РОСЖЕЛДОР

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»

(ФГБОУ ВО РГУПС)

О. В. Игнатьева, А. В. Суханов, В. Р. Хусаинов

ОБЪЕКТНО ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON

Учебно-методическое пособие
для лабораторных работ

Ростов-на-Дону
РГУПС
2021

УДК 004.4(07) + 06

Рецензент – кандидат технических наук, доцент В. В. Жуков

Игнатьева, О. В.

Объектно ориентированное программирование на языке Python : учебно-методическое пособие для лабораторных работ / О. В. Игнатьева, А. В. Суханов, В. Р. Хусаинов ; ФГБОУ ВО РГУПС. – Ростов-на-Дону : РГУПС, 2021. – 112 с.

Рассматриваются типовые задачи программирования, алгоритмы их решения и реализация этих алгоритмов на языке Python.

Для студентов направлений «Информационные системы и технологии» и «Информатика и вычислительная техника» с целью углубленного изучения программирования на языке Python на аудиторных занятиях и самостоятельного изучения материала по дисциплинам «Объектно ориентированное программирование», «Алгоритмизация и программирование», «Основы искусственного интеллекта», «Системы и технологии искусственного интеллекта», а также для обучающихся магистратуры, бакалавриата и специалитета различных направлений, изучающих дисциплины по программированию и спецкурсы.

Одобрено к изданию кафедрой «Вычислительная техника и автоматизированные системы управления».

Учебное издание

Игнатьева Олеся Владимировна
Суханов Андрей Валерьевич
Хусаинов Владимир Романович

ОБЪЕКТНО ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON

Печатается в авторской редакции
Технический редактор Т. И. Исаева

Подписано в печать 01.12.2021. Формат 60×84/16.
Усл. печ. л. 6,51. Тираж экз. Изд. № 5086. Заказ .

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)
Адрес университета: 344038, г. Ростов н/Д, пл. Ростовского Стрелкового Полка Народного Ополчения, д. 2, www.rgups.ru

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно ориентированным – всё является объектами.

Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов.

Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

Python является мультипарадигмальным языком программирования, поддерживающим императивное, процедурное, структурное, объектно ориентированное программирование, метапрограммирование и функциональное программирование. Задачи обобщённого программирования решаются за счёт динамической типизации. Аспектно ориентированное программирование частично поддерживается через декораторы, более полноценная поддержка обеспечивается дополнительными фреймворками. Такие методики как контрактное и логическое программирование можно реализовать с помощью библиотек или расширений.

Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений с глобальной блокировкой интерпретатора (GIL), высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ и являющийся стандартом де-факто языка. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные.

CPython компилирует исходные тексты в высокоуровневый байт-код, который исполняется в стековой виртуальной машине. К другим трём основным реализациям языка относятся Jython (для JVM), IronPython (для CLR/.NET) и PyPy. PyPy написан на подмножестве языка Python (RPython) и разрабатывался как альтернатива CPython с целью повышения скорости исполнения программ, в том числе за счёт использования JIT-компиляции.

Поддержка версии Python 2 закончилась в 2020 году. На текущий момент активно развивается версия языка Python 3. Разработка языка ве-

дётся через предложения по расширению языка PEP (*Python Enhancement Proposal*), в которых описываются нововведения, делаются корректировки согласно обратной связи от сообщества и документируются итоговые решения.

Стандартная библиотека включает большой набор полезных переносимых функций, начиная от функционала для работы с текстом и заканчивая средствами для написания сетевых приложений. Дополнительные возможности, такие как математическое моделирование, работа с оборудованием, написание веб-приложений или разработка игр, могут реализовываться посредством обширного количества сторонних библиотек, а также интеграцией библиотек, написанных на Си или C++, при этом и сам интерпретатор Python может интегрироваться в проекты, написанные на этих языках. Существует и специализированный репозиторий программного обеспечения, написанного на Python, – PyPI. Данный репозиторий предоставляет средства для простой установки пакетов в операционную систему и стал стандартом де-факто для Python. По состоянию на 2019 год, в нём содержалось более 175 тысяч пакетов.

Python стал одним из самых популярных языков, он используется в анализе данных, машинном обучении, DevOps и веб-разработке, а также в других сферах, включая разработку игр. За счёт читабельности, простого синтаксиса и отсутствия необходимости в компиляции язык хорошо подходит для обучения программированию, позволяя концентрироваться на изучении алгоритмов, концептов и парадигм. Отладка же и экспериментирование в значительной степени облегчаются тем фактом, что язык является интерпретируемым. Применяется язык многими крупными компаниями, такими как Google или Facebook.

По состоянию на октябрь 2021 года, Python занимает первое место в рейтинге TIOBE популярности языков программирования с показателем 11,27 %. «Языком года» по версии TIOBE Python объявлялся в 2007, 2010, 2018 и 2020 годах.

ЛАБОРАТОРНАЯ РАБОТА № 1
ВВЕДЕНИЕ В PYTHON

1.1 Методические рекомендации

Установка Python

Вначале необходимо установить на компьютер интерпретатор Python.

1 Для загрузки дистрибутива переходим на страницу https://www.python.org/downloads/ и в списке доступных версий выбираем Python 3.9.6. На открывшейся странице находим раздел Files и щелкаем по гиперссылке Windows installer (32-bit/64bit) в зависимости от разрядности системы. Запускаем загруженный файл.

2 После успешной установки в меню пуск появится папка python 3.9.6. В дальнейшем потребуется IDLE (python 3.9) для написания и отладки программ.

2 Введение в язык Python

Базовые операторы языка Python приведены в табл. 1.1.

Таблица 1.1 – Базовые операторы языка Python

Инструкция	Описание
print()	Инструкция выводит значения в поток данных. По умолчанию выводит на экран
for	Циклы позволяют выполнить одни и те же инструкции многократно
If	Оператор ветвления позволяет в зависимости от значения логического выражения выполнить отдельный участок программы или, наоборот, не выполнить его
bool()	Преобразует объект в логический тип данных
int()	Преобразует объект в число
float()	Преобразует целое число или строку в вещественное число.
str()	Преобразует объект в строку
break	Выполняет выход из цикла
continue	Начинает следующий проход цикла, минуя оставшееся тело цикла

Синтаксис основных конструкций языка Python

FOR

for <Текущий элемент> in <Последовательность>:
    <Инструкции внутри цикла>
[else:
    <Блок, выполняемый, если не использовался оператор break>
]

IF

If <Логическое выражение>:
    <блок, выполняемый, если условие истинно>
[elif <Логическое выражение>:
    <Блок, выполняемый, если условие истинно>
]
[else:
    <Блок, выполняемый, если все условия ложны>
]

Таблица 1.2 – Операторы сравнения

Оператор	Описание
==	Если значения двух операндов равны, условие становится истинным
!=	Если значения двух операндов не равны, то условие становится истинным
<>	Если значения двух операндов не равны, то условие становится истинным
>	Если значение левого операнда больше, чем значение правого операнда, тогда условие становится истинным
<	Если значение левого операнда меньше значения правого операнда, тогда условие становится истинным
>=	Если значение левого операнда больше или равно значению правого операнда, тогда условие становится истинным
<=	Если значение левого операнда меньше или равно значению правого операнда, тогда условие становится истинным

Таблица 1.3 – Арифметические операторы

Оператор	Описание
+	Добавляет значения по обе стороны от оператора
-	Вычитает правый операнд из левого операнда
*	Умножает значения по обе стороны от оператора
/	Делит левый операнд на правый операнд
%	Делит левый операнд на правый операнд и возвращает остаток
**	Выполняет экспоненциальный (степенной) расчет операторов
//	Деление с остатком – деление операндов, результатом которого является частное, в котором удаляются цифры после десятичной точки. Но если один из операндов отрицателен, результат не учитывается, то есть округляется от нуля (в сторону отрицательной бесконечности).

Таблица 1.4. – Операторы присваивания

Оператор	Описание
=	Присваивает значения из правых операндов левому операнду
+=	Он добавляет правый операнд к левому операнду и присваивает результат левому операнду
-=	Он вычитает правый операнд из левого операнда и присваивает результат левому операнду
*=	Он умножает правый операнд на левый операнд и присваивает результат левому операнду
/=	Он делит левый операнд на правый и присваивает результат левому операнду.
%=	Он принимает модуль с использованием двух операндов и присваивает результат левому операнду
**=	Выполняет экспоненциальное (степенное) вычисление операторов и присваивает значение левому операнду
//=	Он выполняет деление с остатком по операторам и присваивает значение левому операнду

Каждая переменная должна иметь уникальное имя, состоящее из латинских букв, цифр и знаков подчеркивания, причем имя переменной не может начинаться с цифры.

3 Создание простейшей программы

Для просмотра работы функций python достаточно написать в IDLE (python 3.9), рис. 1.1.

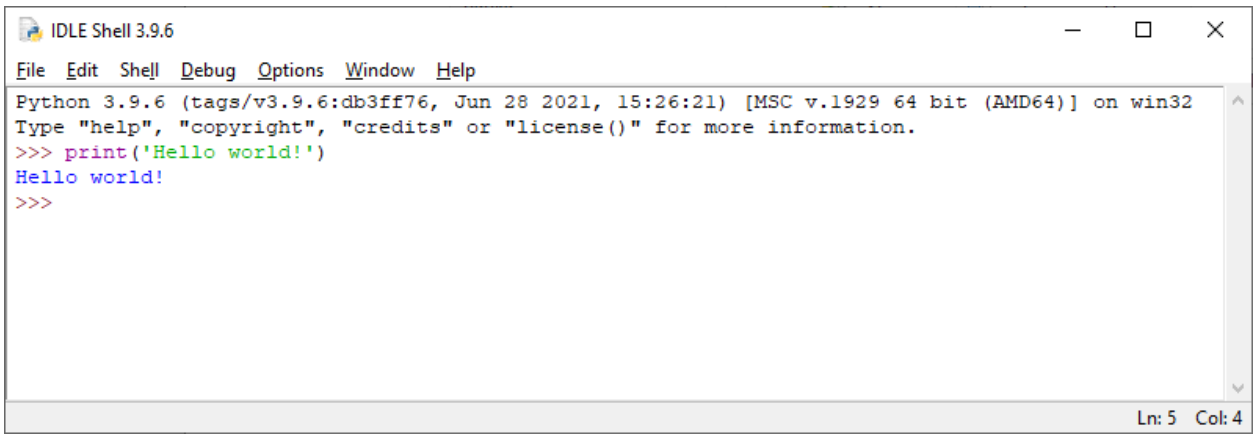

Рисунок 1.1

Так же можно работать с переменными, рис. 1.2.

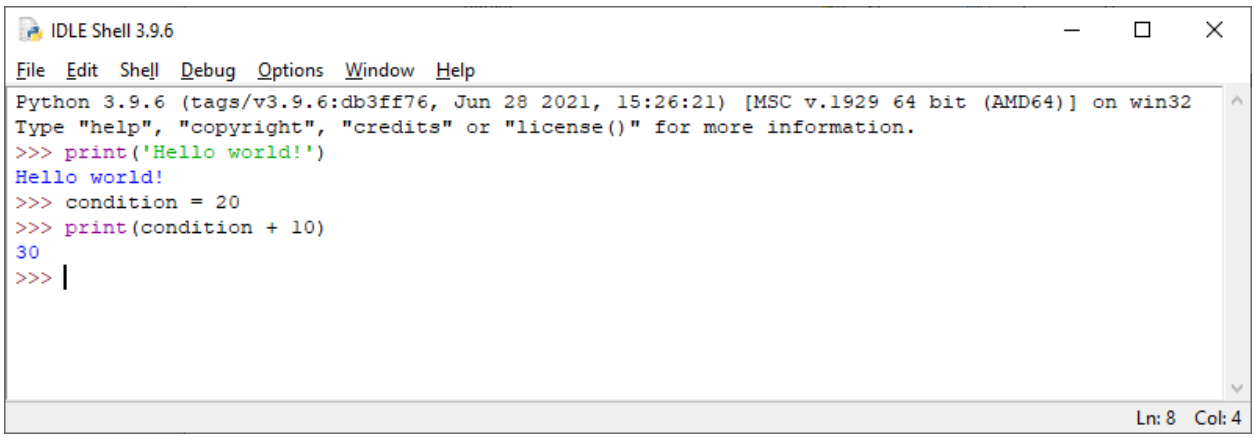

Рисунок 1.2

4 Пример написания программы

Пример 1

Условие: Написать программу, получающую на вход в качестве аргумента два параметра – числа x и y. Меньшее из них разделить на 10, а большее из них возвести в квадрат. Вывести результат на экран.

Решение: Необходимо создать файл, рис. 1.3.

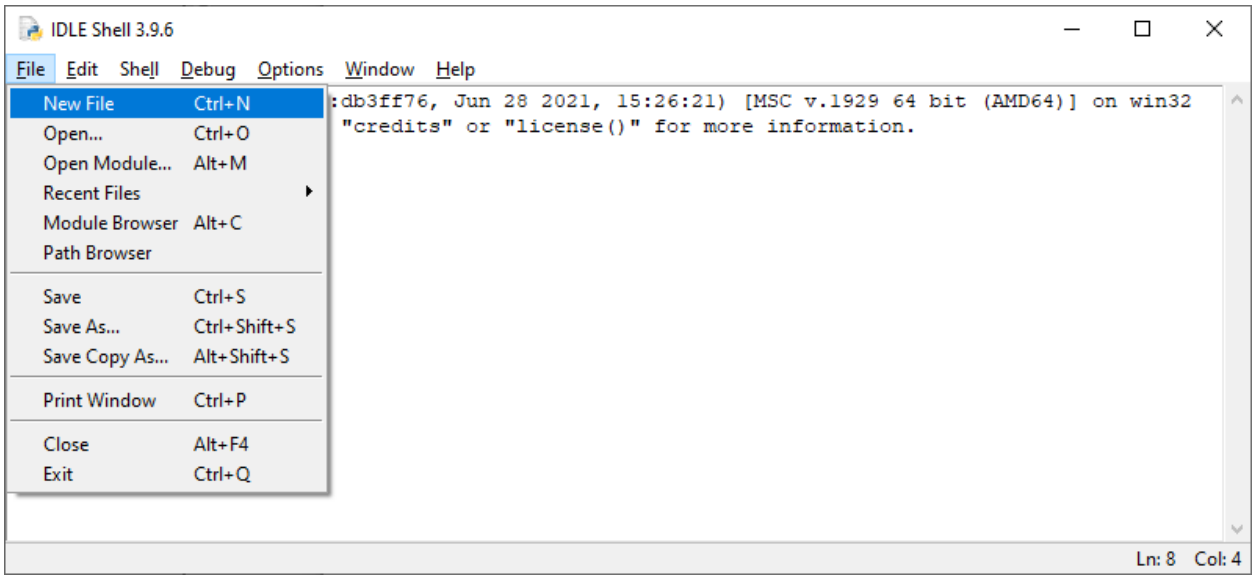

Рисунок 1.3

Код программы:

```
import math as m # подключим библиотеку

x = int(input('Введите x '))
y = int(input('Введите y '))

if x > y:
    resultY = y/10
    resultX = m.pow(x,2) # Возведем во 2 степень x
else:
```

```
    resultY = m.pow(y,2) # возведем во 2 степень x
    resultX = x/10

print('X = ', resultX)
print('Y = ', resultY)
```

Для работы с математическими функциями подключим библиотеку *Import math as m*, где m является сокращением слова math

ВАЖНО: *блоки внутри составной функции выделяются одинаковым количеством пробелов. Концом блока является инструкция, перед которой расположено меньшее количество пробелов, рис. 1.4.*

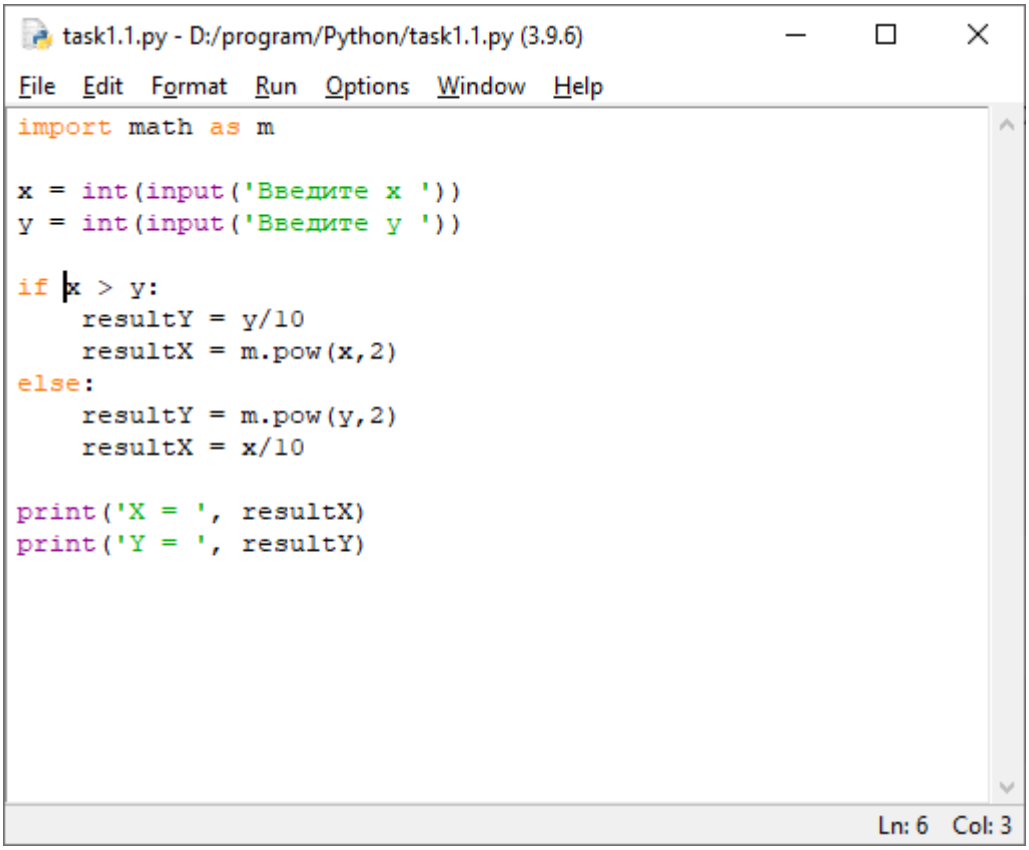

Рисунок 1.4

Пример 2

Условие: Пользователь вводит x и в зависимости от уравнения нужно вычислить Y

$$Y = \begin{cases} x^3 + 1 & \text{если } x <= -3 \\ x(1+2^x) & \text{если } -3 < x <= 4 \\ tg\ x & \text{если } x > 4 \end{cases} \qquad F = \begin{cases} e^{\sin x} & \text{если } x <= -3 \\ x^4 & \text{если } -3 < x <= 4 \\ \sqrt[5]{tgx} & \text{если } x > 4 \end{cases}$$

Решение:

Код программы:

```
import math as m
```

```
x = int(input('Введите x '))

if x <= -3:
   y = m.pow(x,3)+1 #Возведем в 3 степень число x
else:
   if x > 4:
      y = m.tan(x) # Вычисли тангенс x
   else:
      y = x * (1 + m.pow(2,x))

print('Y = ', y)
```

1.2 Задания для самостоятельной работы

Варианты заданий

Задание 1

1 Написать программу, получающую на вход в качестве аргумента два параметра – числа x и y. Если произведение этих чисел больше 50, то вычислить удвоенный корень квадратный первого числа. Вывести результат на экран.

2 Написать программу, получающую на вход в качестве аргумента два параметра – числа, a и b. Если сумма двух чисел больше 100, то вычислить удвоенный синус первого числа. Вывести результат на экран.

3 Написать программу, получающую на вход в качестве аргумента два параметра – числа x и y. Большее из них возвести в квадрат. Вывести результат на экран.

4 Написать программу, получающую на вход в качестве аргумента два параметра – числа a и b. Если произведение двух чисел больше 100, то вычислить утроенный тангенс второго числа, в противном случае первое число умножить на 5. Вывести результат на экран.

5 Написать программу, получающую на вход в качестве аргумента два параметра – числа a и b. Если произведение двух чисел больше 20, то вычислить котангенс второго числа, в противном случае первое число разделить на 3. Вывести результат на экран.

6 Написать программу, получающую на вход в качестве аргумента два параметра – числа x и y. Меньшее из них разделить на 2. Вывести результат на экран.

7 Написать программу, получающую на вход в качестве аргумента два параметра – числа a и b. Если произведение двух чисел больше 30, то вычислить удвоенный котангенс первого числа, в противном случае первое число разделить на 2. Вывести результат на экран.

8 Написать программу, получающую на вход в качестве аргумента два параметра – числа a и b. Если произведение двух чисел больше 40, то вычислить удвоенный тангенс первого числа, в противном случае второе число умножить на 4. Вывести результат на экран.

9 Написать программу, получающую на вход в качестве аргумента два параметра – числа x и y. Из меньшего извлечь корень квадратный. Вывести результат на экран.

10 Написать программу, получающую на вход в качестве аргумента два параметра – числа a и b. Если произведение двух чисел больше 50, то вычислить удвоенный косинус первого числа, в противном случае второе число умножить на 3. Вывести результат на экран.

11 Написать программу, получающую на вход в качестве аргумента два параметра – числа x и y. Если произведение этих чисел больше 100, то вычислить удвоенный куб первого числа и второе число разделить на 2. Вывести результат на экран.

12 Написать программу, получающую на вход в качестве аргумента два параметра – числа x и y. Если сумма этих чисел больше 20, то вычислить утроенный квадрат первого числа и куб второго числа. Вывести результат на экран.

13 Написать программу, получающую на вход в качестве аргумента два параметра – числа x и y. Если произведение этих чисел больше 50, то вычислить удвоенный корень квадратный первого числа и квадрат второго числа. Вывести результат на экран.

14 Написать программу, получающую на вход в качестве аргумента два параметра – числа a и b. Если сумма двух чисел больше 100, то вычислить удвоенный синус первого числа, а первое число умножить на 5. Вывести результат на экран.

15 Написать программу, получающую на вход в качестве аргумента два параметра – числа a и b. Если произведение двух чисел больше 100, то вычислить утроенный тангенс второго числа и синус второго числа, в противном случае первое число умножить на 5. Вывести результат на экран.

Задание 2

Пользователь вводит x, и в зависимости от уравнения нужно вычислить Y

1.

$$Y=\begin{cases} x^3+1 & \text{если } x <= -3 \\ x(1+2^x) & \text{если } -3 < x <= 4 \\ \operatorname{tg} x & \text{если } x > 4 \end{cases} \quad F=\begin{cases} e^{\sin x} & \text{если } x <= -3 \\ x^4 & \text{если } -3 < x <= 4 \\ \sqrt[5]{tgx} & \text{если } x > 4 \end{cases}$$

2.

$$Y=\begin{cases} x^3+1 & \text{если } x <= -3 \\ (1+2^{tgx}) & \text{если } -3 < x <= 0 \\ ctg^2\, x & \text{если } x > 0 \end{cases} \quad F=\begin{cases} e^{x+1} & \text{если } x <= -3 \\ x^4 & \text{если } -3 < x <= 0 \\ \sqrt[5]{tgx} & \text{если } x > 0 \end{cases}$$

3.

$$Y=\begin{cases} x^5+\operatorname{arctg}8x & \text{если } x <= 1 \\ 5x-(1+3^x) & \text{если } 1 < x <= 2 \\ ctg\,(2x+1) & \text{если } x > 2 \end{cases} \quad F=\begin{cases} e^{tg\,x+1} & \text{если } x <= 1 \\ x^4-tg4x & \text{если } 1 < x <= 2 \\ \sqrt[5]{x} & \text{если } x > 2 \end{cases}$$

4.

$$Y=\begin{cases} 1-x^5+tg8x & \text{если } x <= 0 \\ (1+3^x) & \text{если } 0 < x <= 3 \\ \operatorname{arctg}(2x+1) & \text{если } x > 3 \end{cases} \quad F=\begin{cases} e^{2x+1} & \text{если } x <= 0 \\ x^2-\sin 4x & \text{если } 0 < x <= 3 \\ \sqrt[5]{x^2} & \text{если } x > 3 \end{cases}$$

5.

$$Y=\begin{cases} 1-9^{x+1} & \text{если } x <= 0 \\ (1+3^x) & \text{если } 0 < x <= 3 \\ ctg\,(2x+1) & \text{если } x > 3 \end{cases} \quad F=\begin{cases} e^{tg(2x+1)} & \text{если } x <= 0 \\ x^2-\sin^4 x & \text{если } 0 < x <= 3 \\ \sqrt[5]{x^2} & \text{если } x > 3 \end{cases}$$

6.

$$Y=\begin{cases} x^3+1 & \text{если } x <= -3 \\ x(1+2^x) & \text{если } -3 < x <= 4 \\ \operatorname{tg} x & \text{если } x > 4 \end{cases} \quad F=\begin{cases} e^{\sin x} & \text{если } x <= -3 \\ x^4 & \text{если } -3 < x <= 4 \\ \sqrt[5]{tgx} & \text{если } x > 4 \end{cases}$$

7.

$$Y=\begin{cases} x^5+\operatorname{arctg}8x & \text{если } x <= 1 \\ 5x-(1+3^x) & \text{если } 1 < x <= 2 \\ ctg\,(2x+1) & \text{если } x > 2 \end{cases} \quad F=\begin{cases} e^{tg\,x+1} & \text{если } x <= 1 \\ x^4-tg4x & \text{если } 1 < x <= 2 \\ \sqrt[5]{x} & \text{если } x > 2 \end{cases}$$

8.

$$Y=\begin{cases} x^5-\operatorname{tg}(2x-1) & \text{если } x<=-2 \\ 3x(1+e^{x+1}) & \text{если } -2<x<=1 \\ \operatorname{Sin}^5 x & \text{если } x>1 \end{cases} \qquad F=\begin{cases} e^{\sin x} & \text{если } x<=-2 \\ x^2 & \text{если } -2<x<=1 \\ \sqrt{\cos x} & \text{если } x>1 \end{cases}$$

9.

$$Y=\begin{cases} x^3+1 & \text{если } x<=-3 \\ (1+2^{tgx}) & \text{если } -3<x<=0 \\ \operatorname{ctg}^2 x & \text{если } x>0 \end{cases} \qquad F=\begin{cases} e^{x+1} & \text{если } x<=-3 \\ x^4 & \text{если } -3<x<=0 \\ \sqrt[5]{tgx} & \text{если } x>0 \end{cases}$$

10.

$$Y=\begin{cases} x^5+\operatorname{arctg}8x & \text{если } x<=1 \\ 5x-(1+3^x) & \text{если } 1<x<=2 \\ \operatorname{ctg}(2x+1) & \text{если } x>2 \end{cases} \qquad F=\begin{cases} e^{\operatorname{tg}x+1} & \text{если } x<=1 \\ x^4-\operatorname{tg}4x & \text{если } 1<x<=2 \\ \sqrt[5]{x} & \text{если } x>2 \end{cases}$$

11.

$$Y=\begin{cases} 1-x^5+\operatorname{tg}8x & \text{если } x<=0 \\ (1+3^x) & \text{если } 0<x<=3 \\ \operatorname{arctg}(2x+1) & \text{если } x>3 \end{cases} \qquad F=\begin{cases} e^{2x+1} & \text{если } x<=0 \\ x^2-\sin 4x & \text{если } 0<x<=3 \\ \sqrt[5]{x^2} & \text{если } x>3 \end{cases}$$

12.

$$Y=\begin{cases} 1-9^{x+1} & \text{если } x<=0 \\ (1+3^x) & \text{если } 0<x<=3 \\ \operatorname{ctg}(2x+1) & \text{если } x>3 \end{cases} \qquad F=\begin{cases} e^{\operatorname{tg}(2x+1)} & \text{если } x<=0 \\ x^2-\sin^4 x & \text{если } 0<x<=3 \\ \sqrt[5]{x^2} & \text{если } x>3 \end{cases}$$

13.

$$Y=\begin{cases} a-b^{x+1} & \text{если } x<=0 \\ (1+3^{xa}) & \text{если } 0<x<=3 \\ \operatorname{ctg}(ax+1) & \text{если } x>3 \end{cases} \qquad F=\begin{cases} e^{\operatorname{tg}(ax+1)} & \text{если } x<=0 \\ x^a-\sin^b x & \text{если } 0<x<=3 \\ \sqrt[5]{x^2} & \text{если } x>3 \end{cases}$$

14.

$$Y=\begin{cases} x^3+1 & \text{если } x<=-3 \\ x(1+2^x) & \text{если } -3<x<=4 \\ \operatorname{tg}x & \text{если } x>4 \end{cases} \qquad F=\begin{cases} e^{\sin x} & \text{если } x<=-3 \\ x^4 & \text{если } -3<x<=4 \\ \sqrt[5]{tgx} & \text{если } x>4 \end{cases}$$

15.

$$Y=\begin{cases} x^5-\operatorname{tg}(2x-1) & \text{если } x<=-2 \\ 3x(1+e^{x+1}) & \text{если } -2<x<=1 \\ \operatorname{Sin}^5 x & \text{если } x>1 \end{cases} \qquad F=\begin{cases} e^{\sin x} & \text{если } x<=-2 \\ x^2 & \text{если } -2<x<=1 \\ \sqrt{\cos x} & \text{если } x>1 \end{cases}$$

ЛАБОРАТОРНАЯ РАБОТА № 2
РАБОТА СО СПИСКАМИ, КОРТЕЖАМИ, МНОЖЕСТВАМИ

2.1 Методические рекомендации

Списки, кортежи, множества – это нумерованные наборы объектов. Каждый элемент набора содержит лишь ссылку на объект – по этой причине они могут содержать объекты произвольного типа данных и иметь неограниченную степень вложенности.

1 Списки

Списки в Python – упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

Создать список можно несколькими способами. Например, можно обработать любой итерируемый объект (например, строку) встроенной функцией list, рис. 2.1.

