**Лабораторная работа № 1**

**РАБОТА В IDLE. ИСПОЛЬЗОВАНИЕ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ. СОЗДАНИЕ СПИСКОВ И СЛОВАРЕЙ. РАБОТА С ЦИКЛАМИ**

**Вариант №11**

**Цель работы:** научиться использовать Pycharm при работе с Python; изучить списки и словари, освоить основные методы для работы с ними; изучить виды циклов в Python.

**Краткая теория**

Среда разработки Pycharm. После установки Python 3 автоматически устанавливается Pucharm – это интегрированная среда разработки, которая включает подсветку синтаксиса, отладчик, Python Shell и полноценную документацию по Python 3 [6-8]. При запуске IDLE открывается окно командной оболочки Python – REPL-среда. Она запрашивает инструкцию от пользователя отображая подсказку в виде трех знаков больше (>>>). При использовании данной среды после ввода блока кода он незамедлительно выполняется и отображается результат его работы.

Pycharm понимает синтаксис Python и предлагает подсказки по завершению кода, когда используются встроенные функции типа print(). Функция print() выводит сообщение на стандартное устройство вывода (обычно экран). Стоит отметить, что в отличие от С-подобных языков программирования вместо фигурных скобок ({}), определяющих границы блока кода, в Python используются отступы. При этом стоит отметить, что Python 3 запрещает смешивание табуляции и пробелов в отступах.

Pycharm имеет множество инструментов для работы с кодом. Рассмотрим некоторые из них:

− чтобы Pycharm предложил варианты завершения встроенной функции, требуется нажать на кнопку TAB, после чего выбрать необходимую функцию из списка;

− в окне командной оболочки Python используются комбинации клавиш ALT+P для вызова предыдущего блока кода и ALT+N

– для вызова следующего. Таким образом, можно отметить, что PyCharm отличный вариант для написания кода на Python

**Задание**

Написать функцию, которая принимает целочисленный список, состоящий из n элементов, и возвращает True, если в каком-то месте списка по порядку содержатся элементы 1,2,3 или комбинации их перестановок.

Результат работы программы представлен на рисунках 1-2.

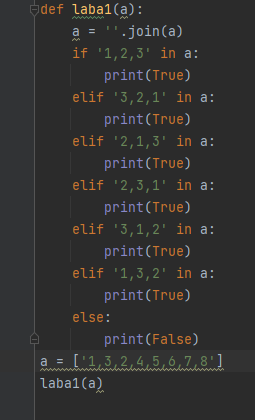


Рисунок 1.1 – программный код.

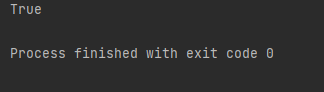


Рисунок 1.2-результат работы программы

**Вывод**: в результате проделанной работы изучен PyCharm при работе с Python; изучили списки и словари, освоили основные методы для работы с ними, а так же изучили виды циклов в Python.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Проверил | Елкин Н.С. |

**Лабораторная работа №2**

**РАБОТА С ОСНОВНЫМИ ВСТРОЕННЫМИ ФУНКЦИЯМИ**

**Вариант №11**

**Цель работы**: рассмотреть основные встроенные функции языка программирования Python и научиться с ними работать.

**Краткая теория**

Стандартная библиотека Python предоставляет огромное количество различных модулей и функций, так называемых Built-in Functions. Все эти инструменты поставляются вместе с интерпретатором и их достаточно для того, чтобы немедленно начать работу с языком сразу после установки. Рассмотрим некоторые из основных встроенных функций:

**print(\*objects, sep=’’, end=’\n’, file=sys.stdout, flush=False).**

Данная функция предназначена, как было описано в лабораторной работе №1, для вывода сообщения на стандартное устройство вывода. Однако стоит дополнительно рассмотреть её параметры:

− \*objects – параметр, который принимает объекты для вывода на устройства вывода. Символ «\*» говорит о том, что параметр принимает переменное количество аргументов, например, строки через запятую;

− sep – параметр, аргумент которого предназначен для разделения аргументов параметра \*objects;

− end – параметр, аргумент которого выводится в конце вывода аргументов параметра \*objects;

− file – параметр, принимающий в качестве аргумента только объекты с методом write(string), если он не указывается или указан как None будет использоваться sys.stdout;

− flush – параметр, который предназначен для дополнительного указания функции при использовании параметра file в случае, если он Истина, то производить немедленный сброс данных из буфера в память.

Дополнительно стоит отметить, что оба параметра sep и end должны принимать только строки. Они могут принимать 37 None, что инициирует использование значений по умолчанию. Также стоит учесть, что если \*objects не будет принимать никаких аргументов, то выведется только значение end.

**len(s)** Данная функция принимает в качестве аргумента последовательность (строка, байты, список, кортежи и другое) или коллекции (словарь, множество и другие). Возвращает длину (число элементов) объекта.

**int([x])** Возвращает целочисленный объект из числа или строки, переданной в параметр x. В случае, если аргументы не заданы, то возвращает 0. У вещественных чисел удаляется дробная часть.

**float([x])** Возвращает число с плавающей точкой из числа или строки, переданной в параметр x. В случае, когда аргументом параметра x является строка, то она должна содержать десятичное число с необязательным предшествующим знаком (+ или -). Если аргумент не задан, то возвращается 0.0.

**list([iterable])** Функция предназначена для конструирования списка. Аргумент параметра iterable может быть последовательностью, контейнером, либо итераторным объектом. Если в качестве аргумента iterable передается уже существующий список, то функция возвращает копию объекта переданного списка. Если заданных аргументов нет, то возвращается новый пустой список.

**dict(\*\*kwarg)** Функция предназначена для создания словаря. Два символа «\*» означают, что параметр принимает переменное значение именованных аргументов (one=1, two=2). В случае, если аргумент передан параметру \*\*kwarg, то будет создаваться новый словарь с теми же парами ключ-значение. Если ключ в аргументе присутствует более одного раза, то будет использоваться последнее значение ключа переданного аргумента. Если аргумент не задан, то функция вернет новый пустой словарь.

**range(start, stop[, step])** Функция предназначена для генерации неизменяемой числовой последовательности. Данную функцию можно вызывать тремя способами:

⎯ range(stop). В этом случае вернется ряд чисел, начинающихся с 0 и включающих каждое число до аргумента параметра stop, но не включающее само значение аргумента;

⎯ range(start, stop). В этом случае вернется ряд чисел, начинающихся со значения аргумента параметра stop и включающих каждое число до значения параметра start, но не включающее само значение его аргумента;

⎯ range(start, stop, step). В этом случае функция будет работать так же, как и в предыдущем случае, за исключением того, что аргумент параметра step будет указывать на разницу между двумя соседними числами.

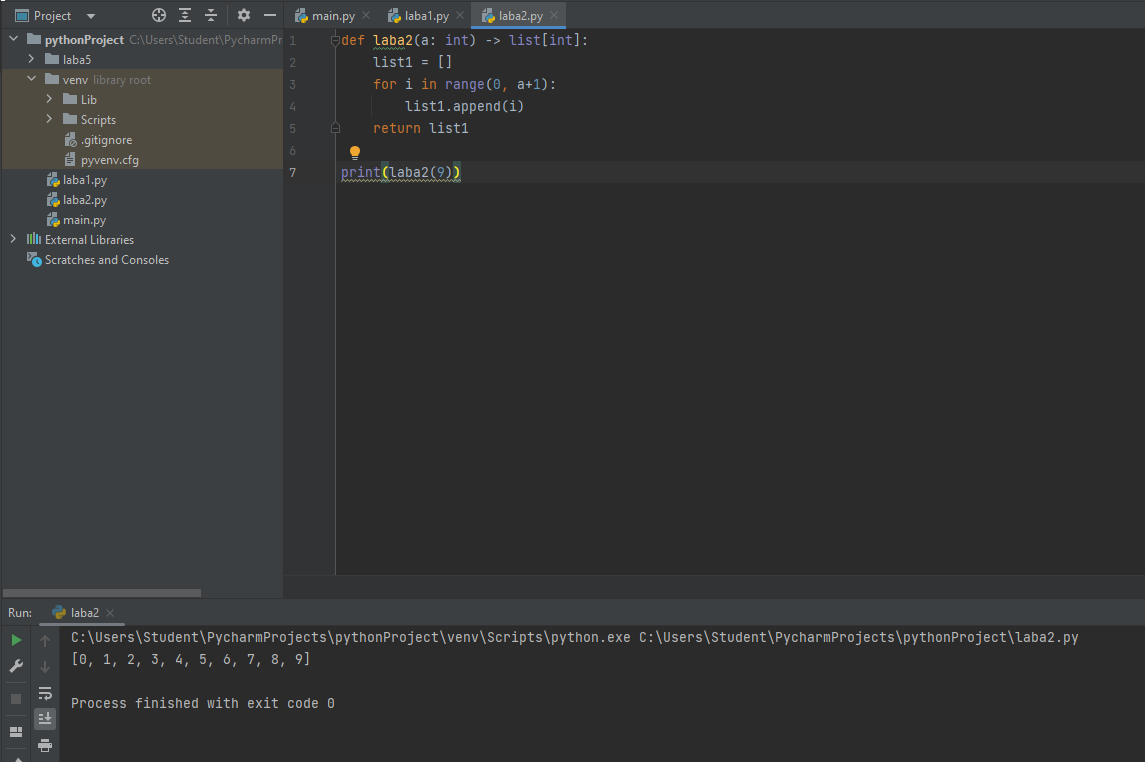
Следует отметить, что данная функция создает не список, а отдельный тип неизменяемой последовательности.

