте. С согласия пользователей, существует возможность получить доступ к информации об их друзьях, фотографиях, аудиозаписях, видеороликах и прочих данных.

В рамках подключения к *Open API* создается приложение, которое позволяет использовать все текущие методы ВКонтакте API.

Рассмотрим самые необходимые методы для получения данных о конкретных пользователях.

1. *Users.Search* – возвращает список пользователей в соответствии с заданными критериями поиска.
2. *Friends.Get* – возвращает список идентификаторов друзей пользователя или расширенную информацию о друзьях пользователя.
3. *Wall.Get* – возвращает список записей со стены пользователя или сообщества

Кратко рассмотрим формат запроса *Users.Search* с критериями «Вася Иванов», который работает в Екатеринбурге (id=49) в ОАО «РЖД».

API.Users.Search(q=”Вася Иванов”, company=”ОАО РЖД”, city=49) В

результате мы получим следующий ответ:

"response": {
"count": 1,
"items": [{
"id": 2943442,
"first_name": "Vasya",
"last_name": "Ivanov"
}]}

Таким образом, в зависимости от требуемого запроса, можно получить те или иные данные в зависимости от необходимых задач. При этом важно учитывать ограничение на количество запросов по времени, а также на то, что некоторые поля данного пользователя могут быть недоступности в связи с настройками приватности.

1.3 Библиотеки Python для работы с данными

Для дальнейшей работы для получения данных, преобразования, обработки и представления основных стандартных библиотек *Python* недостаточно и потребуется дополнительно загрузить определенные модули.

Для загрузки дополнительных библиотек устанавливаем *pip*, а затем загружаем соответствующие библиотеки. Например, для установки библиотеки для работы с *API* ВКонтакте требуется в командной строке прописать:

python -m pip install vk

Аналогичным способом устанавливаются и другие библиотеки.

Рассмотрим библиотеки *Python*, которые нам пригодятся в дальнейшем для работы с данными.

1. *RE* – модуль *Python* для работы с регулярными выражениями. Регулярное выражение – это последовательность символов, используемая для поиска и замены текста в строке или файле. Чаще всего регулярные выражения используются для:

– поиска в строке;

– разбиения строки на подстроки;

– замены части строки.

2. *LXML* – это быстрая и гибкая библиотека *Python* для обработки разметки *XML* и *HTML* на *Python*. С помощью данной библиотеки, мы можем легко работать с документами вышеупомянутого типа, производить поиск в данных документах, отбирать необходимые данные по определенным тегам.

3. *Requests* – это библиотека *Python*, которая элегантно и просто выполняет *HTTP*-запросы. Благодаря данной библиотеки, мы можем легко получить *html*-текст страницы, пройти регистрацию и многое другое.

4. *VK* – это библиотека *Python* для работы с *API* социальной сети ВКонтакте. Библиотека представляет собой все возможные методы данного *API*.

5. *Pandas* – это библиотека *Python*, предоставляющая широкие возможности для анализа данных. Данные, с которыми работают специалисты, часто хранятся в табличном формате – например, в форматах *.csv или .xlsx*. С помощью библиотеки *Pandas* такие табличные данные очень удобно загружать, обрабатывать и анализировать с помощью *SQL*-подобных запросов. А в связке с библиотеками *Matplotlib* и *Seaborn Pandas* предоставляет широкие возможности визуального анализа табличных данных.

6. *MatPlotLib* – библиотека на языке программирования *Python* для визуализации данных двумерной графикой. Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях.

7. *Seaborn* – библиотека для визуализации на языке *Python*. Представляет собой высокоуровневый интерфейс для рисования различных статистических данных.

1.4 Получение данных для анализа

Для дальнейшего изучения инструментария получения данных рассмотрим пример с добычей и обработкой данных с социальных сетей.