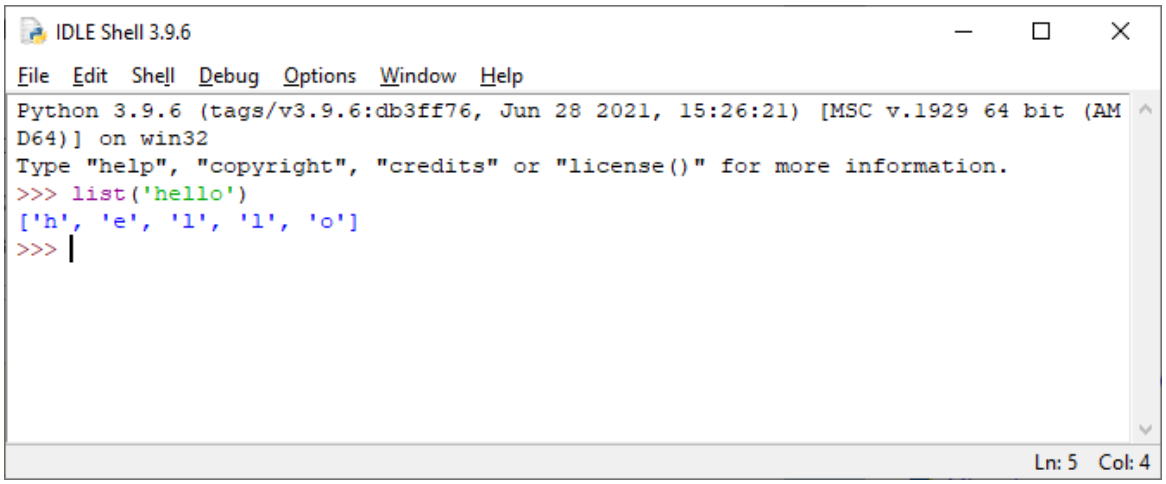

Рисунок 2.1

Либо можно создать список с помощью квадратных скобок, рис. 2.2.

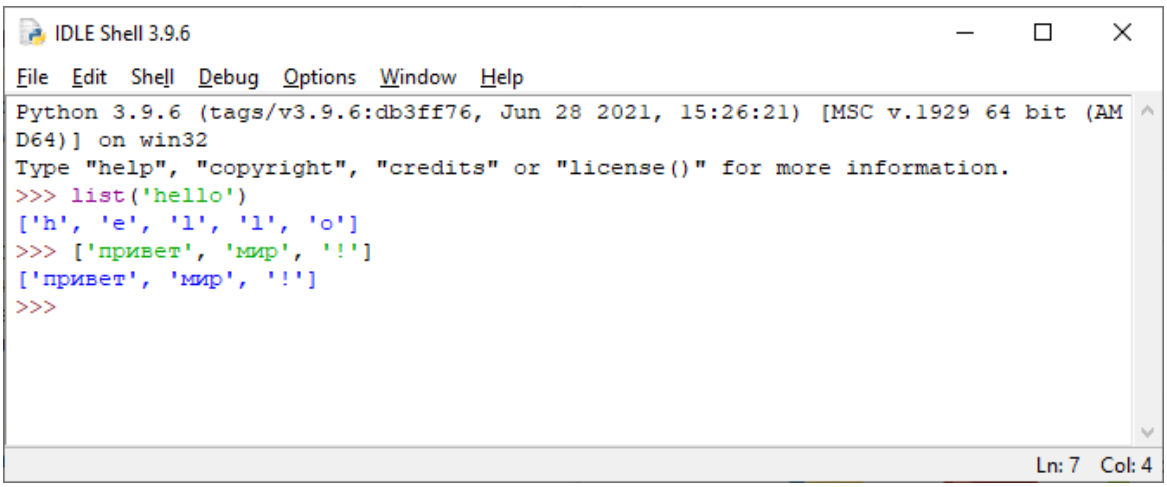

Рисунок 2.2

Также списки могут быть вложенными и разного типа данных, рис. 2.3.

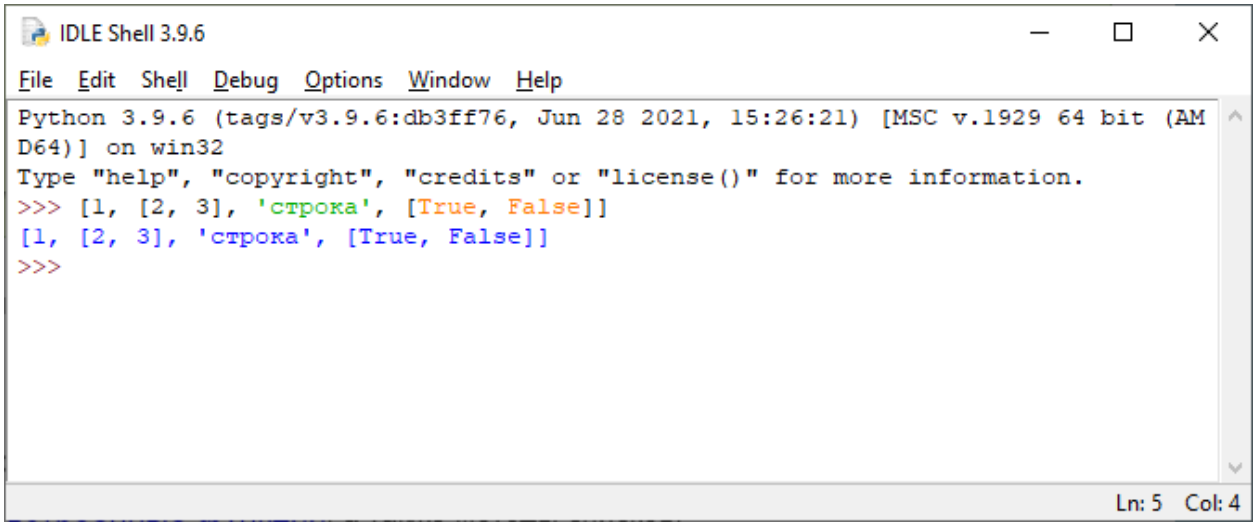

Рисунок 2.3

Методы с листами представлены в табл. 2.1.

Таблица 2.1 – Методы листов

Метод	Что делает
list.append(x)	Добавляет элемент в конец списка
list.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x)	Вставляет на i-й элемент значение x
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
list.pop([i])	Удаляет i-й элемент и возвращает его. Если индекс не указан, удаляется последний элемент
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)
list.count(x)	Возвращает количество элементов со значением x
list.sort([key=функция])	Сортирует список на основе функции
list.reverse()	Возвращает список в обратном порядке
list.copy()	Поверхностная копия списка
list.clear()	Очищает список

2 Кортежи

Кортежи, как и списки, являются упорядоченными последовательностями элементов. Они во много аналогичны спискам, но имеют одно очень важное отличие – изменить кортеж нельзя. Можно сказать, что кортеж – это список, доступный только для чтения.

Создать кортеж можно следующими способами:

– с помощью функции tuple([<последовательность>]), рис. 2.4;

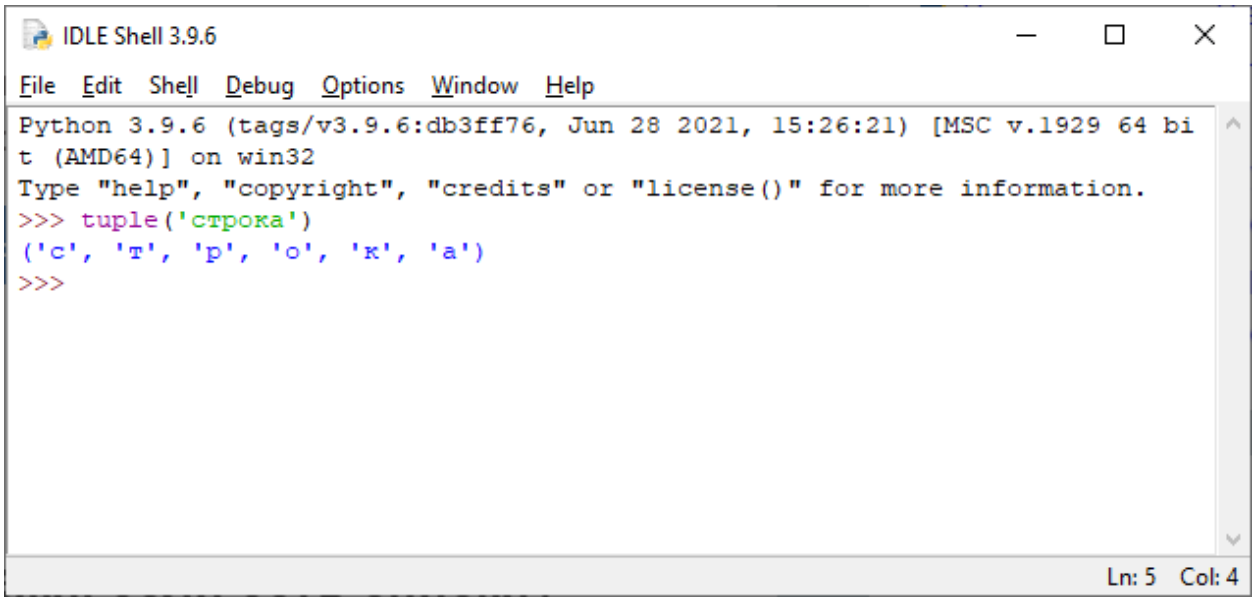

Рисунок 2.4

– с помощью круглых скобок, рис. 2.5.

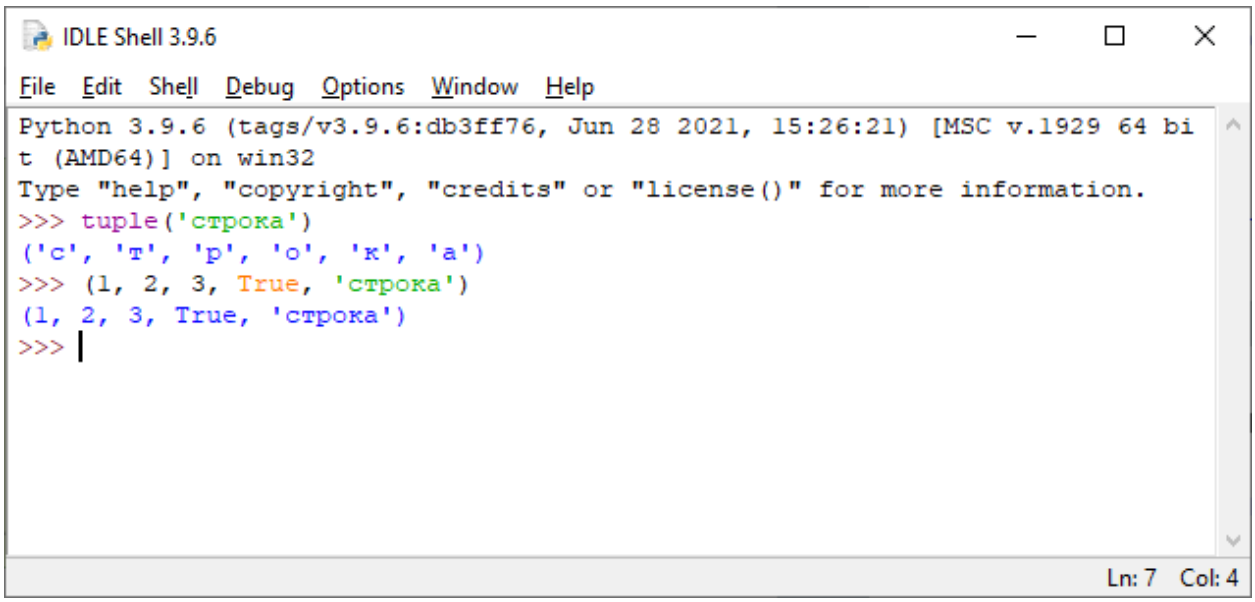

Рисунок 2.5

Операции с кортежами – такие же, как и операции над списками, только не изменяющие список. Можно также по-разному менять элементы местами и так далее.

3 Множества

Множества – это неупорядоченная последовательность уникальных элементов, с которой можно сравнивать другие элементы, чтобы определить, принадлежат ли они этому множеству. Объявить множество можно с помощью функции set([<Последовательность>]), рис. 2.6.

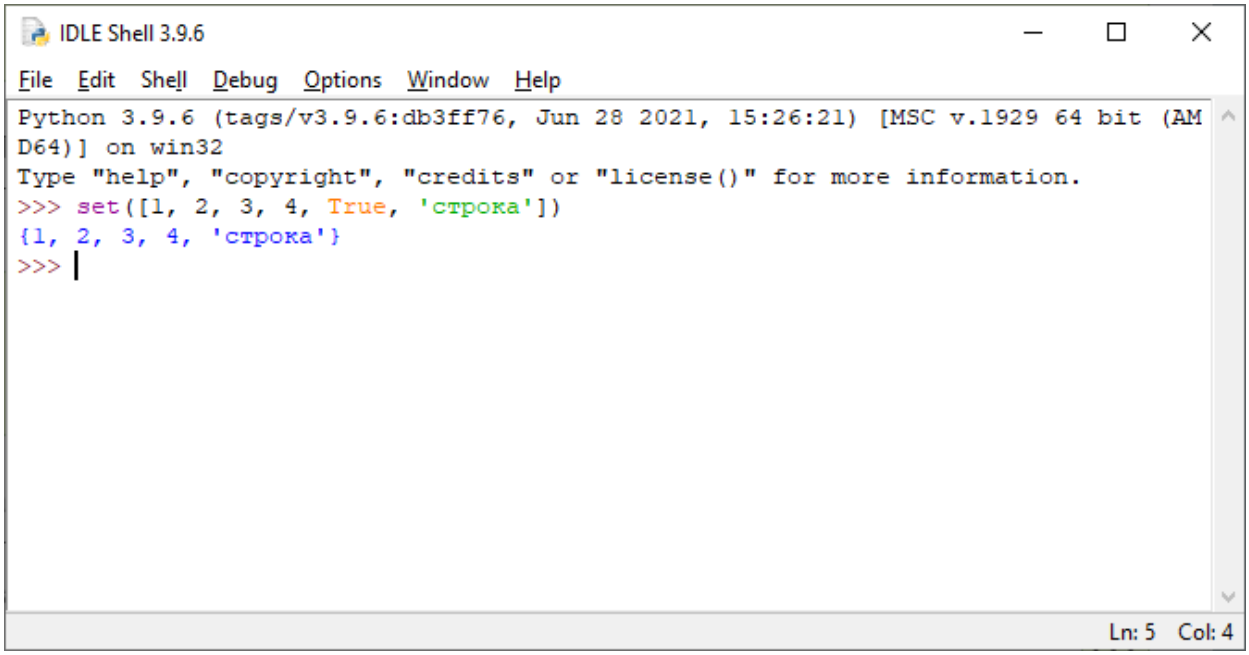

Рисунок 2.6

На рис. 2.7 во втором примере видно, что при создании множества повторяющиеся элементы были удалены.

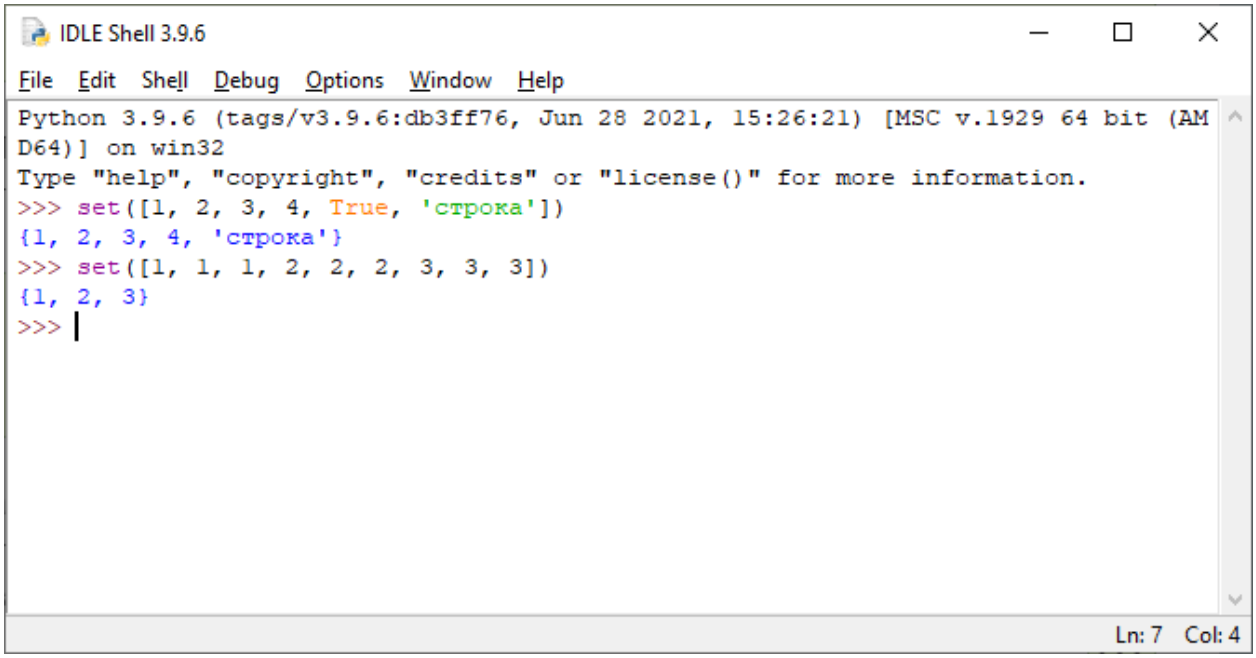

Рисунок 2.7

Методы с множествами:

С множествами можно выполнять множество операций: находить объединение, пересечение.

- len(s) – число элементов в множестве (размер множества).
- x in s – принадлежит ли x множеству s.
- **set.isdisjoint**(other) – истина, если set и other не имеют общих элементов.
- **set == other** – все элементы set принадлежат other, все элементы other принадлежат set.
- **set.issubset**(other) или **set <= other** – все элементы set принадлежат other.
- **set.issuperset**(other) или **set >= other** – аналогично.
- **set.union**(other, ...) или **set | other | ...** – объединение нескольких множеств.
- **set.intersection**(other, ...) или **set & other & ...** – пересечение.
- **set.difference**(other, ...) или **set - other - ...** – множество из всех элементов set, не принадлежащие ни одному из other.
- **set.symmetric_difference**(other); **set ^ other** – множество из элементов, встречающихся в одном множестве, но не встречающихся в обоих.
- **set.copy**() – копия множества.

И операции, непосредственно изменяющие множество:

- **set.update**(other, ...); set |= other | ... – объединение.
- **set.intersection_update**(other, ...); set &= other & ... – пересечение.
- **set.difference_update**(other, ...); set -= other | ... – вычитание.
- **set.symmetric_difference_update**(other); set ^= other – множество из элементов, встречающихся в одном множестве, но не встречающихся в обоих.
- **set.add**(elem) – добавляет элемент в множество.
- **set.remove**(elem) – удаляет элемент из множества. KeyError, если такого элемента не существует.
- **set.discard**(elem) – удаляет элемент, если он находится в множестве.
- **set.pop**() – удаляет первый элемент из множества. Так как множества не упорядочены, нельзя точно сказать, какой элемент будет первым.
- **set.clear**() – очистка множества.

4 Диапазоны

Диапазоны, как следует из самого названия, – это диапазоны целых чисел с заданными начальным и конечным значением и шагом (промежутком между соседними числами). Как и списки, кортежи и множества, диапазоны представляют собой последовательности и, подобно кортежам, являются неизменяемыми.

Для создания диапазона применяется функция range(<начало>, <конец>, [<шаг>]) (рис. 2.8).

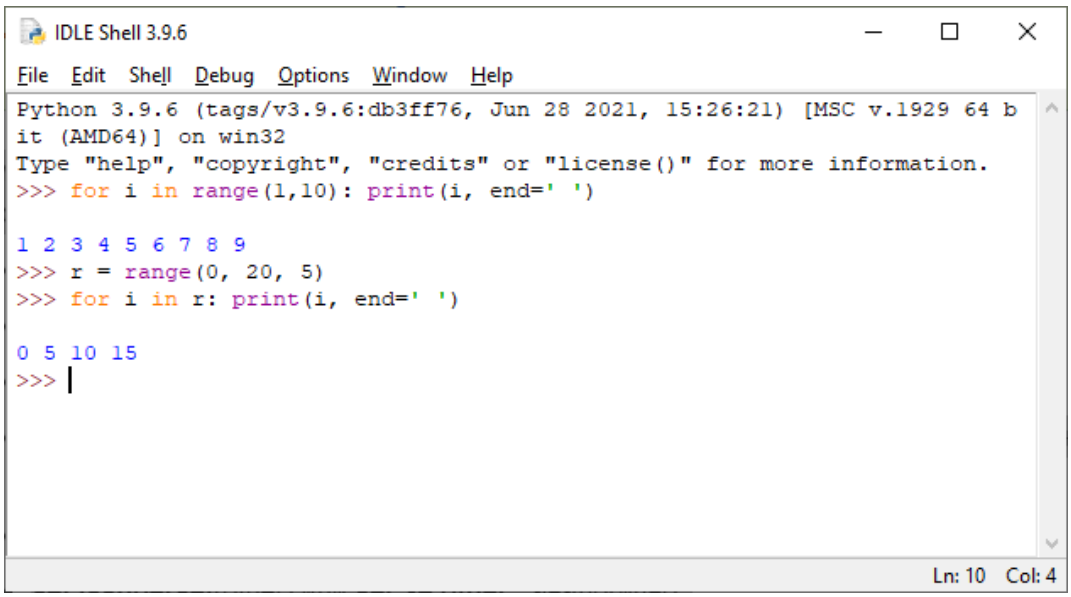

Рисунок 2.8

В первом примере мы с помощью цикла пробежались по диапазону от 1 до 10, при этом последний элемент не учитывается

Во втором примере появился третий параметр {5} – это шаг в диапазоне.

5 Словари

Словари – это наборы объектов, доступ к которым осуществляется не по индексу, а по ключу. В качестве ключа можно указать неизменяемый объект, например число, строку или кортеж. Элементы словаря могут содержать объекты произвольного типа данных и иметь неограниченную степень вложенности. Следует также заметить, что элементы в словарях располагаются в произвольном порядке. Чтобы получить элемент, необходимо указать ключ, который использовался при сохранении значения.

Создать словарь можно следующим способом

С помощью dict() :

dict(<Ключ1>=<Значение>[, …, <КлючN>=<ЗначениеN>]

dict(<Словарь>)

dict(<Список кортежей с двумя элементами (ключ, значение)>)

dict(<Список списков с двумя элементами [ключ, значение]>)

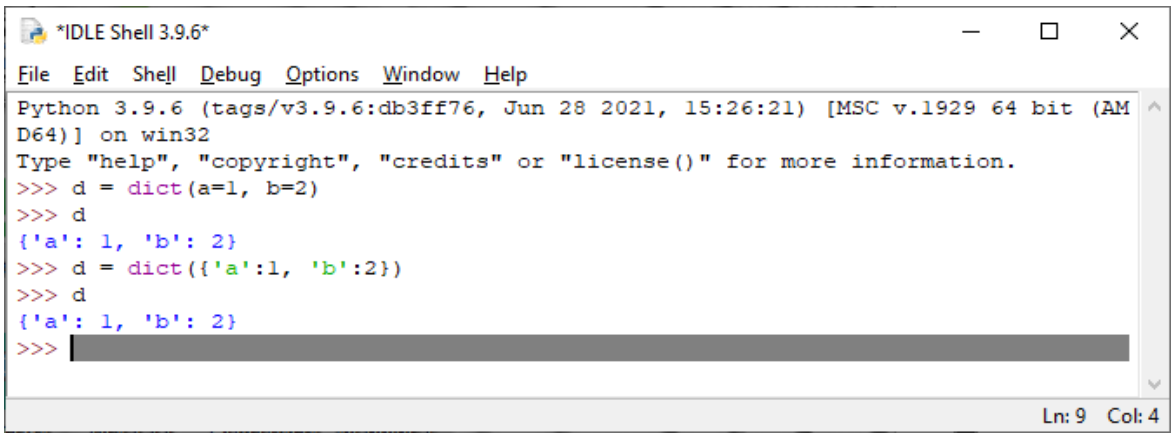

Рисунок 2.9

Пример 1

Условие: Составить и вывести на экран новый список с номерами элементов исходного списка, которые равны заданному значению. Заданное значение определяется константой.

Решение:

Листинг кода:

```
import random as rand # подключаем библиотеку для работы с рандомными числами

const = 4 # инициализация переменной
massBegin = [] # инициализация переменной
massEnd = [] # инициализация переменной

for i in range(15): # цикл в 15 итераций
    massBegin.append(rand.randint(1,7)) # заполняем массив рандомными значениями от 1 до 7

for i in range(len(massBegin)): # цикл от 0 до длины массива
    if massBegin[i] == const: # если элемент массива равен 4 то
        massEnd.append(i) # добавляем во второй массив порядковый номер

print('Исходный массив: ', massBegin) # вывод первго массива
print('Полученный массив', massEnd) # вывод второго массива
```

Результат работы программы:

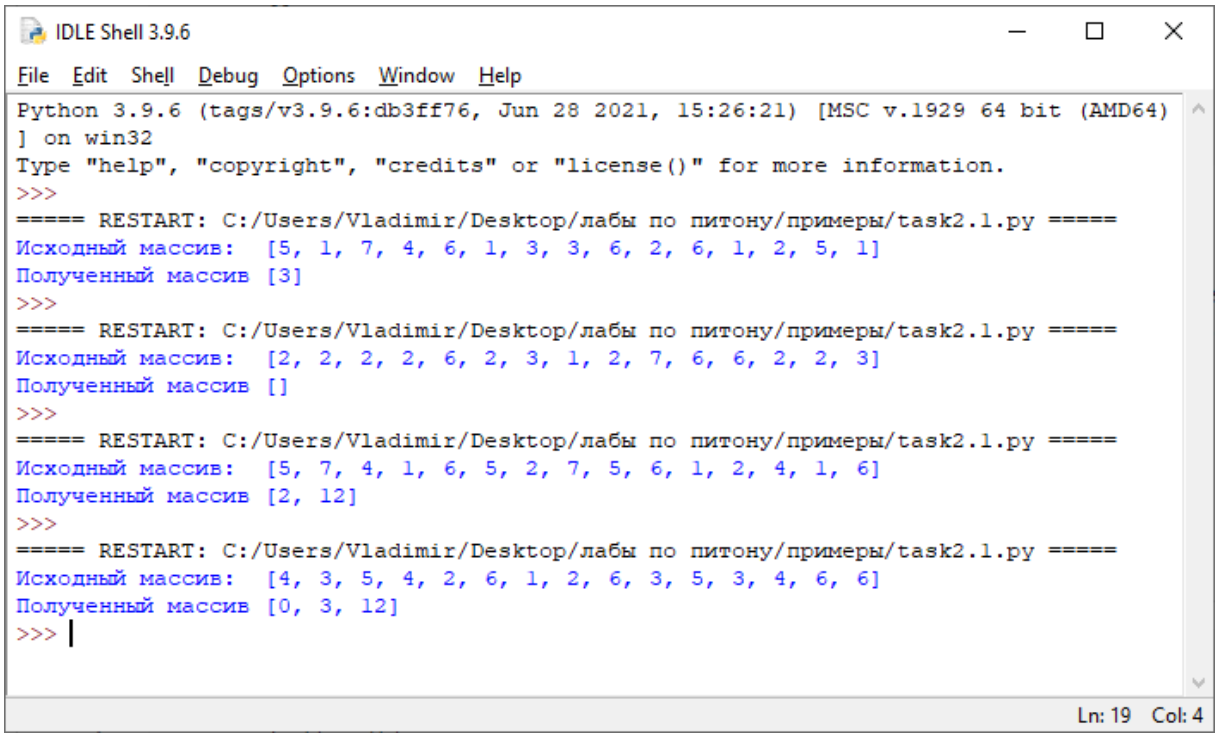

Рисунок 2.10

Пример 2

Условие: Дан список b. Переписать в список G положительные элементы списка b, умноженные на 5. Затем упорядочить методом по возрастанию новый список.

Решение:

Код программы:

```
import random as rand # подключаем библиотеку для работы с рандомными числами


b = [] # инициализация переменной
G = [] # инициализация переменной

for i in range(15):
    b.append(rand.randint(-10,10)) # заполняем массив рандомными значениями от -10 до 10


for i in range(len(b)): # цикл от 0 до длины массива
    if b[i] > 0: # если элемент массива больше 0, то
        G.append(b[i] * 5) # добавляем во второй массив значение первого массива умноженное на 5

print('Исходный массив: ', b)
print('Полученный массив до сортировки', G)
G = sorted(G)
print('Полученный массив после сортировки', G)
```

Результат работы программы:

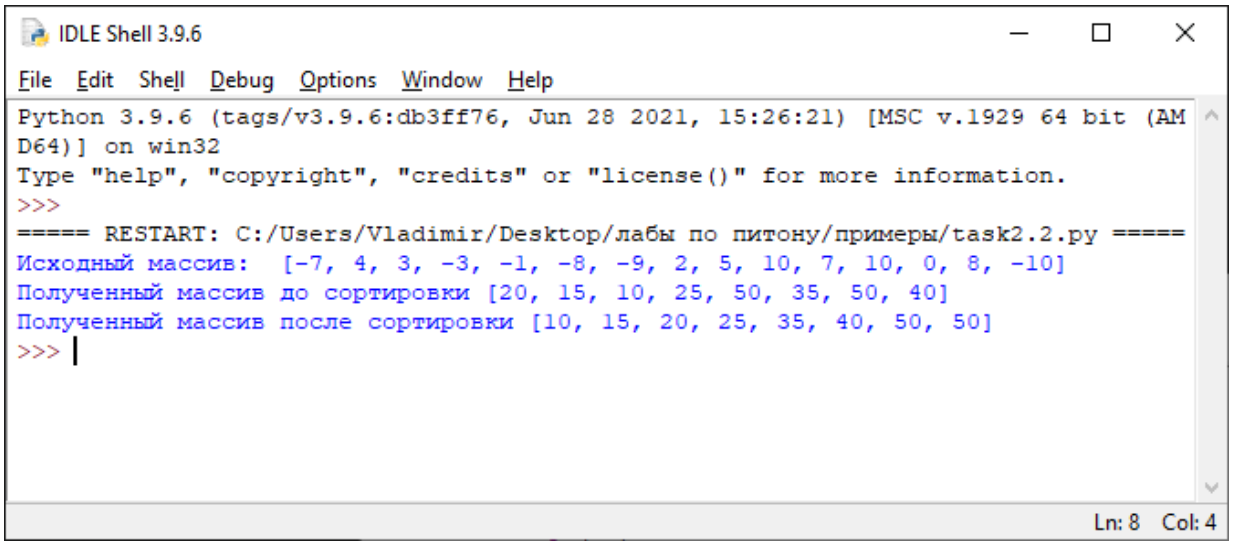

Рисунок 2.11

Пример 3

Условие: Определить матрицу (двумерный массив) и ее заполнить случайными значениями. Определить: число неотрицательных элементов в *i*-й строке.

