МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»
(ТУСУР)

Кафедра технологий электронного обучения (ТЭО)

Д.С. Шульц
А.В. Гураков

ИНФОРМАТИКА

Методические указания по выполнению лабораторных работ для студентов, обучающихся по направлениям подготовки:

27.03.02	Управление качеством, профиль «Управление качеством в информационных системах»
27.03.05	Инноватика, профиль «Управление инновациями»

Томск
2024

УДК 004.432.2
ББК 32.973.2
Ш95

Рецензент:
Перминова М.Ю., доцент кафедры технологий электронного обучения ТУСУР, канд. техн. наук

Шульц Денис Сергеевич
Ш95 **Информатика**: Методические указания по выполнению лабораторных работ для студентов, по направлениям подготовки 27.03.02 Управление качеством, 27.03.05 Инноватика / Д.С. Шульц, А.В. Гураков. – Томск, 2024. – 56 с.

Настоящие методические указания содержат задания и требования по выполнению лабораторных работ по дисциплине «Информатика».

Одобрено на заседании каф. ТЭО протокол № 9 от 27.09.2024 г.

УДК 004.432.2
ББК 32.973.2

Оглавление

1 Введение

Дисциплина «Информатика» является одними из важнейших общих математических и естественнонаучных дисциплин. Современный уровень развития вычислительной техники требует от специалистов высокого уровня знаний и навыков работы с компьютером для решения вопросов получения, хранения, преобразования, передачи и использования информации.

Целью изучения дисциплины является получение представления о предмете информатики, информации и её видах, основных методах передачи, обработки, накопления и управления информацией; получение полного представления о современных языках, системах, алгоритмах программирования.

Основными задачами изучения дисциплины «Информатика» являются:

- формирование практических навыков поиска, обработки, хранения информации посредством современных информационных технологий для решения задач в различных областях профессиональной деятельности;
- формирование потребности обращаться к компьютеру при решении задач из любой предметной области, базирующуюся на осознанном владении информационными технологиями и навыками взаимодействия с компьютером;
- освоение основных приемов разработки алгоритмов с выбором наиболее подходящих алгоритмов в зависимости от постановки задач;
- освоение приемов разработки компьютерных программ на языке высокого уровня, использующихся для решения задач профессиональной деятельности.

Данное руководство содержит методические указания по выполнению лабораторных работ.

2 Методические указания по выполнению лабораторных работ

В рамках изучения курса «Информатика» студентам необходимо выполнить пять лабораторных работ:

1) Решение элементарных задач.
2) Работа со структурами данных
3) Работа с массивами данных. Построение графиков.
4) Построение функций и диаграмм с помощью модуля turtle
5) Создание приложений с помощью модуля Tkinter

Выполнение лабораторных работ включает в себя следующие этапы:

- постановку темы занятий и определение задач лабораторной работы;
- определение порядка выполнения лабораторной работы или отдельных её этапов;
- непосредственное выполнение лабораторной работы студентом и контроль за ходом занятий;
- подведение итогов лабораторной работы и формулирование основных выводов;
- оформление отчета;
- отправка отчета на проверку;
- защита лабораторной работы (для текстовых работ).

При подготовке к лабораторным занятиям необходимо заранее изучить методические рекомендации по его проведению. Обратить внимание на цель занятия, на основные вопросы для подготовки к занятию, на содержание темы занятия.

Если в процессе выполнения лабораторной работы или при изучении теоретического материала у студента возникают вопросы, разрешить которые самостоятельно не удается, необходимо обратиться к преподавателю для получения у него разъяснений или указаний. Сделать это можно лично на занятиях или в часы консультаций, а также на специальном форуме «Консультация», который находится в электронном курсе.

Задание к лабораторным работам состоит из нескольких задач. Студенту необходимо решить эти задачи, разработать алгоритм этого решения и записать на языке программирования Python.

По результатам работы необходимо оформить отчет по лабораторной работе. В отчете необходимо привести:

- текст задания;
- решение задачи и его алгоритм;
- программу на языке программирования Python;
- результаты тестирования программы (в виде скриншота).

В электронном курсе в соответствующем элементе «Задание» необходимо выложить на проверку преподавателю:

- отчет;
- файлы с кодом программы (*.py);
- файлы с результатами работы программы (при наличии).

2.1 Лабораторная работа №1. Решение элементарных задач

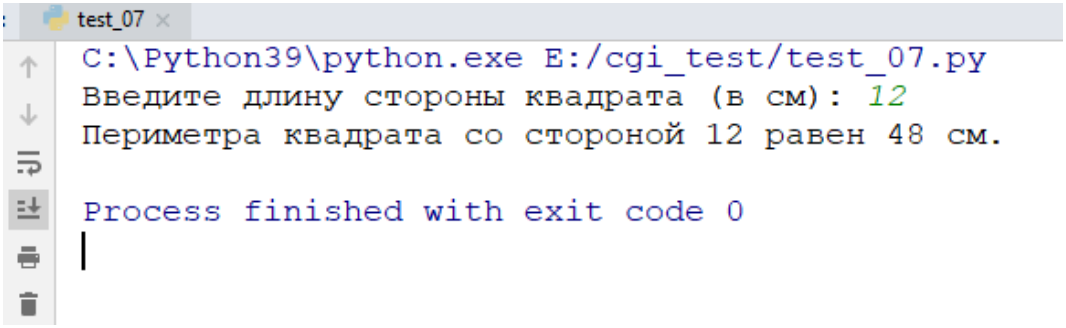
Задание к лабораторной работе

Необходимо разработать алгоритм решения задачи и написать программу на языке программирования Python.

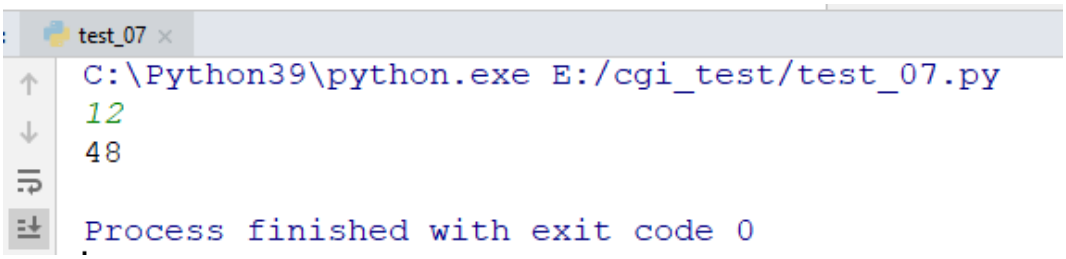
Всего необходимо решить четыре задачи.

Задача 1

Элементарная задача на ввод и вывод данных. Из операторов используется только присваивание. Здесь важно использовать комментарии при организации ввода данных. А также, выводить развернутый ответ. Например



Недопустимо вводить данные и выводить на экран результат следующим образом.



Задача 2

Прежде чем писать программу необходимо внимательно проанализировать выражение. При некоторых значениях переменных вычислить выражение невозможно. Например, выражение

$$\frac{1}{\ln(a)}$$

нельзя вычислить при $a=1$[1]. Поскольку ln(1)=0 возникнет ошибка деление на ноль.

Поэтому прежде чем вычислить значение переменной *y*, следует проверить входные данные с помощью оператора *if*.

Задача 3

Необходимо найти сумму ряда чисел. Для решения этого типа задач используется цикл. Поскольку количество этих чисел известно для решения этой задачи используется цикл *for*.

Задача 4

Задача аналогичная предыдущей. Но, ряд в данном случае бесконечный и количество слагаемых заранее неизвестно. Поэтому лучше всего использовать цикл *while*.

Значение переменной *eps* задается в формате *1E-n*, где *n* – порядковый номер цифры после запятой. Если необходимо обеспечить точность до второго знака после запятой (*n=2*), то *eps=0.01=1E-2*.

Будем считать, что достигнута необходимая точность, если разность сумм *n* слагаемых (S_n) и *n+1* слагаемых (S_{n+1}) меньше *eps*:

$$|S_{n+1} - S_n| <= eps.$$

После выполнения всех заданий необходимо написать отчет по лабораторной работе. В отчете обязательно должны быть: титульный лист, текст задачи, приведен анализ задачи и

[1] В этом выражении есть ещё одно ограничение для переменной *a*. Предлагается найти его самостоятельно.

её решение, алгоритм и/или блок-схема алгоритма, подробное описание всех переменных, код программы с комментариями.

ВАРИАНТ 1

1. Вводятся два числа a и b. Получить их сумму, разность и произведение.
2. С клавиатуры вводится значение переменных x1 и x2 типа float, и r1 и r2 типа int. Необходимо определить значение $y = \frac{\cos(x_1)+4\cdot r_1}{\ln(r_2)-x_2}$. Полученное значение y необходимо вывести на экран.
3. Составить программу вычисления суммы конечного ряда:

$$\sum_{n=1}^{15} \frac{1}{n!}.$$

4. Составить программу вычисления суммы бесконечного ряда с точность eps. Значение eps вводится с клавиатуры.

$$\sum_{n=1}^{\infty} \frac{(-1)^n}{2 \cdot n}.$$

ВАРИАНТ 2

1. Вводится длина ребра куба. Найти объем куба и площадь его боковой поверхности.
2. С клавиатуры вводится значение переменных x1 и x2 типа float, и r1 и r2 типа int. Необходимо определить значение $y = e^{x_1-r_2} \cdot \frac{|x_2-r_1|}{\sin(x_2)}$. Полученное значение y необходимо вывести на экран.
3. Составить программу вычисления суммы конечного ряда.

$$\sum_{n=1}^{13} \frac{1}{(2n)!}.$$

4. Составить программу вычисления суммы бесконечного ряда с точность eps. Значение eps вводится с клавиатуры.

$$\sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n^2}.$$

ВАРИАНТ 3

1. Вводятся два положительных числа. Найти среднее арифметическое и среднее геометрическое этих чисел.
2. С клавиатуры вводится значение переменных x1 и x2 типа float, и r1 и r2 типа int. Необходимо определить значение $y = \ln|arctg\ x_1| \cdot \left(\frac{x_2-r_2}{r_1}\right)^2$. Полученное значение y необходимо вывести на экран.
3. Составить программу вычисления суммы конечного ряда.

$$\sum_{n=1}^{11} \frac{1}{(2 \cdot n-1)!}.$$

4. Составить программу вычисления суммы бесконечного ряда с точность eps. Значение eps вводится с клавиатуры.

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{(n+1)!+2}.$$

ВАРИАНТ 4

1. Вводятся два числа. Найти среднее арифметическое этих чисел и среднее геометрическое их модулей.
2. С клавиатуры вводится значение переменных x1 и x2 типа float, и r1 и r2 типа int. Необходимо определить значение $y = e^{\cos x_1} \cdot \frac{(\ln r_1 - r_2)^2}{tg\ x_2}$. Полученное значение y необходимо вывести на экран.
3. Составить программу вычисления суммы конечного ряда

$$\sum_{n=1}^{14} \frac{1}{(2 \cdot n)! + 2}.$$

4. Составить программу вычисления суммы бесконечного ряда с точность eps. Значение eps вводится с клавиатуры.

$$\sum_{n=1}^{\infty} \frac{(-1)^n}{(n+1) \cdot (n+2)}.$$

ВАРИАНТ 5

1. Вводятся катеты прямоугольного треугольника. Найти его гипотенузу и площадь.
2. С клавиатуры вводится значение переменных x1 и x2 типа float, и r1 и r2 типа int. Необходимо определить значение$y = \sqrt{\frac{r_2 + r_1}{\ln r_2} + \frac{(x_1 - x_2)^2}{|x_2|}}$. Полученное значение y необходимо вывести на экран.
3. Составить программу вычисления суммы конечного ряда

$$\sum_{n=1}^{16} \frac{1}{n! + 4}.$$

4. Составить программу вычисления суммы бесконечного ряда с точность eps. Значение eps вводится с клавиатуры.

$$\sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{(n+2) \cdot n}.$$

ВАРИАНТ 6

1. Смешано v1 литров воды температуры t1 с v2 литрами воды температуры t2. Найти объем и температуру образовавшейся смеси. Значения v1, v2, t1, t2 вводятся с клавиатуры.
2. С клавиатуры вводится значение переменных x1 и x2 типа float, и r1 и r2 типа int. Необходимо определить значение $y = \sqrt{e^{x_1}} \cdot \frac{|x_2 - r_2|}{\mathrm{tg}\, r_2}$. Полученное значение y необходимо вывести на экран.
3. Составить программу вычисления суммы конечного ряда

$$\sum_{n=1}^{18} \frac{1}{2^n}.$$

4. Составить программу вычисления суммы бесконечного ряда с точность eps. Значение eps вводится с клавиатуры.

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}.$$

ВАРИАНТ 7

1. Определить периметр правильного n-угольника, описанного около окружности радиуса r. Значения n и r вводятся с клавиатуры.
2. С клавиатуры вводится значение переменных x1 и x2 типа float, и r1 и r2 типа int. Необходимо определить значение $y = \frac{\sin(x_1)\cdot x_2 + r_1 - 4}{\ln r_2 - (x_2)^2}$. Полученное значение y необходимо вывести на экран.
3. Составить программу вычисления суммы конечного ряда

$$\sum_{n=1}^{19} \frac{1}{n^n}.$$

4. Составить программу вычисления суммы бесконечного ряда с точность eps. Значение eps вводится с клавиатуры.

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n^3}.$$

ВАРИАНТ 8

1. Три сопротивления r1, r2, R3 соединены параллельно. Найти сопротивление соединения. Значения r1, r2, R3 вводятся с клавиатуры.
2. С клавиатуры вводится значение переменных x1 и x2 типа float, и r1 и r2 типа int. Необходимо определить значение $y = |ctg(x_1)| \cdot \frac{\sqrt{|x_2|} + r_2}{r_1}$. Полученное значение y необходимо вывести на экран.
3. Составить программу вычисления суммы конечного ряда

$$\sum_{n=1}^{21} \frac{1}{(n+2)!}.$$

4. Составить программу вычисления суммы бесконечного ряда с точность eps. Значение eps вводится с клавиатуры.

$$\sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n^4}.$$

ВАРИАНТ 9

1. Определить время падения камня на поверхность земли с высоты h. Значение h вводится с клавиатуры.
2. С клавиатуры вводится значение переменных x1 и x2 типа float, и r1 и r2 типа int. Необходимо определить значение $y = \sqrt{\frac{r_1 + r_2}{(\sin r_2)^2} \cdot \frac{(\cos(x_1 \cdot x_2))^2}{|x_2|}}$. Полученное значение y необходимо вывести на экран.
3. Составить программу вычисления суммы конечного ряда

$$\sum_{n=1}^{20} \frac{1}{4^n}.$$

4. Составить программу вычисления суммы бесконечного ряда с точность eps. Значение eps вводится с клавиатуры.

$$\sum_{n=1}^{\infty} \frac{(-1)^n}{2 \cdot n \cdot (n+1)}.$$

ВАРИАНТ 10

1. Вводится сторона равностороннего треугольника. Найти площадь этого треугольника.
2. С клавиатуры вводится значение переменных x1 и x2 типа float, и r1 и r2 типа int. Необходимо определить значение $y = \frac{ctg(|x_1 \cdot r_2|)+x_2-r_2}{\sqrt{r_2+(x_2)^2}}$. Полученное значение y необходимо вывести на экран.
3. Составить программу вычисления суммы конечного ряда
$$\sum_{n=1}^{18} \frac{1}{n! + n}.$$
4. Составить программу вычисления суммы бесконечного ряда с точность eps. Значение eps вводится с клавиатуры.
$$\sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{(n+2)^2}.$$

2.2 Лабораторная работа №2. Работа со структурами данных

Стандартные структуры данных

К стандартным типам данных в Python относятся:

- списки;
- строки;
- кортежи;
- словари;
- файлы;
- множества.