**sum(iterable[, start])** Функция возвращает сумму значений аргумента параметра iterable, начиная со start. По умолчанию параметр start равен 0.

С большим количеством встроенных функций можно ознакомиться в документации по Python в разделе «The Python Standard Library – Built-in Functions» или по ссылке [10].

**Задание**

Написать функцию, которая принимает целое число и возвращает сгенерированный список, содержащий количество элементов от 0 до входящего числа включительно.



2.1 Программный код с результатом в Pycharm.

**Вывод**: в результате были изучены основные встроенные функции языка программирования Python и способы взаимодействия с ним.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Принял | Елкин Н.С. |

**Лабораторная работа №3**

**РАБОТА С ИТЕРАТОРАМИ, ГЕНЕРАТОРАМИ. РАБОТА С ГЕНЕРАТОРНЫМИ ВЫРАЖЕНИЯМИ**

**Вариант №11**

**Цель работы:** изучить понятия итератора и генератора в Python, а также их преимущества; ознакомиться с примерами их пользования.

**Краткая теория**

*Итераторы* – популярный поведенческий паттерн проектирования для последовательного обхода коллекции, который позволяет не раскрывать их внутреннего представления.

*Итерируемый объект* – это такой объект, от которого можно получить итератор. В Python итерируемым объектом является такой объект, от которого встроенная функция iter() возвращает итератор.

*Итератором в Python* является объект, который реализует метод \_\_next\_\_ без аргументов и метод \_\_iter\_\_. Метод - \_\_next\_\_ должен вернуть следующий элемент или ошибку StopIteration.

Преимущества использования итераторов, как было сказано выше, заключается в возможности «указывать» на определенный объект коллекции и при этом скрывать его структуру. Все последовательности (list, tuple, range) в Python являются итерируемыми объектами.

Основным местом работы с итераторами в данной лабораторной работе будет использование цикла for. Например, при переборе элементов списка или другой последовательности, используя цикл for, фактически происходит обращение к итератору данной последовательности с просьбой выдать следующий элемент. Когда элементы в последовательности заканчиваются, очередное обращение к следующему объекту итератора сгенерируют исключение, которое при использовании цикла for обрабатывается незаметно для пользователя. На рис. 3.1 представлен обход списка с помощью цикла for, а на рис. 3.2 непосредственное использование итератора в работе со списком.

Для того, чтобы получить итератор последовательности, необходимо в качестве аргумента x встроенной функции iter(x) 41 передать эту последовательность. Затем для получения следующего элемента последовательности с помощью другой встроенной функции next(iter) в качестве аргумента iter передавать полученный итератор.

*Генератор (генераторная функция)* – это функция, которая возвращает подвид итератора, генерирующий значения. Основное их отличие в том, что они экономят память за счет того, что хранят не все значения, например, списка, а только его предыдущий элемент, предел и формулу, по которой рассчитывается следующий элемент. Данная функция вместо return содержит ключевое слово yield, которое возвращает объект-генератор, а 42 не выполняет сразу весь код. В Python имеется возможность создавать генераторный объект в сокращенной форме. Для этого используются круглые скобки.

Перебрать генератор можно используя цикл for как и при работе с итераторами. Однако стоит заметить, что перебрать второй раз генератор не получится, потому что объектгенератор уже сгенерировал данные по описанной в нем формуле. Поэтом генераторы стоит использовать, когда требуется один раз пройтись по итерируемому объекту.

В Python вводится такое понятие как list comprehension, которое в русскоязычном литераторе встречается в виде генераторных выражений или генераторов списков. В данном пособии будем употреблять первый вариант. Основная задача использования генераторных выражений – это быстрое создание и заполнение списков, словарей или множеств. Отметим, что генераторные выражения – это всего лишь «синтаксический сахар», иначе говоря, они не вносят никаких дополнительных преимуществ, кроме как удобство.

**Задание**

Написать функцию, которая принимает список, состоящий из n элементов, и два целых числа a и b и с помощью генераторной функции выводит на консоль элементы входящего списка от a до b включительно. Требуется предусмотреть ситуацию, когда значения a или b больше длины списка.

Результат работы программы представлен на рисунке.

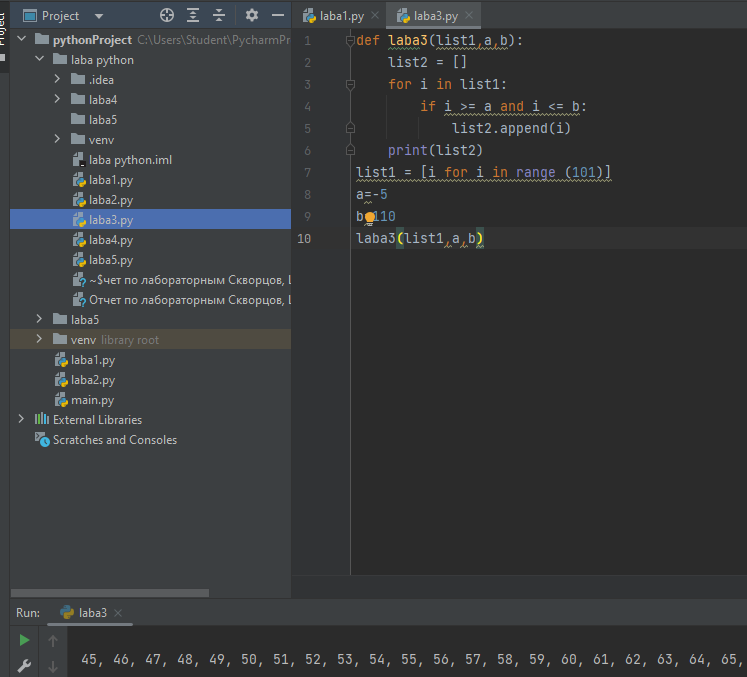


Рисунок 3.1. Программный код с результатом в Pycharm.

**Вывод**: Было освоено понятия итератора и генератора в Python, их преимущества, а так же с примерами их пользования.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Проверил | Елкин Н.С. |

**Лабораторная работа №4**

**РАБОТА С ОСНОВНЫМИ МОДУЛЯМИ**

**Вариант №11**

**Цель работы:** изучить основные модули стандартной библиотеки Python 3; рассмотреть модули os и datetime.

**Краткая теория**

Модули в Python – это файлы с расширением .py, которые содержат исполняемый код на Python. Именем модуля является имя файла без расширения. Система модулей позволяет логически организовывать код программы. Группирование кода в модули значительно облегчает процесс написания, а что более важно – дальнейшее понимание написанной программы. Для того, чтобы использовать файл .py как модуль в другом файле, требуется использовать ключевое слово import.

Стандартная библиотека Python содержит достаточно большое количество встроенных модулей. Рассмотрим некоторые из них:

− модуль os. Представляет множество функций для работы с операционной системой. При этом стоит учитывать, что их поведение, как правило, не зависит от конкретной операционной системы, что позволяет разрабатывать кроссплатформенные решения с использованием данного модуля;

− модуль datetime. Содержит классы для работы с датой и временем. В отличие от других языков программирования данный модуль достаточно прост в освоении и использовании; − модуль array. Представляет собой реализацию массивов на Python. Они похожи на списки, но их различает то, что массивы ограничены одним определенным типом данных и размером каждого элемента;

− модуль itertools. Содержит сборник полезных итераторов;

− модуль sys. Модуль, позволяющий обращаться к некоторым переменным и функциям, которые взаимодействуют с интерпретатором Python;

− модуль random. Данный модуль, как следует из названия, предназначен для генерации случайной последовательности 47 чисел, букв и случайного выбора элементов последовательности;

− модуль math. Предоставляет набор функций для выполнения математических операций;

− модуль json. Содержит функции для работы с форматом передачи данных json. Предоставляет возможности как парсинга данных, так и создания объектов в данном формате;

− модуль gzip и zlib. Предоставляет средства для работы со сжатыми файлами;

− модуль tkinter. Данный модуль позволяет создавать кроссплатформенные программы с графическим интерфейсом.