Решение:

```
import random as rand

b = []
count = 0

for i in range(5):
    b.append([rand.randint(-10,10) for j in range(7)]) # генератор массива со значениями от -10 до 10

i = int(input('Введите i: '))
for j in range(len(b[i])):
    if b[i][j] >= 0:
        count += 1

print('Исходный массив: ', b)
print('Количество неотрицательных элементов в i строчке', count)
```

Результат работы программы:

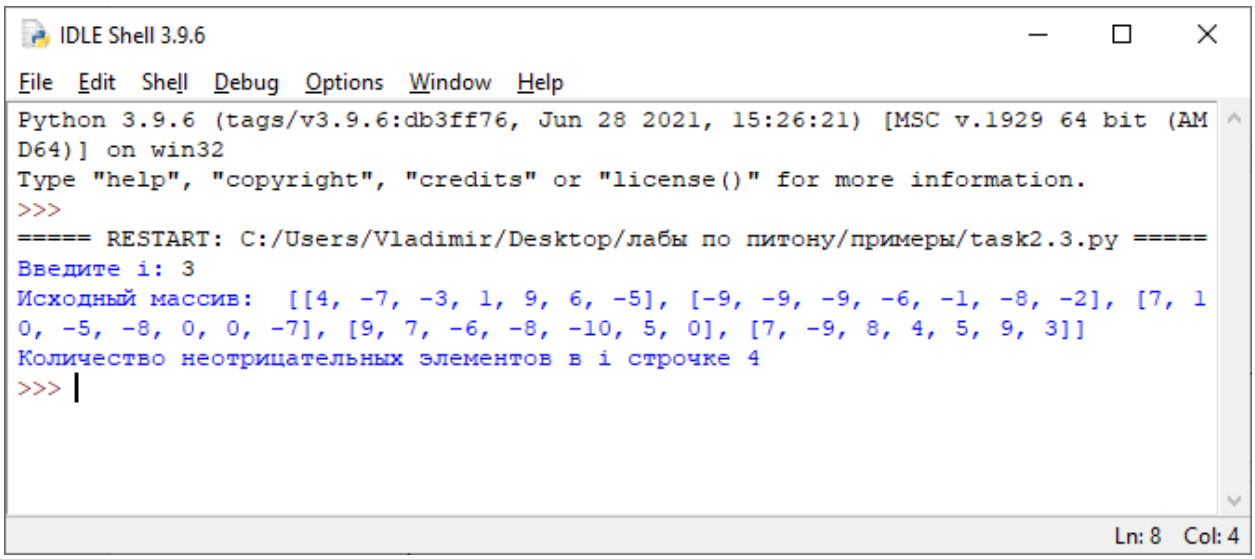

Рисунок 2.12

2.2 Задания для самостоятельной работы

Варианты заданий

Задача 1. Определить список и заполнить его случайными значениями:

1 Составить и вывести на экран список с N максимальными значениями исходного списка. N определяется константой.

2 Переписать элементы списка в обратном порядке. Вывести измененный список на экран.

3 Определить дополнительный список, состоящий из неповторяющихся элементов исходного списка, и вывести его на экран.

4 Составить и вывести на экран список номеров элементов исходного списка, встречающихся один раз.

5 Определить дополнительный массив, состоящий из повторяющихся элементов исходного списка, и вывести его на экран.

6 Вычислить, сколько элементов данного списка больше своего предыдущего элемента.

7 Найти максимальный элемент в одномерном списке x. Затем каждый элемент в списке разделить на максимальный элемент.

8 Дан список b. Переписать в список C положительные элементы списка b умноженные на 5 (сжатие списка).

9 Дан одномерный список A в котором находится единственный нулевой элемент. Найти, где он находится, и вычислить сумму последующих за ним элементов. Выдать на экран номер элемента и сумму.

10 Определить, сколько раз в этом списке меняется знак. Например, в списке 1, –34, 8, 14, –5, –8, –78, 3 знак меняется 4 раза.

11 Найти минимальный элемент в одномерном списке x. Затем каждый элемент в списке умножить на минимальный элемент.

12 Дан список c. Переписать в список x все ненулевые элементы списка, умноженные на 4 (сжатие списка).

13 Определить номера элементов = 5, количество положительных элементов для всего списка и произведение возведенных в квадрат отрицательных элементов.

14 Найти максимальный элемент и переставить его с 1-м элементом списка.

15 Дан список из 16 двоичных цифр (0;1). Определить сколько раз в этом списке меняется число 0 на 1 или 1 на 0. Например, в списке 11110010001101 число меняется 6 раз.

16 Найти минимальный элемент в одномерном списке x. Затем из каждого элемента списка вычесть минимальный элемент.

Задача 2

1 Дан список c. Переписать в список x все ненулевые элементы списка, возведенные в квадрат. Затем упорядочить новый список.

2 Дан список c. Переписать в список x все ненулевые элементы списка. Затем упорядочить новый список.

3 Дан список c. Переписать в список x ненулевые элементы списка c, разделенные на 5. Затем упорядочить новый список.

4 Дан список b. Переписать в список C корни квадратные из положительных элементов списка b, деленные на 5. Затем упорядочить новый список.

5 Дан список b. Переписать в список C корни квадратные из положительных элементов списка b. Затем упорядочить новый список.

6 Дан список x. Переписать в список y элементы списка x, большие 3. Затем упорядочить новый список.

7 Дан одномерный список, a, в котором находится единственный нулевой элемент. Найти, где он находится, и упорядочить по возрастанию элементы, расположенные за ним. Выдать на экран номер элемента и упорядоченный список.

8 Известно, что в списке x есть один элемент = 1. Определить, где он находится, и упорядочить по убыванию элементы, расположенные за ним. Выдать на экран номер элемента и упорядоченный список.

9 В списке z один отрицательный элемент. Найти, где он находится, и упорядочить по возрастанию элементы, расположенные перед ним. Выдать на экран номер элемента и упорядоченный список.

10 Дан одномерный список, a, в котором находится единственный элемент равный 5. Найти, где он находится, и упорядочить по убыванию элементы, расположенные перед ним. Выдать на экран номер элемента и упорядоченный список.

11 Найти максимальный и минимальный элементы в одномерном списке x, а также их порядковые номера. Затем упорядочить по возрастанию элементы, расположенные между максимальным и минимальным элементами.

12 Найти максимальный элемент и его порядковый номер в одномерном списке x. Затем упорядочить по возрастанию элементы, расположенные после максимального элемента.

13 Найти минимальный элемент и его порядковый номер в одномерном списке x. Затем упорядочить по убыванию элементы, расположенные после минимального элемента.

14 Найти максимальный элемент и его порядковый номер в одномерном списке x. Затем упорядочить по возрастанию элементы, расположенные перед максимальным элементом.

15 Найти минимальный элемент и его порядковый номер в одномерном списке x. Затем упорядочить по возрастанию элементы, расположенные перед минимальным элементом.

Задача 3

Определить матрицу (двумерный список) и заполнить случайными значениями.

Определить:

1) количество четных чисел в i-й строке;
2) сумму положительных элементов в каждом столбце матрицы;
3) произведение положительных элементов в каждом столбце матрицы;
4) количество положительных элементов в каждом столбце матрицы;

5) среднее арифметическое положительных элементов в каждом столбце матрицы;

6) среднее геометрическое положительных элементов в каждом столбце матрицы;

7) сумму отрицательных элементов в каждом столбце матрицы;

8) произведение отрицательных элементов в каждом столбце матрицы;

9) минимальный элемент в каждой строке матрицы. Затем каждую строку матрицы разделить на минимальный элемент строки;

10) минимальный элемент в каждой строке матрицы среди положительных элементов;

11) максимальный элемент в каждой строке матрицы среди отрицательных элементов;

12) минимальный элемент в каждом столбце матрицы. Затем каждый столбец матрицы умножить на минимальный элемент;

13) максимальный элемент в каждой строке матрицы. Затем к каждому элементу каждой строки прибавить максимальный элемент строки;

14) минимальный элемент и его номер в каждой строке матрицы. Затем из каждого элемента каждой строки вычесть номер минимального элемента строки;

15) номер минимального элемента в каждой строке матрицы. Затем к каждому элементу каждой строки прибавить номер минимального элемента строки;

16) номер максимального элемента в каждом столбце матрицы. Затем каждый элемент каждого столбца умножить на номер максимального элемента столбца.

ЛАБОРАТОРНАЯ РАБОТА № 3
РАБОТА СО СТРОКАМИ

3.1 Методические рекомендации

Строки в Python – упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

S = 'spam"s'

S = "spam's"

Строки в апострофах и в кавычках – одно и то же. Причина наличия двух вариантов в том, чтобы позволить вставлять в литералы строк символы кавычек или апострофов, не используя экранирование.

Экранированные последовательности позволяют вставить символы, которые сложно ввести с клавиатуры.

Таблица 3.1 – Экранированные последовательности

Экранированная последовательность	**Назначение**
\n	Перевод строки
\f	Перевод страницы
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\N{id}	Идентификатор ID базы данных Юникода
\uhhhh	16-битовый символ Юникода в 16-ричном представлении
\Uhhhh…	32-битовый символ Юникода в 32-ричном представлении
\xhh	16-ричное значение символа
\ooo	8-ричное значение символа
\0	Символ Null (не является признаком конца строки)

Строки в тройных апострофах или кавычках

Главное достоинство строк в тройных кавычках в том, что их можно использовать для записи многострочных блоков текста. Внутри такой строки возможно присутствие кавычек и апострофов, главное, чтобы не было трех кавычек подряд.

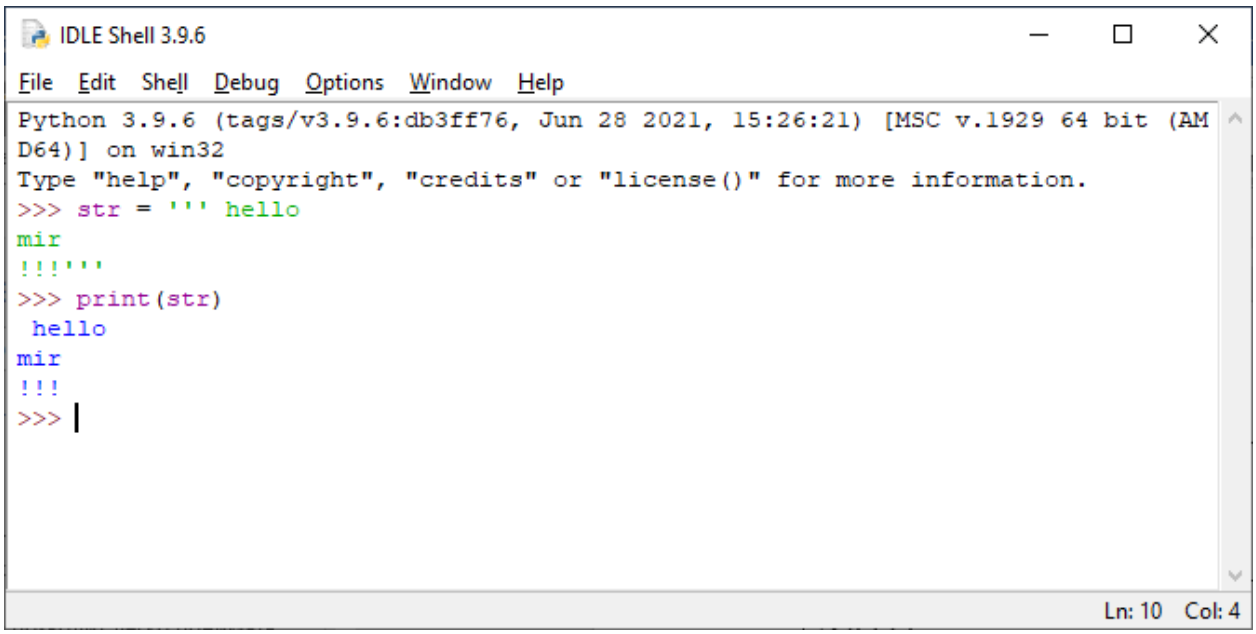

Рисунок 3.1

Таблица 3.2 – Функции и методы строк

Функция или метод	**Назначение**
1	**2**
S = 'str'; S = "str"; S = '''str'''; S = """str"""	Литералы строк
S = "s\np\ta\nbbb"	Экранированные последовательности
S = r"C:\temp\new"	Неформатированные строки (подавляют экранирование)
S = b"byte"	Строка байтов
S1 + S2	Конкатенация (сложение строк)
S1 * 3	Повторение строки
S[i]	Обращение по индексу
S[i:j:step]	Извлечение среза
len(S)	Длина строки
S.find(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или –1
S.rfind(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или –1
S.index(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
S.rindex(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError

Продолжение табл. 3.2

1	2
S.replace(шаблон, замена[, maxcount])	Замена шаблона на замену. maxcount ограничивает количество замен
S.split(символ)	Разбиение строки по разделителю
S.isdigit()	Состоит ли строка из цифр
S.isalpha()	Состоит ли строка из букв
S.isalnum()	Состоит ли строка из цифр или букв
S.islower()	Состоит ли строка из символов в нижнем регистре
S.isupper()	Состоит ли строка из символов в верхнем регистре
S.isspace()	Состоит ли строка из неотображаемых символов (пробел, символ перевода страницы ('\f'), "новая строка" ('\n'), "перевод каретки" ('\r'), "горизонтальная табуляция" ('\t') и "вертикальная табуляция" ('\v'))
S.istitle()	Начинаются ли слова в строке с заглавной буквы
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру
S.startswith(str)	Начинается ли строка S с шаблона str
S.endswith(str)	Заканчивается ли строка S шаблоном str
S.join(список)	Сборка строки из списка с разделителем S
ord(символ)	Символ в его код ASCII
chr(число)	Код ASCII в символ
S.capitalize()	Переводит первый символ строки в верхний регистр, а все остальные в нижний
S.center(width, [fill])	Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию)
S.count(str, [start],[end])	Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию)
S.expandtabs([tabsize])	Возвращает копию строки, в которой все символы табуляции заменяются одним или несколькими пробелами, в зависимости от текущего столбца. Если TabSize не указан, размер табуляции полагается равным 8 пробелам
S.lstrip([chars])	Удаление пробельных символов в начале строки
S.rstrip([chars])	Удаление пробельных символов в конце строки
S.strip([chars])	Удаление пробельных символов в начале и в конце строки

Окончание табл. 3.2

1	2
S.partition(шаблон)	Возвращает кортеж, содержащий часть перед первым шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий саму строку, а затем две пустых строки
S.rpartition(sep)	Возвращает кортеж, содержащий часть перед последним шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий две пустых строки, а затем саму строку
S.swapcase()	Переводит символы нижнего регистра в верхний, а верхнего – в нижний
S.title()	Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний
S.zfill(width)	Делает длину строки не меньшей width, по необходимости заполняя первые символы нулями
S.ljust(width, fillchar=" ")	Делает длину строки не меньшей width, по необходимости заполняя последние символы символом fillchar
S.rjust(width, fillchar=" ")	Делает длину строки не меньшей width, по необходимости заполняя первые символы символом fillchar

ВАЖНО: При вызове методов необходимо помнить, что строки в Python относятся к категории неизменяемых последовательностей, то есть все функции и методы могут лишь создавать новую строку.

Пример 1

Условие: Пользователь вводит текст. Вывести исходный текст, заменив в нем слово программирование на ПрОгРаМиРоВаНиЕ. Вычислить количество всех слов.

Решение:

```
s = input('Введите исходную строку: ')
newS = s.replace('программирование', 'ПрОгРаМиРоВаНиЕ')
print(newS)
```

Результат работы программы приведен на рис. 3.2.

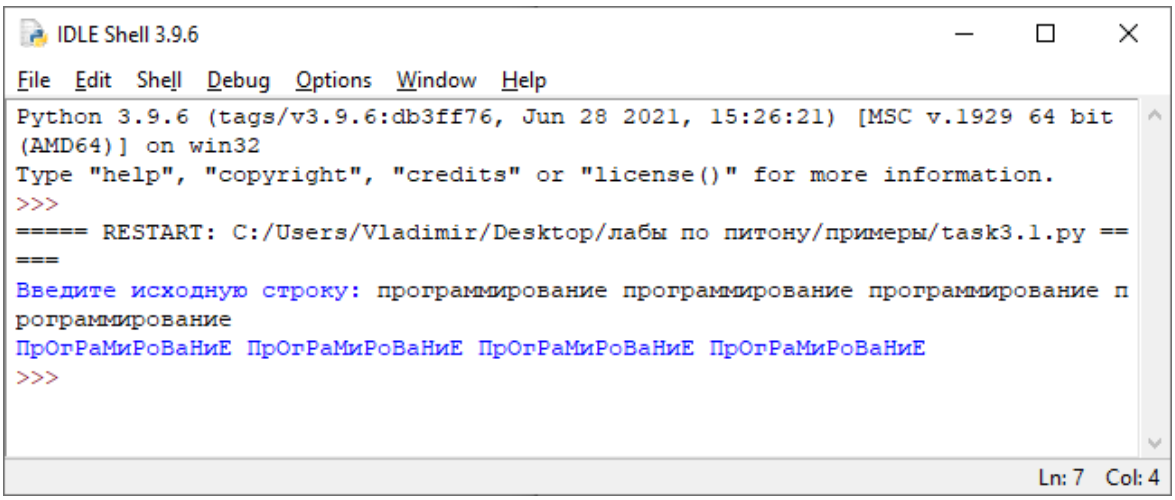

Рисунок 3.2

Пример 2

Условие: Исходный текст набран с ошибками: иногда отсутствуют пробелы после вопросительных знаков. Вставить 1 пробел после каждого вопросительного знака, если он отсутствует перед следующим словом. А также вычислить количество слов «программирование».

Решение:

```
s = input('Введите исходную строку: ')
finish = ''

while s.find('?') != -1 and s.find('?') != len(s)-1:
    finish += s[0:s.find('?')+1] # срез с начала строки до ?
    if s[s.find('?')+1] != ' ':
        finish += ' '

    s = s[s.find('?')+1:len(s)] # перезаписываем строку в которой ищем

finish += s[0:len(s)]
print(finish)
```

Результат программы:

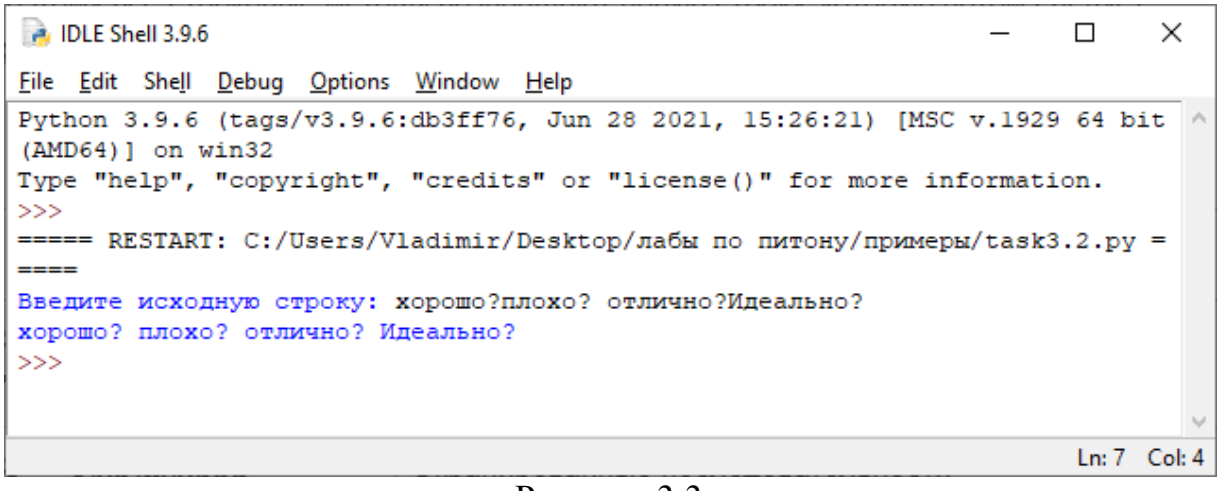

Рисунок 3.3

Пример 3

Условие: Дан список из 5 языков программирования. Переписать в другой список только те, в которые начинаются на ‘p’. Затем упорядочить по алфавиту второй список.

Решение:

```
language = ['python', 'pascal', 'c++', 'php', 'java']
newList = []

for i in range(len(language)):
    if language[i][0] == 'p':
        newList.append(language[i])

print('Второй список до сорттировки: ', newList)
newList.sort()
print('Второй список после сорттировки: ', newList)
```

Результат программы:

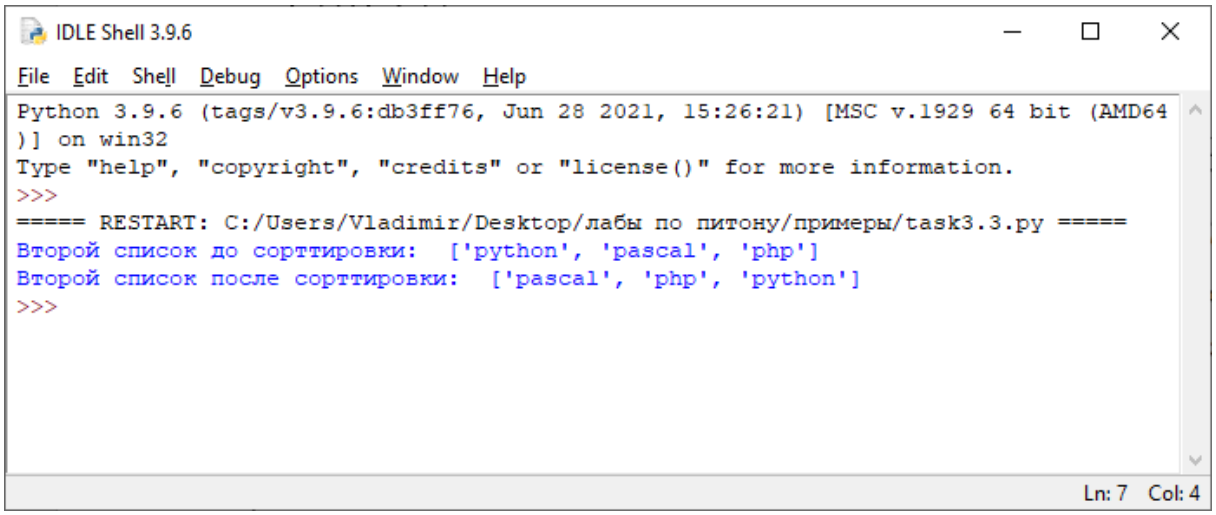

Рисунок 3.4

3.2 Задания для самостоятельной работы

Задание 1

1 Пользователь вводит текст. Вычислить количество слов, начинающихся на «м». Количество слов «Компьютер» или «компьютер», а также количество предложений.

2 Пользователь вводит текст. Заменить в тексте слова «ПК» на «компьютер», подсчитав их количество.

3 Пользователь вводит текст. Вывести исходный текст, заменив в нем слово «Иванов И.И.» на «Сидоров А.А.». Заменить круглые скобки на фигурные, подсчитав их количество.

4 Пользователь вводит текст. Вывести исходный текст, заменив в нем слово «Pascal» на «C++» и подсчитав их количество. Вычислить количество слов «компьютер»,

5 Пользователь вводит текст. Вывести исходный текст, заменив в нем слово «плохо» на «хорошо». Вычислить количество всех слов.

6 Пользователь вводит текст. Вычислить количество слов, начинающихся на «А». Количество слов «мало» или «Мало». Заменить в тексте слова «доллар» на «рубль».

7 Пользователь вводит текст. Заменить в тексте слова «Максимальный» на «Наибольший». Удалить все слова «Иванов И.И.», вычислить количество предложений

8 Пользователь вводит текст. Заменить в тексте слова «кризис» на «проблема», подсчитав их количество. Удалить все слова «компьютер»,

9 Пользователь вводит текст. Вывести исходный текст, заменив в нем квадратные скобки на круглые. Вычислить количество всех слов и количество появления слова «обучаемый».

10 Пользователь вводит текст на. Вывести исходный текст, заменив в нем слово «проблема» на «задача». Удалить все слова «Иванов И.И.».

11 Пользователь вводит текст. Вывести исходный текст, заменив в нем слово «три» на «удовлетворительно». Вычислить количество слов, начинающихся на «к».

12 Пользователь вводит текст. Вывести исходный текст, заменив в нем слово «дублирование» на «копирование». Вычислить количество всех слов.

13 Пользователь вводит текст. Вывести исходный текст, заменив в нем слово «четыре» на «хорошо». Вычислить количество всех слов и предложений. Заменить все скобки на пробелы.

14 Пользователь вводит текст. Вывести исходный текст, заменив в нем слово «Pascal» на «C++». Удалить символы '*'. Заменить все цифра на пробелы.

15 Пользователь вводит текст. Вывести исходный текст, заменив в нем слово «дублирование» на «копирование». Вычислить количество всех слов.

Задание 2

1 Исходный текст набран с ошибками: иногда отсутствуют пробелы после точек. Вставить 1 пробел после каждой точки, если он отсутствует перед следующим предложением. А также вычислить количество слов и предложений.

2 Исходный текст набран с ошибками: Вывести исходный текст, заменив в нем строчные (малые) буквы, следующие за точкой и произвольным количеством пробелов, на прописные (большие) буквы. В исходном тексте может быть много предложений и точек. А также вычислить количество слов и предложений.

3 В тексте заменить цифры на их словесные названия Исходный текст набран с ошибками: между некоторыми словами по несколько пробелов. Заменить в тексте подряд идущие пробелы одним пробелом. (

4 В тексте заменить символы арифметических операций (+-*/) на их словесные названия.

5 Исходный текст набран с ошибками: после слова может находиться один или более пробелов перед точкой (или нет). Вывести исходный текст, убрав в нем эти пробелы перед точкой (между словом и точкой). В исходном тексте может быть много предложений и точек. А также удалить символы '#'.

6 Исходный текст набран с ошибками. Вывести исходный текст, заменив в нем строчные(малые) буквы, следующие за точкой и одним пробелом, на прописные(большие) буквы. В исходном тексте может быть много предложений и точек. Вычислить количество слов, начинающихся на букву «А».

7 Пользователь вводит текст на русском языке. Вывести исходный текст, заменив в нем все заглавные (прописные (большие)) буквы на строчные (малые). Вычислить количество слов, а также количество предложений. А также удалить символы '@'.

8 Исходный текст набран с ошибками: Выражения, заключенные в скобки, имеют один пробел в начале и в конце. Вывести исходный текст, убрав в нем пробелы после открывающей скобки, а также перед закрывающей скобкой. В исходном тексте может быть много выражений, заключенных в скобки. (

9 Исходный текст набран с ошибками. Вывести исходный текст, заменив в нем строчные (малые) буквы, следующие за точкой и произвольным количеством пробелов, на прописные (большие) буквы. В исходном тексте может быть много предложений и точек. А также вычислить количество слов «ПК».

10 Исходный текст набран с ошибками: после слова может находиться один или более пробелов перед запятой (или нет). Вывести исходный текст, убрав в нем пробелы перед запятой (между словом и запятой). В исходном тексте может быть много предложений и запятых. А также удалить символы ‘-'.

11 Исходный текст набран с ошибками: Выражения, заключенные в скобки, имеют один или более пробелов вначале и в конце (или нет). Вывести исходный текст, убрав в нем пробелы после открывающей скобки, а также перед закрывающей скобкой. В исходном тексте может быть много выражений, заключенных в скобки.

12 Исходный текст набран с ошибками: после слова может находиться один пробел перед запятой или нет. убрать в тексте эти пробелы перед запятой (между словом и запятой). В исходном тексте может быть много предложений и запятых. А также удалить символы '$'.

13 Исходный текст набран с ошибками: некоторые слова по ошибке начинаются не с одной первой заглавной буквы, а с двух заглавных букв. Исправить текст.

14 Исходный текст набран с ошибками: иногда отсутствуют пробелы после запятых. Вставить 1 пробел после каждой запятой, если он отсутствует перед следующим словом. А также вычислить количество слов «Информатика».

15 Исходный текст набран с ошибками: иногда отсутствуют пробелы после точек. Вставить 1 пробел после каждой точки, если он отсутствует перед следующим предложением, а также вычислить количество предложений. А также удалить квадратные скобки.

Задание 3

1 Дан список фамилий сотрудников *(массив [.] string)*. Переписать в другой список только фамилии, чья длина больше 7 букв. Затем упорядочить по алфавиту второй список.

2 Дан список фамилий сотрудников *(массив [.] string)*. Переписать в другой список только те фамилии, которые заканчиваются на ‘а’. Затем упорядочить по алфавиту второй список.

3 Дан список фамилий сотрудников *(массив [.] string)*. Переписать в другой список только те фамилии, которые заканчиваются на ‘в’. Затем упорядочить по алфавиту второй список.

4 Дан список фамилий сотрудников. Переписать в другой список только те фамилии, в которых вторая буква ‘л’. Затем упорядочить по алфавиту второй список.

5 Дан список из 10 городов *(массив [.] string)*. Переписать в другой список только те города, в которых третья буква ‘к’. Затем упорядочить по алфавиту второй список.

6 Дан список из 10 городов *(массив [.] string)*. Переписать в другой список только те города, которые заканчиваются на ‘в’. Затем упорядочить по алфавиту второй список

7 Дан список из 10 городов. Переписать в другой список только те города, чье название длиннее 7 букв. Затем упорядочить по алфавиту второй список.

8 Задан список из 20 названий горных вершин *(массив [.] string)*. Переписать в другой список только те вершины, чье название длиннее 7 букв. Затем упорядочить по алфавиту второй список.

9 Задан список из 20 названий горных вершин *(массив [.] string)*. Переписать в другой список только те вершины, название которых оканчивается на «тау» или “tau”. Затем упорядочить по алфавиту второй список.

10 Задан список из 20 названий горных вершин *(массив [.] string)*. Переписать в другой список только те вершины, название которых оканчивается на «рок» или ‘rok’. Затем упорядочить по алфавиту второй список.

11 Задан список из 20 названий горных вершин *(массив [.] string)*. Переписать в другой список только те вершины, название которых вторая буква «М». Затем упорядочить по алфавиту второй список.

12 Задан список из 10 имен девочек *(массив [.] string)*. Переписать в другой список только те имена, в которых есть ровно 1 буква «Р». Затем упорядочить по алфавиту второй список.

13 Задан список из 10 имен девочек *(массив [.] string)*. Переписать в другой список только те имена, в которых нет буквы «Р». Затем упорядочить по алфавиту второй список.

14 Дан список фамилий сотрудников *(массив [.] string)*. Переписать в другой список только те фамилии, которые заканчиваются на ‘ова’ или ‘ova’. Затем упорядочить по алфавиту второй список.