Строки

Строки – последовательности символов, используемые для хранения и представления текстовой информации

Чтобы было понятно, что речь идет о данных строкового типа, их ограничивают апострофами или кавычками. В Python они абсолютно равнозначны.

Помимо обычных в строках можно использовать служебные символы. Они начинаются с символа обратный слеш («палочка», наклоненная влево). Они имеют управляющие функции. Например, разбить одну строку на две и более. Некоторые служебные символы приведены в таблице.

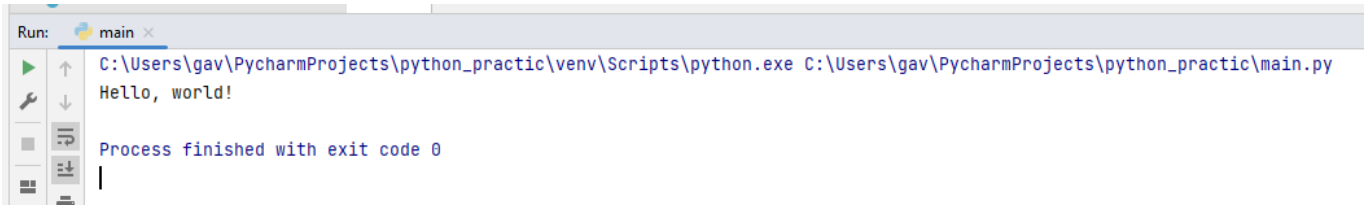
Таблица 3.1 Служебные символы

Символ	**Значение**
\n	Перевод строки
\t	Символ табуляции
\b	BackSpase
\r	Перевод курсора в начало строки
\N{id}	Символ Юникод по названию
\uhhhh	Символ Юникод (16-битовый код)
\Uhhhhhhhh	Символ Юникод (32-битовый код)

Для строк применимы операции конкатенации (объединения), дублирования и отношения (сравнения).

Операция объединения является бинарной, т.е. над двумя операндами. В качестве символа операции используется плюс (+).

```
words_one = 'Hello'
words_two = 'world!'
hi = words_one + ', ' + words_two
print(hi)
```

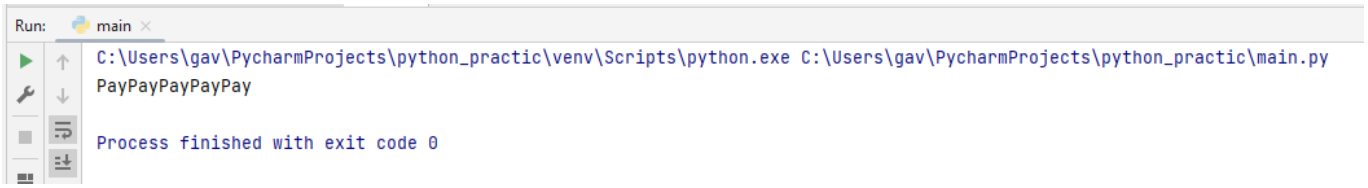


В результате объединения строк получается новая строка, в которой сначала записывается первая строка (слева от знака +), а затем вторая (без добавления каких-либо разделителей.

Объединять можно только строку со строкой. Попытка объединить строку и данные другого типа вызовет ошибку.

Дублирование позволяет создавать строки из одинаковой последовательности символов.

```
pay = 'Pay'
print(pay * 5)
```

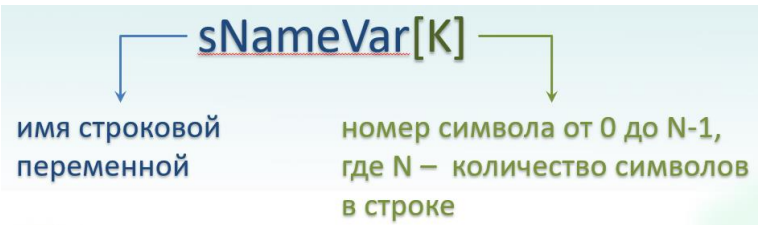


Операция бинарная. Один операнд должен быть строкой. Второй – обязательно целым числом. Между ними ставится звездочка (как у умножения).

Строки можно сравнивать друг с другом. Применимы все операции отношения.

Сравнение строк идет посимвольно, до первых различающихся символов. При этом сравниваются не сами символы, а их коды. Получается, что один символ будет меньше другого, если он идет в таблице раньше. Например, латинская буква «A» с кодом 65 меньше «a», чей код равен 97

Можно обращаться к каждому элементу строки по отдельности. Для этого необходимо написать имя переменной (или саму строку) и в квадратных скобках указать индекс (номер) символа в строке.

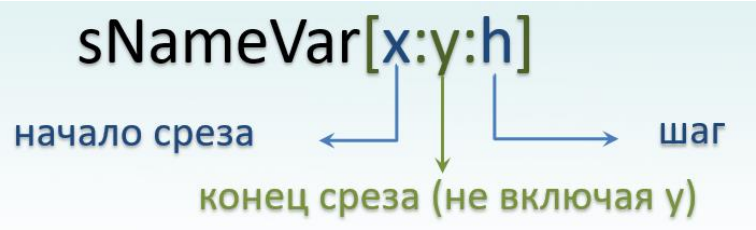


Если считать слева направо, т.е. крайний слева символ имеет нулевой индекс.

Индекс	S[0]	S[1]	S[2]	S[3]	S[4]	S[5]
Строка S	P	y	t	h	o	n
Индекс	S[-6]	S[-5]	S[-4]	S[-3]	S[-2]	S[-1]

Нумерация справа налево начинается с минус единицы.

Срез – это новая строка, состоящая из символов исходной начиная с определенного символ и заканчивая другим символом, стоящим справа. В общем виде это записывается следующим образом:



Несколько функций, которые будут полезными при работе со строками.

Таблица 3.2 Функции для работы со строками

Название	**Описание**
chr()	Преобразует целое число в символ, с соответствующим кодом
ord()	Возвращает код символа
len()	Возвращает длину строки
str()	Изменяет тип объекта на string

Каждая строка в Python является объектом и имеет собственные методы для работы с ней. Обращение к методам осуществляется следующим образом

```
строковый_объект.имя_метода(параметры)
```

Таблица 3.3 Методы строк

Метод	**Описание**
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*
S.replace(str1, str2 [, maxcount])	Замена *строка1* на *строка2*. *maxcount* ограничивает количество замен
S.split(symbol)	Разбиение строки по разделителю *символ*. По умолчанию *символ* равен пробелу.
S.count(str, [start],[end])	Возвращает количество непересекающихся вхождений подстроки в диапазоне [*start*, *end*] (0 и длина строки по умолчанию)
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.startswith(str)	Начинается ли строка S с шаблона str
S.endswith(str)	Заканчивается ли строка S шаблоном str
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру
S.capitalize()	Переводит первый символ строки в верхний регистр, а все остальные в нижний
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.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

Списки

Списки представляют собой последовательность объектов, перечисленных через запятую и ограниченную квадратными скобками.

Список можно создать простым перечислением объектов или с помощью функции list().

```
# Список из целых чисел
int_list = [19, 2, 32, 3, 23, 24, 45, 37, 6, 40]
# Список строк
str_list = ['Red', 'Green', 'Blue', 'Yellow', 'Black']
# Список списков
age_list = [[2, 4], [4, 1], [6, 2]]
# Список из объектов различного типа
dif_list = ['Jon', 32, [75, 176]]
# Пустой список
empty_lst = []
lst_empty = list()
# Список символов из строки
str_to_lst = list('Python')
```

Можно создавать списки, в которых объекты определяются по какому-либо правилу. Для этого используется специальная структура – генератор. В общем виде он записывается следующим образом:

`[выражение for переменная in последовательность if условие].`

Выражение – любое выражение. Результат вычисления которого станет элементом списка.

Цикл for – определяет количество элементов списка (если нет if) и как меняются переменные из выражения.

Оператор if – условие отбора значений для списка из последовательности. Необязательный параметр.

Аналогично строкам можно обращаться к каждому элементу списка по его индексу. Также можно получить часть строки используя срезы.

Индекс	S[0]	S[1]	S[2]	S[3]	S[4]	S[5]
Строка S	[‘P’,	‘y’,	‘t’,	‘h’,	‘o’,	‘n’]
Индекс	S[-6]	S[-5]	S[-4]	S[-3]	S[-2]	S[-1]

Но, есть и отличие. Списки относятся к изменяемым типа данных. Поэтому мы можем изменить любой объект списка на другой.

Кортежи

Кортеж – неизменяемый тип данных, который представляет собой последовательность объектов, перечисленных через запятую и ограниченную круглыми скобками.

```
empty_tuple = ()
point = (1.5, 6.0)
names = ('Timur', 'Ruslan', 'Roman')
info = ('Timur', 'Guev', 28, 170, 60, False)
nested_tuple = (('one', 'two'), ['three', 'four'])
gen = tuple(i**2 for i in range(1, 10))
```

Кортежи аналогичны спискам. Поэтому для них допустимы практически те же операции что и для списков. А именно:

- доступ к элементу по индексу (только для получения значений элементов);
- методы, в частности index(), count();
- встроенные функции, в частности len(), sum(), min() и max();
- срезы;
- оператор принадлежности in;
- операторы конкатенации (+) и повторения (*);
- операции сравнения.

Словари

Словарь - изменяемая коллекция элементов с произвольными индексами – ключами.

{key1: value1, key2: value2, …}.

Здесь key (ключ) – объект любого неизменяемого типа, value (значение) – объект любого типа.

Словарь записывается в фигурных скобках. Через запятую перечисляются пары ключ и значения, разделенные двоеточием.

```
# Ключ и значения целые числа
dict_01 = {1:1, 2:4, 3:9, 4:16, 5:25, 6:36}
# Ключ – строка.
dict_02 = {'A':23, 'B': 21, 'C': 43}
# Ключ строка. Значение – список
dict_03 = {'go' : ['идти', 'ехать', 'ходить'],
           'car': ['машина', 'автомобиль'],
           'tree': ['дерево']
           }
# Пустой словарь
dict_04 = {}
# или
dict_05 = dict()
```

К словарям применимы только две операции сравнения: равно и не равно. А также *in*.

Словари сравниваются по ключам, а затем по их значениям. Порядок следования ключей при этом значения не имеет.

Функции, которые применяются к словарям, в качестве аргументов используют ключи.

Таблица 3.4 Функции для работы со словарями

Название	**Описание**
min()	Минимальное значение ключа
max()	Максимальное значение ключа
len()	Возвращает количество элементов (длина) словаря
dict()	Создание словаря
sum()	Вычисляет сумму *числовых* ключей

Но имеется возможность анализировать и значения.

Таблица 3.5 Методы для работы со словарями

Название	**Описание**
keys()	Возвращает список ключей
values()	возвращает список значений всех элементов словаря
items()	возвращает список всех элементов словаря, состоящий из кортежей пар (ключ, значение).
get()	Возвращает значение по его ключу или значение по умолчанию, если такого нет.
update()	Объединение двух словарей. При совпадении ключей берется значение из аргумента
pop()	Возвращает значение словаря по его ключу и удаляет его из словаря.
popitem()	Удаляет из словаря последний добавленный элемент и возвращает удаляемый элемент в виде кортежа (ключ, значение)
clear()	Удаляет все элементы из словаря
copy()	Создает поверхностную копию словаря
setdefault()	Позволяет получить значение из словаря по заданному ключу, автоматически добавляя элемент словаря, если он отсутствует.

Файлы

Python способен работать с различными типа файлов. Их условно можно разделить на две группы: текстовый и двоичные. К первой группе относятся любые файлы, которые хранят информацию в текстовом виде (txt, html, py и т.п.). Вторая – это изображения, видео и аудио файлы.

Работа с любым файлом разбивается на три этапа:

1. открыть файл;
2. считать или записать данные;
3. закрыть файл.

Открытие файла.

Чтобы открыть файл используется специальная функция

```
open(file_name, mode, encoding)
```

file_name – полный путь к файлу, *mode* – режим открытия файл (для чтения, или для записи), *encoding* – используемая в текстовом файле кодировка.

Таблица 3.6 Режимы работы с файлами

Режим	Обозначение
'r'	открытие на чтение (является значением по умолчанию).
'w'	открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.
'a'	открытие на дозапись, информация добавляется в конец файла.
'b'	открытие в двоичном режиме.

Закрытие файла.

Открытый файл после работы с данными в нем следует закрыть. Для этого используется метод

```
file.close()
```

Запись данных файл

Для записи данных используются методы

Таблица 3.7 Методы для записи данных в файл

Метод	Описание
write	Записывает в файл строку
writelines	Помещает в файл список строк

Пример.

Создать текстовый файл *num.txt* и записать в него числа от одного до десяти.

```
# Открываем файл для записи
file = open('num.txt', 'w', encoding='utf-8')
# Создаем список из чисел от 1 до 10 и преобразуем каждое в строку
lst = [str(i) for i in range(1, 11)]
# Объединяем весь список в одну строку. Разделитель - пробел
txt = ' '.join(lst)
# Записываем значение переменной txt в файл
file.write(txt)
# Закрываем файл
file.close()
```

Чтение файла.

Для чтения данных из файла используются следующие методы:

Таблица 3.8 Методы для чтения данных из файла

Метод	Описание
read	Читает все содержимое и возвращает его в виде строки
readline	Считывает одну строку из файла
readlines	Читает все содержимое файла и возвращает список из строк.

Пример

В текстовом файле *num.txt* записаны числа от одного до десяти. Создать список и поместить в него данные из файла.

```
# Открываем файл для чтения
file = open('num.txt', 'r', encoding='utf-8')
# Считываем данные из файла
txt = file.read()
# Создаем список из целых чисел
lst = [int(i) for i in txt.split()]
# Закрываем файл
file.close()
```

Множества

Множество – это совокупность объектов, понимаемых как единое целое.

Основной особенность множества является то, что все его элементы различны (уникальны), т.е. два элемента не могут иметь одинаковое значение. Кроме этого:

- множества не упорядочены, то есть элементы не хранятся в каком-то определенном порядке;
- элементы множества должны относиться к неизменяемым типам данных;
- хранящиеся в множестве элементы могут иметь разные типы данных.