Данные модули всего лишь небольшая часть стандартной библиотеки Python. Существуют модули для работы со звуком, базами данных, шифрованием, хешированием, логгированием и многим другим. Остановимся более подробно на некоторых модулях, которые понадобятся в дальнейших работах.

**Модуль os**

Как было сказано ранее, данный модуль предоставляет функции для работы с операционной системой:

− os.listdir(path) – функция позволяет получить список файлов и директорий по указанному пути аргумента параметра path;

− os.mkdir(path) – функция позволяет создать новую директорию. В случае, если такая существует, пробрасывает OSError;

− os.remove(path) – функция удаляет файл;

− os.rename(src, dst) – функция переименовывает файл или директорию из src в dst;

− os.rmdir(path) – удаляет пустую директорию.

Стоит выделить также вложенный модуль os.path, который предоставляет средства для работы с путями:

− os.path.exists(path) – функция возвращает True, в случае если path указывает на существующий путь или дескриптор открытого файла;

− os.path.isfile(path) – функция возвращает True, если path указывает на файл;

− os.path.isdir(path) – функция возвращает True, если path указывает на директорию;

− os.path.join(path1[, path2[, ...]]) – функция соединяет пути с учетом особенностей операционной системы;

− os.path.samefile(path1, path2) – функция возвращает True, если path1 и path2 указывают на один и тот же файл или директорию.

**Модуль datetime**

− datetime.date(year, month, day) – класс date создает объект даты по значениям аргументов указанных параметров. В случае, если один из аргументов будет некорректным, сгенерируется исключение ValueError;

− datetime.time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None) – класс time создает объект времени по значениям аргументов указанных параметров;

− datetime.today() – возвращает объект datetime по текущей дате и времени;

− datetime.strptime(date\_string, format) – преобразует строку в datetime;

− datetime.strftime(datetime\_obj, format) – формирует строку из объекта datetime в указанном формате.

Следует заметить, что стандартная библиотека – это не единственное место, где можно найти полезные модули. Сообщество Python поддерживает обширную коллекцию сторонних модулей. Их можно найти в репозитории сообщества по ссылке https://pypi.python.org/. Чтобы установить сторонний модуль, можно воспользоваться системой управления пакетами pip, который устанавливается вместе с интерпретатором Python. Для установки модуля требуется ввести в окне консоли следующую команду pip install <наименование\_модуля>.

**Задание**

Написать функцию, которая принимает путь к директории и возвращает список с наименованиями файлов, которые хранятся в полученной директории.

Результат работы программы представлен на рисунке.

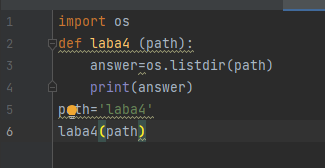


Рисунок 4.1. Программный код.

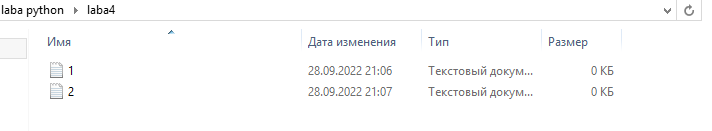


Рисунок 4.2-файлы в директории

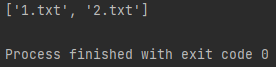


Рисунок 4.3-результат работы программы

**Вывод:** Были изучены основные модули стандартной библиотеки Python 3,а также рассмотрели модули os и datetime.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Проверил | Елкин Н.С. |

**Лабораторная работа №5**

**РАБОТА С ФАЙЛАМИ. РАЗРАБОТКА СИНТАКСИЧЕСКОГО АНАЛИЗАТОРА. ВЫВОД ФОРМАТИРОВАННЫХ ДАННЫХ В ФОРМАТЕ JSON**

**Вариант №11**

**Цель работы:** изучить работу с файлами с помощью функций из стандартной библиотеки; рассмотреть понятие синтаксического анализа текста и ознакомиться с его применением на языке Python; изучить возможности взаимодействия Python с форматом хранения данных JSON.

**Краткая теория**

**Работа с файлами**

Python поддерживает работу с множеством различных типов файлов. Условно их можно разделить на текстовые и бинарные. Текстовые файлы – это те файлы, которые хранят данные в текстовом виде, а бинарные – это, к примеру видео, изображение и прочее.

Python позволяет работать с файлами двумя способами:

− непосредственного открытия и закрытия файла с помощью встроенных функций open(path, mode) и close();

− с применением конструкции with.

Чтение файла и запись в файл осуществляются с помощью методов read() и write() соответственно. Python позволяет работать с файлом в определенных режимах, передаваемых функции open в качестве строкового аргумента для параметра mode:

1) «r» (Read) – файл открывается для чтения. Если файл не найден, то генерируется исключение FileNotFoundError;

2) «w» (Write) – файл открывается для записи. В любом случае создается новый. Следовательно, старые данные будут удалены;

3) «a» (Append) – файл открывается для дополнительной записи. Если файл отсутствует, то создастся новый. При существовании такого файла, данные запишутся в конец;

4) «b» (Binary) – используется для работы с бинарными файлами и применяется вместе с другими режимами («wb», «rb», «ab»);

5) «+» – файл открывается одновременно на чтение и запись. Применяется вместе с другими режимами («r+», «a+», «wb+» и т.д.). Стоит выделить комбинации «r+» и «w+» при таких режимах, оба файла будут открыты на одновременное чтение и запись, но в первом случае при отсутствии файла возникнет исключение, а во втором создастся новый файл.

Как было отмечено ранее, для работы с файлами можно использовать встроенные функции open() и close() и конструкцию with. В первом случае стоит учитывать, что во время работы с файлами могут быть сгенерированы различные исключения, что может привести к не выполнению функции close() и соответственно не закрытию файла. Для предотвращения данной ситуации применяется конструкция try, где в блоке finally обязательно вызовется функция close(). Однако, чтобы избежать данных дополнительных действий существует конструкция with, которая внутри себя уже реализует обработку исключений и обязательного закрытия файла.

**Задание**

Одна строка рандомные буквы. Нужно вывести словарь, ключ- буква(а.б ….), кол-во повторений – значение.

Результат работы программы представлен на рисунке.

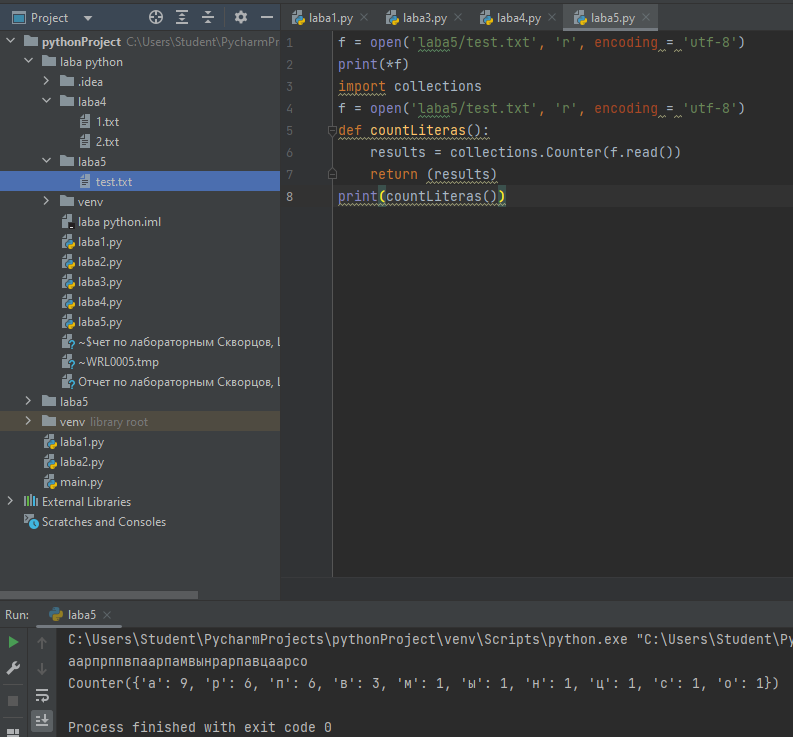


Рисунок 5.1. Программный код с результатом в Pycharm.