В качестве примера, будет рассмотрен код получения данных о друзьях конкретного пользователя социальной сети ВКонтакте. Будут использоваться

методы *API* данной социальной сети и некоторые библиотеки *Python*, рассмотренные выше. Результатом анализа будет визуализация эгоцентрического графа друзей пользователя. Пример является простым, но достаточно показательным для дальнейшего самостоятельного применения при анализе данных [10].

```
#подключаем соответствующие библиотеки
import requests
import networkx
import pylab as plt
#создаем функцию, которая запрашивает список друзей пользователя user_id
def get_friends_ids(user_id):
    friends_url = 'https://api.vk.com/method/friends.get?user_id={}'
        #сохраняем данные в json-объект
    json_response = requests.get(friends_url.format(user_id)).json()
    if json_response.get('error'):
        print json_response.get('error')
        return list()
    return json_response[u'response']

#создаем пустой граф
graph = {}
#вводим id пользователя, для которого строим граф
friend_ids = get_friends_ids(111111)
#последовательно получаем список друзей каждого из
#друзей введенного пользователя и записываем в словарь-граф
for friend_id in friend_ids:
    print 'Processing id: ', friend_id
    graph[friend_id] = get_friends_ids(friend_id)
#с помощью библиотеки networkx строим граф по полученному словарю
g = networkx.Graph(directed=False)
```

```
for i in graph:
    g.add_node(i)
    for j in graph[i]:
        if i != j and i in friend_ids and j in friend_ids:
            g.add_edge(i, j)
#с помощью библиотеки pylab визуализируем этот граф
networkx.draw_networkx(g,font_size=8)
plt.show()
```

В результате выполнения описанного выше кода получаем следующий эгоцентрический граф, отражающий взаимосвязь каждого из друзей конкретного пользователя. Данную информацию можно по-разному использовать в зависимости от поставленных целей[10].

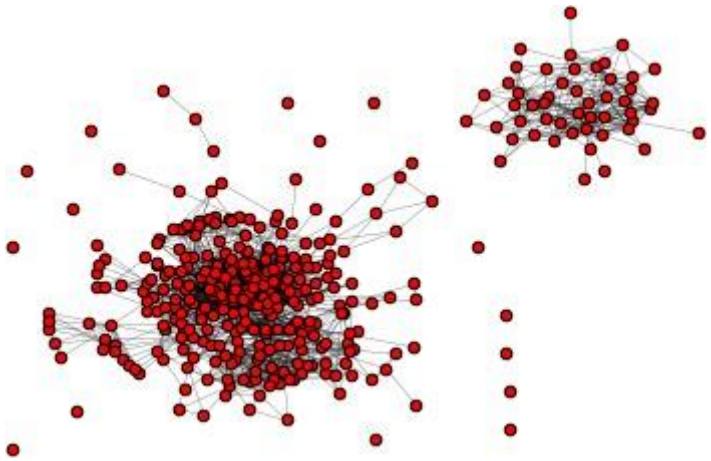


Рисунок 1.3 – эгоцентрический граф друзей

Аналогичным способом можно получить любую информацию, описанную в профиле пользователя. С помощью методов Wall, можно получить информацию со стены пользователя, проанализировать любые комментарии, ссылки, статусы пользователя.

Особое внимание стоит уделить приватности данных. Многие пользователи по-разному настраивают этот параметр для своих данных.

Большую часть информации можно получить с использованием ключа доступа access_token. Данный токен можно получить зарегистрировав свое приложение на сайте разработчиков конкретной социальной сети. Далее идет запрос через протокол авторизации *OAuth* 2.0 для получения соответствующей информации.

В некоторых случаях недостаточно и данного метода вследствие очень строгих настроек приватности пользователя[9].

Другой важной составляющей является ограничение на количество запросов с авторизацией через *API* со стороны приложения. Для клиентского приложения это 3 запроса в секунду. Как следствие, в любой код, через который мы получаем информацию требующую авторизацию нужно всего лишь добавить небольшую задержку[9,10].

1.5 Регулярные выражения

Регулярные выражения представляют собой похожий сильный инструмент для поиска строк, проверки их на соответствие какому-либо шаблону и другой подобной работы. Англоязычное название этого инструмента *Regular Expressions* или просто *RegExp*. Строго говоря, регулярные выражения — специальный язык для описания шаблонов строк. В языке программирования Python для регулярных выражений существует библиотека RE.

Любая строка сама по себе является регулярным выражением, по которому можно находить определенные строки, однако существуют дополнительные операторы, которые делают regexp мощным инструментом для работы с текстом https://habr.com/ru/post/115825/.