Все объекты, которые включаются в множество, перечисляются через запятую и ограничиваются фигурными скобками.

```
# множество цветов
colors = {'Red', 'Green', 'Blue', 'Black'}

# множество четных чисел
even = {2, 4, 6, 8, 10, 12, 14}

# Множество разных значений
info = {'Alex', 1974, 100}
```

Поскольку фигурные скобки используются также и в словарях, создать пустое множество можно только с помощью функции set().

```
nullset = set()
```

Так как элементы множества не упорядочены, то к ним нельзя обращаться по индексу. Соответственно недоступны и срезы.

Среди всех операций к множествам применимы операции сравнения == и !=. А также операции объединения (|), пересечения (&), разности (-) и симметрической разности (^).

Список основных функций приведен в таблице 3.9.

Таблица 3.9 Функции для работы со множествами

Название	**Описание**
min()	Минимальное значение множества
max()	Максимальное значение множества
len()	Возвращает длину множества
sum()	Вычисляет сумму *числовых* элементов множества
in	Оператор, позволяет проверить, содержит ли множество заданное значение

В таблице 3.10 приведены методы множества.

Таблица 3.10 Методы множества

Название	**Описание**
add()	Добавление нового элемента
remove()	Удаляет элемент из множества с генерацией исключения (ошибки) в случае, если такого элемента нет
discard()	Удаляет элемент из множества без генерации исключения (ошибки), если элемент отсутствует.
pop()	Удаляет и возвращает случайный элемент из множества с генерацией исключения (ошибки) при попытке удаления из пустого множества.
clear()	Удаляет все элементы из множества
union() *оператор* |	Объединение множеств
intersection() *оператор* &	Пересечение множеств
difference() *оператор* -	Разность множеств
symmetric_difference() *оператор* ^	Симметрическая разность множеств
clear()	Удаляет все элементы из множества

Задание к лабораторной работе

Имеется текстовый файл, в котором в определенном порядке представлена некоторая информация. Необходимо написать программу, которая будет считывать данные из файла, обрабатывать их и представлять в более удобной форме. Исходный файл в соответствии с вариантом расположен в электронном курсе или его можно получить у преподавателя.

На проверку необходимо предоставить отчет по лабораторной работе и файл с программой. В отчете привести подробное описание решения задачи, код программы и результаты её работы. Ответ на дополнительные вопросы оформлять средствами текстового редактора!

Вариант №1

С некоторого сайта студент скопировал таблицу со списком крупнейших городов мира и вставил её в текстовый файл (towns.txt). Однако, что-то пошло не так и все таблица вставилась в виде одной строки. Необходимо помочь студенту и преобразовать строку в таблицу. А, так как ситуация может повториться, процесс следует автоматизировать, т.е. написать программу.

В результате работы программы должен получиться файл, в котором информация о каждом городе располагается на отдельной строке. В качестве разделителя между столбцами использовать символ табуляции.

Выведите на экран информацию по городам из одной страны. Определите сколько городов России входит в этот список. Сравните его с официальными данными и ответьте на вопрос: какого города нет в списке.

Подсказка:

Список городов, стран, континентов из двух и более слов, разделенных пробелом:

```
lst_double = ['Демократическая Республика Конго', 'Доминиканская
республика',
              'Китайская Республика', 'Новая Зеландия',
              'Объединённые Арабские Эмираты', 'Республика Конго',
              'Республика Корея','Саудовская Аравия','Чешская республика',
              'Северная Америка' , 'Южная Америка',
              'Корейская Народно-Демократическая Республика',
              'Алма-Ата (Алматы)', 'Вадодара (Барода)', 'Варанаси
(Бенарес)',
              'Виджаявада (Безвада)', 'Вишакхапатнам (Визаг)',
              'Вэйхай (Вэйхайвэй)', 'Гвалиор (Гвалияр)', 'Гирин
(Цзилинь)',
              'Даммам (Эд-Даммам)', 'Калькутта (Колката)',
              'Коимбатур (Коямпуттур)', 'Мератх (Мирут)', 'Нави Мумбаи',
              'Нашик (Насик)', 'Нижний Новгород', 'Патна (Паталипутра)',
              'Сюйчан (Жаочжуань)', 'Сямынь (Амой)',
              'Тайчжоу (Вэньлин, пров. Чжэцзян)', 'Тайчжоу (пров.
Цзянсу)',
              'Тхане (Тхана)', 'Урумчи (Дихуа)', 'Ховрах (Хаура)',
              'Хойчжоу (Хуэйчжоу)', 'Цзыбо (Чжандянь)', 'Цзюлун (Коулун)',
              'Чжанцзякоу (Калган)', 'Чифэн (Улан-Хад)', 'Шаньтоу
(Сватоу)',
              'Шэньян (Мукден)', 'Янгон (Рангун)', 'Яньтай (Чифу)']
```

Входные данные:

Файл *towns.txt*

Пример содержимого файла:

№ Город Численность (чел.) Страна Континент 1. Чунцин 32 054 159 Китай Азия 2. Шанхай 26 875 500 Китай Азия 3. Пекин 21 893 095 Китай Азия 4. Чэнду 20 937 757 Китай Азия 5. Гуанчжоу 18 676 605 Китай Азия 6. Шэньчжэнь 17 633 800 Китай Азия

Выходные данные:

1. Файл *towns_out.txt*

Пример содержимого выходного файла:

```
№   Город  Численность (чел.) Страна Континент
1. Чунцин 32 054 159 Китай  Азия
2. Шанхай 26 875 500 Китай  Азия
3. Пекин  21 893 095 Китай  Азия
4. Чэнду  20 937 757 Китай  Азия
5. Гуанчжоу   18 676 605 Китай  Азия
6. Шэньчжэнь  17 633 800 Китай  Азия
```

2. Таблица на экране по городам из одной страны в том же формате, что и таблица в выходном файле.

Вариант №2

С некоторого сайта студент скопировал таблицу со списком крупнейших метрополитенов мира и вставил её в текстовый файл (metro.txt). Однако, что-то пошло не так и все таблица вставилась в виде одной строки. Необходимо помочь студенту и преобразовать строку в таблицу. А, так как ситуация может повториться, процесс следует автоматизировать, т.е. написать программу.

В результате работы программы должен получиться файл, в котором информация о каждом метрополитене располагается на отдельной строке. В качестве разделителя между столбцами использовать символ табуляции.

Добавить в программу возможность выводить список городов по запросу, а также выводить всю информацию о метрополитене конкретного города.

Подсказка:

Список городов из двух слов, разделенных пробелом:

```
lst_double = ['Нижний Новгород', 'Кривой Рог']
```

Входные данные:

Файл *towns.txt*

Пример содержимого файла:

```
№ Город  Пассажиров(млн.чел./год) Длина_линий(км) Линий Станций 1. Шанхай  3880.00 802.0 19 506 2. Пекин 3848.40 807.0 27 478 3. Токио 3217.00 310.3 13 290
```

Выходные данные:

1. Файл *metro_out.txt*

Пример содержимого файла:

```
№   Город  Пассажиров(млн.чел./год)   Длина линий(км)    Линий  Станций
1. Шанхай 3880.00    802.0  19 506
2. Пекин  3848.40    807.0  27 478
3. Токио  3217.00    310.3  13 290
```

2. Пример интерфейса программы.

```
Меню:
1. Список городов
2. Название города
3. Выход
?: 3
```

3. Формат вывод информации по метрополитену конкретного города.

```
Метрополитен города Шанхай.
Пассажиров: 3880.00 млн.чел./год
Общая протяженность: 802.0 км.
Количество линий: 19
Число станций: 506
```

2.3 Лабораторная работа №3. Работа с массивами данных. Построение графиков.

В лабораторной работе необходимо научиться создавать массивы и решать задачи, связанны с обработкой массивов данных. Студентам предлагается решить три задачи, предварительно познакомившись с тремя библиотеками: array, numpy и matplotlib.

Модуль array[2]

Модуль array предназначен для работы с массивами.

Массив – структура, которая объединяет данные одного типа.

Массив очень похож на список, за исключением того, что тип, хранящихся в нем объектов, ограничен. Модуль позволяет создавать массивы: целых, вещественных чисел и символов.

Для создания массива используется специальный класс

array.array (typecode[*, initializer*] *)*

Здесь *typecode* – определяет тип элементов массива; *initializer* – необязательный параметр, которые содержит значения для массива. Параметр *initializer* может быть любым итерируемым по значениям объектом.

```
import array
#Массив целых чисел
A = array.array('b', range(50))
#Массив вещественных чисел
lst = [0.1, 0.2, 0.3, 0.4, 0.5]
B = array.array('f', lst)
# Символьный массив
txt = 'Array'
C = array.array('u', txt)
```

Таблица 3.11 Возможные типы элементов массива

Код	Тип	Размер
'b'	целый со знаком	1
'B'	целый без знака	1
'u'	Символьный	2
'h'	целый со знаком	2
'H'	целый без знака	2
'i'	целый со знаком	4
'I'	целый без знака	4
'l'	целый со знаком	4
'L'	целый без знака	4
'q'	целый со знаком	8
'Q'	целый без знака	8
'f'	вещественный одинарная точность	4
'd'	вещественный двойная точность	8

Объекты массива поддерживают обычные операции списков: индексация, срезы, конкатенация и умножение.

Некоторые методы для работы с массивами приведены в таблице 3.12.

[2] https://docs.python.org/3/library/array.html

Таблица 3.12 Методы для работы с массивами модуля array

Метод	Описание
typecode	TypeCode символ, использованный при создании массива.
itemsize	размер в байтах одного элемента в массиве.
append(x)	добавление элемента в конец массива.
count(x)	возвращает количество вхождений x в массив.
extend(iter)	добавление элементов из объекта в массив.
fromlist(список)	добавление элементов из списка.
index(x)	номер первого вхождения x в массив.
insert(n, x)	включить новый пункт со значением x в массиве перед номером n. Отрицательные значения рассматриваются относительно конца массива.
pop(i)	удаляет i-ый элемент из массива и возвращает его. По умолчанию удаляется последний элемент.
remove(x)	удалить первое вхождение x из массива.
reverse()	обратный порядок элементов в массиве.
tolist()	преобразование массива в список.

Пример 1

Переставить первый элемент массива на место *k*-го элемента. При этом второй, третий, ..., *k*-й элементы сдвинуть влево на 1 позицию.

Решение:

```
import array
import random as rnd

# Создаем массив и заполняем его случайными числами
n = 10
lst = [rnd.randint(1, 100) for _ in range(n)]
A = array.array('i', lst)
print(A)
# Удаляем последний символ (pop) и вставлем его на нулевую
позицию (insert).
A.insert(0, A.pop())
print(A)
```

Пример 2

Одномерный массив A(30) заполнили случайными целыми числами в диапазоне от -100 до 100. Необходимо подсчитать сколько в массиве положительных чисел и сколько отрицательных.

Решение:

```
import array
import random as rnd

# Создаем массив и заполняем его случайными числами
```

```
n = 30
lst = [rnd.randint(-100, 100) for _ in range(n)]
A = array.array('i', lst)
# Обнуляем счетчики neg- отрицательные числа, pos -
положительные числа
neg = 0
pos = 0
# Перебираем все элементы массива и сравниваем каждый с нулем.
for a in A:
    if a < 0:
        neg += 1
    elif a > 0:
        pos += 1
print(f'Положительных чисел: {pos}')
print(f'Отрицательных чисел: {neg}')
```

Модуль numpy[3]

Numpy предоставляет вам огромный набор быстрых и эффективных способов создания массивов и манипулирования числовыми данными внутри них.

Создание массива на основе существующего объекта.

```
numpy.array(object, dtype=None)[4]
```

Здесь *object* - массив, любой объект, предоставляющий интерфейс массива, или любая (вложенная) последовательность. Если объект является скаляром, возвращается 0-мерный массив, содержащий объект; *dtype* – требуемый тип данных элементов массива.

```
#Одномерный массив из элементов типа int32
A = np.array([1, 2, 3, 4])
#Двумерный массив из элементов типа int32
B = np.array([[1, 2, 3] , [4, 5, 6]], dtype=np.int16)
#Одномерный массив из элементов типа float64
C = np.array([1, 2, 3, 4], dtype=float)
# Одномерный массив из элементов типа bool
lst = [True, False, True, False]
T = np.array(lst)
# # Одномерный массив из элементов типа str_
S = np.array(list('Python'))
```

Иногда требуется создать массиву специального типа, например, единичную матрицу. Для этого в модуле *numpy* имеются соответствующие методы.

Таблица 3.13 Методы numpy для создания специальных массивов

Тип	Описание	Пример
zeros	Массив из нулей	a = np.zeros((2,2), dtype=int)
ones	Массив из единиц	b = np.ones((3, 3))
full	Массив из произвольных чисел	c = np.full((8,), 10)
random.rand	Массив случайных вещественных чисел	d = np.random.rand(3, 3)

[3] https://numpy.org/doc/stable/
[4] https://numpy.org/doc/stable/reference/generated/numpy.array.html#numpy.array

random.randint	Массив случайных целых чисел	e = np.random.randint(1, 10, size=(2, 2))
identity	Единичная матрица (квадратная)	h = np.identity(3)
eye	Единичная матрица (прямоугольная)	k = np.eye(3,5,2)
arange	Массив с регулярно увеличивающимися значениями	n = np.arange(1, 10.2, 0.2)
linspace	Массив с регулярно увеличивающимися значениями	m = np.linspace(1, 10, 46)

Индексация в одномерных массивах аналогична спискам. Можно получить (изменить) значение массива по его индексу, получить срез.

```
import numpy as np
lst = [i**2 for i in range(10)]
A = np.array(lst)
print(A)          # [ 0  1  4  9 16 25 36 49 64 81]
print(A[0])       # 0
A[0] = 99
print(A)          # [99  1  4  9 16 25 36 49 64 81]

print(A[:3])      # [99  1  4]
A[::2] = 77
print(A)          # [77  1 77  9 77 25 77 49 77 81]
```

В многомерных массивах для обращения к элементам указывается индекс для каждой оси (измерения) через запятую.

```
import numpy as np
def f(x, y):
    return x + y

B = np.fromfunction(f, (3, 3), dtype=int)
print(B)
'''
[[0 1 2]
 [1 2 3]
 [2 3 4]]
'''
b23 = B[1, 2]
print(b23) # 3

A11 = B[1:, 1:]
print(A11)
'''
[[2 3]
 [3 4]]
'''
```

Перебор многомерных массивов производится относительно первой оси.

```
import numpy as np

A = np.array([[ 0,  1,  2,  3],
       [10, 11, 12, 13],
       [20, 21, 22, 23],
```

```
        [30, 31, 32, 33],
        [40, 41, 42, 43]])

for a in A:
    print(a)
'''
[0 1 2 3]
[10 11 12 13]
[20 21 22 23]
[30 31 32 33]
[40 41 42 43]
'''
```

Чтобы перебрать все элементы массива используется атрибут *flat*.

```
import numpy as np

A = np.array([[ 0,  1,  2,  3],
        [10, 11, 12, 13],
        [20, 21, 22, 23],
        [30, 31, 32, 33],
        [40, 41, 42, 43]])

for a in A.flat:
    print(a, end=' ')
```

```
test_03
C:\Users\gav\PycharmProjects\LR_for_Python\venv\Scripts\py
0 1 2 3 10 11 12 13 20 21 22 23 30 31 32 33 40 41 42 43
Process finished with exit code 0
```

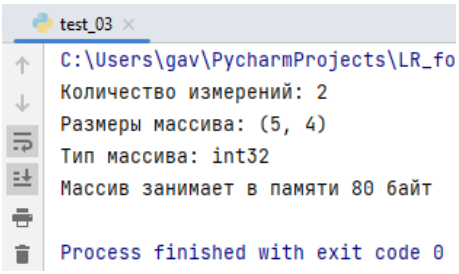
Наиболее важные атрибуты массивов приведены в таблице 3.14.