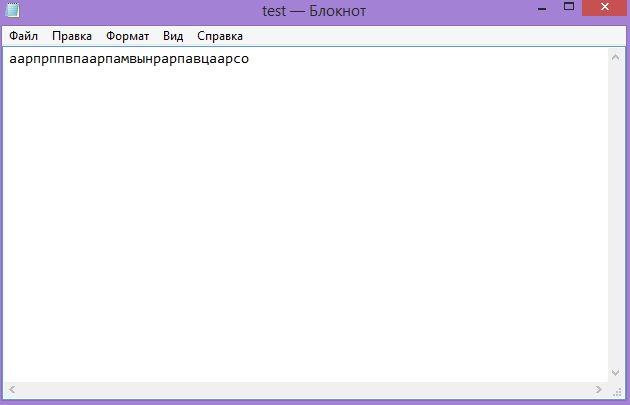


Рисунок 5.2. Текст файла

**Вывод:** Была изучена работа с файлами с помощью функций из стандартной библиотеки; рассмотрено понятие синтаксического анализа текста и его применения на языке Python. Так же изучена возможность взаимодействия Python с форматом хранения данных JSON.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Проверил | Елкин Н.С. |

**Лабораторная работа № 6**

**РАЗРАБОТКА ПРИЛОЖЕНИЯ РАБОТЫ С БАЗОЙ ДАННЫХ**

**Вариант № 11**

**Цель работы:** изучить возможности взаимодействия Python с реляционными базами данных с помощью DB-API 2.0

Почти каждая программа, которая работает с данными, требует места для их хранения. В качестве такого хранилища могут выступать обычные текстовые файлы, файлы JSON или XML. Однако для хранения большого количества данных и удобной организации работы с ними существуют СУБД. В данной лабораторной работе не рассматриваются понятия и термины баз данных, синтаксис SQL. Здесь будут рассмотрены механизмы для работы с базами данных с помощью Python. DB-API 2.0 Несмотря на существующий стандарт SQL (ISO/IEC 9075) каждая СУБД имеет ряд отличий от других. Для того, чтобы программисту не вникать в их реализации в Python существует общее API, описанное стандартом PEP 249. Стоит учесть, что PEP 249 это всего лишь спецификация, реализация которой выполняется самостоятельно. Однако существуют уже готовые реализации данного API для ряда СУБД. Стоит отметить, что Python имеет в качестве встроенного модуля одну из реализаций данного API для работы с SQLite. Ниже приведены некоторые реализации для других СУБД:

− Oracle: cx\_Oracle, mxODBC, pyodbc;

− MySQL: mysql-python, PyMySQL;

− MS SQL Server: adodbapi, pymssql, mxODBC, pyodbc;

− PostgreSQL: psycopg2, txpostgres.

Для подключения реализации рассматриваемого API при работе с Sqlite необходимо импортировать пакет sqlite3 из стандартной библиотеки Python. Чтобы открыть соединение с базой необходимо использовать функцию connect(db\_file\_path), которая в качестве значения аргумента принимает путь к файлу SQLite. При работе с другими СУБД потребуется указать дополнительные параметры функции, такие как хост, имя пользователя, пароль и наименование базы. Для выполнения запросов и получения результатов требуется использовать специальный объект Cursor, который можно получить, вызвав метод cursor() из объекта Connection, который возвращается из функции connect(). Вызов метода execute(query) для объекта Cursor выполняет запрос и в случае наличия результата возвращает ответ, который можно получить методом fetchall(). Стоит отметить, что данный метод возвращает значение только один раз, повторное обращение к нему без предварительного запроса вернет пустой список. Для закрытия соединения с базой данных используется метод close() объекта Connection.

**Задание**

Написать функцию, которая принимает наименование таблицы, поля и его значение и возвращает идентификатор записи, в которой значение полученного поля соответствует переданному функции, или возвращает None.

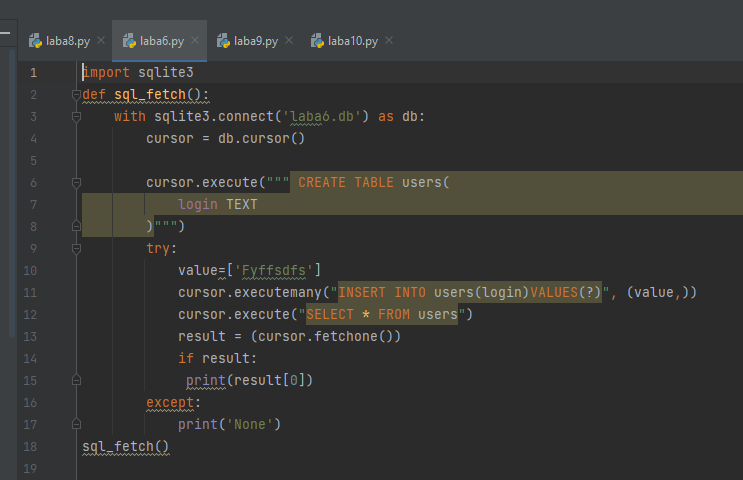


Рисунок 6.1. Программный код с результатом в Pycharm.

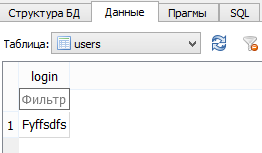


Рисунок 6.2 – результат работы программы.

**Вывод**: в результате проделанной работы было изучено использование возможностей взаимодействия Python с реляционными базами данных с помощью DB-API 2.0.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Проверил | Елкин Н.С. |

**Лабораторная работа № 7**

**РАЗРАБОТКА ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ООП**

**Вариант № 11**

**Цель работы:** ознакомиться с методологией объектно-ориентированного программирования, изучить реализацию данной методологии в языке Python 3.

**Краткая теория**

Объектно-ориентированное программирование (ООП) – это одна из методологий (парадигм) программирования (структурное, функциональное), где компоненты программы представляются в виде объектов. Объект – это экземпляр класса, имеющего поля и методы. Большинство современных языков программирования, таких как Java, C++, C# и другие, реализуют данный подход, в том числе и Python. В данной лабораторной работе не рассматривается теория ООП, а только реализация данной методологии в Python3.

**Модификаторы доступа. Инкапсуляция**

В Python существуют три вида модификаторов доступа: public, protected, private. Модификатор доступа public разрешает доступ к переменным из любой точки вне и внутри класса. Модификатор доступа private допускает обращения к переменным только внутри класса. Модификатор доступа protected разрешает доступ из любой точки вне и внутри класса, внутри пакета, а также в классах-наследниках.

Инкапсуляция – это сокрытие данных одного класса от прямого обращения к его данным другими классами. Суть заключается в том, чтобы предоставлять доступ к переменным класса через методы геттеры и сеттеры, в которых может происходить логика, связанная с работой защищаемой переменной.

**Наследование**

Наследование – это предоставление характеристик классародителя классу-потомку. Данный механизм позволяет в несколько раз сократить количество кода за счет его повторного использования. Стоит также отметить, что в Python существует поддержка множественного наследования.

**Полиморфизм**

Полиморфизм заключается в способности объекта вести себя по-разному. Под полиморфизмом рассмотрим перегрузку и переопределение методов. Перегрузка метода – это способность метода вести себя по-разному при определенном наборе параметров. Переопределение метода заключается в изменении логики метода с таким же наименованием и количеством параметров, что и метод в классе-родителе.

**Задание**

Написать родительский и дочерний классы с методами «display». В каждом методе в консоль выводится различная строка.

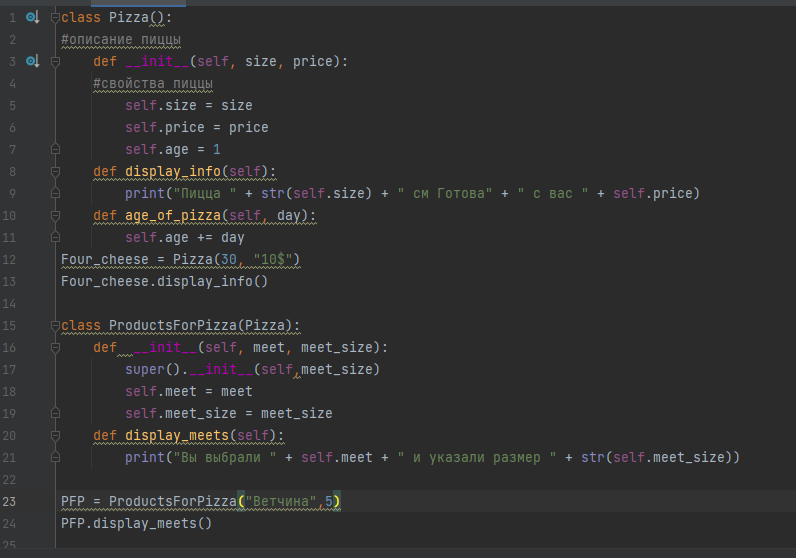
****

Рисунок 7.1. Программный код в Pycharm.

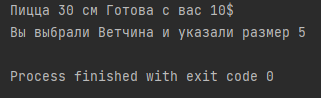


Рисунок 7.2 – результат работы программы.