15 Задан список из 20 названий горных вершин *(массив [.] string)*. Переписать в другой список только те вершины, название которых оканчивается на «рок» или ‘rok’. Затем упорядочить по алфавиту второй список.

ЛАБОРАТОРНАЯ РАБОТА № 4
РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

4.1 Методические рекомендации

Регулярные выражения, также называемые regex, используются практически во всех языках программирования. В python они реализованы в стандартном модуле re.

Он широко используется в естественной обработке языка, веб-приложениях, требующих проверки ввода текста (например, адреса электронной почты), и почти во всех проектах в области анализа данных, которые включают в себя интеллектуальную обработку текста.

Синтаксис RegEx

Синтаксис у регулярных выражений необычный. Символы могут быть как буквами или цифры, так и метасимволами, которые задают шаблон строки.

Таблица 4.1 – Символы для регулярных выражений

Символ	Назначение
.	Один символ кроме новой строки
\.	Просто точка ., обратный слеш \ убирает магию всех специальных символов.
\d	Одна цифра
\D	Один символ кроме цифры
\w	Один буквенный символ, включая цифры
\W	Один символ кроме буквы и цифры
\s	Один пробельный (включая таб и перенос строки)
\S	Один не пробельный символ
\b	Границы слова
\n	Новая строка
\t	Табуляция

Таблица 4.2 – Модификаторы для регулярных выражений

Модификатор	Назначение
$	Конец строки
^	Начало строки
ab|cd	Соответствует ab или de.
[ab-d]	Один символ: a, b, c, d
[^ab-d]	Любой символ, кроме: a, b, c, d
()	Извлечение элементов в скобках
(a(bc))	Извлечение элементов в скобках второго уровня

Таблица 4.3 – Повторы для регулярных выражений

Повторы	Назначение
[ab]{2}	2 непрерывных появления a или b
[ab]{2,5}	от 2 до 5 непрерывных появления a или b
[ab]{2,}	2 и больше непрерывных появления a или b
+	одно или больше
*	0 или больше
?	0 или 1

Регулярные выражения используются для:

– для определения нужного формата, например, телефонного номера или email-адреса;

- для разбивки строк на подстроки;
- для поиска, замены и извлечения символов;
- для быстрого выполнения нетривиальных операций.

Процедуры и функции

re.match(pattern, string)

Этот метод ищет по заданному шаблону в начале строки. Например, если вызвать метод match() на строке «AV Analytics AV» с шаблоном «AV», то он завершится успешно. Но если искать «Analytics», то результат будет отрицательный.

re.search(pattern, string)

Метод похож на match(), но ищет не только в начале строки. В отличие от предыдущего, search() вернёт объект, если пытаться найти «Analytics».

re.findall(pattern, string)

Возвращает список всех найденных совпадений. У метода findall() нет ограничений на поиск в начале или конце строки.

re.split(pattern, string, [maxsplit=0])

Этот метод разделяет строку по заданному шаблону.

re.sub(pattern, repl, string)

Ищет шаблон в строке и заменяет его на указанную подстроку. Если шаблон не найден, строка остается неизменной.

Пример

Условие: извлеките имя пользователя, имя домена и суффикс из данных email-адресов.

Решение:

```
import re

emails = """zuck26@facebook.com
page33@google.com
jeff42@amazon.com"""

pattern = r'(\w+)@([A-Z0-9]+)\.([A-Z]{2,4})' # вводим условия поиска в переменную

result = re.findall(pattern, emails, flags=re.IGNORECASE) #флаг re.IGNORECASE нужен что бы искать без учета регистра символов

print(result)
```

Результат программы приведен на рис. 4.1.

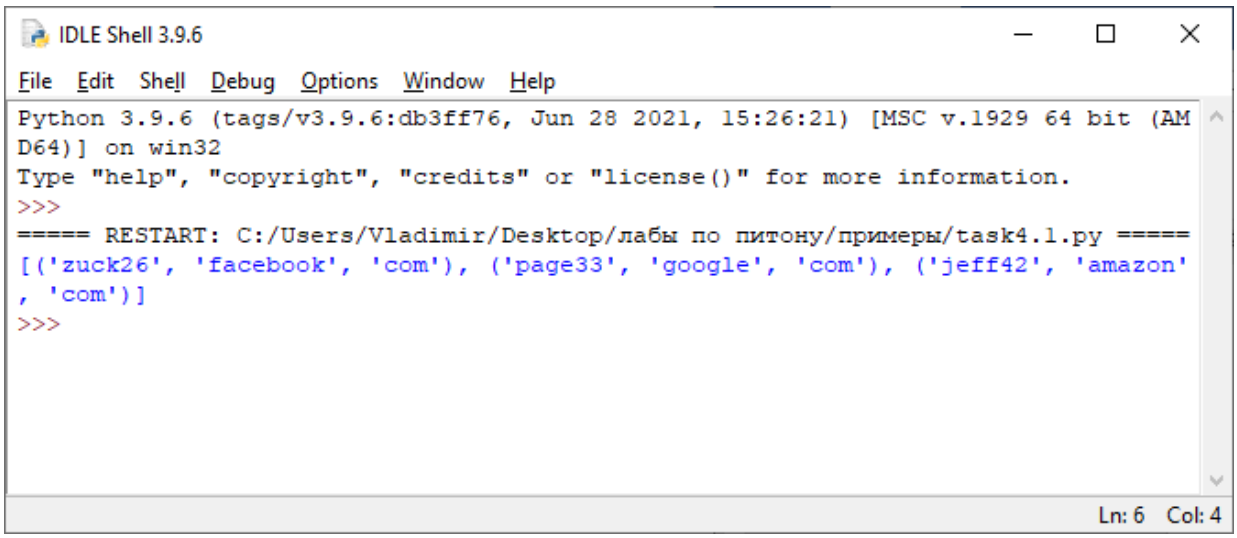

Рисунок 4.1

4.2. Задания для самостоятельной работы

Задание

1 Написать регулярное выражение, определяющее является ли данная строка строкой "abcdefghijklmnopqrstuv18340" или нет.

Пример правильных выражений: abcdefghijklmnopqrstuv18340.
Пример неправильных выражений: abcdefghijklmnoasdfasdpqrstuv18340.

2 Написать регулярное выражение, определяющее является ли данная строка GUID с или без скобок. Где GUID это строчка, состоящая из 8, 4, 4, 4, 12 шестнадцатеричных цифр разделенных тире.

Пример правильных выражений: e02fd0e4-00fd-090A-ca30-0d00a0038ba0.
Пример неправильных выражений: e02fd0e400fd090Aca300d00a0038ba0.

3 Написать регулярное выражение, определяющее является ли заданная строка правильным MAC-адресом.

Пример правильных выражений: aE:dC:cA:56:76:54.
Пример неправильных выражений: 01:23:45:67:89:Az.

4 Написать регулярное выражение, определяющее является ли данная строчка валидным URL адресом. В данной задаче правильным URL считаются адреса http и https, явное указание протокола также может отсутствовать. Учитываются только адреса, состоящие из символов, т.е. IP адреса в качестве URL не присутствуют при проверке. Допускаются поддомены, указание порта доступа через двоеточие, GET запросы с переда-

чей параметров, доступ к подпапкам на домене, допускается наличие якоря через решетку. Однобуквенные домены считаются запрещенными. Запрещены спецсимволы, например «–» в начале и конце имени домена. Запрещен символ «_» и пробел в имени домена. При составлении регулярного выражения ориентируйтесь на список правильных и неправильных выражений, заданных ниже.

Пример правильных выражений: http://www.zcontest.ru, http://zcontest.ru.

Пример неправильных выражений: Just Text, http://a.com.

5 Написать регулярное выражение, определяющее является ли данная строчка шестнадцатиричным идентификатором цвета в HTML. Где #FFFFFF для белого, #000000 для черного, #FF0000 для красного и т.д.

Пример правильных выражений: #FFFFFF, #FF3421, #00ff00.

Пример неправильных выражений: 232323, f#fddee, #fd2.

6 Написать регулярное выражение, определяющее, является ли данная строчка датой в формате dd/mm/yyyy. Начиная с 1600 года до 9999 года.

Пример правильных выражений: 29/02/2000, 30/04/2003, 01/01/2003.

Пример неправильных выражений: 29/02/2001, 30-04-2003, 1/1/1899.

7 Написать регулярное выражение, определяющее является ли данная строчка валидным E-mail адресом согласно RFC под номером 2822.

Пример правильных выражений: mail@mail.ru, valid@megapochta.com.

Пример неправильных выражений: bug@@@com.ru, @val.ru, Just Text2.

8 Составить регулярное выражение, определяющее, является ли заданная строка IP адресом, записанным в десятичном виде.

Пример правильных выражений: 127.0.0.1, 255.255.255.0.

Пример неправильных выражений: 1300.6.7.8, abc.def.gha.bcd.

9 Проверить, надежно ли составлен пароль. Пароль считается надежным, если он состоит из 8 или более символов, где символом может быть английская буква, цифра и знак подчеркивания. Пароль должен содержать хотя бы одну заглавную букву, одну маленькую букву и одну цифру.

Пример правильных выражений: C00l_Pass, SupperPas1.

Пример неправильных выражений: Cool_pass, C00l.

10 Проверить, является ли заданная строка шестизначным числом, записанным в десятичной системе счисления без нулей в старших разрядах.

Пример правильных выражений: 123456, 234567.
Пример неправильных выражений: 1234567, 12345.

11 Есть текст со списками цен. Извлечь из него цены в USD, RUR, EU.

Пример правильных выражений: 23.78 USD.
Пример неправильных выражений: 22 UDD, 0.002 USD.

12 Проверить существуют ли в тексте цифры, за которыми не стоит «+».

Пример правильных выражений: (3 + 5) – 9 × 4.
Пример неправильных выражений: 2 * 9 – 6 × 5.

13 Создать запрос для вывода только правильно написанных выражений со скобками (количество открытых и закрытых скобок должно быть одинаково).

Пример правильных выражений: (3 + 5) – 9 × 4.
Пример неправильных выражений: ((3 + 5) – 9 × 4.

14 Время имеет формат часы:минуты. И часы, и минуты состоят из двух цифр, например 09:00. Напишите регулярное выражение для поиска времени в строке: “Завтрак в 09:00”. Учтите, что “37:98” – некорректное время.

15 Разобрать арифметическое выражение.

Арифметическое выражение состоит из двух чисел и операции между ними, например:

1 + 2
1.2 *3.4
-3/ -6
-2-2

Список операций: “+”, «-», “*” и “/”.

Также могут присутствовать пробелы вокруг оператора и чисел.

Напишите регулярное выражение, которое найдёт как всё арифметическое действие, так и (через группы) два операнда.

ЛАБОРАТОРНАЯ РАБОТА № 5
ФУНКЦИИ

5.1 Методические рекомендации

Функция – это фрагмент кода, который можно вызвать из любого места программы. Создание пользовательских функций, которые позволят уменьшить избыточность программного кода и повысить его структурированность.

Определение функции. Функция определяется с помощью ключевого слова def по следующей схеме:

def <Имя функции>([<Параметры>]):
<Тело функции>
[return <Результат>]

Имя функции должно быть уникальным идентификатором, состоящим из латинских букв, цифр и знаков подчеркивания, причем имя функции не может начинаться с цифры. В качестве имени нельзя использовать ключевые слова, кроме того, следует избегать совпадений с названиями встроенных идентификаторов. Регистр символов в названии функции также имеет значение.

После имени в круглых скобках можно указать один или несколько параметров через запятую, а если функция не принимает параметры, то указываются только круглые скобки. После круглых скобок ставится двоеточие.

Тело функции представляет собой составную конструкцию. Как и в любой составной конструкции, инструкции внутри функции выделяются одинаковым количеством пробелов слева. Концом функции считается инструкция, перед которой находится меньшее количество пробелов. Если тело функции не содержит инструкций, то внутри ее необходимо разместить оператор pass, который не выполняет никаких действий. Этот оператор удобно использовать при отладке программы.

Необязательная инструкция return позволяет вернуть из функции какое-либо значение в качестве результата. После исполнения этой инструкции выполнение функции будет остановлено. Это означает, что инструкции, следующие после оператора return, никогда не будут выполнены.

Пример 1
Код программы:

```
def myFunction():
   print('Это будет выводиться')
   return
   print('это уже не выведет')

myFunction()
```

Результат программы приведен на рис. 5.1.

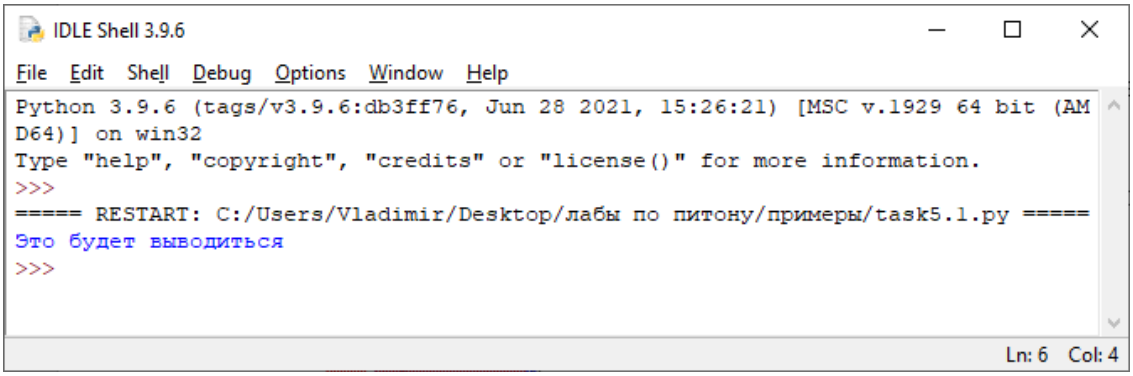

Рисунок 5.1

Аргументы функции

Функция может принимать произвольное количество аргументов или не принимать их вовсе. Также распространены функции с произвольным числом аргументов, функции с позиционными и именованными аргументами, обязательными и необязательными.

Пример 2

Код программы

```
def myFunction(value, default = 20):
    print('value = ', value)
    print('default = ', default)


print('Передадим только value')
myFunction(10)
print('передадим оба параметра')
myFunction(100, 200)
```

Результат работы программы приведен на рис. 5.2.

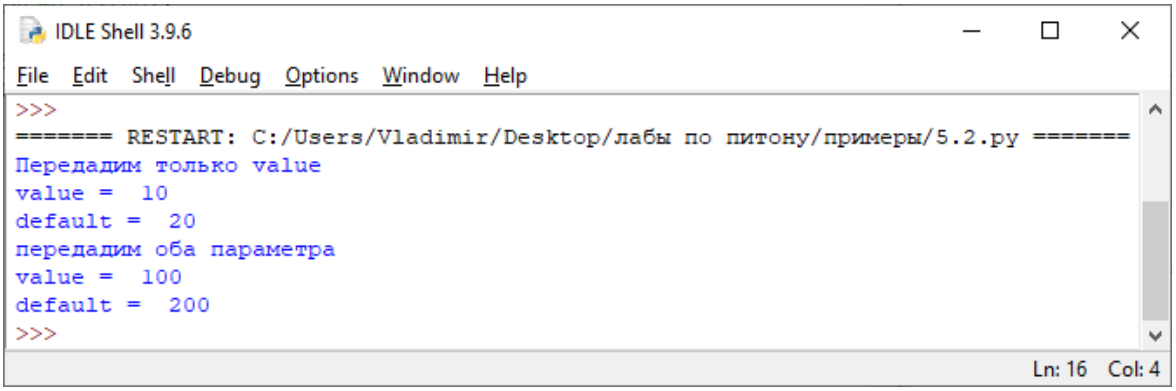

Рисунок 5.2

Рекурсивные функции

Рекурсивная функция — это та, которая вызывает сама себя.

Пример 3

Условие: Написать программу, вычисляющую N факториал.

** Факториал числа — это число, умноженное на каждое предыдущее число вплоть до 1.

Решение:

```
def factorialRecursive(n):
    if n == 1:
        return n
    else:
        return n*factorialRecursive(n-1)

n = 5
res = factorialRecursive(n)
print('Факториал ', n, ' = ', res)
```

Результат программы приведен на рис. 5.3.

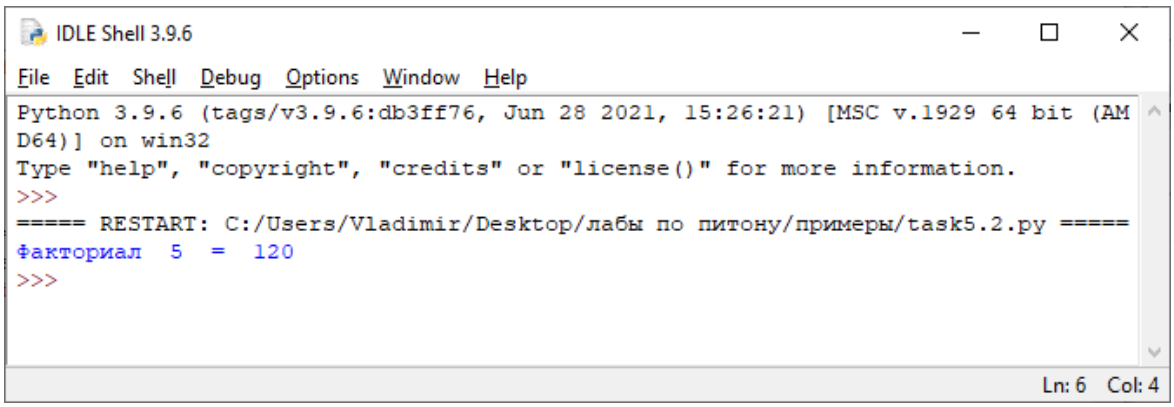

Рисунок 5.3

Анонимные функции

Анонимные функции могут содержать лишь одно выражение, но и выполняются они быстрее. Анонимные функции создаются с помощью инструкции lambda.

Пример 4

Условие: работы lambda

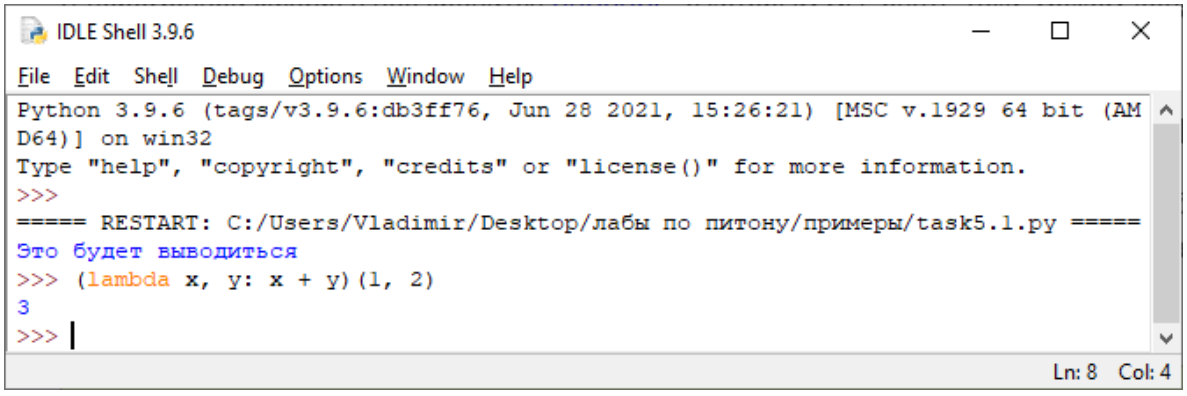

Рисунок 5.4

5.2 Задания для самостоятельной работы

Задание 1

1 Создать функцию, которая возвращает меньшее из двух данных чисел.

2 Создать функцию, которая переводит время, заданное в минутах, в секунды.

3 Создать функцию, которая определяет периметр треугольника по трем его сторонам.

4 Создать функцию, которая возвращает номер квадранта, в котором находится точка.

5 Создать функцию, которая возвращает среднее арифметическое трех данных чисел.

6 Создать функцию, которая определяет площадь круга по его радиусу.

7 Создать функцию, которая возвращает остаток от деления двух натуральных чисел

8 Создать функцию, которая переводит радианы в градусы.

9 Создать функцию, которая определяет длину отрезка по его координатам.

10 Создать функцию, которая возвращает в долларах сумму, заданную в рублях.

11 Создать функцию, которая возвращает большее из двух данных чисел.

12 Создать функцию, которая определяет длину окружности по заданному радиусу.

13 Создать функцию, которая переводит скорость из км/час в м/с.

14 Создать функцию, которая возвращает среднее геометрическое двух данных чисел.

15 Создать функцию, которая возвращает в рублях сумму, заданную в долларах.

Задание 2

Вариант № 1. Реализовать функцию. Функция вычисляет площадь параллелограмма $s= a\ b \cos \alpha$ по заданным двум сторонам и углу между ними. В главной программе задано два параллелограмма. Найти их площади, вызвав функцию 2 раза.

Вариант № 2. Реализовать функцию. Функция вычисляет площадь квадрата по заданной стороне. В главной программе задано два квадрата. Найти их площади, вызвав функцию 2 раза.

Вариант № 3. Реализовать функцию. Функция вычисляет диагональ прямоугольника по заданным двум сторонам. В главной программе задано два прямоугольника. Найти их диагонали, вызвав функцию 2 раза.

Вариант№ 4. Реализовать функцию. Функция вычисляет площадь s = ½ (a +b) h равнобедренной трапеции по заданным двум основаниям a, b и высоте h. В главной программе заданы две равнобедренные трапеции. Найти их площади, вызвав функцию 2 раза.

Вариант № 5. Реализовать функцию. Функция вычисляет площадь окружности по заданному радиусу R. В главной программе заданы две окружности. Найти их площади, вызвав функцию 2 раза.

Вариант№ 6. Реализовать функцию. Функция вычисляет объем сферы v = 4/3πR3 по заданному радиусу R. В главной программе заданы две сферы. Найти их объемы, вызвав функцию 2 раза.

Вариант № 7. Реализовать функцию. Функция вычисляет объем квадратной призмы по заданной высоте H и стороне основания a. В главной программе заданы две квадратные призмы. Найти их объемы, вызвав функцию 2 раза.

Вариант № 8. Реализовать функцию. Функция вычисляет объем правильной треугольной призмы по заданной высоте H и стороне основания a. В главной программе задано две треугольные призмы. Найти их объемы, вызвав функцию 2 раза.

Вариант № 9. Реализовать функцию. Функция вычисляет объем цилиндра по заданной высоте H и радиусу основания R. В главной программе задано два цилиндра. Найти их объемы, вызвав функцию 2 раза.

Вариант № 10. Реализовать функцию. Функция вычисляет площадь боковой поверхности S = π R конуса по заданной высоте H и радиусу основания R . В главной программе задано два конуса. Найти их площади боковых поверхностей, вызвав функцию 2 раза.

Вариант № 11. Реализовать функцию. Функция вычисляет объем v = 1/3 H π R2 конуса по заданной высоте H и радиусу основания R. В главной программе задано два конуса. Найти их объемы, вызвав функцию 2 раза.

Вариант № 12. Реализовать функцию. Функция вычисляет объем v = 1/3 H π (R2 + r2 + R r) усеченного конуса по заданной высоте H и двум радиусам оснований R и r. В главной программе задано два конуса. Найти их объемы , вызвав функцию 2 раза.

Вариант № 13. Реализовать функцию. Функция вычисляет длину вектора на плоскости по заданным координатам x и y. В главной программе задано два вектора. Найти их длины, вызвав функцию 2 раза.

Вариант № 14. Реализовать функцию. Функция вычисляет тангенс угла наклона (между осью OX и вектором) вектора на плоскости по заданным координатам x и y. В главной программе задано два вектора. Найти их углы, вызвав функцию 2 раза.

Задание 3

1 Описать рекурсивную функцию для вычисления n-го члена ряда 3, 0.3, 0.03, 0.003, … 0.00003.

2 Описать рекурсивную функцию для вычисления n-го члена ряда 10, 5, 5/2, 5/4, 5/8, 5/16, … 5/256.

3 Описать рекурсивную функцию для вычисления n-го члена ряда 6, 3, 3/2, ¾, 3/8, … 3/256.

4 Описать рекурсивную функцию для вычисления n-го члена ряда 1, 2, 4, 8, … 256.

5 Описать рекурсивную функцию для вычисления n-го члена ряда 2, 4, 6, … 20.

6 Описать рекурсивную функцию для вычисления n-го члена ряда 1, 3, 5, … 21.

7 Описать рекурсивную функцию для вычисления n-го члена ряда 1, 1.2, 1.4, 1.6, … 3.0.

8 Описать рекурсивную функцию для вычисления n-го члена ряда 0.7; 0.07; 0.007; 0.0007; … 0.00000000007.

9 Описать рекурсивную функцию для вычисления n-го члена ряда 1, 3, 9, 27, 81 … 729.

10 Описать рекурсивную функцию для вычисления n-го члена ряда 7, 17, 27, 37, … 97.

11 Описать рекурсивную функцию для вычисления n-го члена ряда 3, 3.3, 3.6, 3.9, 4.2, … 6.

12 Описать рекурсивную функцию для вычисления n-го члена ряда 20, 10, 5, 5/2, 5/4, 5/8, 5/16, … 5/128.

13 Описать рекурсивную функцию для вычисления n-го члена ряда 5,2; 5,4; 5,6; 5,8; 6; 6,2; … 10.

14 Описать рекурсивную функцию для вычисления n-го члена ряда 40, 20, 10, 5, 5/2, 5/4, … 5/128.

15 Описать рекурсивную функцию для вычисления n-го члена ряда 1, 4, 16, 64, 256, 1024, 4096.

ЛАБОРАТОРНАЯ РАБОТА № 6
МОДУЛИ

6.1 Методические рекомендации

Модулем в языке Python называется любой файл с программным кодом. Каждый модуль может импортировать другой модуль, получая, таким образом, доступ к атрибутам (переменным, функциям и классам), объявленным внутри импортированного модуля. Следует заметить, что импортируемый модуль может содержать программу не только на языке Python – так можно импортировать скомпилированный модуль, написанный на языке C.

Проверить, является модуль главной программы или импортированным модулем, позволяет код:

```
if __name__ == '__main__':
        print("Это главная программа")
else:
        print("Импортированный модуль")
```

Инструкция import

Импортировать модуль позволяет инструкция import. Например, чтобы подключить модуль time для получения текущей даты с помощью функции strftime():

```
import time
print (time.strftime("%d.%m.%Y"))
```

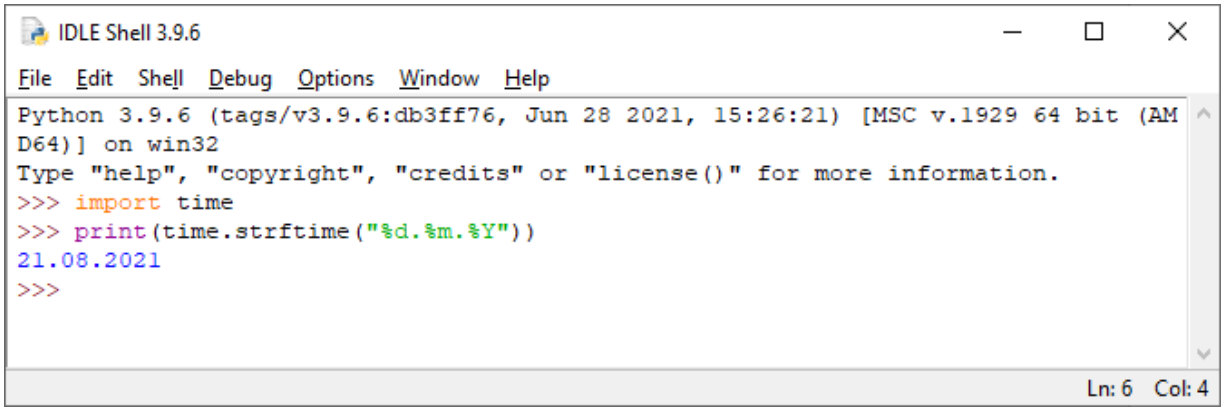

Рисунок 6.1

Инструкция import имеет следующий формат:

Import <Название модуля 1> [as <Псевдоним 1>] [, …,
<Название модуля N> [as <Псевдоним N>]]

После ключевого слова import указывается название модуля. Стоит обратить внимание на то, что название не должно содержать расширения и пути к файлу. При именовании модулей необходимо учитывать, что операция импорта создает одноименный идентификатор, – это означает, что название модуля должно полностью соответствовать правилам именования переменных. Можно создать модуль с именем, начинающимся с цифры, но подключить такой модуль будет нельзя, Кроме того следует избегать совпадений имен модулей с ключевыми словами, встроенными идентификаторами и названиями модулей, входящих в стандартную библиотеку.

За один раз можно импортировать сразу несколько модулей, записав их через запятую. Для примера подключим модули time и math:

```
import time, math

print(time.strftime("%d.%m.%Y"))
print (math.pi)
```

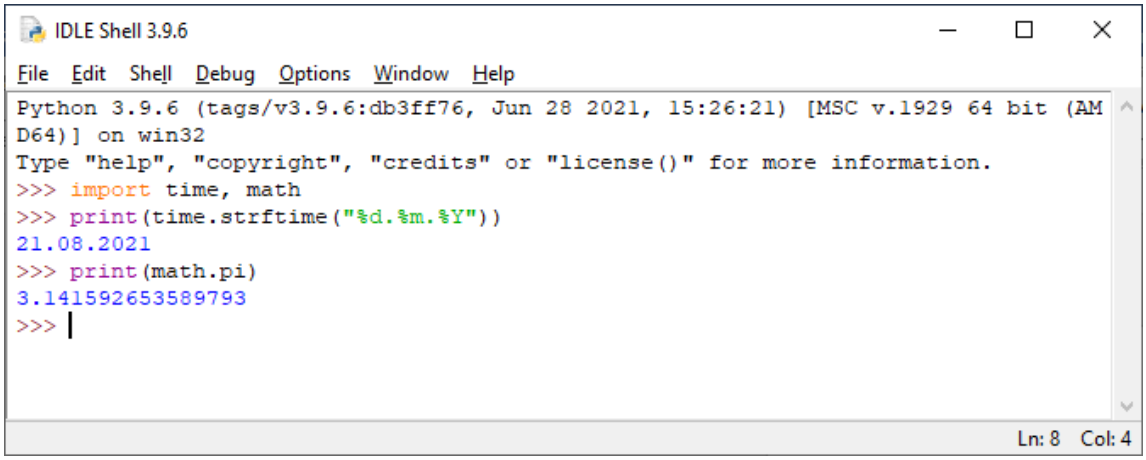

Рисунок 6.2

Если название модуля слишком длинное и его неудобно указывать каждый раз для доступа к атрибутам, то можно создать псевдоним. Псевдоним задается после ключевого слова as. Создадим псевдоним для модуля math:

```
import math as m
print(m.pi)
```

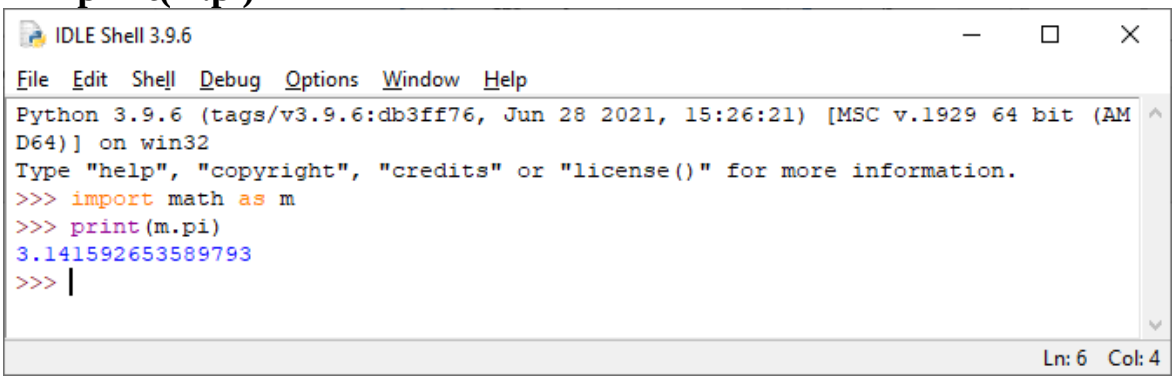

Рисунок 6.3

Теперь доступ к атрибутам модуля math может осуществляться только с помощью идентификатора m. Идентификатор math в этом случае использовать уже нельзя.