Таблица 3.14 Атрибуты массива

Атрибут	Описание
ndim	количество осей (размерностей) массива
shape	размеры массива. Это кортеж целых чисел, указывающий размер массива в каждом измерении.
size	общее количество элементов массива
dtype	тип элементов массива
itemsize	размер в байтах каждого элемента массива
data	буфер, содержащий фактические элементы массива

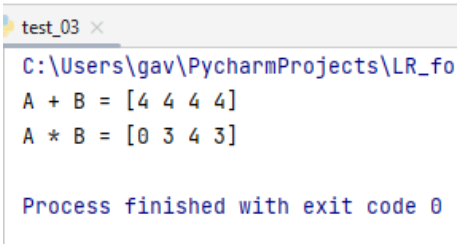
```
import numpy as np

A = np.array([[ 0,  1,  2,  3],
        [10, 11, 12, 13],
        [20, 21, 22, 23],
        [30, 31, 32, 33],
        [40, 41, 42, 43]])
```

```
print(f'Количество измерений: {A.ndim}')
print(f'Размеры массива: {A.shape}')
print(f'Тип массива: {A.dtype}')
print(f'Массив занимает в памяти {A.size * A.itemsize} байт')
```



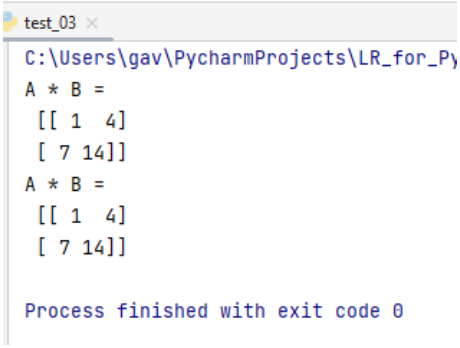
Арифметические операции в массивах выполняются поэлементно. В результате создается новый массив.

```
import numpy as np

A = np.array([ 0,  1,  2,  3])
B = np.array([ 4,  3,  2,  1])

print(f'A + B = {A+B}')
print(f'A * B = {A*B}')
```



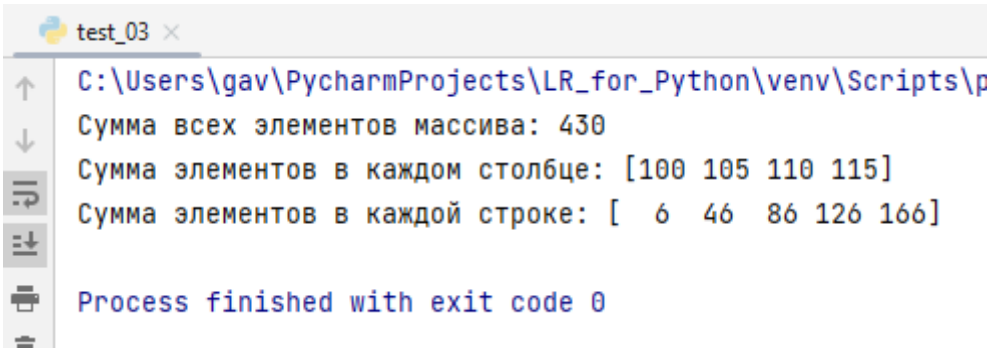
Для выполнения матричных операций используются специальные операции или методы. Например, для матричного умножения используется операция @ или метод *dot*.

```
import numpy as np

A = np.array([[ 0,  1],  [2,  3]])
B = np.array([[ 2,  1],  [1,  4]])

print(f'A * B =\n {A@B}')
print(f'A * B =\n {A.dot(B)}')
```



Многие функции, такие как вычисление суммы всех элементов, поиск максимального элемента и т.д., реализованы с помощью методов. При этом, по умолчанию, они применяются целиком ко всему массиву. Но, если указать параметр *axis*, метод применяется вдоль указанной оси.

```
import numpy as np

A = np.array([[ 0,  1,  2,  3],
        [10, 11, 12, 13],
        [20, 21, 22, 23],
        [30, 31, 32, 33],
        [40, 41, 42, 43]])


print(f'Сумма всех элементов массива: {A.sum()}')
print(f'Сумма элементов в каждом столбце: {A.sum(axis=0)}')
print(f'Сумма элементов в каждой строке: {A.sum(axis=1)}')
```



Модуль matplotlib[5]

Matplotlib — это комплексная библиотека для создания статических, анимированных и интерактивных визуализаций на Python.

С помощью модуля pyplot можно строить следующие графики: линейные графики, диаграммы разброса, столбчатые диаграммы и гистограммы, круговые диаграммы, ствол-лист диаграммы, контурные графики, поля градиентов, спектральные диаграммы.

Построение графических объектов с помощью pyplot осуществляется в три этапа:

- создание массивов данных;
- построение и настройка графика (диаграммы и т.д.);
- отображение на экране.

Исходные данные хранятся в списках, кортежах или массивах. В примере ниже для создания значений по осям используются массивы из модуля numpy.

```
import matplotlib.pyplot as plt
 import numpy as np

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
```

Построение простого графика

Для построения простого графика используется стандартный метод *plot*. В качестве параметров достаточно указать массивы с данными.

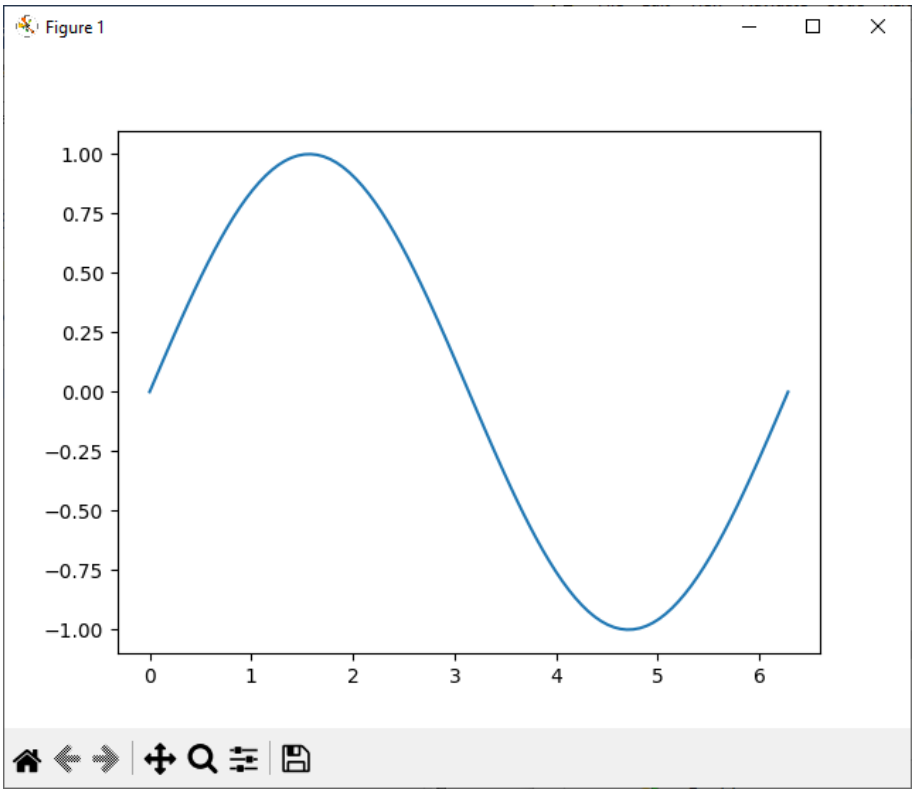
```
plt.plot(x, y)
```

Функция *show*, отвечает за вывод визуализированных данных на экран.

```
plt.show()
```

[5] https://matplotlib.org/3.6.2/index.html

В результате получим график функции *y=sin(x)*, который отображается в специальном окне.



В качестве третьего параметра функции plot можно указать строку, которая определяет формат линии графика. А именно: цвет, тип линии и тип маркеров. Строка формата — это просто аббревиатура для быстрой настройки основных свойств строки. Все это и многое другое можно также контролировать с помощью ключевых аргументов.

Строка формата состоит из трех символов: цвета графика, тип маркера и тип линии.

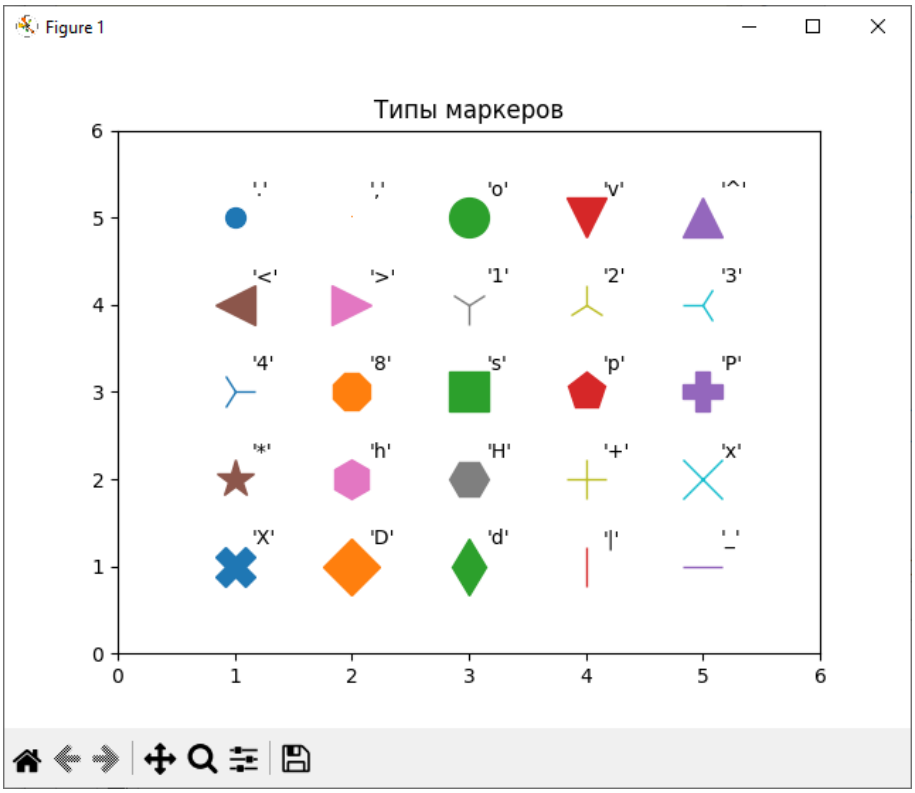
```
fmt = '[marker][line][color]'
```

Каждый из этих символов является необязательным.

Таблица 3.15 Символы для обозначения типа маркера

Символ	Описание
‘.’	точечный маркер
‘,’	пиксельный маркер
‘o’	круглый маркер
‘v’	маркер в виде треугольника вершиной вниз
‘^’	маркер в виде треугольника вершиной вверх
‘<’	маркер в виде треугольника вершиной влево
‘>’	маркер в виде треугольника вершиной вправо
‘1’	символ ⟩, направленный вниз
‘2’	символ ⟩, направленный вверх
‘3’	символ ⟨

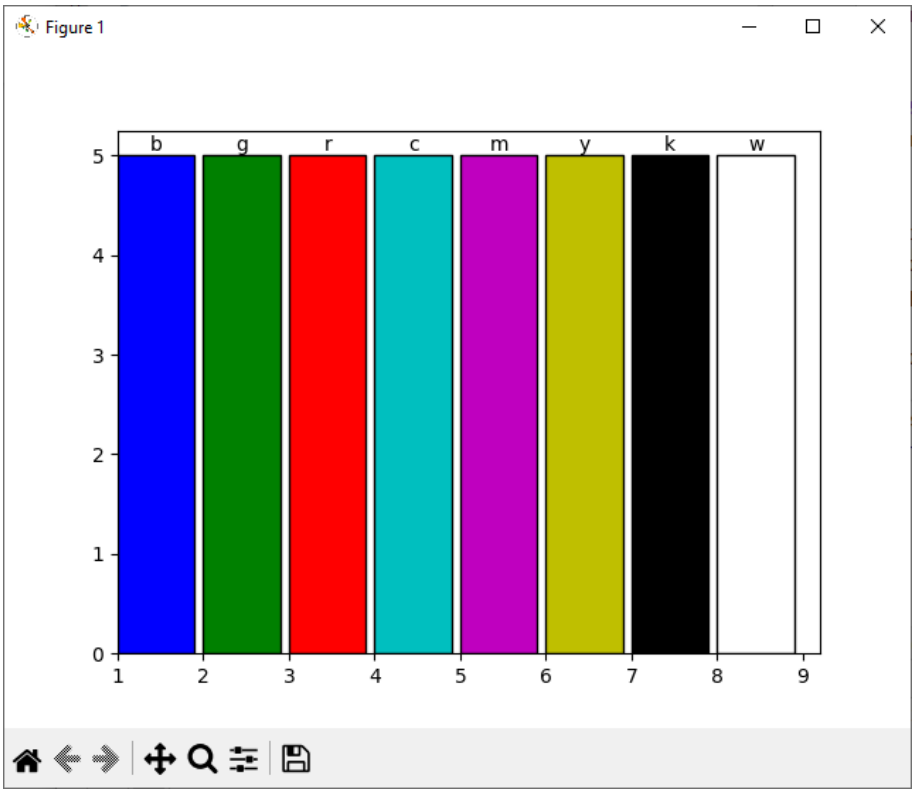
‘4’	символ 〉
‘8’	закрашенный восьмиугольник
‘s’	закрашенный квадрат
‘p’	закрашенный пятиугольник
‘P’	закрашенный плюс
‘*’	закрашенная звезда
‘h’	закрашенный шестиугольник
‘H’	закрашенный шестиугольник
‘+’	маркер в виде плюса
‘x’	маркер в виде крестика
‘X’	закрашенный крестик
‘D’	закрашенный ромб
‘d’	узкий закрашенный ромб
‘|’	маркер в виде вертикальной черты
‘_’	маркер в виде горизонтальной черты



Поддерживаемые цветовые сокращения представляют собой однобуквенные коды.