**Вывод**: в результате проделанной работы было освоена методология объектно-ориентированного программирования и изучена реализация данной методологии в языке Python 3.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Проверил | Елкин Н.С. |

**Лабораторная работа № 8**

**ОБРАБОТКА ИЗОБРАЖЕНИЙ С ПРИМЕНЕНИЕМ БИБЛИОТЕКИ PIL**

**Вариант № 11**

**Цель работы:** рассмотреть возможности работы библиотеки PIL, изучить её базовые инструменты для обработки изображений.

**Краткая теория**

Для Python существует огромное количество полезных библиотек, одной из таких является библиотека для обработки изображений – Python Image Library (PIL). Она обеспечивает работу с разнообразными форматами файлов и мощные инструменты для обработки изображений. С помощью PIL можно создавать миниатюры изображений (thumbnails), конвертировать изображения в различные форматы, изменять размер, вращать изображение, а также с помощью данной библиотеки можно реализовать функцию отправки изображения на печать. Для того, чтобы загрузить PIL в командной строке или терминале, требуется ввести следующую команду: pip install Pillow.

**Основные функции для работы с изображениями.**

Чтобы загрузить изображение из файла, требуется вызвать метод open(path\_to\_file) класса Image, который можно импортировать с помощью команды from PIL import Image, передав в качестве аргумента для параметра path\_to\_file путь к файлу с изображением. В случае успешного чтения файла метод open() вернет объект класса Image, в противном случае возникнет исключение IOError. Из полученного объекта, обращаясь к полям format, size и mode, можно получить формат изображения, его размер (ширину и высоту) в пикселях и схему формирования цвета (Luminance, RGB, CMYK) соответственно. Для того, чтобы конвертировать изображение в другой формат, достаточно его сохранить заново с другим расширением с помощью метода save(new\_file\_name), где в качестве аргумента для параметра new\_file\_name требуется указать путь нового изображения с наименованием и требуемым расширением.

**Обработка изображения**

Часто при обработке изображений требуется обработать его цвета. PIL предоставляет удобные инструменты для работы с различными фильтрами, обработки цветовых каналов изображения, изменения контраста, яркости, цветового баланса и др. 73 PIL предоставляет широкий спектр фильтров для обработки изображений, которыми можно воспользоваться с помощью класса ImageFilter. Рассмотрим обработку изображения с помощью фильтра размытие по Гауссу. Для получения объекта рассматриваемого фильтра требуется вызывать метод GaussianBlur(radius=2), значением параметра которого можно изменить на желаемое число. Данный объект требуется передать в качестве аргумента методу filter(img\_filter) объекта класса Image. Для того, чтобы отобразить результат, можно вызвать метод show() у объекта изображения. Таким образом, Pillow – это мощнейший инструмент для обработки изображений на Python.

**Задание**

Написать класс, который содержит функцию, принимающую путь к gif изображению и делает его раскадровку, сохраняя изображения по тому же пути, что и gif. Вторая функция будет принимать директорию с изображениями и записывать их в gif.

****

Рисунок 8.1. Программный код в Pycharm.

****

Рисунок 8.2. Исходное gif изображение.

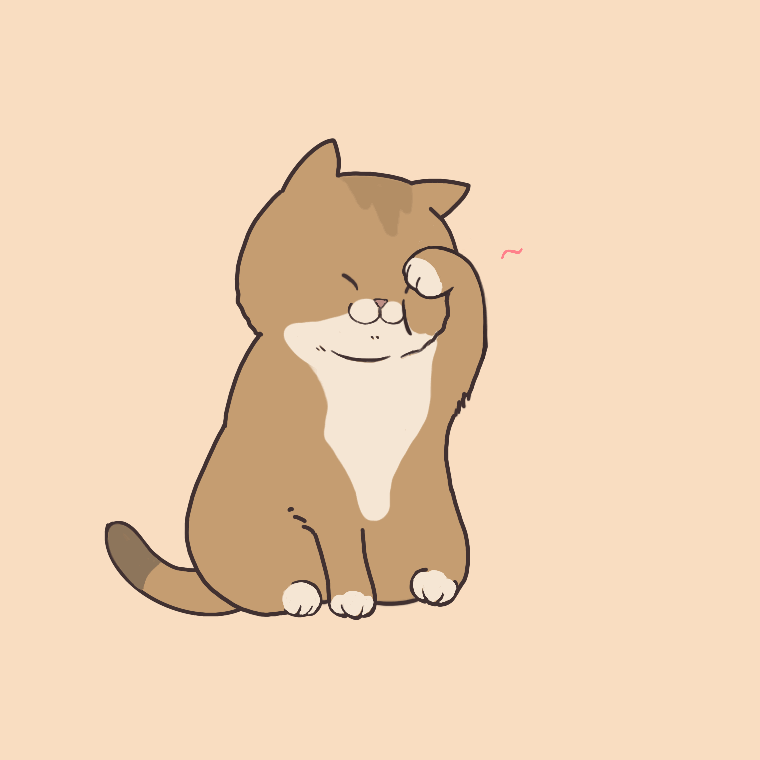


Рисунок 8.3. Исходное gif изображение разбитое на 14 кадров.

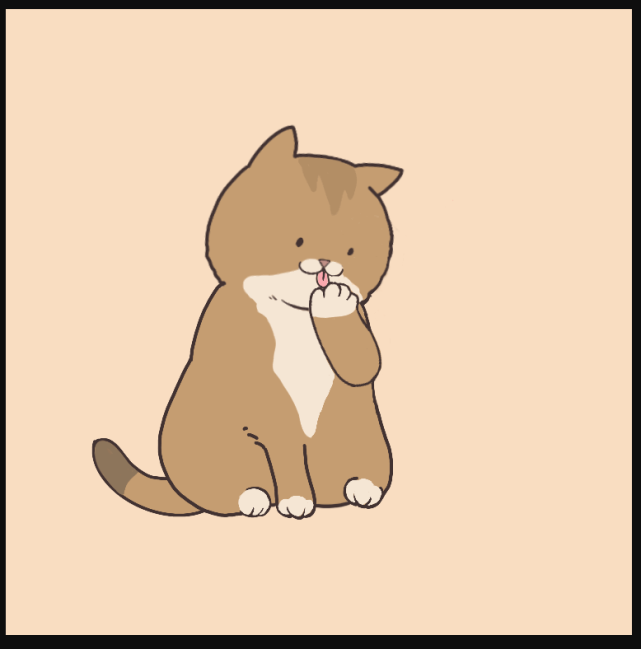


Рисунок 8.4. Gif изображение собранное из разбитых кадров.

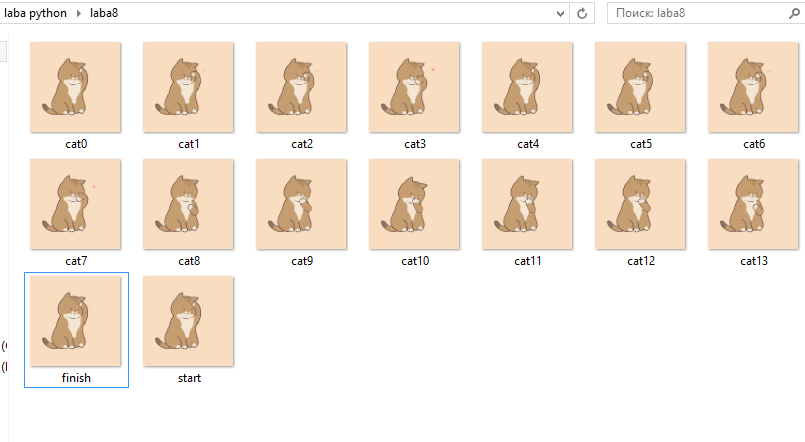


Рисунок 8.4. Папка после выполнения программы.

**Вывод**: в результате проделанной работы были освоены возможности работы библиотеки PIL, и изучены её базовые инструменты для обработки изображений.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Проверил | Елкин Н.С. |

**Лабораторная работа № 9**

**РАЗРАБОТКА GUI ПРИЛОЖЕНИЯ С ПОМОЩЬЮ ГРАФИЧЕСКИХ БИБЛИОТЕК**

**Вариант № 11**

**Цель работы:** рассмотреть возможности библиотеки Tkinter, её базовые виджеты и изучить основные принципы создания приложений с графическим интерфейсом с помощью данной библиотеки.