Все содержимое импортированного модуля доступно только через идентификатор, указанный в инструкции import. Это означает, что любая глобальная переменная на самом деле является глобальной переменной модуля. По этой причине модули часто используют как пространства имен.

Инструкция from

Для импортирования только определенных идентификаторов из модуля можно воспользоваться инструкцией from. Её формат таков:

from <Название модуля> import <Идентификатор 1> [as <псевдоним 1>]
[, …, <Идентификатор N> [as <Псевдоним N>]]

Это позволяет импортировать модуль и сделать доступным только указанные идентификаторы. Для длинных имен можно назначить псевдонимы, указав их после ключевого слова as. В качестве примера сделаем доступными константу pi и функцию floor() из модуля math, А для названия функции создадим псевдоним:

Код программы:

```
from math import pi, floor as f

print(pi)  # Вывод числа ПИ
# Вызываем функцию floor() через идентификатор f
print(f(5.49))  # Выведет 5
```

Результат программы приведен на рис. 6.4.
:

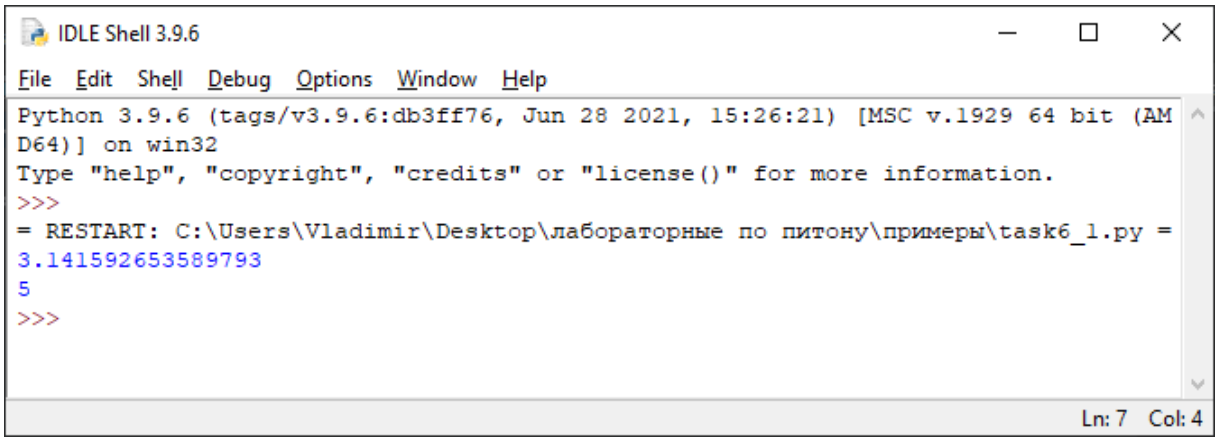

Рисунок 6.4

Обзор модуля time

Модуль time используется для решения задач, связанных со временем.

Таблица 6.1 – Функции модуля time

Функция	Описание
1	2
time.altzone	смещение DST часового пояса в секундах к западу от нулевого меридиана. Если часовой пояс находится восточнее, смещение отрицательно
time.asctime([t])	преобразовывает кортеж или struct_time в строку вида "Thu Sep 27 16:42:37 2012". Если аргумент не указан, используется текущее время
time.clock()	в Unix, возвращает текущее время. В Windows, возвращает время, прошедшее с момента первого вызова данной функции
time.ctime()	преобразует время, выраженное в секундах с начала эпохи в строку вида "Thu Sep 27 16:42:37 2012"
time.daylight	не 0, если определено, зимнее время или летнее (DST).
time.gmtime([сек])	преобразует время, выраженное в секундах с начала эпохи в struct_time, где DST флаг всегда равен нулю
time.localtime([сек])	как gmtime, но с DST флагом
time.mktime(t)	преобразует кортеж или struct_time в число секунд с начала эпохи. Обратна функции time.localtime
time.sleep(сек)	приостановить выполнение программы на заданное количество секунд
time.strftime(формат, [t])	преобразует кортеж или struct_time в строку по формату
time.strptime(строка [, формат])	разбор строки, представляющей время в соответствии с форматом. Возвращаемое значение struct_time. Формат по умолчанию: "%a %b %d %H:%M:%S %Y"
time.struct_time	тип последовательности значения времени. Имеет интерфейс кортежа. Можно обращаться по индексу или по имени. 0. tm_year 1. tm_mon 2. tm_mday 3. tm_hour 4. tm_min

Окончание табл. 6.1

1	2
	5. tm_sec 6. tm_wday 7. tm_yday 8. tm_isdst
time.time()	время, выраженное в секундах с начала эпохи
time.timezone	смещение местного часового пояса в секундах к западу от нулевого меридиана. Если часовой пояс находится восточнее, смещение отрицательно
time.tzname	кортеж из двух строк: первая – имя DST часового пояса, второй – имя местного часового пояса

Таблица 6.2 – Форматы модуля time

Формат	**Значение**
%a	Сокращенное название дня недели
%A	Полное название дня недели
%b	Сокращенное название месяца
%B	Полное название месяца
%c	Дата и время
%d	День месяца [01,31]
%H	Час (24-часовой формат) [00,23]
%I	Час (12-часовой формат) [01,12]
%j	День года [001,366]
%m	Номер месяца [01,12]
%M	Число минут [00,59]
%p	До полудня или после (при 12-часовом формате)
%S	Число секунд [00,61]
%U	Номер недели в году (нулевая неделя начинается с воскресенья) [00,53]
%w	Номер дня недели [0(Sunday),6]
%W	Номер недели в году (нулевая неделя начинается с понедельника) [00,53]
%x	Дата
%X	Время
%y	Год без века [00,99]
%Y	Год с веком
%Z	Временная зона
%%	Знак '%'

6.2 Задания для самостоятельной работы

Вариант 1

Написать программу, которая раз в 10 секунд выводит текущее время.

Вариант 2

Написать программу, которая раз в 15 секунд выводит текущую дату и время.

Вариант 3

Написать программу, которая будет выводить текущую дату по формату, который задаст пользователь (например, D,M;Y выведет 23,05;2020).

Вариант 4

Написать программу, которая будет выводить текущее время по формату, который задаст пользователь (например, H:M: Sc, где H – часы, M – минуты, S – секунды).

Задание 2

Четные варианты:

Написать программу, где пользователю будет предлагаться на выбор 4 действия:

1 Приостановить программу на время, введенное пользователем.

2 Вывести текущую дату в формате (день месяца/сокращенное название месяца/год).

3 Высчитать, сколько дней осталось до Нового года.

4 Завершить программу.

Программа должна быть реализована в виде диалогового окна.

Нечетные варианты

Написать программу, где пользователю будет предлагаться на выбор 4 действия:

1 Приостановить программу на время, введенное пользователем.

2 Вывести текущее время в формате (число секунд: число минут: час(12- часовой формат), информация до полудня или после)).

3 Высчитать сколько дней осталось до 1 мая.

4 Завершить программу.

Программа должна быть реализована в виде диалогового окна.

ЛАБОРАТОРНАЯ РАБОТА № 7
РАБОТА С ФАЙЛАМИ

7.1 Методические рекомендации

Прежде чем работать с файлом, необходимо создать объект файла с помощью функции open(). Функция имеет следующий формат:

open(<Путь к файлу>[, mode='r'][,buffering=-1][, encoding=None][, errors=None][, newline=None][, closefd=True])

В первом параметре указывается путь к файлу. Путь может быть абсолютным или относительным. При указании абсолютного пути в Windows следует учитывать, что в Python слэш является специальным символом. По этой причине слэш необходимо удваивать или вместо обычных строк использовать неформатированные строки. Например,

"C:\\temp\\new\\file.txt"* или *r"C:\temp\new\file.txt"

Необязательный параметр mode в функции open() может принимать значения, приведенные в табл. 7.1.

Таблица 7.1 – Параметр mode в функции open()

Режим	Обозначение
'r'	открытие на чтение (является значением по умолчанию).
'w'	открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.
'x'	открытие на запись, если файла не существует, иначе исключение.
'a'	открытие на дозапись, информация добавляется в конец файла.
'b'	открытие в двоичном режиме.
't'	открытие в текстовом режиме (является значением по умолчанию).
'+'	открытие на чтение и запись

Указать кодировку, которая будет использоваться при записи и чтении файла, позволяет параметр encoding.

Пример:

F = open(r“file.txt”, “w”, encoding=“utf-8”)

В параметре errors можно указать уровень обработки ошибок. Возможные значения:

“strict” – при ошибке возбуждается исключение ValueError – значение по умолчанию;

“replace” – неизвестный символ заменяется символом вопроса или символом с кодом \ufffd;

“ignore” – неизвестные символы игнорируются;

“xmlcharrefreplace” – неизвестный символ заменяется последовательностью &#xxxx;

“backslashreplace” – неизвестный символ заменяется последовательностью \uxxxx.

Параметр newline задает режим обработки символов конца строк. Поддерживаемые им значения:

None – выполняется стандартная обработка символов конца строки. Например, в Windows при чтении символы \r\n преобразуются в символ \n, а при записи производится обратное преобразование.

“” (пустая строка) – обработка символов конца строки не выполняется.

“<Специальный символ>” – Указанный специальный символ используется для обозначения конца строки, и никакая дополнительная обработка не выполняется. В качестве специального символа можно указать лишь \r\n, \r *u* \n

Метод close()

После открытия файла в Python его нужно закрыть. Таким образом освобождаются ресурсы и убирается мусор. Python автоматически закрывает файл, когда объект присваивается другому файлу.

Существуют следующие способы:

1 Проще всего после открытия файла закрыть его, используя метод close(). После закрытия этот файл нельзя будет использовать до тех пор, пока он не будет заново открыт.

2 Также можно написать try/finally, которое гарантирует, что, если после открытия файла операции с ним приводят к исключениям, он закроется автоматически.

Пример:

```
F = open(‘example.txt’, ‘r’)
try:
# работа с файлом
Finally:
        f.close()
```

ВАЖНО! Файл нужно открыть до инструкции try, потому что если инструкция open сама по себе вызовет ошибку, то файл не будет открываться для последующего закрытия.

Второй метод гарантирует, что если операции над файлом вызовут исключения, то он закроется до того, как программа остановится.

Инструкция with

Еще один подход – использовать инструкцию with, которая упрощает обработку исключений с помощью инкапсуляции начальных операций, а также задач по закрытию и очистке.

В таком случае инструкция close не нужна, потому что with автоматически закроет файл.

Пример:

```
with open('example.txt') as f:
    # работа с файлом
```

Методы для работы с файлами:

file.close() – закрывает открытый файл;

file.fileno() – возвращает целочисленный дескриптор файла

file.flush() – очищает внутренний буфер;

file.isatty() – возвращает True, если файл привязан к терминалу;

file.next() – возвращает следующую строку файла;

file.read(n) – чтение первых n символов файла;

file.readline() – читает одну строчку строки или файла;

file.readlines() – читает и возвращает список всех строк в файле;

file.seek(offset[, where]) – устанавливает текущую позицию в файле;

file.seekable() – проверяет, поддерживает ли файл случайный доступ. Возвращает True, если да;

file.tell() – возвращает текущую позицию в файле;

file.truncate(n) – уменьшает размер файла. Если n указана, то файл обрезается до n байт, если нет – до текущей позиции;

file.write(str) – добавляет строку str в файл;

file.writelines(sequence) – добавляет последовательность строк в файл.

Read:

Функция read() используется для чтения содержимого файла после открытия его в режиме чтения (r).

file.read(size)

где file – это объект файла;

size – количество символов, которое нужно прочитать. Если не указывать, то файл прочитается целиком.

Функция readline() и readlines()

Функция *readline()* используется для построчного чтения содержимого файла. Она используется для крупных файлов. С ее помощью можно получать доступ к строке.

Функция readlines() используется для считывания всех строк сразу, при этом формируется list, где каждая строка является элементом list.

Пример 1:

Код программы:

```
with open('temp.txt', 'r', encoding='utf-8') as f:
    #считывание всей строки строки
    print(f.readline())
    #Считывание всего файла
    print(f.readlines())
```

Результат программы:

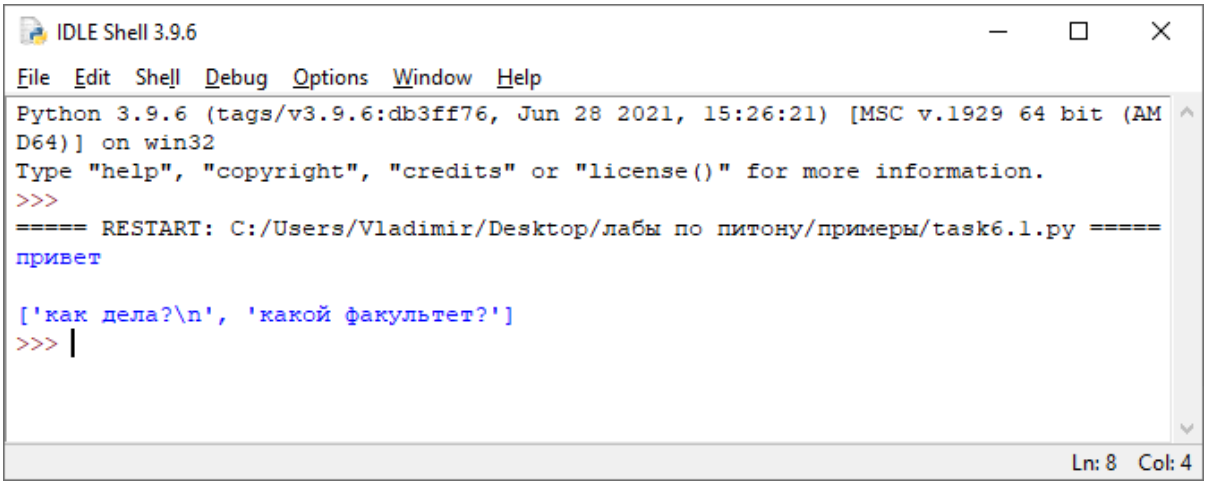

Рисунок 7.1

Функция write()

Функция write() используется для записи в файлы Python, открытые в режиме записи.

Если пытаться открыть файл, которого не существует, в этом режиме, тогда будет создан новый.

Пример 2:

Код программы:

```
with open('writefile.txt', 'w', encoding='utf-8') as f:
    f.write('Привет\nМир!!!')
```

Результат работы приведен на рис. 7.2.

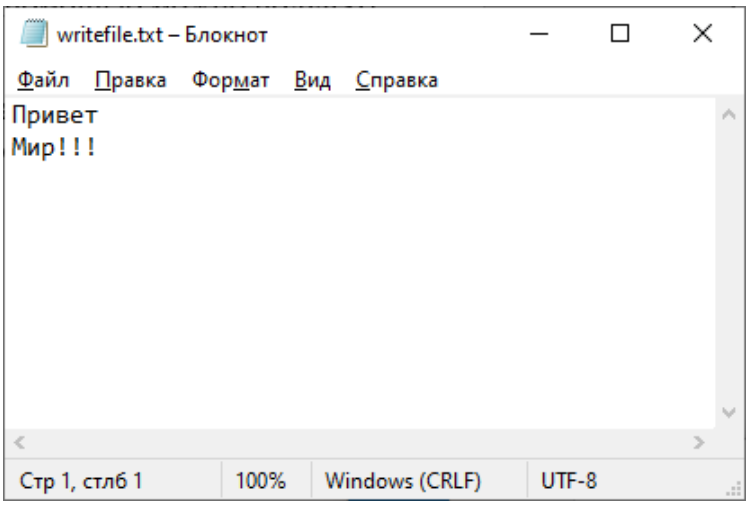

Рисунок 7.2

7.2 Задания для самостоятельной работы

Задание 1

1 Дан файл вещественных чисел a.txt. Найти количество отрицательных и количество положительных элементов.

2 Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и произведение элементов меньших 1 и больших 0.

3 Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и сумму отрицательных элементов.

4 Дан файл вещественных чисел a.txt. Найти количество элементов равных 5 и сумму положительных элементов.

5 Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и сумму положительных элементов.

6 Дан файл вещественных чисел a.txt. Найти количество положительных элементов и сумму положительных элементов.

7 Дан файл вещественных чисел a.txt. Найти количество отрицательных и количество положительных элементов.

8 Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и произведение элементов меньших 1 и больших 0.

9 Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и сумму отрицательных элементов.

10 Дан файл вещественных чисел a.txt. Найти количество элементов равных 5 и сумму положительных элементов.

11 Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и сумму положительных элементов.

12 Дан файл вещественных чисел a.txt. Найти количество положительных элементов и сумму положительных элементов.

13 Дан файл вещественных чисел a.txt. Найти количество положительных элементов и произведение элементов меньших 1 и больших 0.

14 Дан файл вещественных чисел a.txt. Переписать положительные элементы в файл b.txt

15 Дан файл вещественных чисел a.txt. Найти количество отрицательных и количество положительных элементов.

Задание 2

1 Организовать текстовый файл. Заменить в файле все маленькие латинские буквы на большие. (создавая новый дополнительный файл)

2 Из заданного входного файла считать символы и записать в один новый файл только буквы, в другой новый файл только цифры.

3 Организовать текстовый файл. Организовать замену символов в файле. «Старый» символ и «новый» символ запрашиваются и вводятся с клавиатуры (создавая новый дополнительный файл).

4 Организовать текстовый файл. Преобразовать файл, удалив в нем лишние пробелы (создав новый дополнительный файл).

5 Организовать текстовый файл, состоящий из строк. Заменить в файле все большие латинские буквы на маленькие (создав новый дополнительный файл).

6 Организовать текстовый файл. Заменить в файле все цифры на '*' (создав новый дополнительный файл).

7 Организовать текстовый файл. Заменить в файле все буквы (нецифры) на '*' (создав новый дополнительный файл).

8 Организовать текстовый файл. Удалить в файле все цифры (создав новый дополнительный файл).

9 Из заданного входного файла считать символы и записать в один новый файл только большие латинские буквы, в другой новый файл только малые латинские буквы и посчитать количество цифр.

10 Организовать текстовый файл. Удалить в файле все буквы (создав новый дополнительный файл).

11 Из заданного входного файла считать символы и записать в новый файл все символы за исключением символов разделителей: пробелы, точки, запятые, двоеточия и т. д.

12 Организовать текстовый файл, оставив в файле только буквы (создав новый дополнительный файл).

13 Из заданного входного файла считать символы и записать в новый файл только большие буквы латинского алфавита.

14 Организовать файл вещественных чисел. Заменить все положительные компоненты файла их квадратными корнями, а все отрицательные компоненты их квадратами, создав новый дополнительный файл.

15 Организовать файл целых чисел. Удалить из файла все отрицательные компоненты (создав новый дополнительный файл).

16 Организовать файл целых чисел, заменить все элементы файла от –10 до 10 на противоположные, создав новый дополнительный файл.

17 Организовать файл целых чисел Все числа, кратные 3, заменить их удвоенным произведением (создав новый дополнительный файл).

18 Организовать файл целых чисел, заменить все элементы файла от –2 до 5 на противоположные (создав новый дополнительный файл).

ЛАБОРАТОРНАЯ РАБОТА № 8
ОБРАБОТКА ИСКЛЮЧЕНИЙ

8.1 Методические рекомендации

Исключения – это извещения интерпретатора, возбуждаемые в случае возникновения ошибки в программном виде или при наступлении какого-либо события. Если в коде не предусмотрена обработка исключения, то выполнение программы прерывается и выводится сообщение об ошибке.

Существует три типа ошибок в программе:

1) *синтаксические* – это ошибки в имени оператора или функции, отсутствие закрывающей или открывающей кавычек и т. д. ошибки в синтаксисе языка. Как правило, интерпретатор предупредит о наличии ошибки, а программа не будет выполняться совсем;

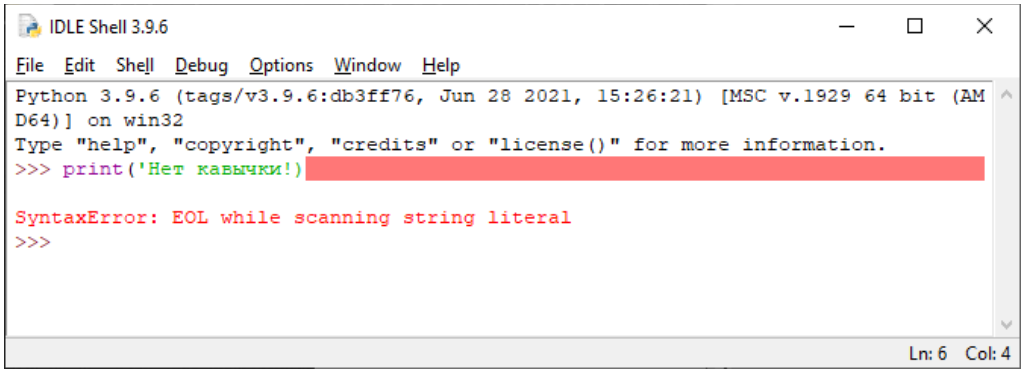

Рисунок 8.1

2) *логические* – это ошибки в логике программы, которые можно выявить только по результатам её работы. Как правило, интерпретатор не предупреждает о наличии такой ошибки, и программа будет успешно выполняться, но результат её выполнения будет не тем, на который мы рассчитывали. Выявить и исправить такие ошибки достаточно сложно;

3*) ошибки времени выполнения* – это ошибки, которые возникают во время работы программы. Причиной являются события, не предусмотренные программистом. Классическим примером служит деление на ноль:

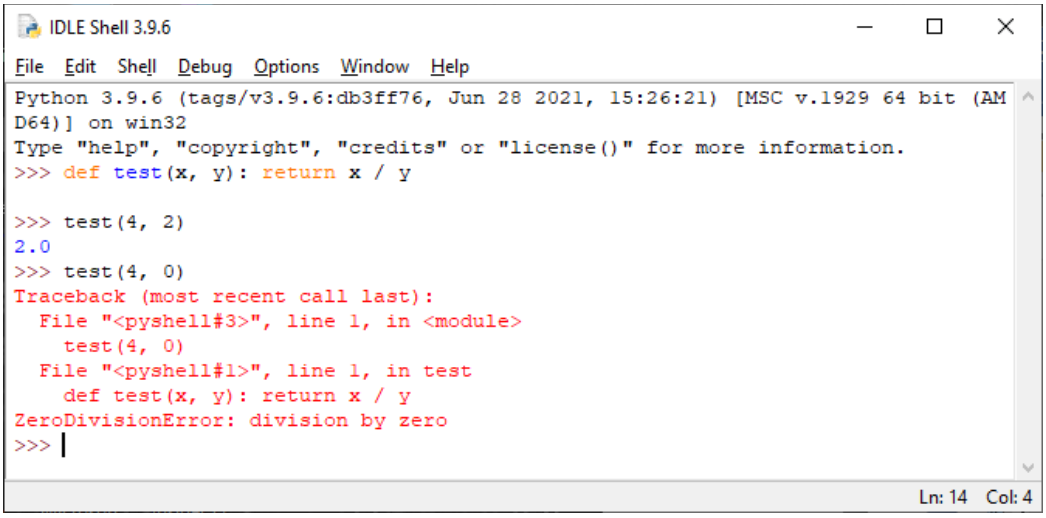

Рисунок 8.2

Необходимо заметить, что в языке Python исключения возбуждаются не только при ошибке, но и как уведомление о наступлении каких-либо событий. Например, метод index() возбуждает исключение ValueError, если искомый фрагмент не входит в строку:

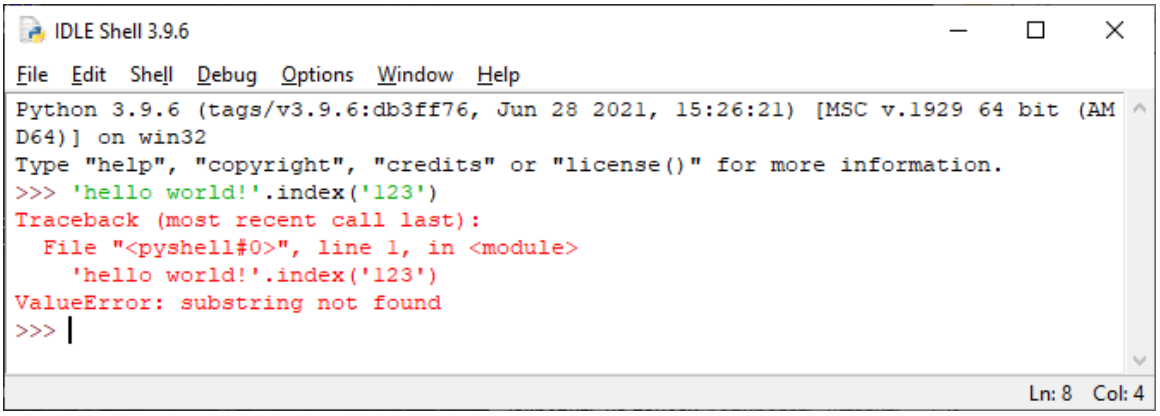

Рисунок 8.3

Инструкции try…except…else…finally

Для обработки исключений предназначена инструкция try. Формат инструкции:

try:
<Блок, в котором перехватываются исключения>
[except [<Исключение1> [as <Объект исключения>]]:
<Блок выполняемый при возникновении исключения>]
[…
except [<Исключение1> [as <Объект исключения>]]:
<Блок выполняемый при возникновении исключения>
[else:
<Блок, выполняемый, если исключение не возникло>]
[finally:
<Блок, выполняемый в любом случае>]

Инструкции, в которых перехватываются исключения, должны быть расположены внутри блока try. В блоке except в параметре <Исключение1> Указывается класс обрабатываемого исключения. Например, обработать исключение, возникающее при делении на ноль, можно так:

Пример 1:

```
try:
   x = 1/0
except ZeroDivisionError:
   print('Обработано исключение деление на ноль')
   x = 0
print(x)
```

Результат работы программы приведен на рис. 8.4.

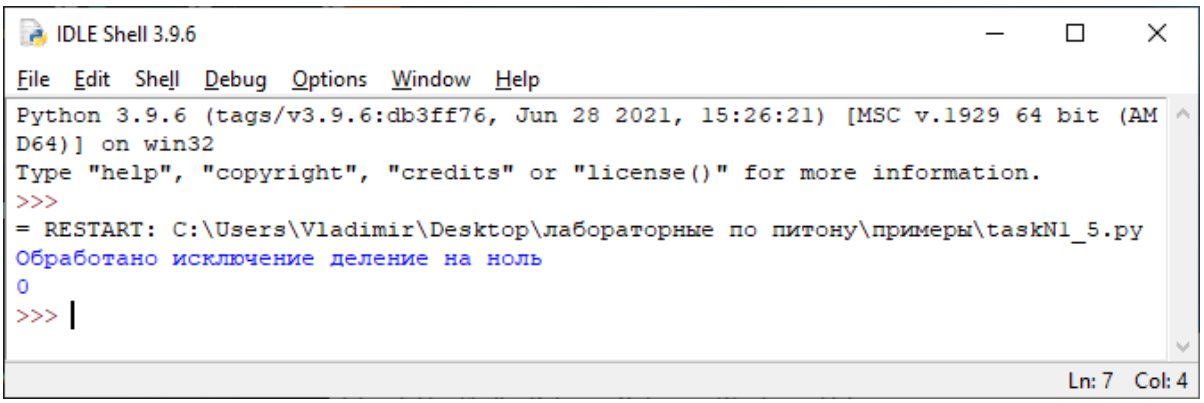

Рисунок 8.4

Если в блоке try возникло исключение, то управление передается блоку except. В случае если исключение не соответствует указанному классу, управление передается следующему блоку except. Если ни один блок except не соответствует исключению, то исключение «всплывает» к обработчику более высокого уровня. Если исключение в программе вообще нигде не обрабатывается, оно передается по умолчанию, который останавливает выполнение программы и выводит стандартную информацию об ошибке. Таким образом, в обработчике может быть несколько блоков except с разными классами исключений. Кроме того, один обработчик можно вложить в другой.

В инструкции except можно указать сразу несколько исключений, перечислив их через запятую внутри круглых скобок. Получить информацию об обрабатываемом исключении можно через второй параметр в инструкции except.