Таблица 3.16 Условное обозначение цвета линий графика

Код	Цвет
‘b’	синий
‘g’	зеленый
‘r’	красный
‘c’	голубой
‘m’	пурпурный
‘y’	желтый
‘k’	черный
‘w’	белый



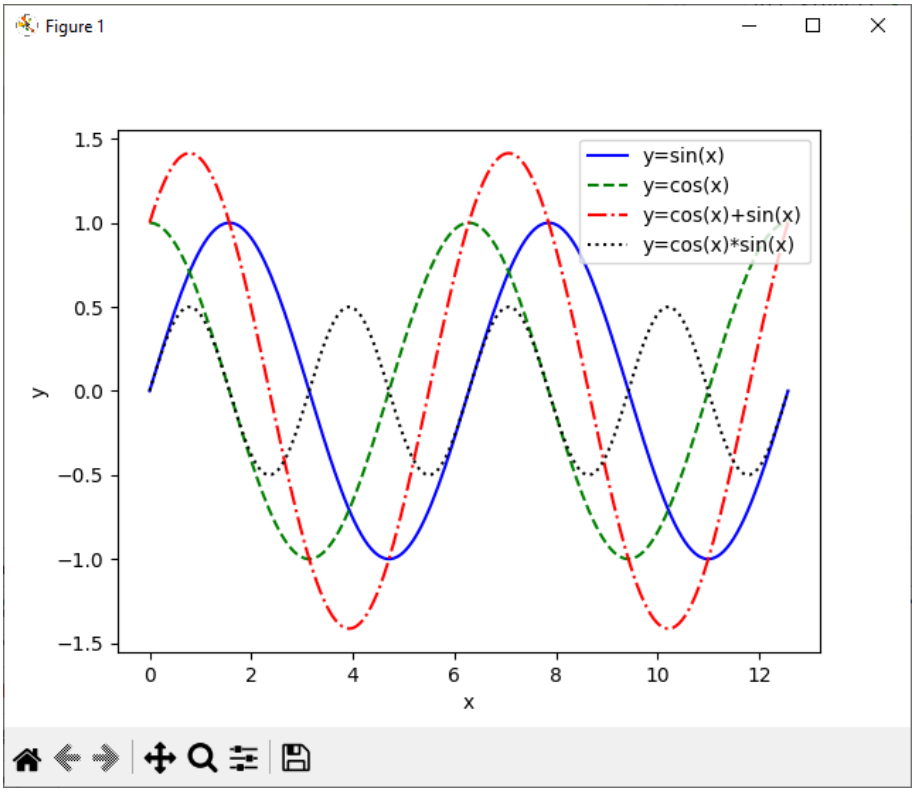
Тип линии определяется следующим набором символов

Таблица 3.17 Условное обозначения типа линии

Код	Линия
‘-’	сплошная
‘--’	штриховая
‘-.’	штрихпунктирная
‘:’	пунктирная

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 4*np.pi, 200)
```

```
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.cos(x) + np.sin(x)
y4 = np.cos(x) * np.sin(x)

plt.xlabel('x')
plt.ylabel('y')

plt.plot(x, y1, '-b', label='y=sin(x)')
plt.plot(x, y2, '--g', label='y=cos(x)')
plt.plot(x, y3, '-.r', label='y=cos(x)+sin(x)')
plt.plot(x, y4, ':k', label='y=cos(x)*sin(x)')

plt.legend()
plt.show()
```



Задание к лабораторной работе

Задача №1

Используя методы модуля array необходимо создать по определенным правилам массив и решить, связанную с ним, задачу в соответствии со своим вариантом.

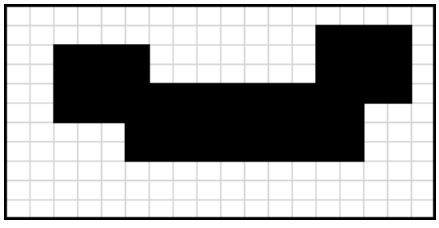
№	Задача
1	Заполнить одномерный массив размером N ($N > 1$) случайными вещественными числами, лежащими в диапазоне от a ($a < 0$) до b ($a < b$). Значения переменных N, a, b – вводятся с клавиатуры. Выполнить с элементами массива следующие действия: а) каждый отрицательный элемент заменить его абсолютной величиной; б) все элементы с нечетными индексами заменить их квадратным корнем.

2	Заполнить одномерный массив размером N ($N > 1$) случайными вещественными числами, лежащими в диапазоне от a ($a > 0$)до b ($a < b$). Значения переменных N, a, b – вводятся с клавиатуры. Выполнить с элементами массива следующие действия: а) каждый элемент, больший M ($a \le M \le b$), заменить его квадратным корнем (значение для M вводится с клавиатуры); б) у всех элементов массива с четными индексами изменить знак на противоположный.
3	Заполнить одномерный массив размером N ($N > 1$) случайными вещественными числами, лежащими в диапазоне от a ($a < 0$) до b ($a < b$). Значения переменных N, a, b – вводятся с клавиатуры. Выполнить с элементами массива следующие действия: а) из всех положительных элементов вычесть элемент с индексом k1, из остальных – элемент с номером k2 (значения $0 \le k1 \le N-1$ и $0 \le k2 \le N-1$ вводятся с клавиатуры); б) все элементы с нечетными индексами увеличить на один, с четными – уменьшить на единицу.
4	Заполнить одномерный массив размером N ($N > 1$) случайными вещественными числами, лежащими в диапазоне от a ($a < 0$) до b ($a < b$). Значения переменных N, a, b – вводятся с клавиатуры. Выполнить с элементами массива следующие действия: а) ко всем отрицательным элементам прибавить элемент с индексом $m1$, к остальным – элемент с индексом m2 (значения $0 \le m1 \le N-1$ и $0 \le m2 \le N-1$ вводятся с клавиатуры); б) все элементы с четными индексами удвоить, с нечетными – уменьшить на число π.
5	Заполнить одномерный массив размером N случайными целыми числами, лежащими в диапазоне от a до b. Значения переменных N, a, b – вводятся с клавиатуры. Выполнить с элементами массива следующие действия: а) все элементы, оканчивающиеся четной цифрой, уменьшить вдвое; б) все нечетные элементы заменить на их квадраты; в) четные элементы увеличить на k (значения для k вводятся с клавиатуры), а из элементов с четными индексами вычесть y (значения для k вводятся с клавиатуры).
6	Заполнить одномерный массив размером N случайными целыми числами, лежащими в диапазоне от a до b. Значения переменных N, a, b – вводятся с клавиатуры. Выполнить с элементами массива следующие действия: а) все элементы, кратные числу 10, заменить нулем; б) все нечетные элементы удвоить, а четные уменьшить вдвое; в) нечетные элементы уменьшить на m (натуральное число), а элементы с нечетными индексами увеличить на N (целое число).
7	Заполнить одномерный массив размером N случайными целыми числами, лежащими в диапазоне от a до b. Значения переменных N, a, b – вводятся с клавиатуры. Составить программу расчета: а) квадратного корня из любого элемента массива; б) среднего арифметического m любых элементов массива, идущих подряд.
8	Заполнить одномерный массив размером N случайными целыми числами, лежащими в диапазоне от a до b. Значения переменных N, a, b – вводятся с клавиатуры. Необходимо проверить: а) является ли i-й элемент массива положительным числом; б) является ли k-й элемент массива четным числом; в) какой элемент массива больше: k-й или i-й.

9	Заполнить одномерный массив размером N ($N > 1$) случайными вещественными числами, лежащими в диапазоне от a ($a < 0$) до b ($a < b$). Значения переменных N, a, b – вводятся с клавиатуры. Требуется найти: а) сумму квадратов всех элементов массива; б) сумму элементов массива с $k1$-го по $k2$-й (значения $k1$ и $k2$ вводятся с клавиатуры; $k2 > k1$); в) среднее арифметическое всех элементов массива.
10	Заполнить одномерный массив размером N ($N > 1$) случайными вещественными числами, лежащими в диапазоне от a ($a < 0$) до b ($a < b$). Значения переменных N, a, b – вводятся с клавиатуры. Требуется найти: а) произведение всех элементов массива; б) сумму первых m элементов массива; в) среднее арифметическое элементов массива с $s1$-го по $s2$-й (значения $s1$ и $s2$ вводятся с клавиатуры; $s2 > s1$).

Задача №2

Для решения второй задачи необходимо создать двумерный массив с помощью модуля numpy.

№	Задача
1	Имеется квадратный двумерный массив вещественных чисел размером N. Необходимо найти сумму чисел в каждой строке, в каждом столбце и по диагонали.
2	Прямоугольное минное поле разбито на квадратные сектора и представляет собой матрицу размером NxM. В каждом секторе размещается от нуля до k мин. В соответствии с приказом министра обороны количество мин, размещенных по периметру, должно быть не менее, чем число взрывных устройств внутри него (периметра). Необходимо написать программу, которая проверяет соответствует ли размещение мин существующим требованиям. На вход подается сначала два числа N и M. Затем N строк по M чисел через пробел. *3 4* *2 3 4 0* *3 2 1 2* *5 4 2 1*
3	Художник авангардист рисует только картины только черными квадратами. Но, поскольку он далеко не Малевич, делает это следующим способом: сначала разбивает весь холст на квадраты, в результате получается сетка размером NxM (координаты ячейки в верхнем левом углу (0, 0)); затем закрашивает соседние ячейки сетки таким образом, что в результате получается прямоугольник со сторонами параллельными сторонам холста. 

На вход программе подается два числа – размер холста в клетках. На следующей строке количество квадратов *k*. Ещё ниже *k* строк из четырех чисел через пробел – координаты верхнего левого угла и нижнего правого угла прямоугольников.

11 18

3

2 2 5 5

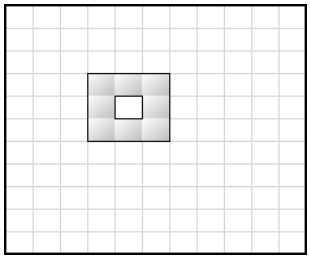
4 5 7 14

1 13 4 16

Необходимо написать программу, которая подсчитывает площадь закрашенной поверхности холста

4 Имеется квадратный двумерный массив из целых чисел размером *NxM*.

Для произвольной ячейки с координатами (*i*, *j*), где $0 \le i, j < N$, на сумму чисел в соседних ячейках.



На вход программе сначала подается два числа (размер массива) через пробел. Затем координаты ячейки (два числа через пробел).

5 Создать три массива размером *NxM* и заполнить их целыми числами следующими способами:

а) змейкой

1	2	3	4	5
10	9	8	7	6
11	12	13	14	15
20	19	18	17	16
21	22	23	24	25

б) по спирали

1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

в) по диагонали

1	2	3	4	5
2	1	2	3	4
3	2	1	2	3
4	3	2	1	2
5	4	3	2	1

6 Дан массив *A* размером *NxM*. Создать новый массив *B*, поворотом массива *A* на угол *alpha*. (*alpha = ±90, 180, ±270*).

Пример поворота влево на 90° показан на рисунке.

32	25	33	19	10
43	55	44	55	34
25	19	27	20	17
34	36	39	33	53
42	25	18	46	33

10	34	17	53	33
19	55	20	33	46
33	44	27	39	18
25	55	19	36	25
32	43	25	34	42

7 Поле, по которому движется робот, разбито на квадратные сектора. По одной стороне *N* секторов, по другой – *M*. Движением робота управляет программа, которая состоит из подряд идущих символов:

F – вперед на один сектор;

L – повернуть налево на 90°;

R – повернуть направо на 90°.

На вход программы подаются размеры поля, адрес ячейки, с которой стартует робот, и программа его движения.

Определить номер ячейки, в которой остановился робот после выполнения программы. Выведите маршрут движения робота.

Примечание. В самом начале, когда робот размещается на поле, он «смотрит» направо. Команда, которая не может быть выполнена в данный момент – игнорируется.

8 Результаты эксперимента занесли в двумерный массив следующим образом: в первую ячейку каждой строки записано значение аргумента x_i, в остальных – результаты измерений y_{ij}.

x_1	y_{11}	y_{11}	…	y_{1M}
x_2	y_{21}	y_{21}	…	y_{2M}
…				
x_N	y_{N1}	y_{N1}	…	y_{NM}

Для каждого значения x_i найти среднее значение y_i и поместить результаты в двумерный массив.

x_i
y_i

На вход программы сначала подаются значения *N* и *M* через пробел. Затем *N* строк с *M+1* вещественным числом через пробел.

9	Рисунок представлен в виде матрицы размером *NxM*. В каждой ячейке находится целое число от 0 до 255 – яркость пиксела. Необходимо преобразовать картинку к черно-белому варианту. Для этого необходимо выполнить следующее: 1. найти среднее арифметическое всех чисел; 2. если число в ячейке меньше среднего, заменить его нулем; 3. если число в ячейке больше или равно среднему, заменить его на 255. Вывести на экран массив с использованием символов: ■ - код 11035; □ - код 11036.
10	Поле разбито на *NxM* клеток. В каждой клетке может находиться либо ноль, либо единица. Для произвольной ячейки необходимо подсчитать общее количество единиц в строке, столбце и диагоналях, на пересечении которых она находится.

Задача №3

С помощью модуля matplotlib построить график функции в соответствии со своим вариантом. Для вычисления функций используйте методы модуля numpy.

№	Функция	Границы
1	$y(x) = \begin{cases} 1 + \sin(3{,}5x) + \left\lvert cos\left(3{,}5x - \frac{\pi}{6}\right)\right\rvert, & \text{если } x \geq \frac{\pi}{2} \\ \frac{\lvert cos(2x)\rvert}{sin(x)+1} + \frac{\lvert sin(3{,}96x)\rvert}{sin(-1{,}2x)}, & \text{если } x < \frac{\pi}{2} \end{cases}$	[0.1; 3.2]
2	$y(x) = \begin{cases} 2.5 \cdot sin(2{,}6x) \cdot \frac{2 - 3{,}1 \cdot sin(\lvert 2{,}5(x-1)\rvert)}{sin(\lvert 2{,}5x\rvert + \pi) + \pi}, & \text{если } x \leq 1.8 \\ 1.6 \cdot tg(\lvert x - 2.1\rvert) + \frac{ctg(\lvert 0{,}2x\rvert)}{0{,}95x}, & \text{если } x > 1.8 \end{cases}$	[0.1; 3.2]
3	$y(x) = \begin{cases} \lvert cos(\pi x)\rvert + cos(2.2x + \frac{\pi}{7}) + 0.7, & \text{если } x \geq \frac{\pi}{2} \\ \lvert sin(\pi x)\rvert + cos(\lvert 2.1x\rvert, & \text{если } x < \frac{\pi}{2} \end{cases}$	[0.1; 3.2]
4	$y(x) = \begin{cases} \lvert sin(\pi x)\rvert + cos(\lvert 2x\rvert), & \text{если } x \geq \frac{\pi}{2} \\ \lvert cos(\pi x)\rvert + cos\left(2.2\lvert x\rvert + \frac{\pi}{7} + 0.7\right), & \text{если } x < \frac{\pi}{2} \end{cases}$	[0.1; 3.2]
5	$y(x) = \begin{cases} 1 + \sin(3.5x) + \left\lvert \cos\left(3.5x - \frac{\pi}{6}\right)\right\rvert, & \text{если } x \geq \frac{\pi}{2} \\ \lvert \cos(\pi x)\rvert + \cos(2.2\lvert x\rvert + \frac{\pi}{7} + 0.7), & \text{если } x < \frac{\pi}{2} \end{cases}$	[0.1; 3.2]
6	$y(x) = \begin{cases} 0.55x^{1.2} - 0.34 \cdot e^{\sqrt{0.72x}} + 0.08 \cdot \cos(4x - \pi), & \text{если } x \geq \frac{\pi}{2} \\ 0.5 \cdot tg\big(0.6 \cdot (x + 3.5)\big) + 0.1 \cdot \cos(5.5x), & \text{если } x < \frac{\pi}{2} \end{cases}$	[0.1; 3.2]
7	$y(x) = \left(2 \cdot \sin\left(\frac{2}{x}\right) \cdot \cos\left(\frac{2}{x}\right)\right)^2$	[0.1; 0.8]

8	$y(x) = \|0.25 \cdot x + 3 \cdot cos(10 \cdot x) \cdot \sin(x)\|$	[-2; 2]
9	$y(x) = \cos(\cos(\cos(\cos(x^6)))) \cdot 3x$	[-6; 6]
10	$y(x) = 8 \cdot \sin(3x) \cdot \cos(5x)$	[-4; 4]

2.4 Лабораторная работа №4. Построение функций и диаграмм с помощью модуля turtle

Черепашья графика — это реализация популярных инструментов геометрического рисования, представленных в Logo[6].

В Python графика черепахи представляет собой физическую «черепаху» (маленького робота с ручкой), которая рисует на листе бумаги.

Это эффективный и хорошо зарекомендовавший себя способ познакомиться с концепциями программирования и взаимодействия с программным обеспечением, поскольку он обеспечивает мгновенную и наглядную обратную связь. Он также обеспечивает удобный доступ к графическому выводу в целом [1].

Начало работы

Сначала необходимо импортировать все объекты модуля turtle в программу с помощь команды