**Краткая теория**

Python не является популярным языком для написания приложений с графическим интерфейсом (GUI) под Windows, однако его часто используют при написании таких приложений под Linux. В настоящее время существует достаточно большое количество графических библиотек для Python 3. Например, некоторые из них: − Tkinter; − wxPython; − PyQt; − PyGTK; − Kivy; − Flexx. В данной лабораторной работе не рассматривается применение всех вышеуказанных библиотек, а только работа с Tkinter. Tkinter (Tk interface) – это библиотека, которая поставляется вместе с интерпретатором Python и как и другие предназначена для создания кросс-платформенных графических интерфейсов. Стоит отметить, что она поставляется в том случае, если вместе с интерпретатором устанавливается IDLE, потому что, как не трудно догадаться, он разработан с помощью данной библиотеки. Для подключения библиотеки в файл скрипта достаточно выполнить команду import tkinter. Tkinter предоставляет достаточное количество различных графических элементов, таких как кнопки, метки, текстовые контейнеры и другие. В рамках рассматриваемой библиотеки они называются виджетами (widgets).

Ниже приведены некоторые из них:

− Button. Виджет, используемый для отображения кнопок в приложении.

− Canvas. Предназначен для фигур, таких как линии, овалы, треугольники и другие.

− Checkbutton. Реализует функции выбора вариантов с поддержкой множественного выбора. Аналогичен работе компонента Checkbox в других средах.

− Frame. Используется в качестве контейнера для других виджетов.

− Label. Представляет собой однострочную строковую метку. Также может содержать изображения.

− Listbox. Используется для отображения списков.

− Text. Предназначен для отображения многострочного текста.

− tkMessageBox. Предназначен для отображения сообщений пользователю в новом окне.

− Radiobutton. Аналогичен назначению виджета Checkbutton с учетом того, что не поддерживается множественный выбор.

Проблему при создании приложений с графическим интерфейсом может вызывать тот факт, что они работают в одном потоке, т.е. все операции, обрабатывающиеся в таком приложении, будут выполняться последовательно. Это приведет к тому, что, например, при написании программы, которая будет обращаться к веб-серверу или выполнять долгие математические расчеты графический интерфейс будет неактивен. Для этого требуется воспользоваться модулем threading. Таким образом, Tkinter представляет собой удобный и мощный инструмент для создания кроссплатформенных приложений с графическим интерфейсом «из коробки».

**Задание**

Написать GUI приложение для работы с базой данных. Приложение позволяет выполнять соединение с БД, а также выполнять запросы и получать результаты ответов.

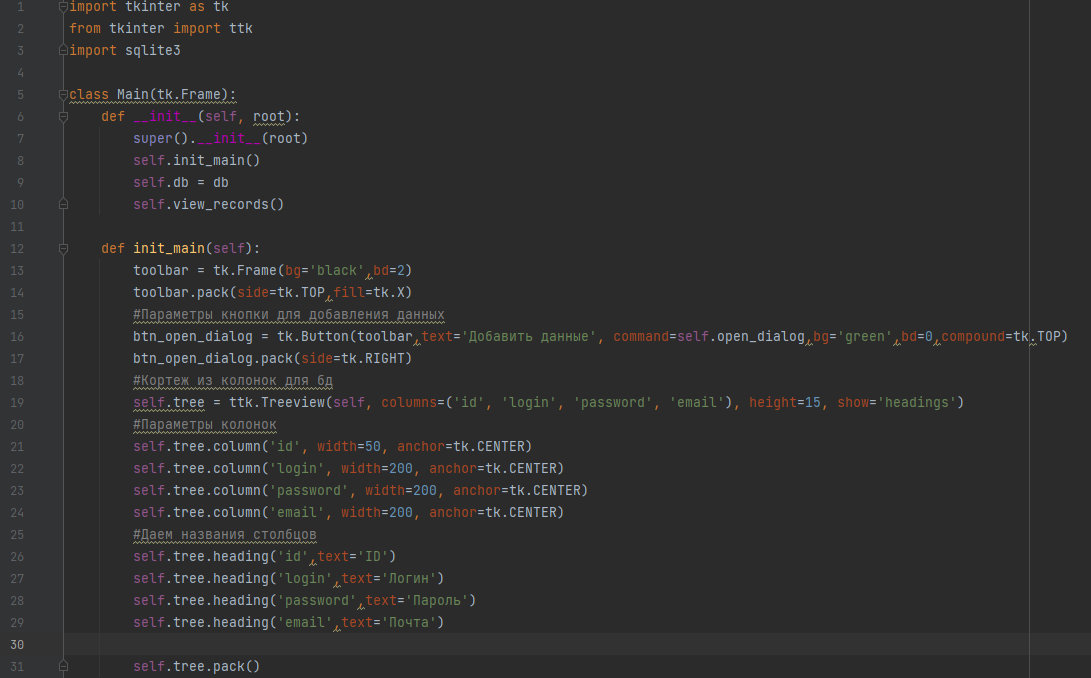


Рисунок 9.1. Программный код в Pycharm.

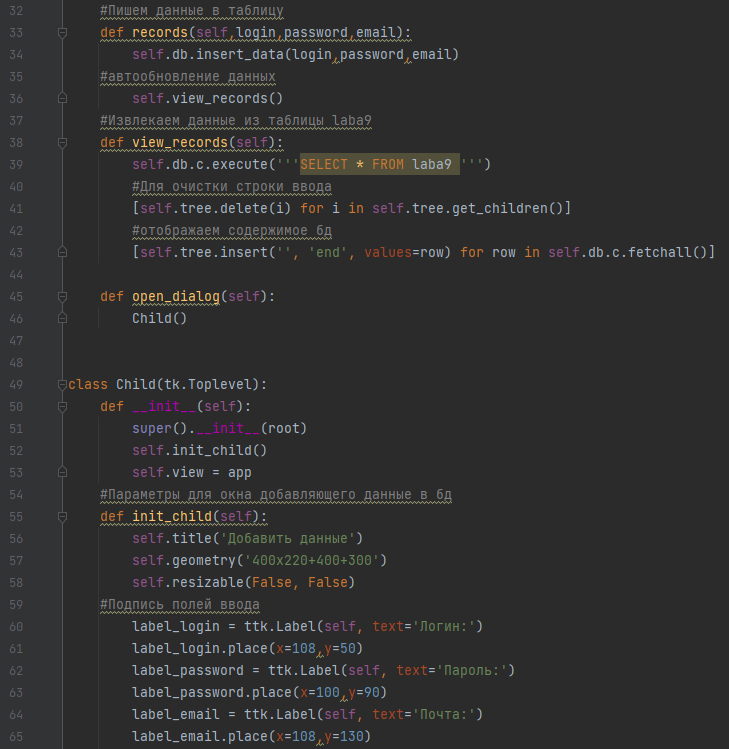


Рисунок 9.2. Программный код в Pycharm.

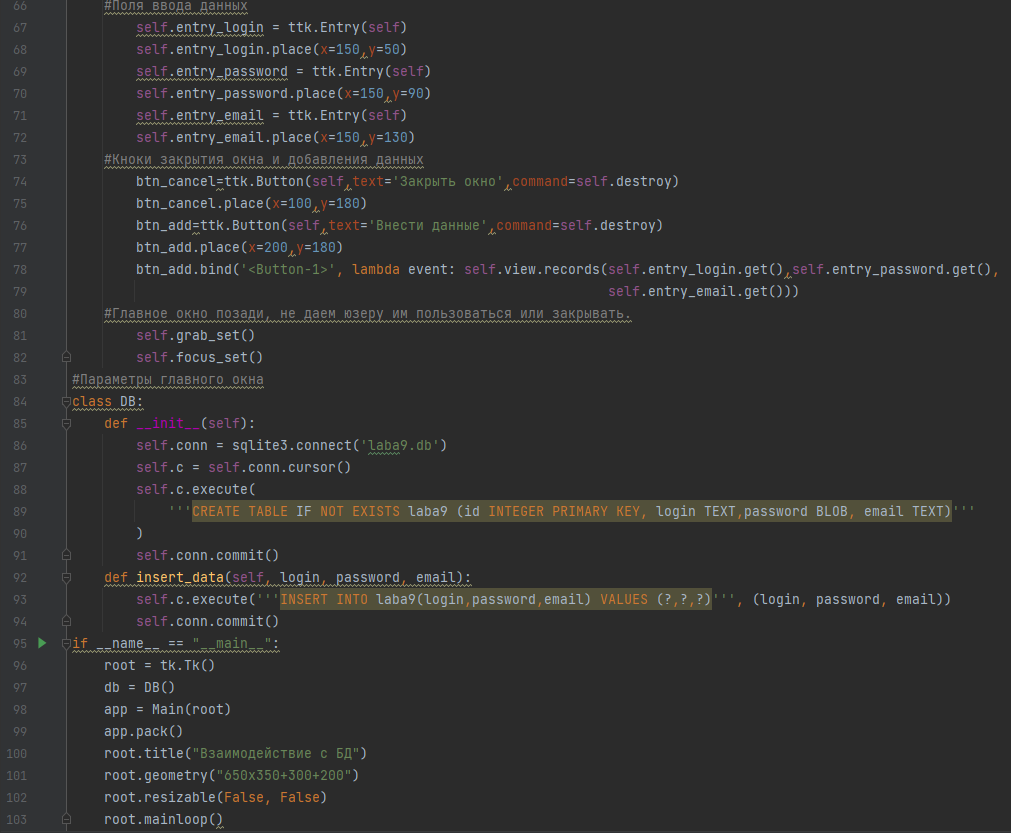


Рисунок 9.3. Программный код в Pycharm.