Таблица 1.1

Основные операторы регулярных выражений

Класс знаков	Описание	Шаблон
[*группа_символов***]**	Соответствует любому одиночному символу, входящему в *группу_символов*. По умолчанию при сопоставлении учитывается регистр.	[ae]
[^*группа_символов***]**	Отрицание: соответствует любому одиночному символу, НЕ входящему в *группу_символов*.	[^aei]
[*first* **-** *last* **]**	Диапазон символов: соответствует одному символу в диапазоне от *first* до *last*.	[A-Z]
.	Подстановочный знак: соответствует какому-либо одному знаку, кроме "\n".	a.e
\w	Соответствует алфавитно-цифровому знаку.	\w

\W	Соответствует любому символу, НЕ являющимся алфавитно-цифровым знаком.	\W
\d	Соответствует любой десятичной цифре.	\d

Другой составляющей регулярных выражений являются привязки. Привязки приводят к успеху сопоставления, в зависимости от текущей позиции в строке, но не предписывают обработчику перемещаться по строке или обрабатывать символы [11,12].

Таблица 1.2

Основные привязки регулярных выражений

Утверждение	Описание	Шаблон
^	Соответствие должно начинаться в начале строки или после знака переноса.	^\d{3}
$	Соответствие должно обнаруживаться в конце строки или до символа ***\n*** в конце строки.	\d{3}$
\b	Соответствие должно обнаруживаться на границе между символом ***\w*** (алфавитно-цифровым) и символом ***\W*** (не алфавитно-цифровым).	\b\w+\s\w+\b

В языке программирования Python существуют следующие методы для работы с регулярными выражениями.

Таблица 1.3

Основные методы Python для работы РВ

Метод	**Описание**
match()	Определить, начинается ли совпадение регулярного выражения с начала строки
search()	Сканировать всю строку в поисках всех мест совпадений с регулярным выражением
findall()	Найти все подстроки совпадений с регулярным выражением и вернуть их в виде списка
finditer()	Найти все подстроки совпадений с регулярным выражением и вернуть их в виде итератора

2 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ №3

1. Получаем *Access Token* для приложения *API* ВКонтакте. Регистрируем свое приложение на сайте разработчиков.
2. Формируем эгоцентрический граф своих друзей в социальной сети ВКонтакте (аналогично рассмотренному примеру):
3. Получаем список всех своих друзей по введенному своему *ID*.
4. Получаем список друзей всех ваших друзей.
5. Формируем граф по полученным спискам. Делаем представление полученных результатов.
6. Контрольным заданием является получение данных со стены определенного круга людей с дальнейшим применением фильтрации с использованием регулярных выражений:
7. Аналогично методу, рассмотренному выше получаем все сообщения со стен (*Wall.Get*) у каждого из ваших друзей (50 сообщений с одного профиля, рассматриваем минимум 20 друзей). Следует уделить внимание рассмотренным выше ограничениям данного метода (не более трех запросов в сек).
8. Последовательно записываем информацию в один из типов файлов, которые были рассмотрены в пункте 1.3 данного методического пособия. Приоритетом является более удобный вид данных для данной лабораторной работы.
9. Получив и сохранив информацию, последовательно начинаем обработку всего объема информации. Для выполнения отчетного задания требуется выполнить следующие задачи с использованием регулярных выражений (*RE*):
10. Вывести список всех пользователей (*User ID*), которые делали записи на конкретной стене пользователя.
11. Вывести все даты создания объектов на конкретной стене.
12. Самостоятельно придумать критерий отбора для данного набора

данных, имеющий научную ценность. Сформировать регулярное выражение для данного критерия. Вывести список всех строк, удовлетворяющих данному критерию.