```
from turtle import *
```

В принципе, этого уже достаточно, чтобы при запуске программы открывалось окно с черепашкой. Однако, после того как будут выполнены все команды в программе, графическое окно сразу закроется. Увидеть результаты работы программы в данном случае будет затруднительно. Поэтому необходимо добавить метод

```
mainloop()
```

до окончания программы. Теперь программа будет ждать закрытия и не прекратит свою работу пока, например, не будет закрыто окно.

Рисование примитивов

Напишите команду

```
forward(200).
```

Черепашка двинется вперед и нарисует горизонтальную линию длиной в 200 точек.

По умолчанию в самом начале программы черепашка расположена в центре экрана (в точке с координатами (0, 0) и «смотрит» вправо. Что нарисовать линию в другом направлении нужно повернуть черепашку, например, направо. Напишите команду

```
right(90)
forward(200)
```

Нарисована еще одна линия такой же длины, но уже вертикальная. Допишем ещё дважды последние две команды и построим квадрат со сторонами 200.

```
right(90)
forward(200)
right(90)
forward(200)
```

[6] Logo — образовательный язык программирования, в которой команды для перемещения и рисования создают линейную или векторную графику либо на экране, либо с помощью небольшого робота, называемого черепахой.

На рисунке 3.1 приведен код программы и результат работы. Обратите внимание. Код программы на рисунке несколько отличается того, что мы набрали ранее. Сравните два кода и скажите какой лучше? Почему?

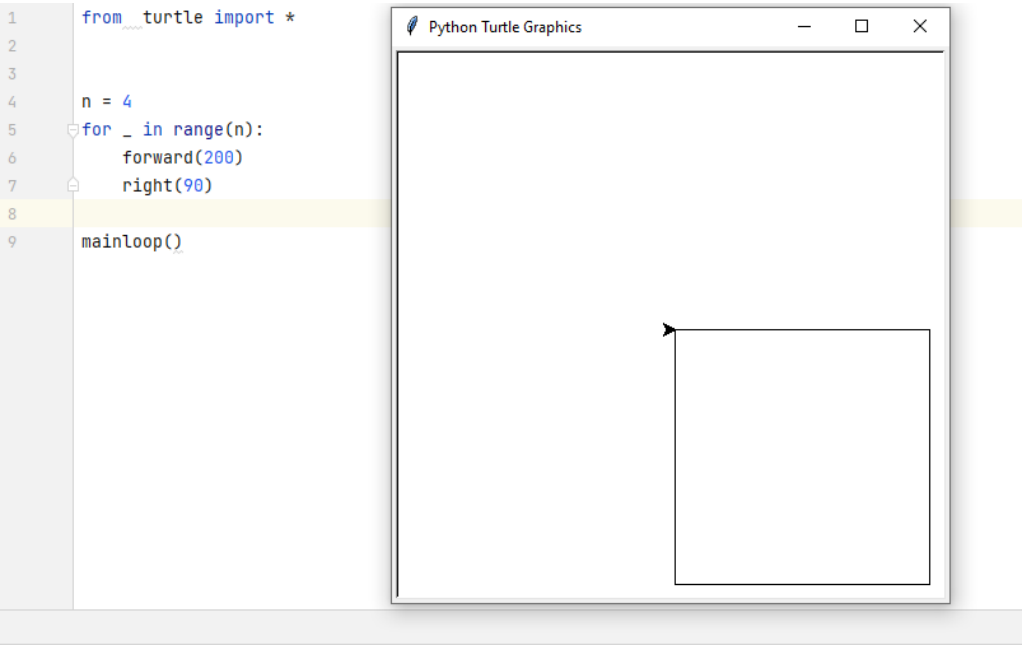


Рисунок 3.1 Программа рисования прямоугольника

Список основных методов перемещения черепашки приведен в таблице 3.18.

Таблица 3.18 Методы для управления черепашкой

Метод	**Описание**	**Синоним**
forward(***distance***)	Переместить черепаху вперед на расстояние *distance*.	fd()
backward(***distance***)	Переместить черепаху назад на расстояние *distance*.	back() | bk()
right(***angle***)	Повернуть черепаху вправо на *angle* градусов.	rt()
left(***angle***)	Повернуть черепаху влево на *angle* градусов.	lt()
goto(***x, y=None***)	Переместить черепаху в точку с координатами (*x, y*). Если *y=None*, то *x* это пара значений или вектор *Vec2D*.	setpos() | setposition()
teleport(***x, y=None, *, fill_gap=False***)	Переместить черепаху в точку с координатами (*x, y*) без рисования линии.	-
home()	Переместить черепаху в начало координат.	
circle(***radius, extent=None, steps=None***)	Рисует окружность радиусом *radius* или многоугольник, вписанный в эту окружность.	

	Здесь *extent* – угол сектора круга (для рисования дуг), *steps* – количество сторон многоугольника)
dot(***size=None, *color***)	Рисует точку диаметром *size* цвета *color*.

Настройки параметров окна

Окно с холстом открывается всегда с дефолтными значениями ширины и высоты. Узнать их можно с помощью функций window_width() и window_height() соответственно.

Но, размеры, заданные по умолчанию, не всегда удобно. Если рисунок небольшой, то не нужно большого окна для его отображения.

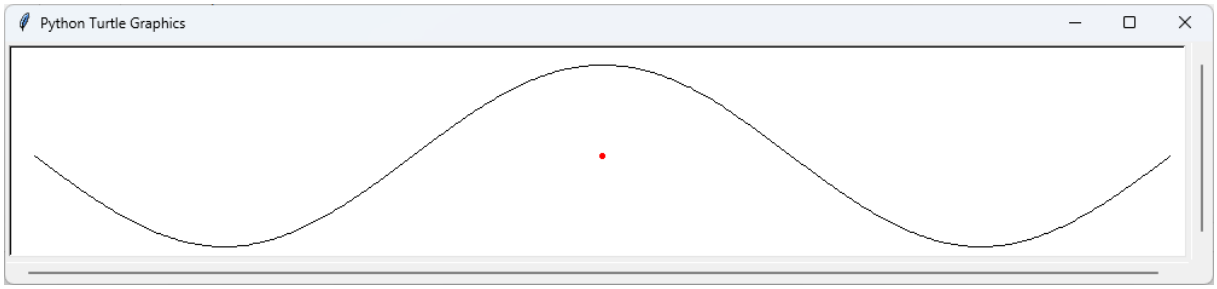


Рисунок 3.2 Окно размером 1000x200

Определите необходимые размеры окна и установите их с помощью метода

```
setup(width, height, startx, starty)
```

где *width* – ширина окна, *height* – высота окна, *startx* и *starty* – координаты верхнего левого угла окна.

Очень часто при построении графиков (например), координатная сетка холста не совпадает с декартовой. Это можно исправить с помощью метода

```
setworldcoordinates(llx, lly, urx, ury).
```

Здесь *llx* и *lly* – координаты нижнего левого угла, *urx* и *ury* – верхнего правого.

Предположим необходимо построить график функции f(x)=cos(x) на интервале от –π до +π. Область значений косинуса находится в интервале [-1, 1]. Соответственно график разместим в прямоугольной области с координатами (-π, -1) – нижний левый угол и (π, 1) – верхний правый (см. рис. 3.3).

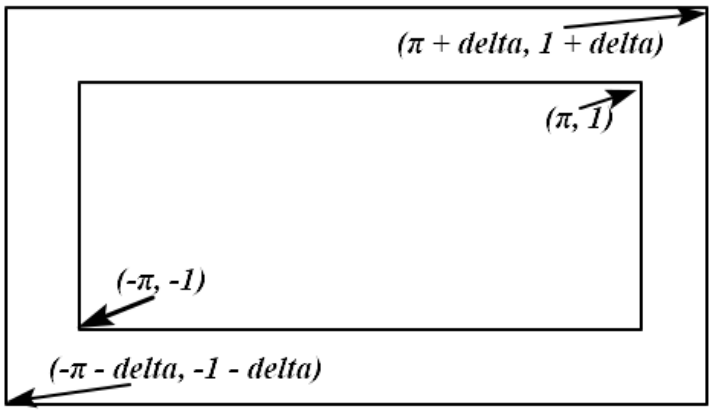


Рисунок 3.3 Расчет координатной сетки

Чтобы график не касался границ не помешает отодвинуть их на небольшое расстояние *delta*.

Параметры черепашки

Существует ряд функций, которые позволяют изменять характеристики черепашки. Например, её отображение или видимость, скорость перемещения, положение пера и т.п.

Таблица 3.19 Методы для изменения параметров черепашки

Метод	**Описание**	**Синоним**
penup()	Поднимает перо. При перемещении черепахи не остается следа.	pu() | up()
pendown()	Опускает перо вниз. Во время движения будет происходить рисование.	pd() | down()
width(*width=None*)¶	Устанавливает толщину линии или возвращает её. Width – положительное число. Если аргумент не задан, то возвращается текущая толщина пера	pensize()
pencolor(**args*)	Устанавливает или возвращает цвет пера. Возможно четыре формата записи аргументов. **pencolor**() – возвращает текущий цвет; **pencolor**(colorstring) – в качестве аргумента передается строка с названием цвета (‘green’, blue’) или его шестнадцатеричным кодом(‘#abcdef’); **pencolor**(r, g, b) или **pencolor**((r, g, b)) – цвет задается в формате RGB.	color()
hideturtle()	Делает черепашку невидимой	ht()
showturtle()	Делает черепашку видимой	st()
speed(*speed=None*)	Устанавливает скорость рисования. Здесь speed – либо целое число от 0 до 10, либо строка: “fastest”: 0, “fast”: 10, “normal”: 6, “slow”: 3, “slowest”: 1	

Задание к лабораторной работе

Необходимо написать функцию, которая рисует на экране график математической функции.

Входными данными к подпрограмме являются: имя функции с формулой, диапазоны по осям.

Функция должна уметь самостоятельно рассчитывать координатную сетку; строить оси координат, строить график функции.

№	**Функция**	**Границы**	**Y_{max}**	**Y_{min}**
1	$y(x) = \left(2 \cdot \sin\left(\frac{2}{x}\right) \cdot \cos\left(\frac{2}{x}\right)\right)^2$	[0.1; 0.8]	1	0

2	$y(x) = \lvert 0.25 \cdot x + 3 \cdot cos(10 \cdot x) \cdot \sin(x) \rvert$	[-2; 2]	4	0
3	$y(x) = \cos(\cos(\cos(\cos(x^6)))) \cdot 3x$	[-6; 6]	10	-10
4	$y(x) = 8 \cdot \sin(3x) \cdot \cos(5x)$	[-4; 4]	8	-8
5	$y(x) = \dfrac{x}{\left\lvert \cos\left(\dfrac{x}{\sin(4x)}\right) \right\rvert}$	[3.6; 4.1]	80	0
6	$y(x) = \cos^2\left(\pi\dfrac{x}{2}\right) + \sin^3(\pi x)$	[0; 3]	1.65	-0.6
7	$y(x) = \lvert 5x^2 + 1 \rvert \cdot (x^3 + x^2 - 2) \cdot \sqrt{2 - x^2}$	[-1.45; 1.45]	10.5	-15
8	$y(x) = x \cdot \sin\left(\dfrac{1}{\cos(x)}\right)$	[1.7; 4.65]	5	-5
9	$y(x) = \begin{cases} -4 \cdot \cos(20\pi x) - 0.35, \text{если} \lvert x \rvert \le 0.01 \\ cox(x) + \ln(\lvert x \rvert), \text{иначе} \end{cases}$	[-4; 4]	1	-4.5
10	$y(x) = 0.25x^{-2} + \left\lvert \dfrac{1}{2x} cos(12x) \cdot \sin(x - 3.1) \right\rvert$	[0.5; 4]	0.6	-0.3

2.5 Лабораторная работа №5. Создание приложений с помощью модуля Tkinter

Tkinter[7] – это пакет для Python, предназначенный для работы с компонентами графического интерфейса пользователя (graphical user interface – GUI).

Под графическим интерфейсом пользователя (GUI) подразумеваются все те окна, кнопки, текстовые поля для ввода, скроллеры, списки, радиокнопки, флажки и другие элементы, которые пользователь видит на экране, открывая то или иное приложение. С их помощью происходит взаимодействие с программой, управление её поведением. Все эти элементы интерфейса будем называть виджетами.

Начало работы с модулем

Для создания самого простого окна достаточно трех строчек кода.