Пример 2:

```
try:
    x = 1/0
except (NameError, IndexError, ZeroDivisionError) as err:
    print(err.__class__.__name__)
    print(err)
```

Результат программы:

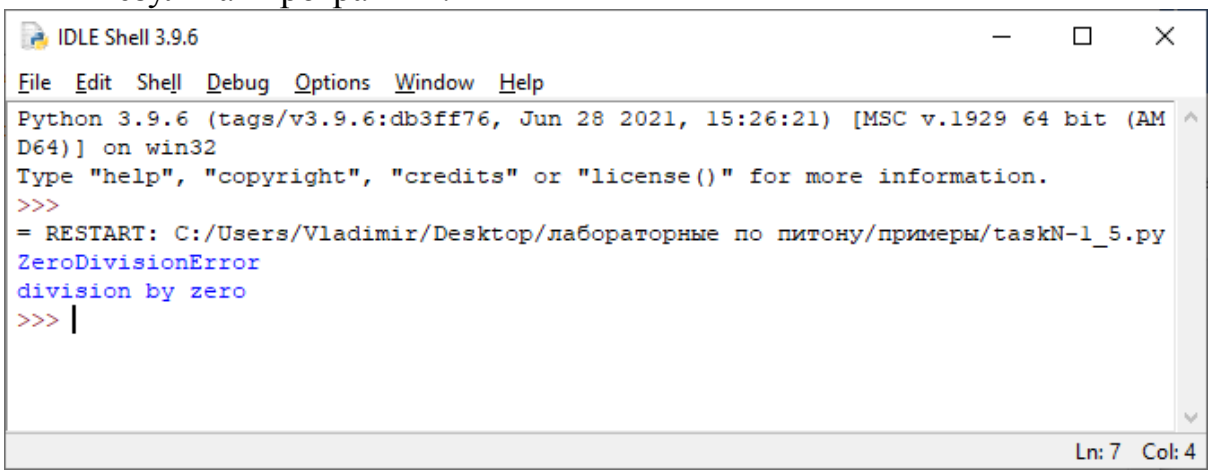

Рисунок 8.5

Классы встроенных исключений

Все встроенные исключения в языке Python представляют собой классы. Основное преимущество использования классов для перехвата всех исключений соответствующих классов потомков

Таблица 8.1 – Основные классы встроенных исключений

Класс	Описание
1	**2**
BaseException	базовое исключение, от которого берут начало все остальные
SystemExit	исключение, порождаемое функцией sys.exit при выходе из программы
KeyboardInterrupt	порождается при прерывании программы пользователем (обычно сочетанием клавиш Ctrl+C)
GeneratorExit	порождается при вызове метода close объекта generator
Exception	Базовый класс для большинства встроенных в Python исключений. Именно его необходимо наследовать при создании пользовательского класса исключения
StopIteration	порождается встроенной функцией next, если в итераторе больше нет элементов
ArithmeticError	арифметическая ошибка
FloatingPointError	порождается при неудачном выполнении операции с плавающей запятой. На практике встречается нечасто
OverflowError	возникает, когда результат арифметической операции слишком велик для представления. Не появляется при обычной работе с целыми числами (так как python поддерживает длинные числа), но может возникать в некоторых других случаях
ZeroDivisionError	деление на ноль
AssertionError	выражение в функции assert ложно
AttributeError	объект не имеет данного атрибута (значения или метода)
BufferError	операция, связанная с буфером, не может быть выполнена
EOFError	функция наткнулась на конец файла и не смогла прочитать то, что хотела
ImportError	не удалось импортирование модуля или его атрибута
LookupError	некорректный индекс или ключ
IndexError	индекс не входит в диапазон элементов
KeyError	несуществующий ключ (в словаре, множестве или другом объекте)
MemoryError	недостаточно памяти
NameError	не найдено переменной с таким именем

Окончание табл. 8.1

1	2
UnboundLocalError	сделана ссылка на локальную переменную в функции, но переменная не определена ранее
OSError	ошибка, связанная с системой
ChildProcessError	неудача при операции с дочерним процессом
ConnectionError	базовый класс для исключений, связанных с подключениями
FileExistsError	попытка создания файла или директории, которая уже существует
FileNotFoundError	файл или директория не существует
InterruptedError	системный вызов прерван входящим сигналом
IsADirectoryError	ожидался файл, но это директория
NotADirectoryError	ожидалась директория, но это файл
PermissionError	не хватает прав доступа
ProcessLookupError	указанного процесса не существует
TimeoutError	закончилось время ожидания
ReferenceError	попытка доступа к атрибуту со слабой ссылкой
RuntimeError	возникает, когда исключение не попадает ни под одну из других категорий
NotImplementedError	возникает, когда абстрактные методы класса требуют переопределения в дочерних классах
SyntaxError	синтаксическая ошибка
IndentationError	неправильные отступы
TabError	смешивание в отступах табуляции и пробелов
SystemError	внутренняя ошибка
TypeError	операция применена к объекту несоответствующего типа
ValueError	функция получает аргумент правильного типа, но некорректного значения
UnicodeError	ошибка, связанная с кодированием / раскодированием unicode в строках
UnicodeEncodeError	исключение, связанное с кодированием unicode
UnicodeDecodeError	исключение, связанное с декодированием unicode
UnicodeTranslateError	исключение, связанное с переводом unicode
Warning	предупреждение

8.2 Задания для самостоятельной работы

Вариант 1

Написать программу отлавливающую ошибку BaseException. Вывести в консоль название класса исключения.

Вариант 2

Написать программу отлавливающую ошибку ArithmeticError. Вывести в консоль название класса исключения.

Вариант 3

Написать программу отлавливающую ошибку AttributeError. Вывести в консоль название класса исключения.

Вариант 4

Написать программу отлавливающую ошибку IndexError. Вывести в консоль название класса исключения.

Вариант 5

Написать программу отлавливающую ошибку KeyError. Вывести в консоль название класса исключения.

Вариант 6

Написать программу отлавливающую ошибку NameError. Вывести в консоль название класса исключения.

Вариант 7

Написать программу отлавливающую ошибку UnboundLocalError. Вывести в консоль название класса исключения.

Вариант 8

Написать программу отлавливающую ошибку FileExistsError. Вывести в консоль название класса исключения.

Вариант 9

Написать программу отлавливающую ошибку FileNotFoundError. Вывести в консоль название класса исключения.

Вариант 10

Написать программу отлавливающую ошибку IsADirectoryError. Вывести в консоль название класса исключения.

Вариант 11

Написать программу отлавливающую ошибку NotADirectoryError. Вывести в консоль название класса исключения.

Вариант 12

Написать программу отлавливающую ошибку UnicodeError. Вывести в консоль название класса исключения.

Вариант 13

Написать программу отлавливающую ошибку ValueError. Вывести в консоль название класса исключения.

ЛАБОРАТОРНАЯ РАБОТА № 9
ВЗАИМОДЕЙСТВИЕ С ИНТЕРНЕТОМ

9.1 Методические рекомендации

Интернет прочно вошёл в нашу жизнь. Очень часто нам необходимо передать информацию на Web-сервер или, наоборот, получить с него какие-либо данные, например, котировки валют или прогноз погоды, проверить наличие писем на почтовом ящике и т. д. В состав стандартной библиотеки Python входит множество модулей, позволяющих работать практически со всеми протоколами Интернета. В данной лабораторной работе будут рассмотрены разбор URL-адреса и строки запроса на составляющие, обмен данными по протоколу HTTP с помощью модулей http.client и urllib.request.

Протоколы

Таблица 9.1 – Протоколы Интернета, входящие в состав стандартной библиотеки Python

Название	Описание
FTP	Предназначен для передачи файлов через Интернет. FTP позволяет переносить с машины на машину не только файлы, но и целые папки, включающие поддиректории на любую глубину вложений
SMTP	Используется для отправки почты и передачи почты между почтовыми серверами
HTTP	Обеспечивает передачу с удаленных серверов на локальный компьютер документов, содержащих код разметки гипертекста, написанный на языке HTML или XML, то есть веб-страниц. Данный прикладной протокол ориентирован прежде всего на предоставление информации программам просмотра веб-страниц, веб-браузерам
POP3 и IMAP	Используют для получения почты с сервера почтового ящика
TELNET	Предназначен для организации терминального доступа к удаленному узлу посредством обмена командами в символьном формате ASCII. Как правило, для работы с сервером по протоколу TELNET на стороне клиента должна быть установлена специальная программа, называемая telnet-клиентом, которая, установив связь с удаленным узлом, открывает в своем окне системную консоль операционной оболочки сервера

Разбор URL-адреса и строки запроса

С помощью модуля urllib.parse можно манипулировать URL-адресом – например, разобрать его на составляющие или получить абсолютный URL-адрес, указав базовый и относительный адреса. URL состоит из следующих элементов:

<Потокол>://<Домен>:<Порт>/<Путь>;<Параметры>?<Запрос>

Схема URL для протокола FTP выглядит по-другому:

<Протокол>://<Пользователь>:<Пароль>@<Домен>

Разобрать URL-адрес на составляющие позволяет функция urlparse()

urlparse(<URL-адрес>[, <Схема>[, разбор якоря]])

Функция возвращает объект ParseResult с результатом разбора URL-адреса. Получить значения можно с помощью атрибутов или индексов. Объект можно преобразовать в кортеж из следующих элементов: (scheme, netloc, path, params, query, fragment). Элементы соответствуют схеме URL-адреса:

Объект ParseResult, возвращаемый функцией urlparse(), содержит следующие атрибуты:

<scheme>://<netloc>/<path>;<params>?<query>#<fragment>

scheme – название протокола. Значение доступно также по индексу 0;

netloc – название домена вместе с номером порта. Значение доступно также по индексу 1;

hostname – название домена в нижнем регистре;

port – номер порта;

path – путь, значение также доступно по индексу 2;

params – параметры, значение доступно по индексу 3;

query – строка запроса;

fragment – якорь;

username – имя пользователя;

password – пароль.

Рисунок 9.1

Выполнить обратную операцию (собрать URL-адрес из отдельных значений) позволяет функция urlunparse(<Последовательность>).

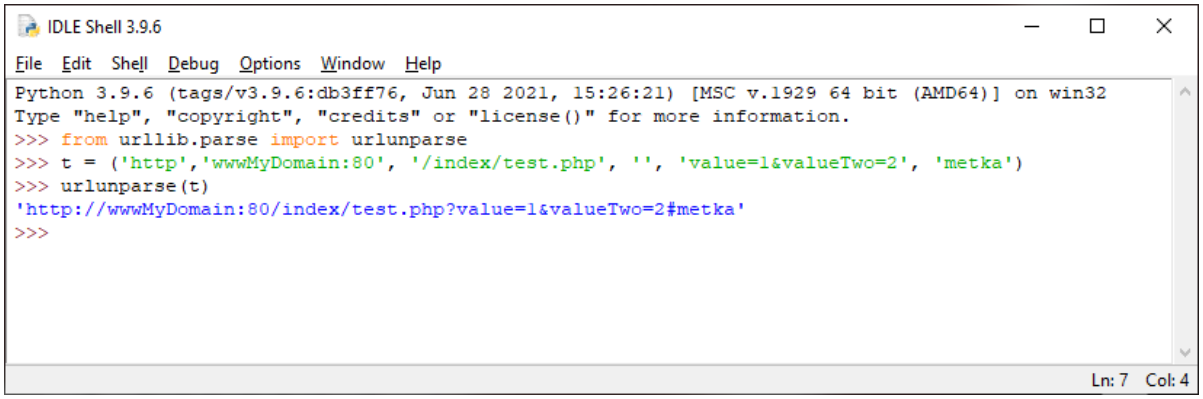

Рисунок 9.2

Кодирование и декодирование

Строка запроса является составной конструкцией, содержащей пары параметр=значение. Все специальные символы внутри названия параметра и значения кодируются последовательностями %nn. Например, для параметра str, имеющего значение “Строка” в кодировке Windows-1251, строка запроса будет выглядеть так:

str=%D1%F2%F0%EE%EA%E0

Если строка запроса содержит несколько пар параметр=значение, то они разделяются символом &. Пример:

str=%D1%F2%F0%EE%EA%E0&value=10

Разобрать строку запроса на составляющие и декодировать данные позволяет следующие функции из модуля urllib.parse

parse_qs() – разбирает строку запроса и возвращает слова с ключами, представляющими собой названия параметров, и значениями, которыми станут значения этих параметров.

parse_qs(<Строка запроса>[, keep_blank_values=False][, strict_parsing=False][, encoding=’utf-8’][, errors=’replace’])

Если в параметре keep_blank_values указано значение True, то параметры, не имеющие значений внутри строки запроса, также будут добавлены в результат. По умолчанию пустые параметры игнорируются. Если в параметре strict_parsing указано значение True, то при наличии ошибки возбуждается исключение ValueError. По умолчанию ошибки игнорируются. Параметр encoding позволяет указать кодировку данных, а параметр errors – уровень обработки ошибок.

Пример использования указанных параметров приведен на рис. 9.3.

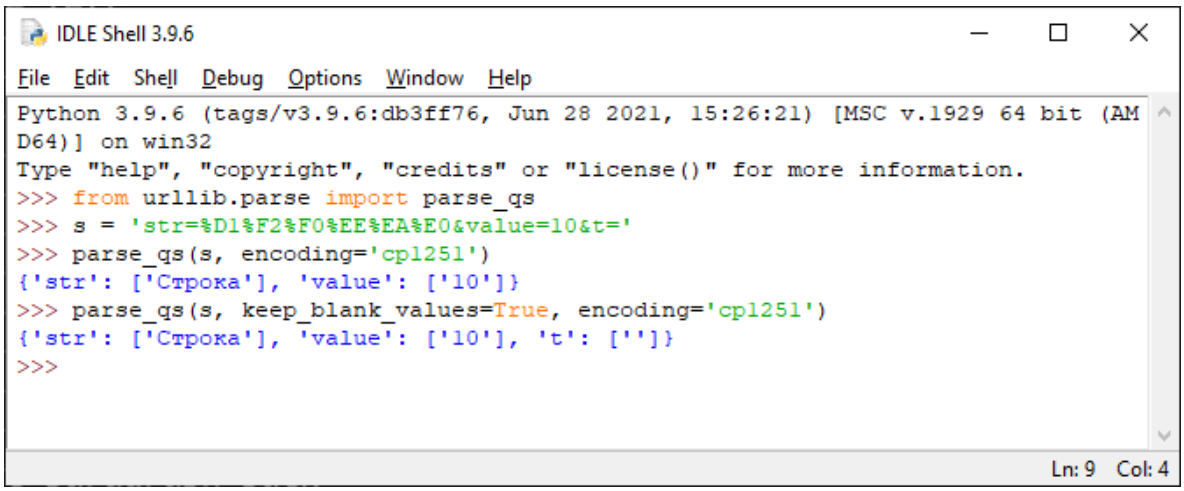

Рисунок 9.3

Преобразование относительного URL-адреса в абсолютный

Относительный путь ссылается на местоположение относительно текущей страницы. Абсолютный путь – это полный адрес страницы.

Очень часто в коде веб-страниц указываются не абсолютные URL-адреса, а относительные. При относительных URL-адрес путь определяется с учетом местоположения страницы, на которой находится ссылка, или значения параметра href тега <base>. Преобразовать относительную ссылку в абсолютный URL-адрес позволяет функция urljoin() из модуля urllib.parse. Формат функции:

urljoin(<Базовый URL-адрес>, <относительный или абсолютный Url-адрес> [, <Разбор якоря>])

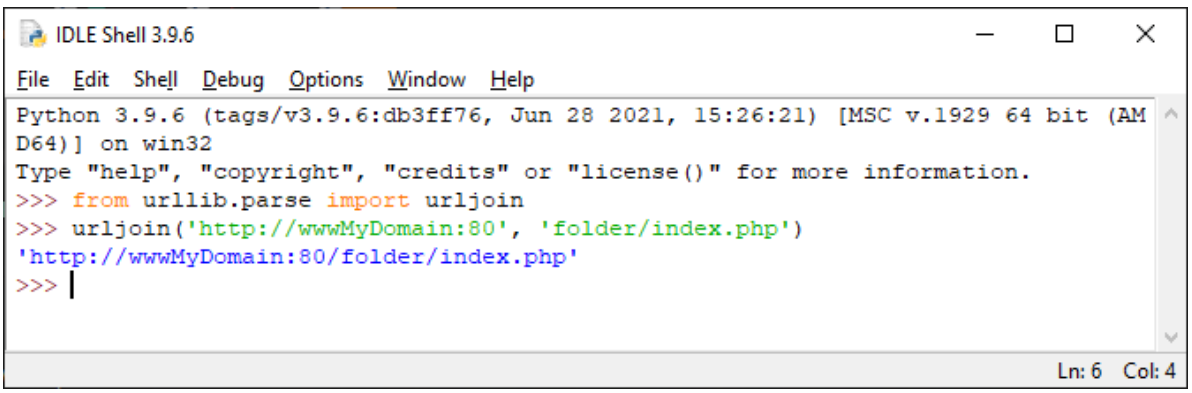

Рисунок 9.4

Обмен данными по протоколу HTTP

Модуль http.client позволяет получить информацию из Интернета по протоколам HTTP и HTTPS. Отправить запрос можно методами GET, POST и HEAD.

Для создания объекта соединения, использующего протокол HTTP, предназначен класс HTTPConnection. Его конструктор имеет следующий формат:

HTTPConnection(<Домен>[, <Порт>[, timeout[, source_address]]])

В первом параметре указывается название домена без протокола. Во втором параметре записывается номер порта – если параметр не указан, используется, используется порт 80. Номер порта можно также задать после названия домена через двоеточие

После создания объекта соединения необходимо отправить запрос, возможно, с параметрами, вызвав метод request() класса HTTPConnection. Формат метода:

Request(<Метод>, <Путь>[, body=None][, headers=<Заголовки>])

В первом параметре указывается метод передачи данных (GET, POST или HEAD). Второй параметр задает путь к запрашиваемому файлу или вызываемой программе, отсчитанный от корня сайта. Если для передачи данных используется метод GET, то после вопросительного знака можно указать передаваемые данные. В необязательном третьем параметре задаются данные, которые передаются методом POST, – допустимо указать строку, файловый объект или последовательность. Четвертый параметр задает в виде словаря HTTP-заголовки, отправляемые на сервер.

Получить объект результата запроса позволяет метод getresponse(). Он возвращает результат выполненного запроса, представленный в виде объекта класса HTTPresponse. Из него можно получить ответ сервера.

Прочитать ответ сервера можно с помощью метода read([<количество байт>]) класса HTTPResponse. Если параметр не указан, метод read() возвращает все данные, а при наличии параметра – только указанное количество байтов при каждом вызове. Если данных больше нет, метод возвращает пустую строку. Прежде чем выполнить другой запрос, данные должны быть получены полностью. Метод read() возвращает последовательность байтов, а не строку. Закрыть объект соединения позволяет метод close() класса HTTPConnection.

Пример 1:

Код программы:

```
from http.client import HTTPConnection

headers = {
   'User-Agent': 'Chrome/92.0.4515.131',
   'Accept': 'text/html, text/plain, application/xml',
   'Accept-Language': 'ru, re-RU',
   'Accept-Charset': 'windows-1251',
} # Задаем заголовки
con = HTTPConnection('www.rgups.ru') # Устанавливаем соединение с сайтом
```

con.request('GET', '/index.php', headers=headers) # Делаем запрос путь которого равен /index.php и заголовки headers

result = con.getresponse() # Получаем результат запроса

print(result.read().decode()) #Выводим результат запроса

Результат программы:

Рисунок 9.5

9.2 Задания для самостоятельной работы

Задание 1

Варианты:

1 Разобрать URL-адрес

https://yandex.ru/news/?utm_source=main_stripe_big

и вывести в консоль название протокола.

2 Разобрать URL-адрес
https://yandex.ru/news/?utm_source=main_stripe_big
и вывести в консоль название домена.

3 Разобрать URL-адрес
https://yandex.ru/news/?utm_source=main_stripe_big
и вывести в консоль путь URL-адреса.

4 Разобрать URL-адрес
https://yandex.ru/news/?utm_source=main_stripe_big
и вывести в консоль строку запроса

5 Разобрать URL-адрес
http://mehelps.ru/yakor-v-html.html#yakor_kod
и вывести в консоль название протокола.

6 Разобрать URL-адрес
http://mehelps.ru/yakor-v-html.html#yakor_kod
и вывести в консоль название домена.

7 Разобрать URL-адрес
http://mehelps.ru/yakor-v-html.html#yakor_kod
и вывести в консоль путь URL-адреса.

8 Разобрать URL-адрес
http://mehelps.ru/yakor-v-html.html#yakor_kod
и вывести в консоль якорь.

9 Разобрать URL-адрес
https://portal.rgups.ru/index.php?r=achievement/profile/index
вывести название протокола.

10 Разобрать URL-адрес
https://portal.rgups.ru/index.php?r=achievement/profile/index
вывести название домена.

11 Разобрать URL-адрес
https://portal.rgups.ru/index.php?r=achievement/profile/index
вывести путь URL-адреса.

12 Разобрать URL-адрес
https://portal.rgups.ru/index.php?r=achievement/profile/index
вывести строку запроса.

Задание 2

1 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /university/dokumenty/

2 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /oop/magistratura/

3 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /university/kontaktnaia-informatciia-rgups-353/

4 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /students/uchastie-rgups-v-mezhdunarodnom-k-1544/

5 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /services/iuridicheskaia-klinika/

6 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /services/time/

7 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /media/rgups-v-smi-1087/

8 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /services/progress/

9 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /services/elektronno-bibliotechnye-sistemy/

10 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /abitur/informirovanie/

11 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /science/granty-1763/

12 Установить соединение с доменом www.rgups.ru по протоколу http, сделать GET запрос где путь равен /university/kontaktnaia-informatciia-rgups-353/

ЛАБОРАТОРНАЯ РАБОТА № 10
ООП: КЛАССЫ

10.1 Методические рекомендации

Объектно ориентированное программирование (ООП) – это способ организации программ, позволяющей использовать один и тот же код многократно. В отличие от функций и модулей, ООП позволяет не только разделить программу на фрагменты, но и описать предметы реального мира в виде удобных сущностей – объектов, а также организовать связи между этими объектами.

Основой ООП является класс. *Класс* – это сложный тип данных, включающий набор переменных и функций для управления значениями, хранящимися в этих переменных. Переменные называют атрибутами, а функции – методами. Класс является производителем объектов, т. е. позволяет создать неограниченное количество экземпляров, основанных на этом классе.

Определение класса

Класс описывается с помощью ключевого слова class по следующей схеме:

class <Название класса>[(<Класс1>[, …, <КлассN>])]:
<Описание атрибутов и методов>

Инструкция создает новый класс и присваивает ссылку на его идентификатор, указанный после ключевого слова class. Это означает, что название класса должно полностью соответствовать правилам именования переменных. После названия класса в круглых скобках можно указать один или несколько базовых классов через запятую. Если же класс не наследует базовые классы, то круглые скобки можно не указывать. Следует отметить, что все выражения внутри инструкции *class* выполняются при создании класса, а не его экземпляра.

Создание атрибута класса аналогично созданию обычной переменной. Метод внутри класса создается так же, как и обычная функция, с помощью *def*. Методам класса первый параметр следует указать явно, автоматически передается ссылка на экземпляр класса. Общепринято этот параметр называть self, хотя это и не обязательно. Доступ к атрибутам и методам класса внутри определяемого метода производится через переменную self с помощью точечной нотации – к атрибуту x из метода класса можно обратиться так: *self.x*

Чтобы использовать атрибуты и методы класса, необходимо создать экземпляр класса согласно следующему синтаксису:

<Экземпляр класса> = <Название класса>([Параметры])

При обращении к методам класса используется такой формат:

<Экземпляр класса>.<Имя метода>([Параметры])

Пример 1

```
class MyClass:
    def printMessage(self):
        print('Вызван метод printMessage()')

    def printValue(self, value):
        print('Вы передали в метод value. Он равен ', value)

test = MyClass()
test.printMessage()
test.printValue(20)
test.printValue('привет мир')
```

Результат программы приведен на рис. 10.1.

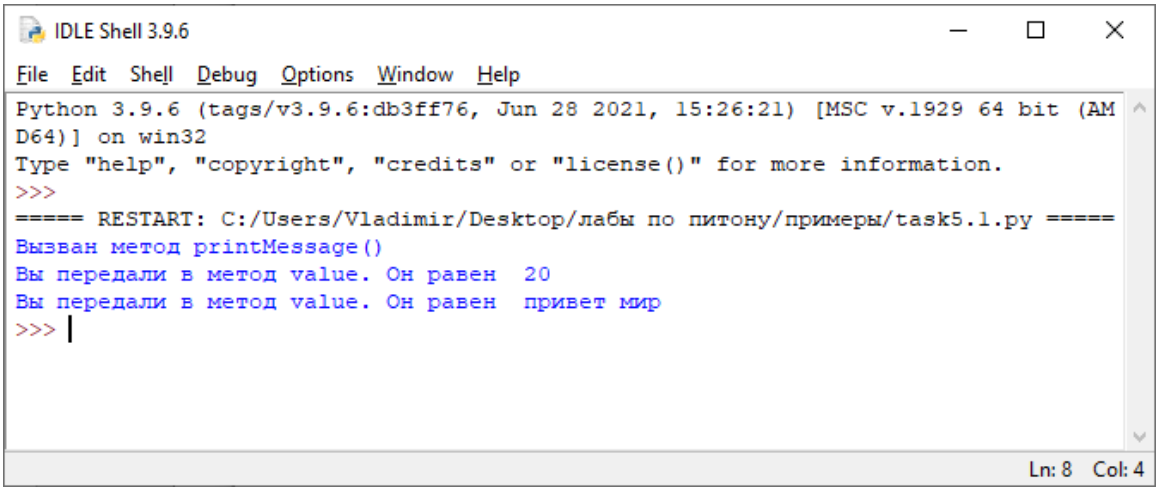

Рисунок 10.1

Все атрибуты класса в языке Python является открытыми (public) т. е. доступными для непосредственного изменения как из самого класса, так и из других классов и из основного кода программы.

Динамическое определение атрибутов

В Python допускается динамическое определение атрибутов после создания класса – можно создать как атрибут объекта класса, так и атрибут экземпляра класса.

Очень важно понимать разницу между атрибутами объекта класса и атрибутами экземпляра класса. Атрибут объекта класса доступен всем экземплярам класса, но после изменения атрибута значение изменится во всех экземплярах класса. Атрибут экземпляра класса может хранить уникальное значение для каждого экземпляра, и изменение его в одном экземпляре класса не затронет значения одноименного атрибута в других экземплярах того же класса.

Пример 2

```
class MyClass:
    x = 30
    y = 40
    def __init__(self):
        print('Сработал конструктор')

temp1 = MyClass()
temp2 = MyClass()
print('temp1.x = ', temp1.x)
print('temp2.x = ', temp2.x)
print('temp1.y = ', temp1.y)
print('temp2.y = ', temp2.y)
#Изменим атрибут у экземпляра класса temp1
temp1.x = 32
print('Изменили значение атрибута "x" у temp1')
print('temp1.x = ', temp1.x)
print('temp2.x = ', temp2.x)
#Изменим атрибут у экземпляра класса temp1
MyClass.y = 90
print('Изменили значение атрибута "y" у MyClass')
print('temp1.y = ', temp1.y)
print('temp2.y = ', temp2.y)
#Изменим атрибут для класса MyClass
MyClass.x = 64
print('Изменили значение атрибута "x" у MyClass')
print('temp1.x = ', temp1.x)
print('temp2.x = ', temp2.x)
```

Результат работы программы приведен на рис. 10.2.

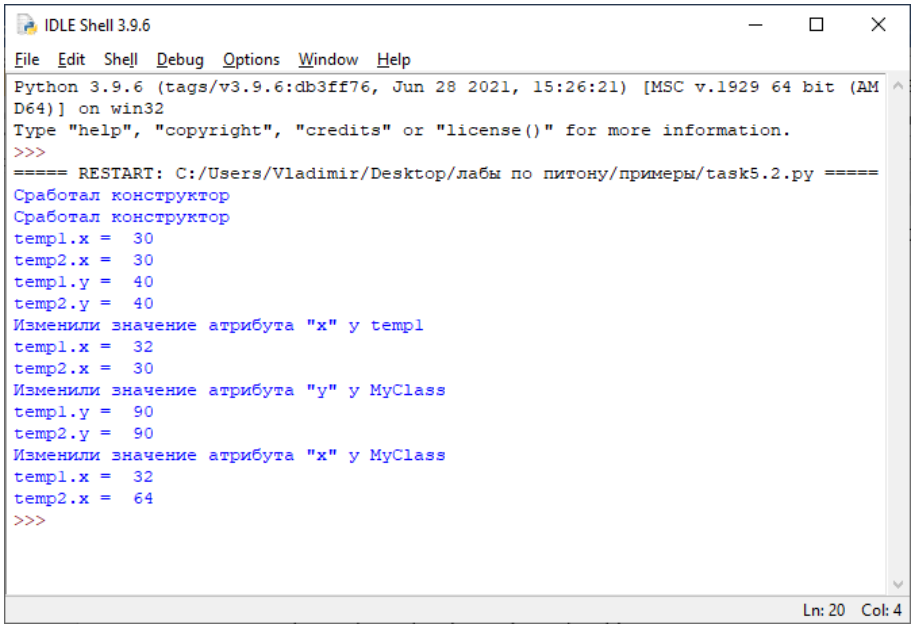

Рисунок 10.2

Методы __init__() и __del__()

При создании экземпляра класса интерпретатор автоматически вызывает метод инициализации __init__(). В других языках программирования такой метод принято называть конструктором класса. Формат метода:

def __init__(self[, <Значение1>[, …, <ЗначениеN>]])

С помощью метода __init__() можно присвоить начальные значения атрибутам класса. При создании экземпляра класса параметры этого метода указываются после имени класса в круглых скобках:

<Экземпляр класса> = <Имя класса>([<Значение1>[, …, <ЗначениеN>]])

Если конструктор вызывается при создании объекта, то перед уничтожением объекта автоматически вызывается метод, называемый деструктором. В языке Python деструктор реализуется в виде предопределенного метода __del__().

ВАЖНО: Если на экземпляр класса существует хотя бы одна ссылка, то метод не будет вызван.

Пример 3

Необходимо по вариантам создать классы. Определить конструктор деструктор и показать, как отрабатывает деструктор. Динамически определить атрибут для экземпляра класса и вывести его.

Написать метод, который будет выводить информацию (по вариантам), и вызвать его.

Система управления доставкой товара.

Order – заявка:

Свойства:

- Id – идентификатор;
- name – название товара;
- courier – курьер (ответственный за доставку);

} Конструктор

- dateTime – дата и время
- type – тип заказа (1 – срочный заказ; 2 – обычный заказ).

Код программы

```
class Order:
   dateTime = ''
   def __init__(self, Id, name, courier):
      self.Id = Id
      self.name = name
      self.courier = courier
      self.typeOrder = 'Доставка'
   def __del__(self):
      print('Сработал деструктор')
```

```
    def printData(self):
        print('Информация о заказе')
        print('id = ', self.Id)
        print('name = ', self.name)
        print('courier = ', self.courier)
        print('dateTime = ', self.dateTime)
        print('type = ', self.typeOrder)

order = Order(10, 'Организация', 'Курьер номер 1')
order.dateTime = '12.12.2010'
order.printData()
order.newValue = 'Новый атрибут'
print(order.newValue)
order = 290
print(order)
```

Результат программы приведен на рис. 10.3.