13. Сформировать отчет по проделанной лабораторной работе. К защите данной лабораторной работы должны быть предоставлены следующие данные:
14. Выполненные задания по синтаксису языка *Python*.
15. Эгоцентрический граф, построенный по конкретным пользователям. Код для построения данного графа.
16. Выполненное задание по сбору и обработке данных со стен пользователей. Список полученных регулярных выражений и список выходных данных по их критериям.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите основные особенности работе IDE Microsoft Visual Studio?
2. Охарактеризуйте язык программирования Python?
3. Какие операторы используются в языке программирования Python?
4. Какие типы данных в языке программирования Python?
5. Назовите основные виды источников открытых данных?
6. Перечислите основные типы текстовых данных? Где они используются?
7. Какие библиотеки Python входят в стандартный набор данного языка программирования?
8. Какие дополнительные библиотеки были рассмотрены в данном методическом пособии? Перечислите их.
9. Для чего служит утилита pip?
10. Как подключаются дополнительные модули к основной программе?
11. Что такое API интернет-сервиса? Перечислите известные Вам?
12. Перечислите основные методы API ВКонтакте. Какие бывают ограничения у методов?
13. Что такое Access Token? Для чего он используется?
14. Что такое эгоцентрический граф? Что мы можем увидеть с помощью него?
15. Какие библиотеки, которые использовались в данном примере?
16. Что такое регулярные выражения? Для чего они используются?
17. Перечислите основные операторы регулярных выражений?
18. Перечислите основные привязки регулярных выражений. Чем они отличаются от операторов?
19. Какой модуль отвечает за регулярные выражения в Python?

20. Перечислите основные методы модуля Python для работы с regexp?

ЗАКЛЮЧЕНИЕ

Методические рекомендации по выполнению лабораторной работы по дисциплине «Системы автоматизированного проектирования информационных систем» для направления подготовки 09.04.02 –Информационные системы и технологии (квалификация «магистр»).

Методические рекомендации по выполнению лабораторной работы на тему «Выполнение моделирования информационной системы на этапах эскизного и технического проектирования в среде разработки Microsoft Visual Studio» содержит краткую теорию относительно основных вопросов, затрагиваемых во время выполнения данной лабораторной работы.

В методических указаниях последовательно описан процесс выполнения лабораторной работы с отсылками на конкретные вопросы, затрагиваемые в теории.

Окончательным результатом выполнения лабораторной работы является оформление и защита отчета по данным методическим рекомендациям. Структура отчета описана в пункте 2 данного пособия. Защита лабораторной работы производится путем ответа на вопросы контрольные вопросы, описанные в данном пособии, а также на вопросы, возникающие во время выполнения лабораторной работы.

По результатам выполнения лабораторной работы у магистров формируются практические навыки работы в среде разработки Microsoft Visual Studio при выполнении моделирования информационной системы на этапах эскизного и технического проектирования.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Форта Бен. Освой самостоятельно регулярные выражения. 10 минут на урок : пер. с англ. – М. Издательский дом «Вильямс», 2005. – 184 с.: ил. Парал. тит. англ.
2. Лутц М. Программирование на Python, том I. – 4-е изд.: пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с., ил.
3. Паклин Н.Б., Орешков В.И. Бизнес-аналитика: от данных к знаниям. – СПб.: Питер, 2013. – 704 с.: ил.
4. https://pythonru.com/baza-znanij/kak-ustanovit-anaconda-na-windows [2].
5. https://www.python.org/downloads/windows/ [3].
6. https://conda.io/projects/conda/en/latest/user-guide/getting-started.html

1. Задание для практических (лабораторных) работ

Задание 1. Разработка базы данных социальной сети (2 часа)

Используя графовую СУБД Neo4j, разработайте базу данных модельной социальной сети. Узел социальной сети имеет следующие атрибуты: ФИО, пол, возраст, город, список групп по интересам (например: спорт, игры, компьютеры). Семантика связи между узлами – дружеские отношения соответствующих персон («А является другом Б»). См. Указания к заданию 1. Разработка базы данных социальной сети.

Задание 2. Разработка запросов к графам (2 часа)

Разработайте и протестируйте запросы на выборку данных из созданной графовой базы данных. См. Указания к заданию 2. Разработка запросов к графам.

Задание 3. Визуализация графа (2 часа)

Разработайте и протестируйте запросы на обновление свойств узлов графа и выполните с их помощью визуализацию графа.

Каждое задание должно быть оформлено на листах формата А4 и представлять собой отдельный отчет. Отчет должен содержать: титульный лист, лист с техническим заданием, отчет о выполненной работе, выводы.

2. Справочная информация о графовой СУБД Neo4j

В качестве основного источника справочной информации по графовой СУБД Neo4j и языку Cypher используйте приведенную ниже краткую справку и ресурс http://docs.neo4j.org/.