```
import tkinter as tk
win = tk.Tk()
win.mainloop()
```

В первой строке импортируем библиотеку tkinter в программу. Далее создаем объект класса Tk. Он также создает окно верхнего уровня, которое служит главным окном приложения.

В конце указывается метод *mainloop*, который выводит окно на экран и обрабатывает все действия, которые совершает пользователь с программой до тех пор, пока программа не завершится.

В результате получим окно (рис. 3.4) размером 200x200 в левом верхнем углу экрана с небольшим смещением.

[7] https://younglinux.info/tkinter/tkinter

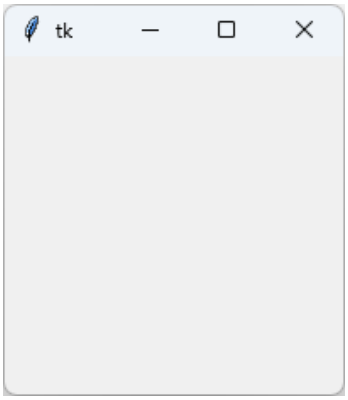


Рисунок 3.4 Окно приложения по умолчанию

Затем настраиваем окно в соответствии с требованиями к программе. Методы, которые для этого используются перечислены в таблице 3.19.

Таблица 3.19 Методы настройки окна

Свойство	**Значение**
title()	Заголовок окна
geometry()	Размеры окна и начальная позиция на экране
resizable()	Запрет/разрешение изменения размеров окна
minsize()	Минимальные размеры окна
maxsize()	Максимальные размеры окна
attributes()	Установка атрибутов окна ('-alpha', '-transparentcolor', '-disabled', '-fullscreen', '-toolwindow', '-topmost')
iconbitmap(default="")	Установка иконки

Размеры окна и смещение задается с помощь строки формата

widthx**height**+**x**+**y**

где width – ширина окна, height – его высота, x – смещение по горизонтали относительно левой границы экрана, y – смещение по горизонтали вниз относительно верхней стороны экрана.

```
#Новые размеры окна и пользовательское смещение
win.geometry('500x200+300+300')
#Стандартные размеры окна и пользовательское смещение
win.geometry('+300+300')
#Новые размеры окна со стандартным смещением
win.geometry('500x200')
```

Виджеты

Tkinter предлагает довольно большой выбор элементов управления, которые называются виджетами.

Button: кнопка

Label: текстовая метка

Entry: однострочное текстовое поле

Text: многострочное текстовое поле

Menu: элемент меню

Scrollbar: полоса прокрутки

Treeview: позволяет создавать древовидные и табличные элементы

Checkbutton: флажок

Radiobutton: переключатель или радиокнопка

Frame: фрейм, который организует виджеты в группы

Listbox: список

Combobox: выпадающий список

Scale: текстовая метка

Spinbox: список значений со стрелками для перемещения по элементам

Progressbar: текстовая метка

Canvas: текстовая метка

Notebook: панель вкладок

Имеется два набора виджетов. Первый располагается непосредственно в пакете *tkinter*. Второй – виджеты из пакета *tkinter.ttk*. Оба пакета предлагают практически одинаковый набор виджетов. Например, кнопки (Button) и текстовые метки (Label) имеются в обоих пакетах (рис. 3.5).

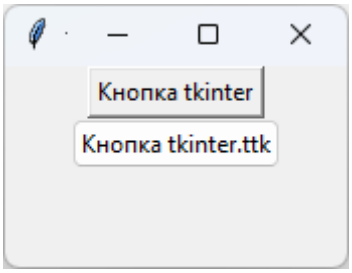


Рисунок 3.5 Разница между кнопками из разных пакетов

Каждый виджет имеет свой набор атрибутов, которые позволяет устанавливать внешний его вид и настроить поведение.

Обычно атрибуты виджета задаются при создании объекта.

```
btn_tk = tk.Button(win, text='Кнопка', width=10)
```

Можно определять значение атрибутов и после создания, используя синтаксис словаря.

```
btn_tk = tk.Button()
btn_tk['text'] = 'Кнопка'
btn_tk['width'] = 10
```

Третий способ изменения значений атрибутов виджета – использование метода *config*.

```
btn_tk = tk.Button()
btn_tk.config(text='Кнопка', width=10)
```

Размещение виджетов

Для размещения виджетов в окне используются различные способы. Первый способ – использование метода *pack*.

В таблице 3.20 представлены атрибуты этого метода.

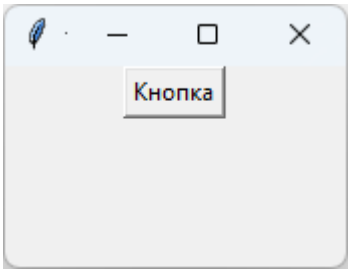
Таблица 3.20 Атрибуты метода pack

Атрибут	Описание
expand	если равно True, то виджет заполняет все пространство контейнера.
fill	определяет, будет ли виджет растягиваться, чтобы заполнить свободное пространство вокруг. Этот параметр может принимать следующие значения: NONE (по умолчанию, элемент не растягивается), X (элемент растягивается только по горизонтали), Y (элемент растягивается только по вертикали) и BOTH (элемент растягивается по вертикали и горизонтали).
anchor	помещает виджет в определенной части контейнера. Может принимать значения n, e, s, w, ne, nw, se, sw, c, которые являются

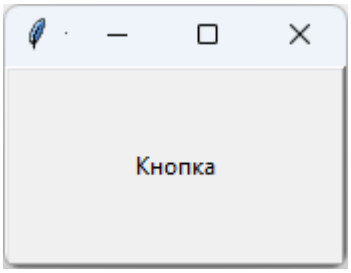
	сокращениями от Noth(север - вверх), South (юг - низ), East (восток - правая сторона), West (запад - левая сторона) и Center (по центру
side	выравнивает виджет по одной из сторон контейнера. Может принимать значения: TOP (по умолчанию, выравнивается по верхней стороне контейнера), BOTTOM (выравнивание по нижней стороне), LEFT (выравнивание по левой стороне), RIGHT (выравнивание по правой стороне).
ipadx	устанавливает отступ содержимого виджета от его границы по горизонтали.
ipady	устанавливают отступ содержимого виджета от его границы по вертикали.
padx	устанавливает отступ виджета от границ контейнера по горизонтали.
pady	устанавливает отступ виджета от границ контейнера по вертикали.

Примеры размещения кнопок в окне с помощью метода *pack*.

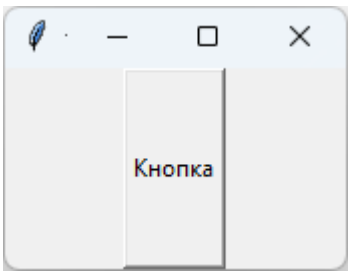
```
btn = tk.Button(text="Кнопка")
btn.pack()
```



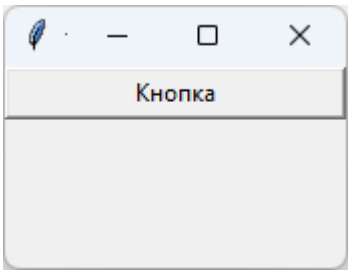

```
btn = tk.Button(text="Кнопка")
btn.pack(expand=True, fill='both')
```



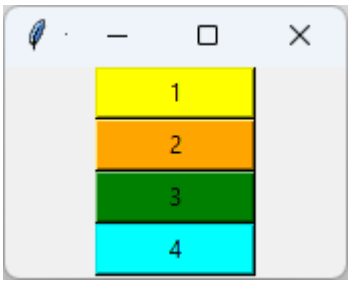

```
btn = tk.Button(text="Кнопка")
btn.pack(expand=True, fill='y')
```



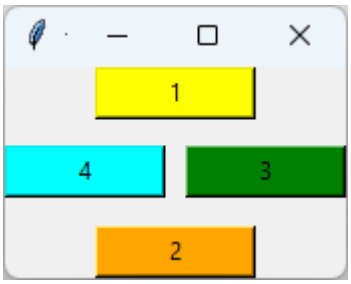

```
btn = tk.Button(text="Кнопка")
btn.pack(fill='x')
```



```
btn_1 = tk.Button(width=10, bg='yellow',
text='1')
btn_2 = tk.Button(width=10, bg='orange',
text='2')
btn_3 = tk.Button(width=10, bg='green',
text='3')
btn_4 = tk.Button(width=10, bg='cyan',
text='4')

btn_1.pack()
btn_2.pack()
btn_3.pack()
btn_4.pack()
```




```
btn_1 = tk.Button(width=10, bg='yellow',
text='1')
btn_2 = tk.Button(width=10, bg='orange',
text='2')
btn_3 = tk.Button(width=10, bg='green',
text='3')
btn_4 = tk.Button(width=10, bg='cyan',
text='4')

btn_1.pack(side='top')
btn_2.pack(side='bottom')
btn_3.pack(side='right')
btn_4.pack(side='left')
```



Второй метод (*place*) позволяет более точно настроить координаты и размеры виджета. Атрибуты этого метода приведены в таблице 3.21.

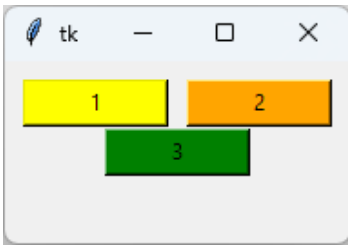
Таблица 3.21 Атрибуты метода place

Атрибут	**Описание**
height и width	устанавливают соответственно высоту и ширину элемента в пикселях
relheight и relwidth	также задают соответственно высоту и ширину элемента, но в качестве значения используется число float в промежутке между 0.0 и 1.0, которое указывает на долю от высоты и ширины родительского контейнера
x и y	устанавливают смещение элемента по горизонтали и вертикали в пикселях соответственно относительно верхнего левого угла контейнера
relx и rely	также задают смещение элемента по горизонтали и вертикали, но в качестве значения используется число float в промежутке между 0.0 и 1.0, которое указывает на долю от высоты и ширины родительского контейнера
bordermode	задает формат границы элемента. Может принимать значение INSIDE (по умолчанию) и OUTSIDE
anchor	устанавливает опции растяжения элемента. Может принимать значения n, e, s, w, ne, nw, se, sw, c, которые

	являются сокращениями от North(север - вверх), South (юг - низ), East (восток - правая сторона), West (запад - левая сторона) и Center (по центру).

Для размещения виджетов в окне (контейнере) указываются абсолютные (x и y) или относительные (relx и rely) координаты. Отсчет координат ведется относительно верхнего левого угла окна.

```
btn_1 = tk.Button(bg='yellow', text='1')
btn_2 = tk.Button(bg='orange', text='2')
btn_3 = tk.Button(bg='green', text='3')

btn_1.place(width=80, x=10, y=10)
btn_2.place(width=80, x=100, y=10)
btn_3.place(width=80, relx=.5, rely=.5,
anchor='center')
```



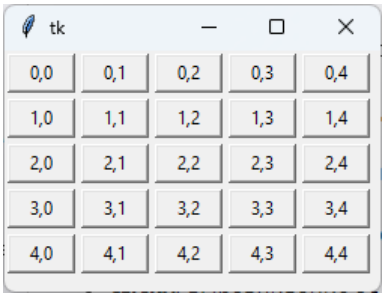
Третий способ размещения виджетов предполагает, что окно разбито на ячейки. В этом случае метод *grid* позволяет поместить виджет в определенную ячейку. В таблице 3.22 перечислены атрибуты метода *grid*.

Таблица 3.22 Атрибуты метода grid

Атрибут	**Описание**
column	номер столбца, отсчет начинается с нуля
row	номер строки, отсчет начинается с нуля
columnspan	сколько столбцов должен занимать элемент
rowspan	сколько строк должен занимать элемент
ipadx и ipady	отступы по горизонтали и вертикали соответственно от границ элемента до его содержимого
padx и pady	отступы по горизонтали и вертикали соответственно от границ ячейки грида до границ элемента
sticky	выравнивание элемента в ячейке, если ячейка больше элемента. Может принимать значения n, e, s, w, ne, nw, se, sw, которые указывают соответствующее направление выравнивания

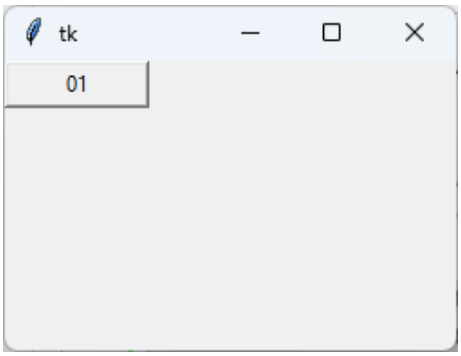
Этот метод очень удобен для симметричного размещения большого количества виджетов.

```
n = 5
w = 250 // n
for r in range(n):
  for c in range(n):
      btn = tk.Button(width=n,
text=f"{r},{c}")
      btn.grid(row=r, column=c, padx = 2,
pady=2)
```



Однако, если необходимо сразу поместить виджет в определенную ячейку, то это может не получиться.

```
btn_01 = tk.Button(width=10, text='01')

btn_01.grid(row=2, column=2)
```



Как видно на рисунке, кнопка располагается в верхнем левом углу, а не в ячейке с координатами (2, 2).

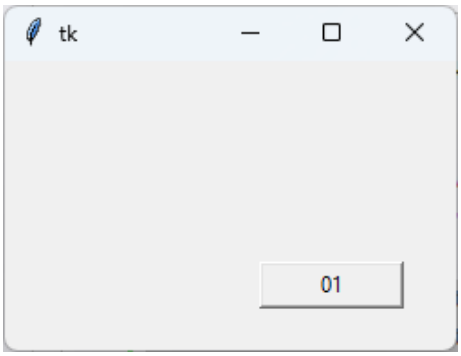
Чтобы это исправить необходимо предварительно создать сетку. Для этого используются два метода *columnconfigure(index, weight)* и *rowconfigure(index, weight)*.

Здесь *index* – индекс столбца (строки), weight – вес столбца (строки).

Метод *columnconfigure* используется для настройки столбца. Соответственно, метод *rowconfigure* настраивает строку. Столбцы (строки) распределяются по всей ширине (высоте) контейнера в соответствии со своим весом.

Изменим предыдущий код программы.

```
btn_01 = tk.Button(width=10, text='01')

for c in range(3):
    win.columnconfigure(index=c,
weight=1)
for r in range(3):
    win.rowconfigure(index=r, weight=1)

btn_01.grid(row=2, column=2)
```



Кнопки

Кнопка – это наиболее часто используемый виджет. В *tkinter* она представлена классом *Button*.

Кнопки необходимы чтобы выполнить какие-то действия. После нажатия на кнопку должен запускаться специальный код, который оформляется в виде функции. Связывается функция и кнопка с помощью атрибута *command*.