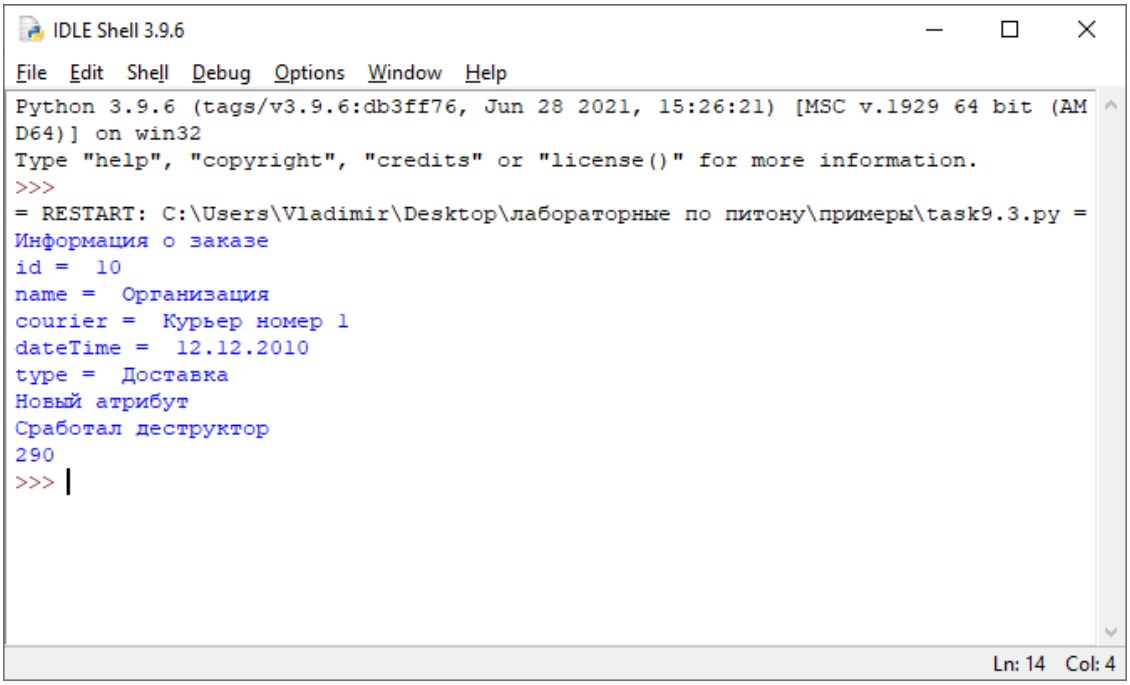

Рисунок 10.3

10.2 Задания для самостоятельной работы

Задание

Необходимо по вариантам создать классы. Определить конструктор деструктор и показать, как отрабатывает деструктор. Динамически определить атрибут для класса и экземпляра класса и вывести их.

Написать метод, который будет выводить информацию (по вариантам), и вызвать его.

***Вариант 1*. Реализация готовой продукции**
Commodity – Товар:
Свойства:

- id – идентификатор;
- productCode – код товара;
- name – наименование;
- wholesalePrice – оптовая цена;
- retailPrice – розничная цена;
- description – описание.

***Вариант 2*. Успеваемость студентов ВУЗА**
Students – студент:
Id_studenta – номер зачетной книжки;
Fam – фамилия;
Name – имя;
Groupa – группа;
Department – кафедра;
discipline – дисциплина;
mark – оценка;
NameTeacher – фамилия преподавателя.

***Вариант 3*. Деканат**
NameFaculty – факультет;
Room – аудитория;
corps – корпус;
Telephone – контактный телефон;
NameDean – фамилия декана.

***Вариант 4*. Супермаркет**
Supermarket:
Свойства:

- nameotdela – название отдела;
- productCode – код товара;
- name – наименование товара;
- cuntry – страна-производитель;
- retailPrice – розничная цена;
- namesource – поставщик.

***Вариант 5*. Военный состав**
Command:
Свойства:

- фамилия;
- рота;
- звание;
- дата рождения;

- дата поступления на службу;
- часть.

***Вариант 6*. Литература**
Literature:
Свойства:
- код источника литературы;
- тип литературы;
- название;
- год издательства;
- название издательства;
- количество страниц;
- автор.

***Вариант 7*. Продажа путевок**
Tourist:
Свойства:
- код путевки;
- фамилия клиента;
- название пансионата;
- номер;
- вид жилья;
- дата заезда;
- дата выезда;
- количество человек;
- цена.

***Вариант 8*. Станция техобслуживания**
ServiceCenter:
Свойства:
- название станции;
- адрес станции;
- название автотранспорта на ремонте;
- вид ремонта;
- дата поступления;
- дата выдачи;
- результат ремонта;
- фамилия персонала;
- сумма ремонта.

***Вариант 9*. Медицинское обслуживание пациентов**
Polyclinic:
Свойства:
- название поликлиники;

- адрес поликлиники;
- фамилия пациента;
- номер полиса;
- дата осмотра;
- фамилия врача;
- должность врача;
- диагноз.

***Вариант 10*. Выдача литературы**
Library:
Свойства:
- название библиотеки;
- название читательского зала;
- фамилия читателя;
- название литературы;
- дата выдачи;
- срок выдачи;
- сумма залога.

***Вариант 11*. Продажа автомобилей**
Car:
Свойства:
- марка автомобиля;
- год выпуска;
- цена автомобиля;
- комплектация;
- страна-производитель;
  дата продажи;
- ФИО покупателя.

***Вариант 12*. Интернет-магазин**
ElectronicShopping:
Свойства:
- название магазина;
- название товара;
- страна-производитель;
- вид оплаты;
- сумма покупки;
- дата продажи;
- ФИО покупателя.

***Вариант 13*. Больница**
Hospital:
Свойства:
- название больницы;

- Название отделения;
- ФИО пациента;
- номер полиса;
- дата поступления;
- дата выписки;
- диагноз;
- дата проведения операции;
- название операции;
- стоимость лечения.

***Вариант 14*. Военно-морские учения**
Marine:
Свойства:

- название военной части;
- название корабля;
- тип корабля;
- дата проведения учения;
- наработка корабля;
- количество личного состава;
- место проведения учений;
- результат учений.

***Вариант 15*. Туристические туры**
Tourist trip:
Свойства:

- название тура;
- страны;
- города;
- тип передвижения;
- тип питания;
- цена тура;
- тип проживания;
- дата выезда.

***Вариант 16*. Представление цирка**
Circus:
Свойства:

- название представления;
- город;
- дата премьеры;
- период проведения;
- цена билета;
- автор;
- жанр;
- количество актеров.

ЛАБОРАТОРНАЯ РАБОТА № 11
ООП: НАСЛЕДОВАНИЕ

11.1 Методические рекомендации

Наследование

Наследование – один из четырёх важнейших механизмов объектно ориентированного программирования (наряду с инкапсуляцией, полиморфизмом и абстракцией), позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.

Другими словами, класс-наследник реализует спецификацию уже существующего класса (базовый класс). Это позволяет обращаться с объектами класса-наследника точно так же, как с объектами базового класса.

Простое наследование

Класс, от которого произошло наследование, называется базовым или родительским (англ. base class). Классы, которые произошли от базового, называются потомками, наследниками или производными классами (англ. derived class).

Все в Python является объектом. Модули – это объекты, определения классов и функции — это объекты, и, конечно, объекты, созданные из классов, тоже являются объектами.

Наследование является обязательной функцией каждого объектно ориентированного языка программирования. Это означает, что Python поддерживает наследование. А также это один из немногих языков, который поддерживает множественное наследование.

Чтобы обратиться к родительским классам, можно воспользоваться инструкцией super().

Пример:

Необходимо создать класс «Транспорт» с полями (количество колес, тип двигателя, вес), и дочерние классы:

– «Мотоцикл» (максимальная скорость)
– «Автомобиль» (количество дверей)

Реализовать класс для хранения списка товаров с методом добавления нового товара и методом печати списка товаров.

Код программы:

```
class Vehicle:
    def __init__(self, countWheel=4, typeEngine='disel', weight=1):
        self.countWheel = countWheel
        self.typeEngine = typeEngine
        self.weight = weight
```

```
    def showInfo(self):
        print('Кол-во колес: ', self.countWheel)
        print('Тип двигателя: ', self.typeEngine)
        print('Вес транспорта: ', self.weight)

class Bike(Vehicle):
    def __init__(self, maxSpeed = 60, countWheel=4, typeEngine='disel',
weight=1):
        super().__init__(countWheel, typeEngine, weight)
        self.maxSpeed = maxSpeed

    def showInfo(self):
        print('Bike: ')
        super().showInfo()
        print('Максимальная скорость: ', self.maxSpeed)

class Car(Vehicle):
    def __init__(self, countDoor =4, countWheel=4, typeEngine='diesel',
weight=1):
        super().__init__(countWheel, typeEngine, weight)
        self.countDoor = countDoor

    def showInfo(self):
        print('Car: ')
        super().showInfo()
        print('Количество дверей: ', self.countDoor)

class Garage:
    def __init__(self):
        self.vehicles = []

    def addVehicle(self, vehicle):
        self.vehicles.append(vehicle)

    def showGarage(self):
        for i in self.vehicles:
            i.showInfo()

bike = Bike(countWheel=2)
car = Car(typeEngine='petrol')
garage = Garage()
garage.addVehicle(bike)
garage.addVehicle(car)
garage.showGarage()
```

Результат программы:

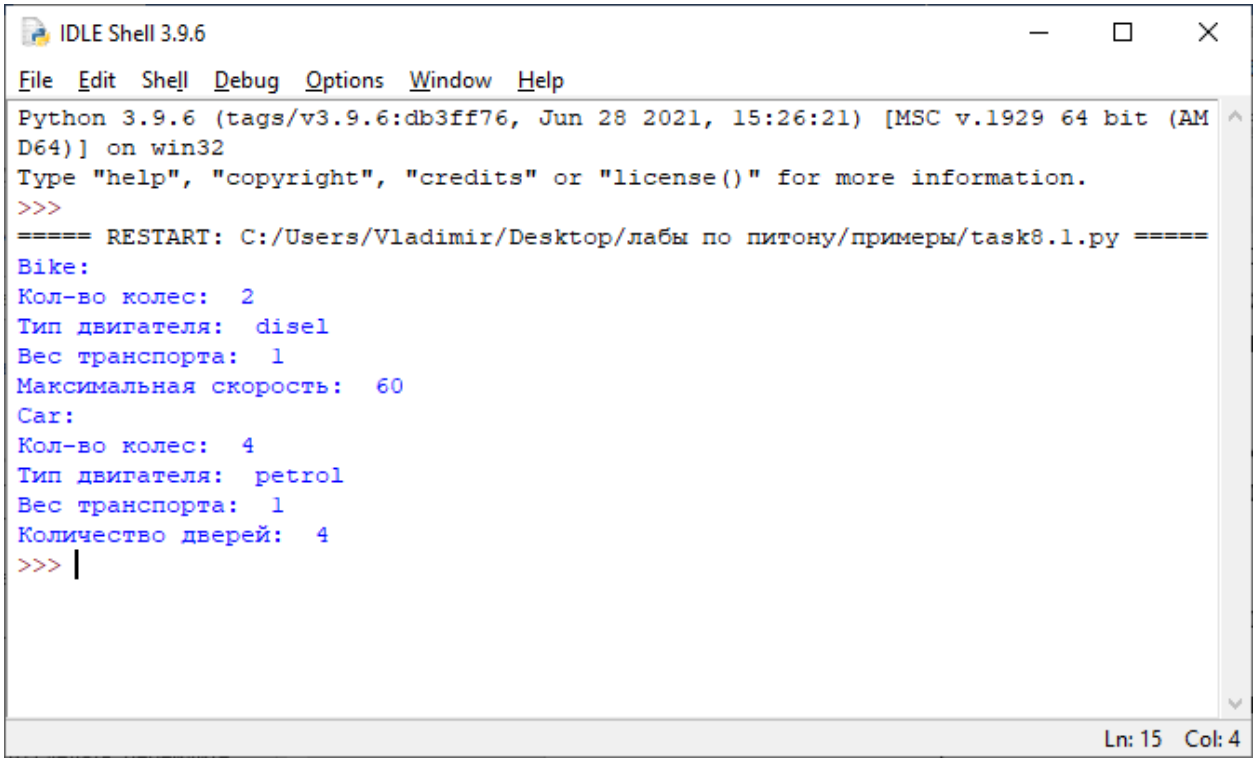

Рисунок 11.1

11.2 Задания для самостоятельной работы

Вариант 1. Учет успеваемости студентов вуза. Создать родительский класс «Студент» (номер зачетной книжки, фамилия, имя, группа, кафедра, дисциплина, оценка, фамилия преподавателя) и дочерние классы:

- «Очное отделение» (бал ЭГЕ, средний бал аттестата);
- «Заочное отделение» (место работы, должность, сумма обучения);
- «Целевое обучение» (название целевого предприятия, сумма обучения).

Реализовать класс для хранения списка студентов с методом добавления нового студента и методом печати списка студентов.

Вариант 2. Автоматизация работы факультета. Создать родительский класс «Факультет» (аудитория, корпус, контактный телефон, фамилия декана) и дочерние классы:

- «Состав факультета» (ФИО, должность);
- «Кафедры» (название, направление подготовки);
- «Преподаватели кафедры» (ФИО преподавателя, кафедра, должность, стаж работы, читаемые дисциплины).

Реализовать класс для хранения списка факультетов с методом добавления нового факультета и его кафедр и преподавателей и методом печати списка факультетов с полной информацией о его составе.

Вариант 3. Продажа товаров супермаркета. Создать родительский класс «Супермаркет» (название отдела, название товара, страна-производитель, розничная цена, поставщик) и дочерние классы:

- «Игрушки» (возрастная группа, тип);
- «Фрукты» (мах время хранения, температура хранения);
- «Габаритный товар» (высота, ширина, длина).

Реализовать класс для хранения списка товаров с методом добавления нового товара и методом печати списка товаров.

Вариант 4. Учет военного состава. Создать родительский класс «Военный состав» (фамилия, рота, звание, дата рождения, дата поступления на службу, часть) и дочерние классы:

- «Органы военного управления» (название округа, должность, выслуга лет, сумма надбавки);
- «Военная служба по контракту» (период договора, дата договора, номер протокола, сумма зарплаты);
- «Награжденные» (название награды, премия, сумма надбавки).

Реализовать класс для хранения списка военных с методом добавления нового военного и методом печати списка военных.

Вариант 5. Учет литературы. Создать родительский класс «Литература» (код источника литературы, тип литературы, название, год издательства, название издательства, количество страниц, автор) и дочерние классы:

- «научно-техническая литература» (область науки, количество экземпляров);
- «периодика» (вид периодики, период издательства);
- «справочники» (направление, том, часть).

Реализовать класс для хранения списка литературы с методом добавления нового источника и методом печати списка литературы.

Вариант 6. Учет продажи путевок. Создать родительский класс «Путевки» (код путевки, фамилия клиента, название пансионата, номер, вид жилья, дата заезда, дата выезда, количество человек, цена) и дочерние классы:

- «Зарубежные путевки» (загранпаспорт, страховка);
- «Санатории» (медполис, диагноз, направление);
- «Детские оздоровительные» (возраст ребенка, свидетельство о рождении, пол).

Реализовать класс для хранения списка путевок с методом добавления путевки и методом печати списка путевок.

Вариант 7. Учет выполненных работ станции техобслуживания. Создать родительский класс «ТехОбслуживание» (название станции, адрес станции, название автотранспорта на ремонте, вид ремонта, дата поступле-

ния, дата выдачи, результат ремонта, фамилия персонала, сумма ремонта) и дочерние классы:

- «планово-предупредительный осмотр для легкового транспорта» (вид (плановый/капитальный), год проведения, пробег, период);
- «неисправности» (название неисправности, описание выполненных работ);
- «планово-предупредительный осмотр для грузового транспорта» (вид (ТО-1, ТО-2, ТО-3), год проведения, пробег, период, объем двигателя).

Реализовать класс для хранения списка выполненных работ с методом добавления ремонта и методом печати списка ремонтов.

Вариант 8. **Медицинское обслуживание пациентов.** Создать родительский класс «МедОбслуживание» (название поликлиники, адрес поликлиники, фамилия пациента, номер полиса, дата осмотра, фамилия врача, должность врача, диагноз) и дочерние классы:

- «планово-предупредительный осмотр» (вид (амбулаторный/стационарный), год проведения, период действия, результат);
- «вакцинация» (название вакцины, дата вакцинации, период действия);
- «медобсуживание детей и подростков» (свидетельство о рождении, пол, возраст ребенка).

Реализовать класс для хранения списка медицинского обслуживания пациентов с методом добавления и методом печати списка.

Вариант 9. **Библиотека.** Создать родительский класс «Библиотека» (название библиотеки, адрес, город, ФИО директора) и дочерние классы:

- «читательский зал» (название зала, количество источников литературы, этаж, кабинет);
- «читатели» (фамилия, имя, отчество, место работы, возраст, пол).
- «выдача литературы» (название читательского зала, фамилия читателя, название литературы, дата выдачи, срок выдачи, сумма залога)

Реализовать класс для хранения списка литературы с методом добавления и методом печати списка.

Вариант 10. **Продажа автомобилей**. Создать родительский класс «Автомобили» (марка автомобиля, год выпуска, цена автомобиля, комплектация, страна-производитель, дата продажи, ФИО покупателя) и дочерние классы:

- «Поддержанные авто» (степень сохранности, ФИО владельца, пробег);

– «Спортивные» (кол-во секунд до набора скорости, объем двигателя, мощность);
– «Спецтехника» (вид (строительная, грузовая, дорожная и т. д.), масса, габаритные размеры).

Реализовать класс для хранения списка проданных автомобилей с методом добавления нового автомобиля и методом печати списка автомобилей.

Вариант 11. **Учет продаж через интернет-магазин**. Создать родительский класс «Интернет-магазин» (название магазина, название товара, страна-производитель, вид оплаты, сумма покупки, дата продажи, ФИО покупателя) и дочерние классы:

– «Мебель для гостиных» (название, цена, тип мебели, производитель);
– «Мебель для кухни» (название, цена, длина, высота, ширина, материал);
– «Мебель для ванн» (название, цена).

Реализовать класс для хранения списка товаров с методом добавления нового товара и методом печати списка товаров.

Вариант 12. Создать родительский класс «Больница» (название больницы, заведующий, город, адрес) и дочерние классы:

– «отделения» (название отделения, корпус, этаж, ФИО заведующего);
– «пациенты» (название отделения, ФИО пациента, номер полиса, дата поступления, дата выписки, диагноз, дата проведения операции, название операции, стоимость лечения);
– «врачи» (название отделения, ФИО врача, должность, научное звание, стаж работы).

Реализовать класс для хранения списка больниц, с отделениями, пациентами и врачами, с методом добавления и методом печати списка.

Вариант 13. Создать родительский класс «Военно-морские учения» (название военной части, дата проведения учения, количество личного состава, место проведения учений, результат учений) и дочерние классы:

– «корабли» (название корабля, тип корабля, наработка корабля, название военной части);
– «личный состав» (ФИО военного, звание, стаж, должность, название военной части);
– «место учений» (название, область, район).

Реализовать класс для хранения списка военно-морских учений, с методом добавления и методом печати списка.

***Вариант 14*. Учет оформления туров**. Создать родительский класс «Туры» (название тура, тип питания, цена тура, дата выезда) и дочерние классы:

- «Зарубежные туры» (страны, города, тип передвижения, тип проживания, длительность);
- «Походные туры» (название экскурсии, длительность, тип проживания);
- «Физкультурно-оздоровительные» (спортивное мероприятие, город, тип участия, тип проживания).

Реализовать класс для хранения списка туров с методом добавления тура и методом печати списка туров.

***Вариант 15*. Учет продажи лекарств в аптеке**. Создать родительский класс «Аптека» (название аптеки, адрес, город, ФИО директора) и дочерние классы:

– «клиенты аптеки» (название аптеки, фамилия, имя, отчество клиента, процент скидки);

– «лекарственный фонд аптеки» (название аптеки, название лекарства, тип лекарства, цена лекарства, страна-производитель);

– «продажи» (название аптеки, название лекарства, цена лекарства, ФИО клиента, количество, сумма к оплате).

Реализовать класс для хранения списка лекарств с методом добавления нового лекарства и методом печати списка лекарств.

***Вариант 16*. Реализация готовой продукции.** Создать родительский класс «Товар» (идентификатор, код, наименование, цена, описание) и дочерние классы:

- «Хрупкий товар» (коэффициент хрупкости);
- «Скоропортящийся товар» (мах время хранения);
- «Габаритный товар» (высота, ширина, длина).

Реализовать класс для хранения списка товаров с методом добавления нового товара и методом печати списка товаров.

ЛАБОРАТОРНАЯ РАБОТА № 12
ООП: МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ

12.1 Методические рекомендации

Множественное наследование позволяет одному дочернему классу иметь несколько родителей. Предположим, что мы хотим написать программу для отслеживания работы учителей. Учитель — это Human. Тем не менее он также является Сотрудником (Employee).

В определении класса в круглых скобках можно указать сразу несколько базовых классов через запятую.

Пример 1

Код программы

```
class ClassOne:
    def func1(self):
        print('Метод func1() класса ClassOne')

class ClassTwo(ClassOne):
    def func2(self):
        print('Метод func2() класса ClassTwo')

class ClassThree(ClassOne):
    def func1(self):
        print('Метод func1() класса ClassThree')
    def func2(self):
        print('Метод func2() класса ClassThree')
    def func3(self):
        print('Метод func3() класса ClassThree')
    def func4(self):
        print('Метод func4() класса ClassThree')

class ClassFour(ClassTwo, ClassThree):
    def func4(self):
        print('Метод func4() класса ClassFour')

c = ClassFour()
c.func1()
c.func2()
c.func3()
c.func4()
```

Результат программы приведен на рис. 12.1.

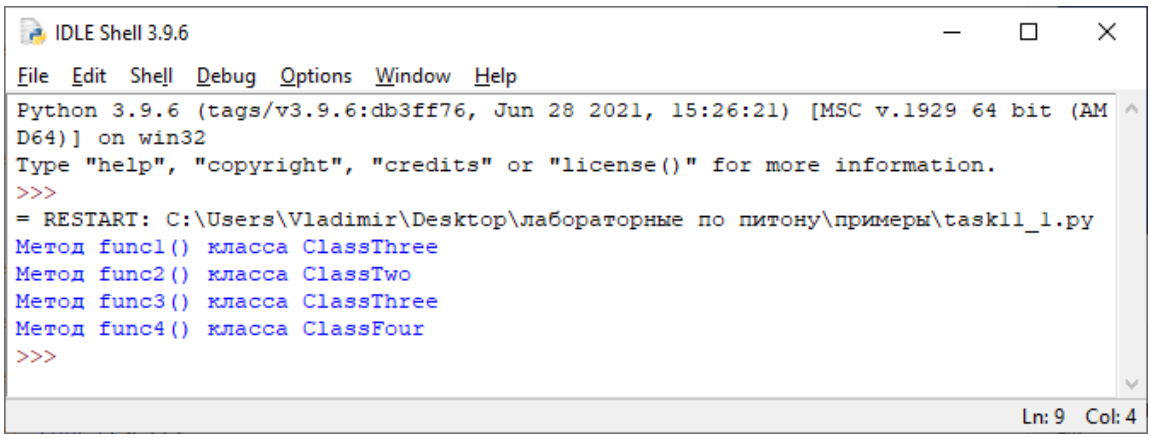

Рисунок 12.1

Метод func1() определен в двух классах: ClassOne и ClassTwo. Так как вначале просматриваются все базовые классы, непосредственно указанные в определении текущего класса, метод func1() будет найден в классе ClassThree (Поскольку он указан в числе базовых классов в определении ClassFour), а не в классе ClassOne.

Метод func2() также определен в двух классах: ClassTwo и ClassThree. Так как класс ClassTwo стоит первым в списке базовых классов, то метод будет найден именно в нем. Чтобы наследовать метод из класса ClassThree, следует указать это явным образом. Метод func3() определен только в классе ClassThree, поэтому метод наследуется от этого класса. Метод func4(), определенный в классе ClassThree, переопределяется в произвольном классе. Если же искомый метод в произвольном классе, то вся иерархия наследования просматриваться не будет. Переделаем определение класса ClassFour из предыдущего примера и наследуем метод func2() из класса ClassThree.

Пример 2

Код программы

```
class ClassOne:
    def func1(self):
        print('Метод func1() класса ClassOne')

class ClassTwo(ClassOne):
    def func2(self):
        print('Метод func2() класса ClassTwo')

class ClassThree(ClassOne):
    def func1(self):
        print('Метод func1() класса ClassThree')
    def func2(self):
        print('Метод func2() класса ClassThree')
    def func3(self):
        print('Метод func3() класса ClassThree')
```

```
    def func4(self):
        print('Метод func4() класса ClassThree')

class ClassFour(ClassTwo, ClassThree): #множественное наследование
    # Наследуем func2() из класса Classthree, а не из класса ClassTwo
    func2 = ClassThree.func2
    def func4(self):
        print('Метод func4() класса ClassFour')

c = ClassFour()
c.func1()
c.func2()
c.func3()
c.func4()
```

Результат работы программы:

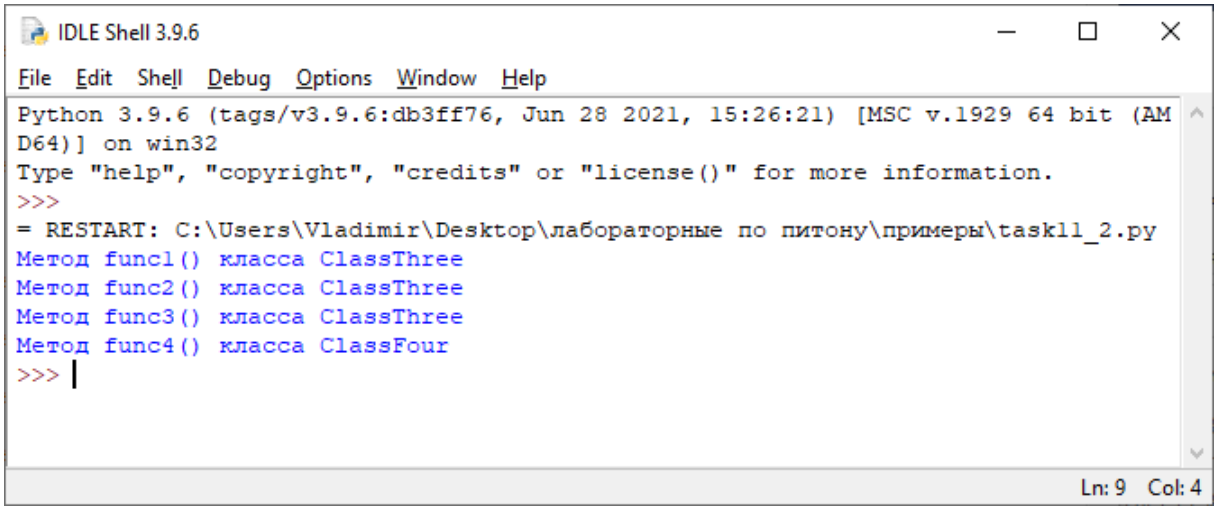

Рисунок 12.2

Специальные методы

Специальные методы в Python – это методы, которые отвечают за «стандартные» возможности объектов и вызываются автоматически при использовании этих возможностей.

Таблица 12.1 – Специальные методы в Python

Метод	Описание
1	**2**
__call__(self[, <Параметр1>[, <ПараметрN>]])	Позволяет обработать вызов экземпляра класса как вызов функции.
__getattr__(self, <Атрибут>)	Вызывается при обращении к несуществующему атрибуту класса.

Окончание табл. 12.1

1	2
__getattribute__(self, <Атрибут>)	Вызывается при обращении к любому атрибуту класса. Необходимо учитывать, что использование точечной нотации (для обращения к атрибуту класса) внутри этого метода приведет к зацикливанию. Чтобы избежать зацикливания, следует вызвать метод __getattribute__() объекта object. Внутри метода нужно вернуть значение атрибута или возбудить исключение AttributeError.
__setattr__(self , <Атрибут>, <Значение>)	Вызывается при попытке присваивания значения атрибуту экземпляра класса. Если внутри метода необходимо присвоить значение атрибуту, то следует использовать словарь __dict__, иначе при точечной нотации метод __setattr__() будет вызван повторно, что приведет к зацикливанию.
__delattr__(self, <Атрибут>)	Вызывается при удалении атрибута с помощью инструкции del <Экземпляр класса>.<Атрибут>
__len__(self)	Вызывается при использовании функции len(), а также для проверки объекта на логическое значение при отсутствии метода __bool__(). Метод должен возвращать положительное число.
__bool__(self)	Вызывается при использовании функции bool()
__int__(self)	Вызывается при преобразовании объекта в целое число с помощью функции int()
__float__(self)	Вызывается при преобразовании объекта в вещественное число с помощью функции float()
__complex__(self)	Вызывается при преобразовании объекта в комплексное число с помощью функции complex()
__round__(self)	Вызывается при использовании функции round()
__index__(self)	Вызывается при использовании функции bin(), hex(), oct()
__repr__(self) и __str__(self)	Служат для преобразования объекта в строку. Метод __repr__() вызывается при выводе в интерактивной оболочке, а так же при использовании функции repr(). Метод __str__() вызывается при выводе с помощью функции print(), а также при использовании функции str().
__hash__()	Этот метод следует переопределить, если экземпляр класса планируется использовать в качестве ключа словаря или внутри множества.

12.2 Задания для самостоятельной работы

Вариант 1

Необходимо создать класс A с методом methodOne, который возвращает название метода и класса. Создать класс B с методом methodTwo, который возвращает название метода и класса. Создать класс D который наследует класс A и B. Переопределить в классе D метод methodOne и добавить methodThree который выводит ФИО студента, выполняющего лабораторную. Написать в класс D метод __str__(). Проверить работоспособность всех методов класса D.

Вариант 2

Необходимо создать класс A с методом methodOne, который возвращает название метода и класса. Создать класс B с методом methodTwo, который возвращает название метода и класса. Создать класс D который наследует класс A и B. Переопределить в классе D метод methodTwo и добавить methodThree который выводит ФИО студента, выполняющего лабораторную. Написать в класс D метод __int__(). Проверить работоспособность всех методов класса D.