Краткая справка по языку Cypher

Cypher представляет собой декларативный язык запросов СУБД Neo4j. С помощью Cypher возможно сопоставление паттерны узлов и отношений в графе для извлечения и модификации информации, поддержка системы идентификаторов для обозначения именованных узлов, граничных узлов и их па- раметров, а так же создание, обновление и удаление узлов, связей и их свойств. При указании на части паттернов используются условные имена (называемые *идентификаторами*).

Пример: `START n=node(1) MATCH n-->b RETURN b`

Здесь `n` и `b` являются идентификаторами. Идентификаторы регистрозависимы, могут содержать буквы, числа и символы подчеркивания, но должны начинаться с буквы. Если идентификатор должен содержать другие символы, то его необходимо заключить в кавычки.

Комментарии добавляются с помощью двойного слэша.

Допустимые *выражения* в Cypher:

1) *Численный литерал* (целый или вещественный): `13`, `40000`, `3.14`.
2) *Строковый литерал*: `"Hello"`, `'World'`.
3) *Булевский литерал*: `true`, `false`, `TRUE`, `FALSE`.
4) *Идентификатор*: `n`, `x`, `rel`, `myFancyIdentifier`, `` `A name with weird stuff in it[]!` ``.
5) *Свойство*: `n.prop`, `x.prop`, `rel.thisProperty`, `myFancyIdentifier.`(weird property name)`.
6) *Nullable-свойство* (свойство со знаком вопроса или восклицания): `n.prop?`, `rel.thisProperty!`.
7) *Параметр*: `{param}`, `{0}`
8) *Коллекция* выражений:
   `["a", "b"]`, `[1,2,3]`, `["a", 2, n.property, {param}]`, `[ ]`.
9) *Вызов функции*: `length(p)`, `nodes(p)`.
10) *Агрегирующая функция*: `avg(x.prop)`, `count(*)`.
11) *Типы отношений*: `:REL_TYPE`, `:`REL TYPE``, `:REL1|REL2`.
12) *Паттерн пути*: `a-->()<--b`.
13) *Предикат*: `a.prop = "Hello"`, `length(p) > 10`, `has(a.name)`

В Cypher поддерживаются следующие *форматы запросов*:

Вид запроса	Синтаксис запроса
Чтение данных	START [MATCH] [WHERE] RETURN [ORDER BY] [SKIP] [LIMIT]
Запись данных	CREATE [UNIQUE]* [SET|DELETE|FOREACH]* [RETURN [ORDER BY] [SKIP] [LIMIT]]
Чтение и запись данных	START [MATCH] [WHERE] [CREATE [UNIQUE]]* [SET|DELETE|FOREACH]* [RETURN [ORDER BY] [SKIP] [LIMIT]]

Язык запросов Cypher состоит из следующих *предложений*:

1) `START`: начальные точки запроса в графе, полученные с помощью индексов или идентификаторов узлов (ID).

Пример	Семантика
`START n=node(*)`	Стартовая точка будет установлена на все узлы.
`START n=node({ids})`	Стартовая точка будет установлена на один или более узлов, указанных в списке идентификаторов.
`START n=node({id1}), m=node({id2})`	Несколько стартовых точек.

2) `MATCH`: графовый паттерн, удовлетворяющий запросу, ограниченный начальными точками из оператора `START`. Любой паттерн может использоваться в предложении `MATCH`, за исключением паттернов, содержащих карты свойств.

   Пример: `MATCH (n)-->(m)`

3) `WHERE`: критерий фильтрации.

   Пример: `WHERE n.property <> {value}`

4) `RETURN`: возвращаемое значение.

Пример	Семантика
`RETURN *`	Вернуть все идентификаторы.
`RETURN n AS columnName`	Использовать псевдоним в качестве имени колонки.
`RETURN DISTINCT n`	Вернуть уникальные строки.
`ORDER BY n.property`	Отсортировать по свойству.

Пример	Семантика
`ORDER BY n.property DESC`	Отсортировать по свойству в убывающем порядке.
`SKIP {skip_number}`	Пропустить заданное количество строк в выдаче.
`LIMIT {limit_number}`	Ограничить выдачу заданным количеством строк.
`SKIP {skip_number} LIMIT {limit_number}`	Пропустить сколько-то строк, а потом ограничить выдачу.

5) `CREATE`: создание узлов и связей.