```
def quit():
    win.destroy()

win = tk.Tk()
win.geometry('200x200+500+100')

btn_exit = tk.Button(win, text='Выход',
                     command=quit,
                     font='"Arial" 14 italic',
                     width=10,
                     height=5)
btn_exit.configure(bg='lightgreen')
btn_exit.pack()
```

Таблица 3.23 Атрибуты виджета Button

Атрибут	Описание
command	функция, которая вызывается при нажатии на кнопку
compund	устанавливает расположение картинки и текста относительно друг друга
cursor	курсор указателя мыши при наведении на кнопку
font	параметры шрифта, которые используются при отображении текста кнопки
image	ссылка на изображение, которое отображается на кнопке
pading	отступы от границ кнопки до её текста
state	состояние кнопки
text	устанавливает текст кнопки
textvariable	устанавливает привязку к элементу StringVar
underline	указывает на номер символа в тексте кнопки, который подчеркивается. По умолчанию значение -1, то есть никакой символ не подчеркивается
width	ширина кнопки
height	высота кнопки

Текстовая метка

Виджет Label используется в основном для отображения текста в специальном поле. Его атрибуты схожи с атрибутами кнопки (см. таблицу 3.24)

Таблица 3.24 Атрибуты виджета Label

Атрибут	Описание
anchor	устанавливает позиционирование текста
background	фоновый цвет
borderwidth	толщина границы метки
cursor	курсор указателя мыши при наведении на метку
font	шрифт текста
foreground	цвет текста
height	высота виджета
image	ссылка на изображение, которое отображается на метке
justify	устанавливает выравнивание текста. Значение LEFT выравнивает текст по левому краю, CENTER - по центру, RIGHT - по правому краю
pading	отступы от границ виджета до его текста
relief	определяет тип границы, по умолчанию значение FLAT
text	устанавливает текст метки
textvariable	устанавливает привязку к элементу StringVar
underline	указывает на номер символа в тексте метки, который подчеркивается. По умолчанию значение -1, то есть никакой символ не подчеркивается
width	ширина виджета
wraplength	при положительном значении строки текста будут переносится для вмещения в пространство виджета

```
lbl_01 = tk.Label(win)
lbl_01.config(text='Label 01. Font: "Curier New", 10.',
```

```
                  font=('Curier New', 10))
lbl_01.place(x=10, y=10)

lbl_02 = tk.Label(win, text='Label 02. Font: "Arial" 10 bold. Text color
- green.',
                      font='"Arial" 10 bold', fg='green')
lbl_02.place(x=10, y=30)

tk.Label(win, text='Label 03. Font: "Times New Roman" 12 bold italic.',
         font='"Times New Roman" 12 bold italic').place(x=10, y=50)
```

Однострочное поле ввода

Виджет *Entry* используется для организации ввода данных пользователем. Элемент, помимо атрибутов аналогичных другим виджетам (таблица 3.25), имеет несколько отличных методов (таблица 3.26).

Таблица 3.25 Атрибуты виджета Entry

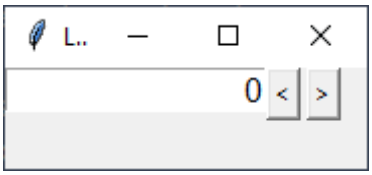
Атрибут	Описание
background	фоновый цвет
cursor	курсор указателя мыши при наведении на текстовое поле
foreground	цвет текста
font	шрифт текста
justify	устанавливает выравнивание текста. Значение LEFT выравнивает текст по левому краю, CENTER - по центру, RIGHT - по правому краю
show	задает маску для вводимых символов
state	состояние элемента, может принимать значения NORMAL (по умолчанию) и DISABLED
textvariable	устанавливает привязку к элементу StringVar
width	ширина элемента

Методы виджета Entry получить значение, которое находится в поле, или изменить его.

Таблица 3.26 Методы виджета Entry

Метод	Описание
insert(index, str)	вставляет в текстовое поле строку по определенному индексу
get()	возвращает введенный в текстовое поле текст
delete(first, last=None)	удаляет символ по индексу first. Если указан параметр last, то удаление производится до индекса last. Чтобы удалить до конца, в качестве второго параметра можно использовать значение END.
focus()	установить фокус на текстовое поле

Напишем программу, которая открывает окно с полем Entry с начальным значением равным нулю и двумя кнопками. Первая кнопка уменьшает значение в поле на единицу. Вторая – увеличивает.



Код программы

```
import tkinter as tk

def add():
    # Получаем значение из поля ent и увеличиваем его на единицу
    x = int(ent.get()) + 1
    # Очищаем поле
    ent .delete(0,tk.END)
    # Помещаем в поле новое значение
    ent.insert(0, x)

def sub():
    # Получаем значение из поля ent и уменьшаем его на единицу
    x = int(ent.get()) - 1
    # Очищаем поле
     ent .delete(0,tk.END)
    # Помещаем в поле новое значение
    ent.insert(0, x)

win = tk.Tk()
win.title('Label')
win.geometry('180x50+600+200')

ent = tk.Entry(win, justify=tk.RIGHT, font='"Curier New" 12',
width=14)
ent.insert(0, 0)
ent.place(x=0, y=0)

btn_01 = tk.Button(win, text='<', width=1, command=sub, height=1)
btn_01.place(x=130, y=0)

btn_02 = tk.Button(win, text='>', width=1, command=add, height=1)
btn_02.place(x=150, y=0)
win.mainloop()
```

Привязка виджетов к переменным

В модуле *tkinter* описан специальный класс *Variable* и его дочерние классы *IntVar*, *DoubleVar*, *StringVar*, *BooleanVar*.

Они необходимы для проверки состояния виджетов и позволяют упростить обработку данных. Если переменную, относящуюся к данным классам, связать с виджетом, то отпадает необходимость писать код для обработки события ввода данных.

Сначала необходимо создать переменную. Затем с помощью параметра *textvariable* привязать её к виджету или нескольким виджетам.

```
import tkinter as tk

win = tk.Tk()
win.title('Variable')
win.geometry('280x50+600+200')

st = tk.StringVar()

btn = tk.Button(win, textvariable=st)
btn.grid(row=0,column=0)

ent = tk.Entry(win, textvariable=st)
ent.insert(0, 'Кнопка')
ent.grid(row=0, column=1)

win.mainloop()
```

Если запустить эту программу и написать что-нибудь в поле *Entry*, то одновременно будет изменяться и название кнопки, т.к. переменная *st* связана с двумя виджетами одновременно.

Переменные *tkinter* имеют два метода

- get() – возвращает значение переменной;
- set(value) – устанавливает значение для переменной, которое передано через параметр.

```
st = tk.StringVar()
st.set('Кнопка')

btn = tk.Button(win, textvariable=st)
btn.grid(row=0,column=0)

ent = tk.Entry(win, textvariable=st)
ent.grid(row=0, column=1)
```

Однако, нет необходимости использовать метод *set* при создании переменной. Можно использовать атрибут *value*.

```
st = tk.StringVar(value='Кнопка')
```

Модуль Matplotlib

Matplotlib — это комплексная библиотека для создания статических, анимированных и интерактивных визуализаций[8].

Matplotlib отображает ваши данные на графиках (например, окнах, виджетах Jupyter и т. д.), каждый из которых может содержать одну или несколько областей (плоскостей), где точки могут быть указаны в декартовых координатах (или в полярных координатах, или

[8] https://matplotlib.org/stable/

трехмерные координаты и т. д.). Самый простой способ создания фигуры с осями — использовать метод pyplot.subplots. Затем мы можем использовать методы объекта Axes для передачи данных для осей (plot) и установки различных параметров, например, подписи осей.

Чтобы отобразить график понадобится метод plt.show().

Ниже приведен код пользовательской функции plot(). Для использования функции в своей программе необходимо создать две переменные vara и varb класса DoubleVar. Связать их с виджетами типа Entry, которые используются для ввода чисел, определяющих границы рисования графика.

В цикле используется обращение к ещё одной пользовательской функции f(x). Она также должна быть описана в программе и возвращать значение математической функции в точке x.

Саму функцию plot() необходимо связать с кнопкой «График».

```
def plot():
    import matplotlib.pyplot as plt
    #vara и varb глобальные переменные класса DoubleVar,
    # связанные с виджетами класса Entry
    # vara - левая граница и varb - правая граница по оси OX
    a = float(vara.get())
    b = float(varb.get())
    h = 0.01    #
    # Формируем два списка с исходными данными
    # x - аргумент, y - значение функции
    x, y = [], []
    while a <= b:
        x.append(a)
        y.append(f(a))
        a += h
    # Создаем и показываем график функции
    fig, ax = plt.subplots()
    ax.plot(x, y)
    ax.set_title('График функции. Вариант №0.')
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    plt.show()
```

Результат работы функции после нажатия кнопки показан на рисунке 3.6.

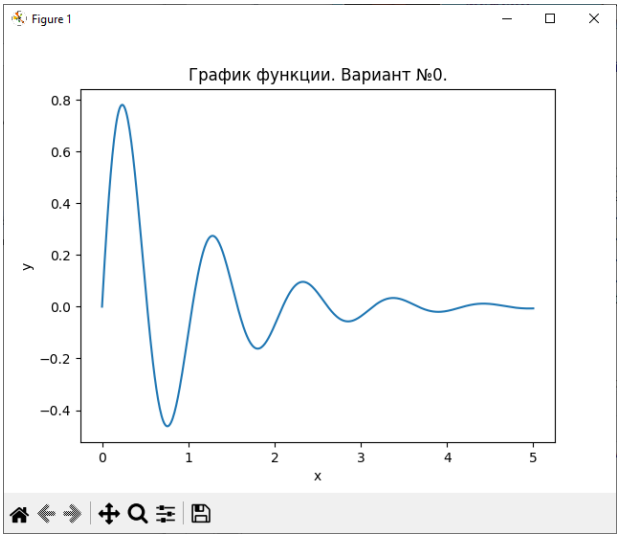
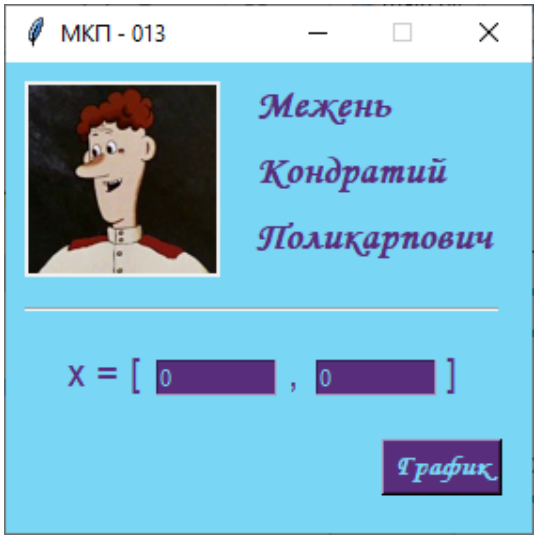


Рисунок 3.6 Результат работы функции plot()

Задание к лабораторной работе

Необходимо создать небольшое приложение по образцу.



Требования к приложению.

1) В строке заголовка указать свои инициалы автора и номер его группы через дефис.
2) Рекомендуемые размеры окна: высота – 250 px, ширина – 250 px. Можно увеличить ширину окна до 300 px, если не будет хватать места для ФИО. Размеры окна должны быть зафиксированы.
3) Расстояние от края окна и до любого объекта, а также между виджетами должно равняться 10-15 px.
4) Для фото использовать файл формата png размером 100x100. Выкладывать только свою фотографию!
5) Справа от фотографии разместить фамилию, имя и отчество разработчика приложения.
6) Для надписей использовать шрифты Monotype Corsiva, Arial
7) Цвета фона и текста берется из таблицы в соответствии со своим вариантом.
8) График функции строится с помощью модуля matplotlib. Используется функция plot() (см. выше).
9) Функция для построения графика берется из таблицы с вариантами.

№	Цвета	Функция
1	Фон: #D59FE6 Текст: #0619A2	$y = x^2 - \|2x\| - 6$
2	Фон: #58BD90 Текст: #1F3C5E	$y = \|x^2 - x - 8\|$
3	Фон: #85F720 Текст: #2D490A	$y = x(\|x\| - 4)$
4	Фон: #38F10C Текст: #125F93	$y = x^2 + 4x + 3$
5	Фон: #1F3C5E Текст: #58BD90	$y = \frac{\|x\|}{x}(x^2 + 3x + 5)$
6	Фон: #2D490A Текст: #85F720	$y = \frac{\|x-2\|}{2-x}(x^2 - 2x)$
7	Фон: #0B1A68 Текст: #4CA53E	$y = \big\|\|\|x\| - 2\big\| - 1\big\|$
8	Фон: #A8F46C Текст: #3E4920	$y = \big\|\|1 - x^2\| - 3\big\|$
9	Фон: #125F93 Текст: #38F10C	$y = 2 - \sqrt{\|x-3\|}$
10	Фон: #A3E658 Текст: #4C13AD	$y = \begin{cases} 3, & \text{если } x \le 4 \\ \|x^2 - 4\|x\| + 3\|, & \text{если } -4 < x \le 4 \\ 3 - (x-4)^2, & \text{если } x > 4 \end{cases}$

3 Заключение

Выполнение методических указаний к лабораторным работам по дисциплине «Информатика» способствует успешному ее освоению и развитию у обучающихся навыков написания программ с использованием языка Python.

4 Список рекомендуемой литературы

1. Чернышев, С. А. Основы программирования на Python: учебное пособие для вузов / С. А. Чернышев. — 2-е изд., перераб. и доп. — Москва: Издательство Юрайт, 2023. — 349 с. [Электронный ресурс]: — Режим доступа: https://urait.ru/bcode/532446 (дата обращения: 15.10.2024).
2. Стивенсон, Б. Python. Сборник упражнений: учебное пособие / Б. Стивенсон; перевод с английского А. Ю. Гинько. — Москва: ДМК Пресс, 2021. — 238 с. [Электронный ресурс]: — Режим доступа: https://e.lanbook.com/book/241025 (дата обращения: 15.10.2024).
3. Федоров, Д. Ю. Программирование на языке высокого уровня Python: учебное пособие для вузов / Д. Ю. Федоров. — 5-е изд., перераб. и доп. — Москва: Издательство Юрайт, 2023. — 227 с. [Электронный ресурс]: — Режим доступа: https://urait.ru/bcode/532868
4. Северанс, Ч. Р. Python для всех / Ч. Р. Северанс; перевод с английского А. В. Снастина. — Москва: ДМК Пресс, 2021. — 262 с. [Электронный ресурс]: — Режим доступа: https://e.lanbook.com/book/241115 (дата обращения: 15.10.2024).
5. Лутц, Марк. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с. : ил.
6. Лутц, Марк. Изучаем Python, том 2, 5-е изд.: Пер. с англ. — СПб. : ООО “Диалектика”, 2020. — 720 с. : ил.
7. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с., ил.
8. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с., ил.
9. Python 3.12.5 documentation: сайт / turtle — Turtle graphics © Copyright 2001-2024, Python Software Foundation –. URL: https://docs.python.org/3/library/turtle.html (дата обращения: 01.09.2024).