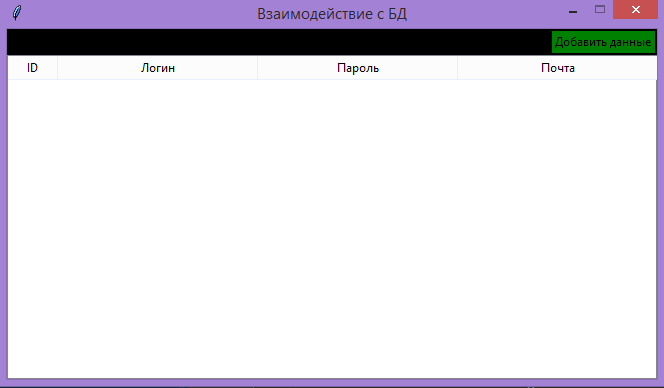


Рисунок 9.4. Окно для отображения данных в БД.

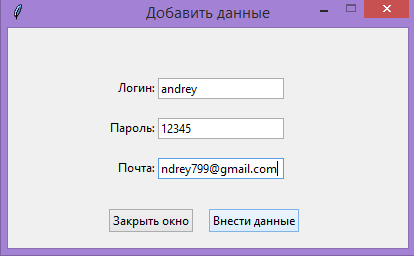


Рисунок 9.5. Окно для добавления данных в БД.

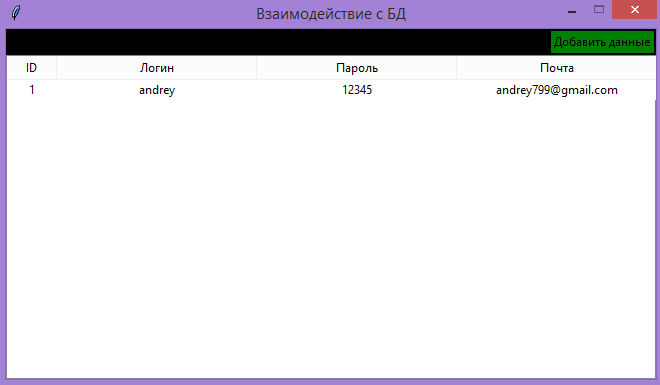


Рисунок 9.6. Отображение внесенных в БД данных.

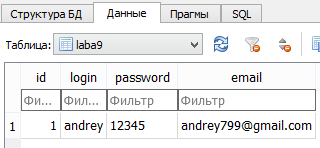


Рисунок 9.7. Записанные в БД данные.

**Вывод**: в результате проделанной работы было изучено использование возможностей библиотеки Tkinter, её базовые виджеты и изучены основные принципы создания приложений с графическим интерфейсом с помощью данной библиотеки.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Проверил | Елкин Н.С. |

**Лабораторная работа № 10**

**ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ РАБОТЫ МАТЕМАТИЧЕСКИХ АЛГОРИТМОВ С ИСПОЛЬЗОВАНИЕМ NUMPY И MATPLOTLIB**

**Вариант № 11**

**Цель работы:** рассмотреть возможности пакета NumPy, его основные составляющие и возможности для обработки многомерных массивов данных; изучить библиотеку Matplotlib и её основные инструменты для создания графиков и диаграмм.

**Краткая теория**

NumPy – это пакет для научных расчётов на Python. Он содержит ряд особенностей:

− удобная и эффективная работа с обработкой многомерных данных;

− инструменты для интеграции кода на C/C++;

− работа с линейной алгеброй;

− поддержка огромного количества полезных алгоритмов. Основным объектом рассматриваемого модуля является однородный многомерный массив (ndarray). Массивы аналогичны спискам, за исключением того, что они хранят только определенный тип данных, а работают они в несколько раз быстрее, чем списки. Создать простой массив в numpy можно с помощью функции numpy.array(x), где в качестве аргумента для параметра x можно передать список или кортеж. Стоит отметить, что данная функция возвращает объект типа ndarray. Вложенные списки или кортежи numpy сама формирует в многомерные массивы. Ниже приведены наиболее важные атрибуты объекта ndarray:

− ndim. Число измерений массива;

− shape. Размерность массива. Представляет собой кортеж, состоящий из двух чисел (m,n)

− size. Общее количество элементов массива;

− dtype. Тип элементов массива.

Matplotlib – это open-source библиотека для построения графиков. С помощью данной библиотеки можно просто и быстро генерировать огромное количество различных графиков, гистограмм, диаграмм и многое другое, укладываясь всего лишь в несколько строк кода. С помощью модуля pyplot данная библиотека предоставляет интерфейс похожий на интерфейс Matlab. Данный модуль предоставляет полный контроль над стилями линий, свойствами шрифта и осей и т.д. через объектно-ориентированный интерфейс или набор функций, которые знакомы пользователям Matlab. Matplotlib поддерживает работу с NumPy.

В данной лабораторной работе будет рассмотрен модуль pyplot, который имеет следующие функции:

⎯ scatter(x, y) – строит точку на графике с возможностью изменения размера маркера и цвета;

⎯ plot(arr\_x, arr\_y) – строит ломаную линию, принимает в качестве аргументов массив координат по оси X и по оси Y соответственно;

⎯ text(x, y, str) – добавляет текст на график. Функция принимает координаты текста по двум осям и саму строку, которую надо отобразить;

⎯ bar(arr\_x, arr\_height, width=0.8) – создает столбчатую диаграмму. Принимает массив координат по оси X и мас- 85 сив высот, так же можно указать ширину графика, которая по умолчанию равно 0.8;

⎯pie(arr\_x, labels=None, colors=None) – создает круговую диаграмму. Принимает массив размеров каждой из частей диаграммы, также дополнительно можно указать массив меток и цветов для каждой части.

**Задание**

Написать функцию, которая строит гистограмму популярных языков программирования. Данные можно взять из последних данных TIOBI.

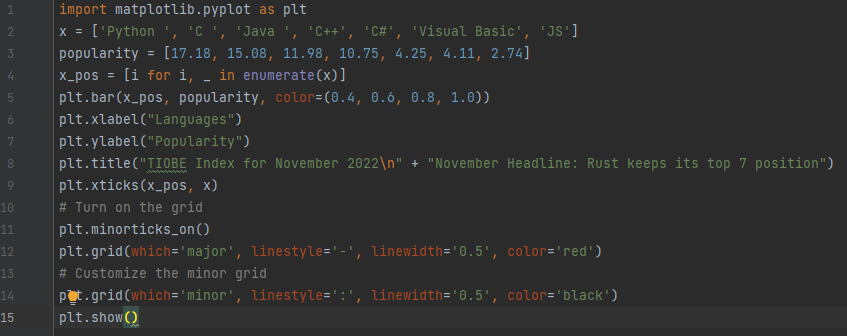
****

Рисунок 10.1. Программный код в Pycharm.

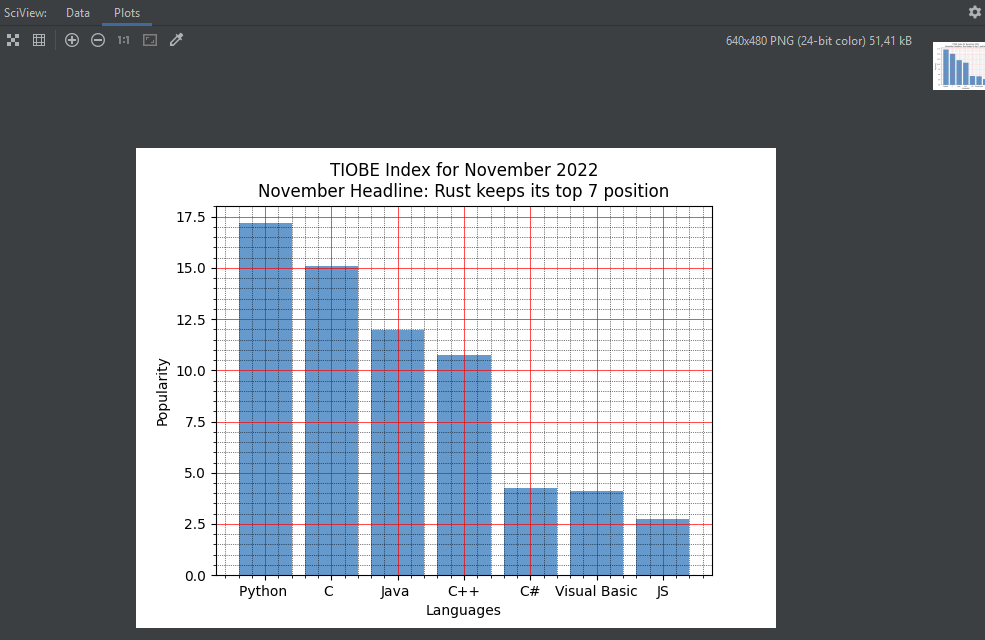
****

Рисунок 10.2. Результат работы программы.