Пример	Семантика
`CREATE (n {name: {value}})`	Создать узел с заданными свойствами.
`CREATE n = {map}`	Создать узел с заданными свойствами.
`CREATE n = {collectionOfMaps}`	Создать узел с заданными свойствами.
`CREATE (n)-[r:KNOWS]->(m)`	Создать связь заданного типа и направления; присвоить ему идентификатор.
`CREATE (n)-[:LOVES {since: {value}}]->(m)`	Создать отношение из заданного типа, направления и набора свойств.

6) `DELETE`: удаление узлов, связей и свойств.
   Примеры: `DELETE n, r` - удалить узел или отношение. `DELETE n.property` - удалить свойство.
7) `SET`: установка значений свойств.
   Примеры: `SET n.property={value}` - обновить или создать свойство. `SET n={map}` - задать все свойства (удалит все существующие свойства).
8) `FOREACH`: выполнение операций обновления для каждого элемента из списка.
9) `WITH`: разбивает запрос на несколько независимых частей.
   При выполнении запросов на языке Cypher необходимо задавать искомые паттерны отношений между узлами. Паттерны составляются по следующим правилам:

Правило	Семантика
`(n)-->(m)`	Проверка на существование отношения от `n` к `m`.

Правило	Семантика
`(n)--(m)`	Проверка на существование отношения между `n` и `m` (направление игнорируется).
`(m)<-[:KNOWS]-(n)`	Проверка на существование отношения от n к m типа `KNOWS`.
`(n)-[:KNOWS|LOVES]->(m)`	Проверка на существование отношения от n к m типа `KNOWS` или `LOVES`.
`(n)-[r]->(m)`	Присвоить идентификатор отношению.
`(n)-[r?]->(m)`	Опциональное отношение.
`(n)-[*1..5]->(m)`	Переменная для пути отношений между `n` и `m`.
`(n)-[*]->(m)`	Нет ограничений на глубину отношений.
`(n)-[:KNOWS]->(m {property: {value}})`	Сопоставить набор свойств в предложениях `CREATE` или `CREATE UNIQUE`.

В языке Cypher предусмотрены *функции для работы с коллекциями и массивами*:

Функция	Семантика
`NODES(path)`	Список узлов в указанном пути.
`RELATIONSHIPS(path)`	Отношения в указанном пути.
`EXTRACT(x IN collection: x.prop)`	Коллекция значений выражения для каждого элемента коллекции.
`FILTER(x IN coll: x.prop <> {value})`	Возвращает коллекцию элементов, для которых предикат истинен.
`TAIL(collection)`	Все элементы коллекции, кроме первого.
`RANGE({begin}, {end}, {step})` `REDUCE(str = "", n IN coll : str + n.prop)`	Создать набор чисел с указанным шагом. Вычислить выражение для каждого элемента коллекции и аккумулировать результат.
`FOREACH (n IN coll : SET n.marked = true)`	Выполнить операцию обновления для каждого элемента в коллекции.

Вспомогательные функции:

Функция	Семантика
`LENGTH(collection)`	Возвращает длину коллекции.
`TYPE(a_relationship)`	Строковое представление типа отношения.
`COALESCE(n.property?, {defaultValue})`	Первое ненулевое выражение.
`HEAD(collection)`	Первый элемент коллекции.
`LAST(collection)`	Последний элемент коллекции.

Функция	Семантика
`TIMESTAMP()`	Количество прошедших миллисекунд от полуночи 1 января 1970 года по … .
`ID(node_or_relationship)`	Внутренний идентификатор отношения или узла.

При работе с примитивами языка Cypher используется развитая система *предикатов*:

Предикат	Семантика
`n.property <> {value}`	Использование операторов сравнения.
`HAS(n.property) AND n.property = {value}`	Использование булевых операторов для комбинирования предикатов.
`HAS(n.property)`	Использование функций.
`identifier IS NULL`	Проверка на null.
`n.property? = {value}`	Возвращает истину, если свойство не существует.
`n.property! = {value}`	Возвращает ложь, если свойство не существует.
`n.property =~ {regex}`	Регулярные выражения.
`(n)-[:KNOWS]->(m)`	Убедиться, что паттерн имеет хотя бы одно совпадение.
`n.property IN [{val1}, {val2}]`	Проверка, что элемент существует в коллекции.