Вариант 3

Необходимо создать класс A с методом methodOne, который возвращает название метода и класса. Создать класс B с методом methodOne, который возвращает название метода и класса. Создать класс D который наследует класс A и B. Переопределить в классе D метод methodOne и добавить methodThree который выводит ФИО студента, выполняющего лабораторную. Написать в класс D метод __float__(). Проверить работоспособность всех методов класса D.

Вариант 4

Необходимо создать класс A с методом methodOne, который возвращает название метода и класса. Создать класс B с методом methodTwo, который возвращает название метода и класса. Создать класс D который наследует класс A и B. Переопределить в классе D метод methodTwo и добавить methodThree который выводит ФИО студента, выполняющего лабораторную. Написать в класс D метод __bool__(). Проверить работоспособность всех методов класса D.

Вариант 5

Необходимо создать класс A с методом methodOne, который возвращает название метода и класса. Создать класс B с методом methodTwo, который возвращает название метода и класса. Создать класс D который наследует класс A и B. Переопределить в классе D метод methodOne и добавить methodThree который выводит ФИО студента, выполняющего лабораторную. Написать в класс D метод __len__(). Проверить работоспособность всех методов класса D.

Вариант 6

Необходимо создать класс A с методом methodOne, который возвращает название метода и класса. Создать класс B с методом methodOne, который возвращает название метода и класса. Создать класс D который наследует класс A и B. Переопределить в классе D метод methodOne и добавить methodThree который выводит ФИО студента, выполняющего лабораторную. Написать в класс D метод __complex__(). Проверить работоспособность всех методов класса D.

Вариант 7

Необходимо создать класс A с методом methodOne, который возвращает название метода и класса. Создать класс B с методом methodOne, который возвращает название метода и класса. Создать класс D который наследует класс A и B. Переопределить в классе D метод methodOne и добавить methodThree который выводит ФИО студента, выполняющего лабораторную. Написать в класс D метод __round__(). Проверить работоспособность всех методов класса D.

Вариант 8

Необходимо создать класс A с методом methodOne, который возвращает название метода и класса. Создать класс B с методом methodTwo, который возвращает название метода и класса. Создать класс D который наследует класс A и B. Переопределить в классе D метод methodOne и добавить methodThree который выводит ФИО студента, выполняющего лабораторную. Написать в класс D метод __getattr__(). Проверить работоспособность всех методов класса D.

Вариант 9

Необходимо создать класс A с методом methodOne, который возвращает название метода и класса. Создать класс B с методом methodOne, который возвращает название метода и класса. Создать класс D который наследует класс A и B. Переопределить в классе D метод methodTwo и добавить methodThree который выводит ФИО студента, выполняющего лабораторную. Написать в класс D метод __delattr__(). Проверить работоспособность всех методов класса D.

Вариант 10

Необходимо создать класс A с методом methodOne, который возвращает название метода и класса. Создать класс B с методом methodOne, который возвращает название метода и класса. Создать класс D который наследует класс A и B. Переопределить в классе D метод methodTwo и добавить methodThree который выводит ФИО студента, выполняющего лабораторную. Написать в класс D метод __repr__(). Проверить работоспособность всех методов класса D.

ЛАБОРАТОРНАЯ РАБОТА № 13
ООП: ПЕРЕГРУЗКА ОПЕРАТОРОВ И СТАТИЧЕСКИЕ МЕТОДЫ

13.1 Методические рекомендации

Перегрузка операторов позволяет экземплярам классов участвовать в обычных операциях. Чтобы перегрузить оператор, необходимо в классе определить метод со специальным названием. Для перегрузки математических операторов используются методы, приведенные в табл. 13.1 и 13.2.

Таблица 13.1 – Методы перегрузки математических операторов

Операция	**Метод**
Сложение obj +y	__add__(self, <параметр>)
Сложение (экземпляр класса справа) y + obj	__radd__(self, <параметр>)
Сложение и присваивание obj +=y	__iadd__(self, <параметр>)
Вычитание obj - y	__sub__(self, <параметр>)
Вычитание (экземпляр класса справа) y - obj	__rsub__(self, <параметр>)
Вычитание и присваивание obj –=y	__isub__(self, <параметр>)
Умножение obj *y	__mul__(self, <параметр>)
Умножение (экземпляр класса справа) y * obj	__rmul__(self, <параметр>)
Умножение и присваивание obj *=y	__imul__(self, <параметр>)
Деление obj /y	__truediv__(self, <параметр>)
Деление (экземпляр класса справа) y/obj	__rtruediv__(self, <параметр>)
Деление и присваивание obj /=y	__itruediv__(self, <параметр>)
Остаток от деления obj %y	__mod__(self, <параметр>)
Остаток от деления (Экземпляр класса справа) y%obj	__rmod__(self, <параметр>)
Остаток от деления и присваивание obj %y	__rmod__(self, <параметр>)
Двоичное И obj & y	__and__(self, <параметр>)
Двоичное И (экземпляр класса справа) True & obj	__rand__(self, <параметр>)
Двоичное И и присваивание obj&=y	__iand__(self, <параметр>)
Двоичное ИЛИ obj | y	__or__(self, <параметр>)
Двоичное ИЛИ (экземпляр класса справа) y|obj	__ror__(self, <параметр>)
Двоичное ИЛИ и присвоение obj|=y	__ior__(self, <параметр>)
Двоичная инверсия ~obj	__invert__(self)

Таблица 13.2 – Перегрузка операторов сравнения

Операция	**Метод**
obj==y	__eq__(self, <параметр>)
obj != y	__ne__(self, <параметр>)
obj < y	__lt__(self, <параметр>)
obj > y	__gt__(self, <параметр>)
obj >= y	__le__(self, <параметр>)
obj <= y	__ge__(self, <параметр>)
y in obj	__contains__(self, <параметр>)

Пример 1

Код программы

```
class Test:
    """
    Метод инициализируект переменные
    """

    def __init__(self):
        self.a = 20
        self.b = 30

    """
    Метод перегрузки оператора сложения
    """

    def __add__(self, other):
        self.a += other
        return self.a

    """
    метод перегрузки оператора сравнения ==
    """

    def __eq__(self, other):
        return self.b == other


obj = Test()
print(obj + 10)
print(obj == 17)
print(obj == 30)
```

Результат работы программы приведен на рис. 13.1.

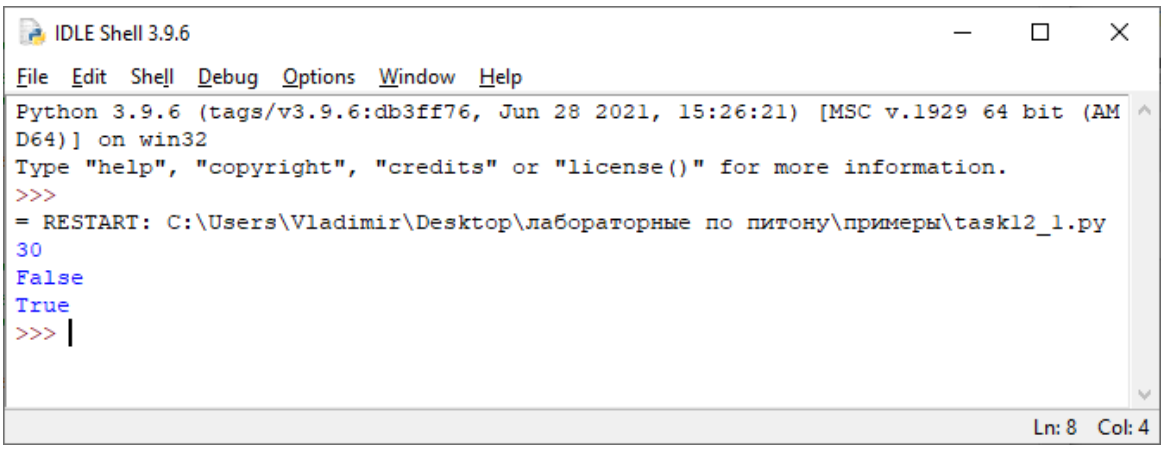

Рисунок 13.1

Статические методы и методы класса

Внутри класса можно создать метод, который будет доступен без создания экземпляра класса (статический метод). Для этого перед определением метода внутри класса следует указать декоратор @staticmethod. Вызов статического метода без создания экземпляра класса осуществляется следующим образом:

<Название класса> .<Название метода>(<Параметры>)

Кроме того, можно вызвать статический метод через экземпляр класса:

<Экземпляр класса>.<Название метода>(<Параметры>)

Пример 2

Код программы

```
class MyClass:
    """
    Статический метод который
    возвращает сумму двух чисел
    """

    @staticmethod
    def func1(x, y):
        return x + y

    """
    Метод который возвращает
    сумму двух чисел
    """

    def func2(self, x, y):
        return x + y

print(MyClass.func1(20, 30))
c = MyClass()
print(c.func2(30, 40))
```

Результат программы:

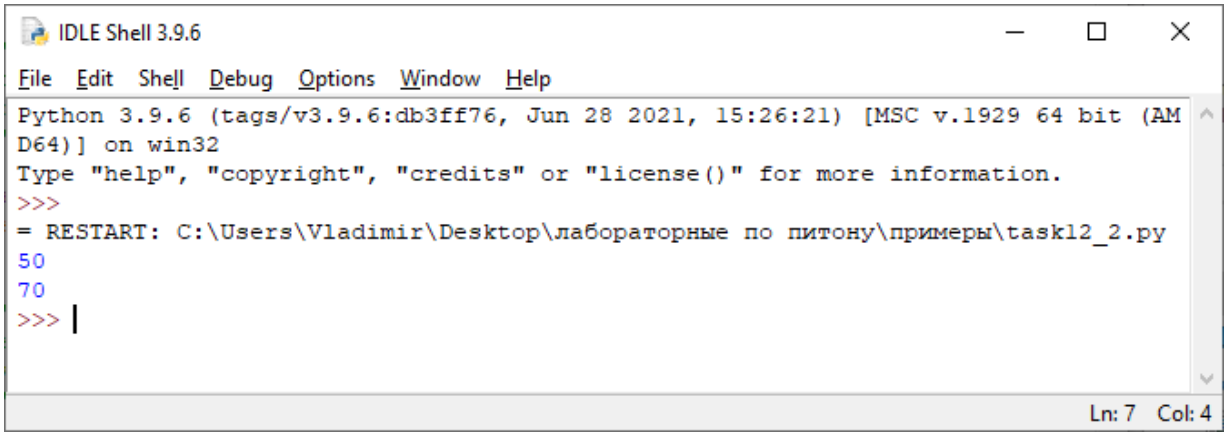

Рисунок 13.2

Стоит обратить внимание на то, что в определении статического метода нет параметра self. Это означает, что внутри статического метода нет доступа к атрибутам и методам экземпляра класса.

13.2 Задания для самостоятельной работы

Задание 1

Вариант 1

Необходимо создать класс и переопределить в нем метод __radd__(). Продемонстрируйте работу данного метода.

Вариант 2

Необходимо создать класс и переопределить в нем метод __sub__(). Продемонстрируйте работу данного метода.

Вариант 3

Необходимо создать класс и переопределить в нем метод __mul__(). Продемонстрируйте работу данного метода.

Вариант 4

Необходимо создать класс и переопределить в нем метод __rmul__(). Продемонстрируйте работу данного метода.

Вариант 5

Необходимо создать класс и переопределить в нем метод __itruediv__(). Продемонстрируйте работу данного метода.

Вариант 6

Необходимо создать класс и переопределить в нем метод __and__(). Продемонстрируйте работу данного метода.

Вариант 7

Необходимо создать класс и переопределить в нем метод __invert__(). Продемонстрируйте работу данного метода.

Вариант 8

Необходимо создать класс и переопределить в нем метод __ge__(). Продемонстрируйте работу данного метода.

Вариант 9

Необходимо создать класс и переопределить в нем метод __ne__(). Продемонстрируйте работу данного метода.

Вариант 10

Необходимо создать класс и переопределить в нем метод __lt__(). Продемонстрируйте работу данного метода.

Вариант 11

Необходимо создать класс и переопределить в нем метод __isub__(). Продемонстрируйте работу данного метода.

Вариант 12

Необходимо создать класс и переопределить в нем метод __imul__(). Продемонстрируйте работу данного метода.

Задание 2

Вариант 1

Создать вспомогательный класс, который содержит 2 статических метода, вычисляющих:

1) сумму 2 чисел;

2) разность 2 чисел.

Продемонстрировать работу данных методов

Вариант 2

Создать вспомогательный класс, который содержит 2 статических метода, вычисляющих:

1) деление 2 чисел;

2) умножение 2 чисел.

Продемонстрировать работу данных методов

Вариант 3

Создать вспомогательный класс, который содержит 2 статических метода, вычисляющих:

1) возведение числа в указанную степень;

2) корень числа.

Продемонстрировать работу данных методов

Вариант 4

Создать вспомогательный класс, который содержит 2 статических метода, вычисляющих:

1) сумму элементов списка;

2) умножение элементов списка.

Продемонстрировать работу данных методов

Вариант 5

Создать вспомогательный класс, который содержит 2 статических метода:

1 Параметры метода это 2 строки. Вернуть ту строку, где больше символов.

2 В параметрах передаются список и строка. Проверить, является ли последний и первый элемент списка этой строкой.

Продемонстрировать работу данных методов

Вариант 6

Создать вспомогательный класс, который содержит 2 статических метода вычисляющих:

1 В параметрах передаются список и число. Необходимо вернуть количество элементов списка меньших переданного числа.

2 В параметрах передаются список и число. Необходимо вернуть количество элементов списка больших переданного числа.

Продемонстрировать работу данных методов

Вариант 7

Создать вспомогательный класс, который содержит 2 статических метода, вычисляющих:

1) число повторений слова в тексте;

2) тангенс угла.

Продемонстрировать работу данных методов

Вариант 8

Создать вспомогательный класс, который содержит 2 статических метода, вычисляющих:

1) синус угла;

2) косинус угла.

Вариант 9

Создать вспомогательный класс, который содержит 2 статических метода:

1 В параметрах передаются строка и число. Вернуть строку если количество символов больше или равно числу, если число больше чем количество символов в строке, то вернуть число.

2 В параметрах передаются два списка. Необходимо вернуть объединенные списки.

Продемонстрировать работу данных методов.

Вариант 10

Создать вспомогательный класс, который содержит 2 статических метода, вычисляющих:

1) площадь прямоугольника;

2) объем прямоугольника.

Продемонстрировать работу данных методов.

ЛАБОРАТОРНАЯ РАБОТА № 14
РАБОТА PYQT

14.1 Методические рекомендации

Что такое PyQt5

PyQt – это библиотека, которая позволяет использовать фреймворк Qt GUI (GUI – это графический интерфейс пользователя) в Python. Сам Qt, как известно, написан на C++. Используя его в Python, вы можете создавать приложения намного быстрее, не жертвуя при этом значительной частью производительности C++.

Установка PyQt

Необходимо запустить командную строку и вписать команду *“pip install PyQt5”*. После успешной установки должна появиться надпись *Successfully installed PyQt5-5.15.4 PyQt5-Qt5-5.15.2 PyQt5-sip-12.9.0*

Пример 1

```
from PyQt5 import QtWidgets
import sys

app = QtWidgets.QApplication(sys.argv)
window = QtWidgets.QWidget()
window.setWindowTitle("Первая графическая программа!")
window.resize(300, 70)
label = QtWidgets.QLabel("<center>Привет мир!!!</center>")
btnQuit = QtWidgets.QPushButton("&Закрыть окно")
vbox = QtWidgets.QVBoxLayout()
vbox.addWidget(label)
vbox.addWidget(btnQuit)
window.setLayout(vbox)
btnQuit.clicked.connect(app.quit)
window.show()
sys.exit(app.exec_())
```

Результат работы:

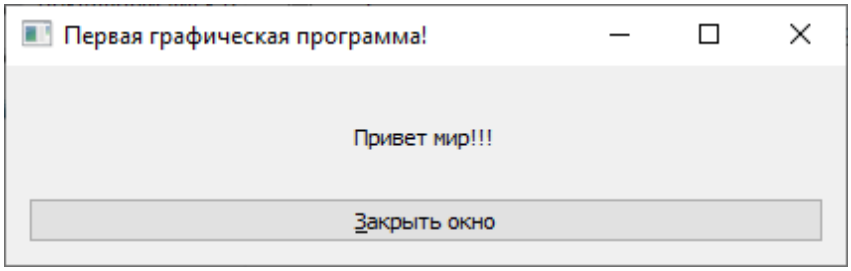

Рисунок 14.1

Теперь рассмотрим поподробнее код. В первой строчке подключается модуль QtWidgets – он содержит классы, реализующие компоненты графического интерфейса: окна, надписи, кнопки, текстовые поля и др. Во второй строчке производится подключение модуля sys, из которого нам потребуется список параметров, переданных в командной строке (argv), а так же функция exit(), позволяющая завершить выполнение программы.

Код *app = Qtwidgets.QApplication(sys.argv)* создает объект приложения в виде экземпляра класса QApplication. Конструктор этого класса принимает список параметров, переданных в командной строке. Следует помнить, что в программе всегда должен быть объект приложения, причем обязательно только один и инициализировать нужно сразу.

Выражение *windows = QtWidgets.QWidget()* создает объект окна в виде экземпляра класса QWidget. Этот класс наследует практически все классы, реализующие компоненты графического интерфейса.

window.setWindowTitle("первая графическая программа!")

Задает текст, который будет выводиться в заголовке окна, для этого используется метод *setWindowTitle()*

window.resize(300, 700) - задает минимальные размеры окна. В первом параметре метода resize() указывается ширина окна, а во втором параметре – его высота. При этом надо учитывать, что метод resize() указывает размеры не самого окна, а его клиентской области – размеры являются рекомендацией, – то есть, если компоненты не помещаются в окне, оно будет увеличено.

label = QtWidgets.QLabel(" Привет мир!!! ") – создает объект надписи. Текст надписи задается в качестве параметра в конструкторе класса QLabel. Стоит обратить внимание, что внутри строки указывается HTML-теги, а именно: с помощью тега получилось выравнивание текста по центру компонента. Возможность использования HTML-тегов и CSS-атрибутов является отличной чертой библиотеки PyQt – например, внутри надписи можно вывести таблицу или отобразить изображение.

btnQuit = QtWidgets.QPushButton("&Закрыть окно") – Создает объект кнопки. Текст, который будет отображаться на кнопке, задается в качестве параметра в конструкторе класса QPushButton. Символ & перед буквой З – таким образом задаются клавиши горячего доступа. Если нажать одновременно клавишу <Alt> и клавишу с буквой, перед которой в строчке указан символ &, то кнопка сработает.

vbox = QtWidgets.QVBoxLayout() – Создает вертикальный контейнер. Все компоненты, добавляемые в этот контейнер, будут располагаться по вертикали сверху вниз в порядке добавления. Внутри контейнера автоматически производится подгонка размеров добавляемых компонентов под размеры контейнера. При изменении размеров контейнера будет произведено изменение размеров всех компонентов.

vbox.addWidgets(label) и *vbox.addWidgets(btnQuit)* с помощью метода addWidgets() производится добавление созданных ранее объектов надписи

и кнопки в вертикальный контейнер. Так как объект надписи добавляется первым, он будет расположен над кнопкой. При добавлении компонентов в контейнер они автоматически становятся потомками контейнера.

window.setLayout(vbox) – добавляет контейнер в основное окно с помощью метода setLayout(). Таким образом, контейнер становится потомком основного окна.

btnQuit.clicked.connect(app.quit) – назначает обработчик сигнала clicked() кнопки, который генерируется при её нажатии. Этот сигнал доступен через одноименный атрибут класса кнопки и поддерживает метод connect(), который и назначает для него обработчик, переданный первым параметром метода. Обработчик представляет собой метод quit() объекта приложения, выполняющий немедленное завершение его работы. Такой метод принято называть *слотом*.

Сигналом в PyQt называется особое уведомление, генерируемое при наступлении какого-либо события в приложении: нажатия, ввода символа в текстовое поле, закрытия окна и т. д.

window.show() – выводит на экран окно и все компоненты, которые были ранее в него добавлены.

sys.exit(app.exec_()) – запускает бесконечный цикл обработки событий в приложении.

Код, расположенный после вызова метода exec_(), будет выполнен только после завершения работы приложения, – поскольку результат выполнения метода exec_() мы передаем функции exit(), дальнейшее выполнение программы будет прекращено, а код возврата передан операционной системе.

Пример 2

Код программы

```
import sys
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import (
    QCheckBox,
    QComboBox,
    QDateEdit,
    QDateTimeEdit,
    QDial,
    QDoubleSpinBox,
    QFontComboBox,
    QLabel,
    QLCDNumber,
    QLineEdit,
    QProgressBar,
    QPushButton,
```

```
    QRadioButton,
    QSlider,
    QSpinBox,
    QTimeEdit,
    QVBoxLayout,
    QWidget,
)

app = QtWidgets.QApplication(sys.argv)
layout = QVBoxLayout()
widgets = [
    QCheckBox,
    QComboBox,
    QDateEdit,
    QDateTimeEdit,
    QDial,
    QDoubleSpinBox,
    QFontComboBox,
    QLCDNumber,
    QLabel,
    QLineEdit,
    QProgressBar,
    QPushButton,
    QRadioButton,
    QSlider,
    QSpinBox,
    QTimeEdit,
]

for w in widgets:
    layout.addWidget(w())

widget = QWidget()
widget.setLayout(layout)
window = QtWidgets.QWidget()
window.setWindowTitle('Обзор всех виджетов')
window.setLayout(layout)
window.resize(300, 400)
window.show()
sys.exit(app.exec_())
```

Результат программы приведен на рис. 14.2.

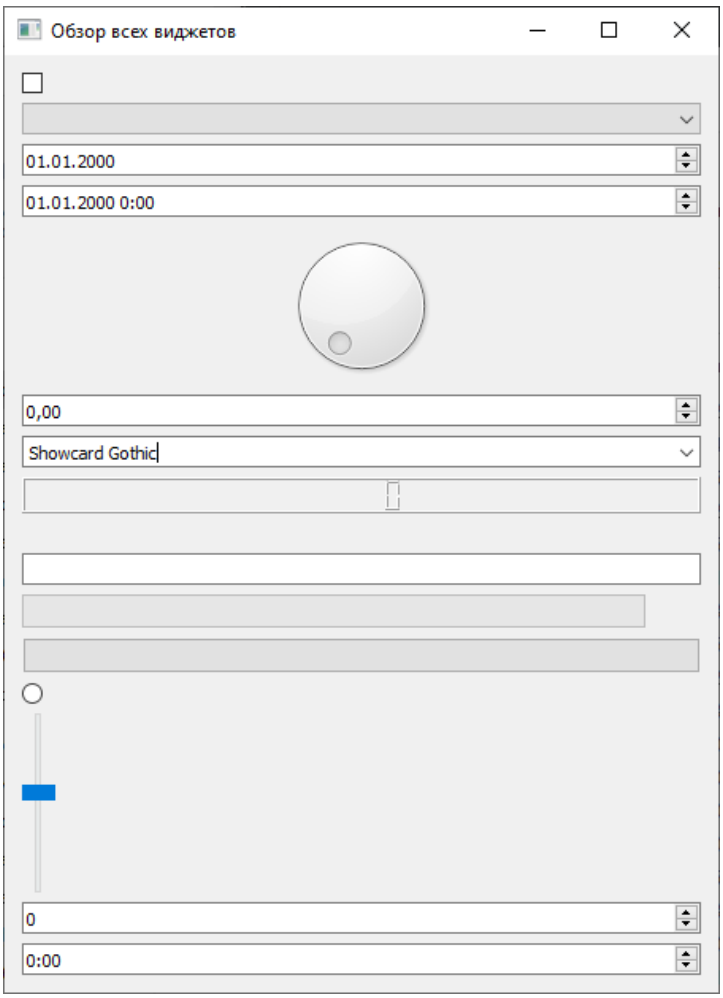

Рисунок 14.2

Разбор кода

```
import sys
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import (
    QCheckBox,
    QComboBox,
    QDateEdit,
    QDateTimeEdit,
    QDial,
    QDoubleSpinBox,
    QFontComboBox,
    QLabel,
    QLCDNumber,
    QLineEdit,
    QProgressBar,
    QPushButton,
    QRadioButton,
    QSlider,
    QSpinBox,
    QTimeEdit,
```

QVBoxLayout,
QWidget,
)

В этом фрагменте кода происходит импорт всех необходимых библиотек.

app = QtWidgets.QApplication(sys.argv) – инициализация объекта приложения, без него ничего не будет работать.

layout = QVBoxLayout() – создает вертикальный контейнер.

widgets = [
QCheckBox,
QComboBox,
QDateEdit,
QDateTimeEdit,
QDial,
QDoubleSpinBox,
QFontComboBox,
QLCDNumber,
QLabel,
QLineEdit,
QProgressBar,
QPushButton,
QRadioButton,
QSlider,
QSpinBox,
QTimeEdit,
]

Данным кодом создается список с названиями классов виджетов.

for w in widgets:
layout.addWidget(w())

В данном фрагменте перебирается список виджетов, где создается экземпляр класса и добавляется в контейнер.

widget.setLayout(layout) – добавляет контейнер в основное окно

window = QtWidgets.QWidget() - создает объект окна в виде экземпляра класса

window.setWindowTitle('Обзор всех виджетов') – устанавливает заголовок окна

window.setLayout(layout) – добавляет контейнер в основное окно

window.resize(300, 400) – задает минимальные размеры окна.

window.show() – выводит на экран окно и все его компоненты

sys.exit(app.exec_()) – запускает бесконечный цикл обработки событий в приложении

***Подробнее о виджетах можно почитать*:**

https://doc.qt.io/qt-5/gallery.html

https://www.pythonguis.com/tutorials/pyqt-basic-widgets

14.2 Задания для самостоятельной работы

Задания

Написать программу на PyQt, где будет (по вариантам) использоваться определенный виджет. В заголовке программы написать название виджета. Контейнер для объектов выбрать на свой выбор.

1 Виджет QCheckbox, нужно создать 3 или более выборов checkbox.

2 Виджет QComboBox, в списке должно быть 3 или более элементов.

3 Виджет QDateEdit, установить по умолчанию текущую дату.

4 Виджет QDateTimeEdit, установить по умолчанию текущую дату и время.

5 Виджет QLabel, по умолчанию в поле должно быть написано название виджета.

6 Виджет QProgressBar, задать свое значение, отличное от значения по умолчанию.

7 Виджет QPushButton, задать название кнопки и при нажатии на кнопку должно закрываться приложение.

8 Виджет QRadioButton

9 Виджет QLineEdit, установить в поле на фоне “Введите какое либо значение”.

10 Виджет QListWidget, добавить 3 элемента списка.

Перечень рекомендуемых учебных изданий, интернет-ресурсов, дополнительной литературы

Основная литература

1 Чернышев, С. А. Основы программирования на Python : учебное пособие для вузов / С. А. Чернышев. – Москва : Юрайт, 2021. – 286 с. – (Высшее образование). – ISBN 978-5-534-14350-8. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/477353 (дата обращения: 16.11.2021)

2 Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для вузов / Д. Ю. Федоров. – 3-е изд., перераб. и доп. – Москва : Юрайт, 2021. – 210 с. – (Высшее образование). – ISBN 978-5-534-14638-7. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/478098 (дата обращения: 16.11.2021).

3 Тузовский, А. Ф. Объектно ориентированное программирование : учебное пособие для вузов / А. Ф. Тузовский. – Москва : Юрайт, 2020. – 206 с. – (Высшее образование). – ISBN 978-5-534-00849-4. – Текст : электронный // ЭБС Юрайт [сайт]. – URL: https://urait.ru/bcode/451429 (дата обращения: 17.03.2021).

4 Гниденко, И. Г. Технологии и методы программирования : учебное пособие для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. – Москва : Юрайт, 2021. – 235 с. – (Высшее образование). – ISBN 978-5-534-02816-4. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/469759 (дата обращения: 16.11.2021).

5 Гниденко, И. Г. Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. – Москва : Юрайт, 2021. – 235 с. – (Профессиональное образование). – ISBN 978-5-534-05047-9. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/472502 (дата обращения: 16.11.2021).

6 Бизли, Д. М. Язык программирования Python : справочник : перевод с английского / Д. М. Бизли. – Киев : ДиаСофт, 2000.

7 Гифт, Н. Python в системном администрировании UNIX и Linux : перевод с английского / Н. Гифт, Д. Джонс. – Санкт-Петербург : СимволПлюс, 2009.

8 Лейнингем, И. Освой самостоятельно Python за 24 часа : перевод с английского / И. Лейнингем. – Москва : Изд. дом «Вильямс», 2001.

9 Лесса, А. Python. Руководство разработчика : перевод с английского / А. Лесса. – Санкт-Петербург : ДиасофтЮП, 2001.

10 Лутц, М. Изучаем Python : перевод с английского / М. Лутц. – Санкт-Петербург : Символ-Плюс, 2009.

11 Лутц, М. Программирование на Python : перевод с английского / М. Лутц. – Санкт-Петербург : Символ-Плюс, 2002.

12 Саммерфельд, М. Программирование на Python 3. Подробное руководство : перевод с английского / М. Саммерфельд. – Санкт-Петербург : Символ-Плюс, 2009.

13 Сузи, Р. А. Python / Р. А. Сузи. – Санкт-Петербург : БХВ-Петербург, 2002.

14 Сузи, Р. А. Язык Python и его применения : учебное пособие / Р.А. Сузи. – Москва : Интернет-Университет информационных технологий: БИНОМ. Лаборатория знаний, 2006.

15 Язык программирования Python / Г. Россум [и др.]. – Санкт-Петербург : АНО «Институт логики» – Невский диалект, 2001.

Дополнительные источники:

1 Хорошая шпаргалка по Python 3 на русском. – URL: https://pythonworld.ru/uploads/mementopython3-russian.pdf.

2 Подборка книг о python для начинающих (на русском языке). – URL: https://pythonworld.ru/bookshop/category/beginners.

3 Стандартная библиотека python. – URL: https://pythonworld.ru/moduli.

4 Примеры программ на языке python. – URL: https://pythonworld.ru/primery-programm.

5 PythonRu – Уроки по Python для начинающих. – URL: https://pythonru.com/uroki/vvedenie-uroki-po-python-dlja-nachinajushhih.

6 Учите Питон. Бесплатный курс по программированию с нуля. – URL: https://pythontutor.ru.

7 Основные возможности Python. – URL: https://pythonchik.ru/osnovy.

8 Лаборатория линуксоида – Python. Обучение языку программирования. – URL: https://younglinux.info/python.