**Вывод**: в результате проделанной работы было освоено использование возможности пакета NumPy, его основные составляющие и возможности для обработки многомерных массивов данных; изучены библиотеки Matplotlib и её основные инструменты для создания графиков и диаграмм.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Проверил | Елкин Н.С. |

**Лабораторная работа № 11**

**ОСНОВЫ РАБОТЫ С ВЕБ-ФРЕЙМВОРКОМ DJANGO**

**Вариант № 11.**

**Цель работы:** рассмотреть возможности веб-фреймворка Django для построения веб-приложений на языке Python, ознакомиться с понятием ORM и его реализации в данном фрейм-ворке.

**Краткая теория**

Django – это веб-фреймворк для Python, основным преимуществом которого является высокая скорость разработки и «чистая» архитектура проекта. Данный фреймворк является open-source проектом. Архитектура Django предоставляет возможность быстрого и гибкого масштабирования. Данный фреймворк несет в себе множество дополнительного функционала «из коробки», такого как аутентификация пользователей, администрирования контента, панель администратора для работы с базой данных и многое другое. Для работы с базой данных Django, как и многие другие веб-фреймворки используют ORM (Object-Relational Mapping). Это технология, позволяющая связать базы данных с парадигмой ООП. Данная технология позволяет не только не привязываться к определенной базе данных, но и генерировать архитектуру базы данных из написанных классов. В ORM предполагается, что каждая таблица является классом, а каждый атрибут (поле) класса является атрибутом таблицы. Таким образом, ORM представляет собой как бы прослойку между БД и ООП.

**Создание проекта на Django.**

Перед началом создания проекта требуется установить сам веб-фреймворк. Это делают аналогично с установкой любой библиотеки через менеджера pip. Данная команда выполняет скачивание и установку фреймворка: pip install django. После установки фреймворка можно приступить к созданию самого проекта. Для этого в командной строке или терминале требуется перейти в нужную директорию, где будет создаваться проект и выполнить команду django-admin 91 startproject newproject, где newproject наименование создаваемого проекта. Данная команда создаст следующую структуру директорий и файлов: − newproject – корневая директория проекта. − manage.py – утилита командной строки, позволяющая взаимодействовать с проектом различными способами; − newproject – директория, которая является основной для проекте и хранит в себе файл конфигурации (settings.py), менеджер URL(urls.py) и точку входа для WSGI-совместимых вебсерверов (wsgi.py). В отличие от других платформ для разработки вебприложений Django «из коробки» имеет собственный однопоточный веб-сервер для разработки приложения. Запустить на нем созданный проект можно, находясь в корневой директории проекта, с помощью команды python manage.py runserer. После чего при успешном старте сервера веб-приложение будет доступно по адресу http://127.0.0.1:8000/. Для остановки вебсервера достаточно будет нажать комбинацию клавиш Ctrl+C, находясь в командной строке или терминале, где была введена предыдущая команда. Стоит отметить, что данный механизм не следует использовать на этапе работы приложения, а только во время его разработки. Создание приложения в созданном проекте Приложением называется модуль Django проекта. В них заключается мощь «чистой» архитектуры, которую предоставляет данный веб-фреймворк. Приложения удобно можно создавать самостоятельно, а можно заимствовать модули из иных проектов. Следует отметить, что для более правильной работы с предоставляемой архитектурой необходимо разделять создаваемые или заимствованные модули и стараться не допускать зависимостей между ними. Изолированные модули в будущем можно будет легко интегрировать в другие проекты, а ненужные удалять из проекта, не принося ущерба работоспособности приложения. Итак, для того, чтобы создать новое приложение в созданном проекте необходимо, находясь в корневой директории, вы- 92 полнить команду python manage.py startapp newapp, где newapp наименование создаваемого приложения. Данная команда создаст новую директорию newapp, которая содержит следую щую структуру: − migrations – директория, которая в будущем будет содержать информацию о миграциях; − admin.py – файл для настройки отображения созданных моделей на панели администратора; − apps.py – файл для конфигурации самого приложения; − models.py – файл для создания моделей; − tests.py – файл для автоматизированных тестов; − views.py – файл, содержащий бизнес-логику приложения. К выше приведенной структуре следует и отнести файл urls.py, который создается вручную и аналогично одноименному файлу из основной директории проекта представляет собой менеджер URL для созданного приложения. После этого необходимо добавить созданное приложение в проект, добавив его наименование в список INSTALLED\_APPS файла settings.py. Работа с базой данных Ранее было сказано о миграциях и моделях. Под моделями в Django следует понимать классы, описывающие таблицы базы данных. Стандартная модель должна наследоваться от класса models.Model и содержать определенный набор атрибутов, причем каждому атрибуту присваивается определенный тип. Стоит отметить, что атрибут суррогатного ключа (id) устанавливается по умолчанию. Ниже приведены несколько типов для объявления полей в моделях: − CharField() – строковый тип, предназначенный для небольших строк; − TextField() – тип для хранения больших объемов текста; − IntegerField(), FloatField() – типы для хранения целых чисел и числе с плавающей точкой соответственно; − ImageField() – тип для хранения изображений; − EmailField() – тип для хранения адресов электронной почти; 93 − DateField(), DateTimeField() – типы для хранения даты и даты и времени соответственно; − ForeignKey() – тип для связки моделей по внешнему ключу. Требует в качестве аргумента для параметра to наименование модели, с которой происходит связь (для рекурсивной зависимости следует использовать строковое значение ‘self’) и аргумент для параметра on\_delete задающий логику действий при удалении зависимой записи.

**Шаблоны**

Под шаблонами в Django понимаются динамически сгенерированные HTML страницы. Шаблоны содержат как статическую часть, так и специальный синтаксис, описывающий, где и как будет располагаться динамический контент. Для этого используются шаблонизаторы. В Django можно использовать один или несколько шаблонизаторов. Встроенный шаблонизатор в Django называется Django template language (DTL). Настройка поддержки DTL устанавливается при создании проекта. Хранить создаваемые шаблоны необходимо в директориях под названием templates. Желательно хранить шаблоны, используемые приложениями в их директориях, а в самой директории templates создавать директорию с наименованием приложения. Это способствует устранению ошибок, т.к. при рендере шаблона фреймворк ищет его по указанному пути во всех директориях под названием templates. Синтаксис DTL содержит четыре конструкции: − переменные. Вывод значений из контекста, обрамляемый двумя фигурными скобками «{{ }}»; − тэги. Предназначены для обеспечения логики в процессе рендера. Например, использование условий или циклов. Обрамляются фигурной скобкой и знаком процента «{% %}»; − фильтры. Трансформируют значения переменных и аргументы тэгов. Указываются через вертикальную черту после значения, которое необходимо изменить « | » − комментарии. Однострочные обрамляются фигурной скобкой и знаком решетки «{# #}», а многострочные выглядят следующим образом «{% comment %} … {% comment %}»

Представления

Однако для того, чтобы провести рендер страницы, необходимо вызывать его из функции, находящейся в файле views.py. Функции, которые исполняются при вызове определенного URL, называются представлениями и указываются в файле views.py. Для этого, в файле urls.py, лежащем в корневой директории проекта, требуется указать адрес доступа к urls, находящимся в директории разработанного приложения. Например, в рассматриваемом примере требуется добавить в список urlpatterns строку path('blog/', include('blog.urls')). После этого требуется указать адрес и функцию, которая будет выполняться при обращении к заданному адресу.

**Задание**

Написать веб-приложение, которое включает регистрацию пользователей на сайте, который выполняет случайную ежедневную рассылку вдохновляющих цитат на e-mail зарегистрированных пользователей.

**Вывод**: были рассмотрены возможности веб-фреймворка Django для построения веб-приложений на языке Python, изучено понятие ORM и его реализации в данном фрейм-ворке.

|  |  |
| --- | --- |
| Выполнили | Скворцов А.А., Щупакевич М.Е. |
| Проверил | Елкин Н.С. |