В языке Cypher также имеются *функции для вычисления предикатов над коллекциями*:

Предикат	Семантика
`ALL(x IN collection WHERE HAS(x.property))`	Возвращает истину, если предикат выполняется для всех элементов коллекции.
`ANY(x IN collection WHERE HAS(x.property))`	Возвращает истину, если предикат выполняется хотя бы для одного элемента коллекции.
`NONE(x IN collection WHERE HAS(x.property))`	Возвращает истину, если предикат не выполняется для всех элементов коллекции.
`SINGLE(x IN collection WHERE HAS(x.property))`	Возвращает истину, если предикат выполняется строго для одного элемента коллекции.

Операторы: математические `+`, `-`, `*`, `/`, `%`; сравнения `=`, `<>`, `<`, `>`, `<=`, `>=`, булевы `AND`, `OR`, `NOT`; строковые `+`; операторы коллекций: `+`, `IN`; регулярные выражения: `=~`; операторы свойств `?`, `!`.

Агрегирующие функции: `COUNT`, `SUM`, `AVG`, `MAX`, `MIN`, `COLLECT`, `PERCENTILE_DISC`, `PERCENTILE_CONT`.

Математические функции: `ABS`, `ROUND`, `SQRT`, `SIGN`.

Строковые функции: `STR`, `REPLACE`, `SUBSTRING`, `LEFT`, `RIGHT`, `LTRIM`, `RTRIM`, `TRIM`, `LOWER`, `UPPER`.

Указания к заданию 1. Разработка базы данных социальной сети

1. Разработайте скрипт с запросами на языке Cypher, которые загружают базу данных социальной сети в СУБД Neo4j. База должна содержать не менее 10 узлов и не менее 15 связей между ними.
2. Загрузите базу данных в СУБД Neo4j при помощи консольного клиента (Neo4jShell.bat).

Указания к заданию 2. Разработка запросов к графам

1. Изучите справочную информацию о языке Cypher по теме Reading Clauses и разработайте следующие запросы:
    1) Выдать упорядоченный список ФИО персон.
    2) Выдать список ФИО мужчин с указанием возраста, упорядоченный по убыванию возраста.
    3) Выдать упорядоченный список ФИО друзей персоны заданными ФИО.
    4) Выдать упорядоченный список ФИО друзей друзей персоны заданными ФИО.
    5) Выдать упорядоченный по алфавиту список ФИО персон, в котором для каждой персоны указано количество друзей.
2. Изучите справочную информацию о языке Cypher по теме Functions и раработайте следующие запросы:
    1) Выдать упорядоченный список групп социальной сети.
    2) Выдать упорядоченный список групп персоны с заданными ФИО.
    3) Выдать список групп социальной сети с указанием количества членов каждой группы, упорядоченный по убыванию количества членов группы.
    4) Выдать список ФИО персон, в котором для каждой персоны указано количество групп, в которые она входит, упорядоченный по убыванию количества групп.
    5) Выдать общее количество групп, в которых состоят друзья друзей персоны с заданными ФИО.

Указания к заданию 3. Визуализация графа

1. Изучите справочную информацию по средствам визуализации Neo4j. Отобразите граф.
2. Добавьте свойство *friend* всем друзьям одной персоны и настройте профиль визуализации таким образом, чтобы узлы со свойством *friend* отображались красным цветом.

3. Добавьте свойство *twoHandFriend* всем друзьям друзей одной персоны и настройте профиль визуализации таким образом, чтобы узлы со свойством *twoHandFriend* отображались желтым цветом.
4. Добавьте свойство *manyFriends* персонам, имеющим больше 5 друзей, свойство *fewFriends* – персонам, у которых меньше 3 друзей. Настройте профиль визуализации таким образом, чтобы узлы со свойством *manyFriends* отображались зеленым цветом, узлы со свойством *fewFriends* отображались красным цветом, а все остальные узлы – желтым цветом.
5. Добавьте свойство *group1* персонам, состоящим в некоторой группе, *group2* – персонам, состоящим в другой группе, а свойство *bothGroups* -- персонам, состоящим в обеих группах. Настройте профиль визуализации таким образом, чтобы узлы со свойством *group1* отображались синим цветом, узлы со свойством *group2* отображались красным цветом, а узлы со свойством *bothGroups* – фиолетовым цветом.