

Колледж космического машиностроения и технологии

ОТЧЕТ

По учебной практике УП.01.01 Разработка программных модулей программного обеспечения для компьютерных систем

специальность 09.02.03 Программирование в компьютерных системах

Оформил студент: Яковлев Прокопий Максимович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

Проверил преподаватель: Гусятинер Леонид Борисович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2020

**Оглавление**

[**Раздел 1. Техника решения задач с использованием структурного и объектно-ориентированного программирования** 3](#_Toc74681519)

[1.1. Установка интерпретатора Python 3 и настройка окружения 3](#_Toc74681520)

[1.2. Техника работы в командной строке и среде IDLE 5](#_Toc74681521)

[1.3. Техника работы с линейными и разветвляющимися программами 7](#_Toc74681522)

[1.4. Техника работы с циклическими программами, цикл while 11](#_Toc74681523)

[1.5. Техника работы с числами 17](#_Toc74681524)

[1.6. Техника работы со строками 18](#_Toc74681525)

[1.7. Техника работы со списками 20](#_Toc74681526)

[1.8. Техника работы с циклом for и генераторами списков 22](#_Toc74681527)

[1.9. Техника работы с функциями 23](#_Toc74681528)  
1.10. Техника работы со словарями…………………………………………………………………………………………………24

[1.11. Техника работы с множествами 25](#_Toc74681529)

[1.12. Техника работы с кортежами 26](#_Toc74681530)

[1.13. Техника работы с файлами 27](#_Toc74681531)

[1.14. Техника работы с модулями 29](#_Toc74681532)

[1.15. Техника работы с классами 33](#_Toc74681533)

[**Раздел 2. Техника решения задач с использованием библиотек** 37](#_Toc74681534)

[2.1 Установка и настройка среды JetBrains PyCharm 37](#_Toc74681535)

[2.2 Техника работы с базами данных 44](#_Toc74681536)

[2.3 Техника работы с библиотекой tkinter 48](#_Toc74681537)

[2.4 Техника работы с библиотекой NumPy 53](#_Toc74681538)

[2.5 Техника работы с библиотекой Matplotlib 55](#_Toc74681539)

[2.6 Элементы работы с библиотекой PyQt 58](#_Toc74681540)

[2.7 Элементы работы с библиотекой PyGame 73](#_Toc74681541)

[**Раздел 3. Разработка проекта с графическим интерфейсом** 78](#_Toc74681542)

[3.1 Изучение входной и выходной документации 78](#_Toc74681543)

[3.2 Разработка требований к проекту. Построение диаграммы использования. 80](#_Toc74681544)

[3.3 Разработка сценария проекта 81](#_Toc74681545)

[3.4 Построение диаграммы классов 81](#_Toc74681546)

[3.5 Разработка базы данных 82](#_Toc74681547)

[3.6 Разработка главного модуля 84](#_Toc74681548)

[3.7 Тестирование и отладка 90](#_Toc74681549)

[3.8 Дневник 91](#_Toc74681550)

[Приложения 93](#_Toc74681551)

# **Раздел 1. Техника решения задач с использованием структурного и объектно-ориентированного программирования**

# Установка интерпретатора Python 3 и настройка окружения

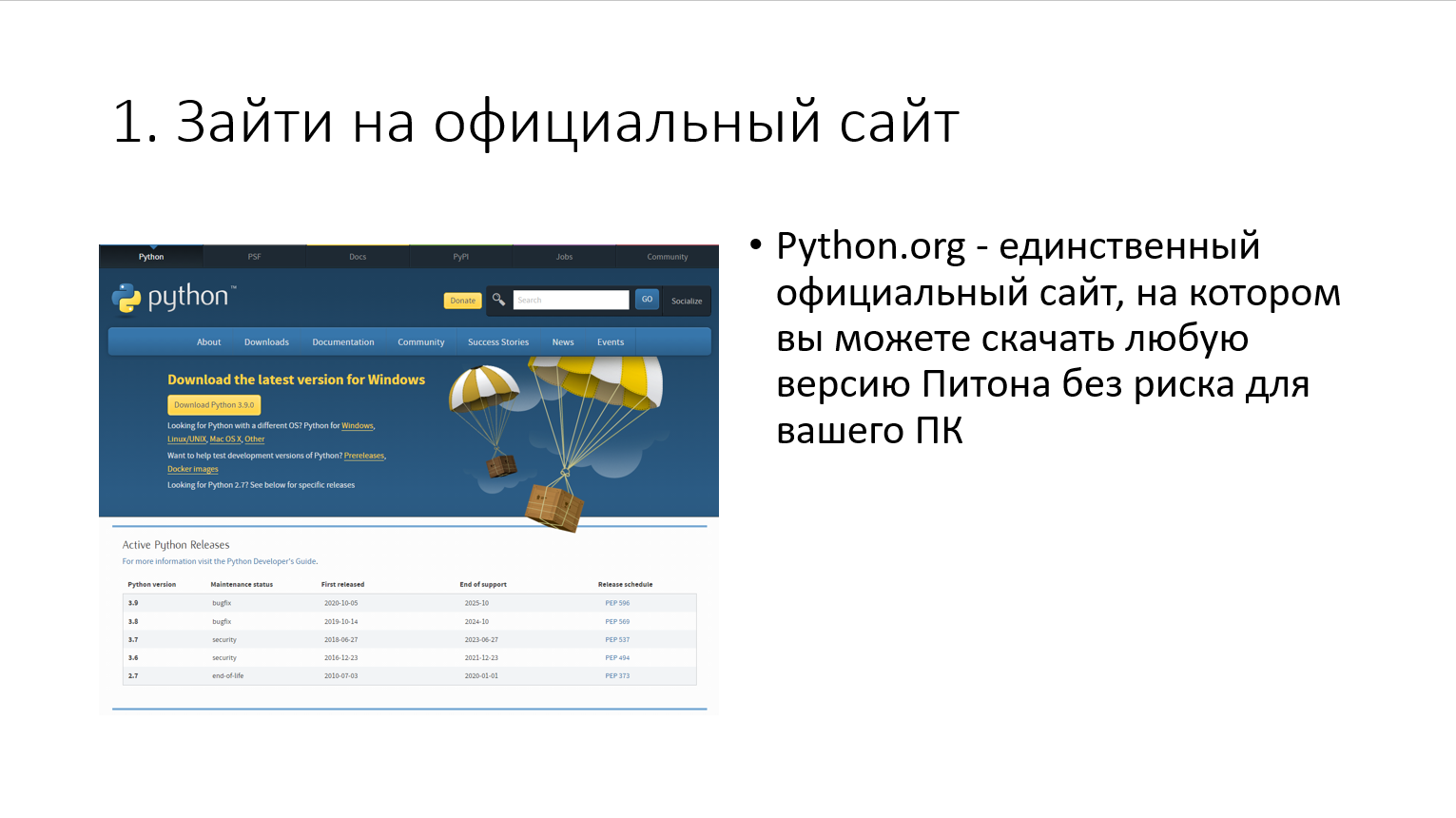


Рисунок 1. Первый этап

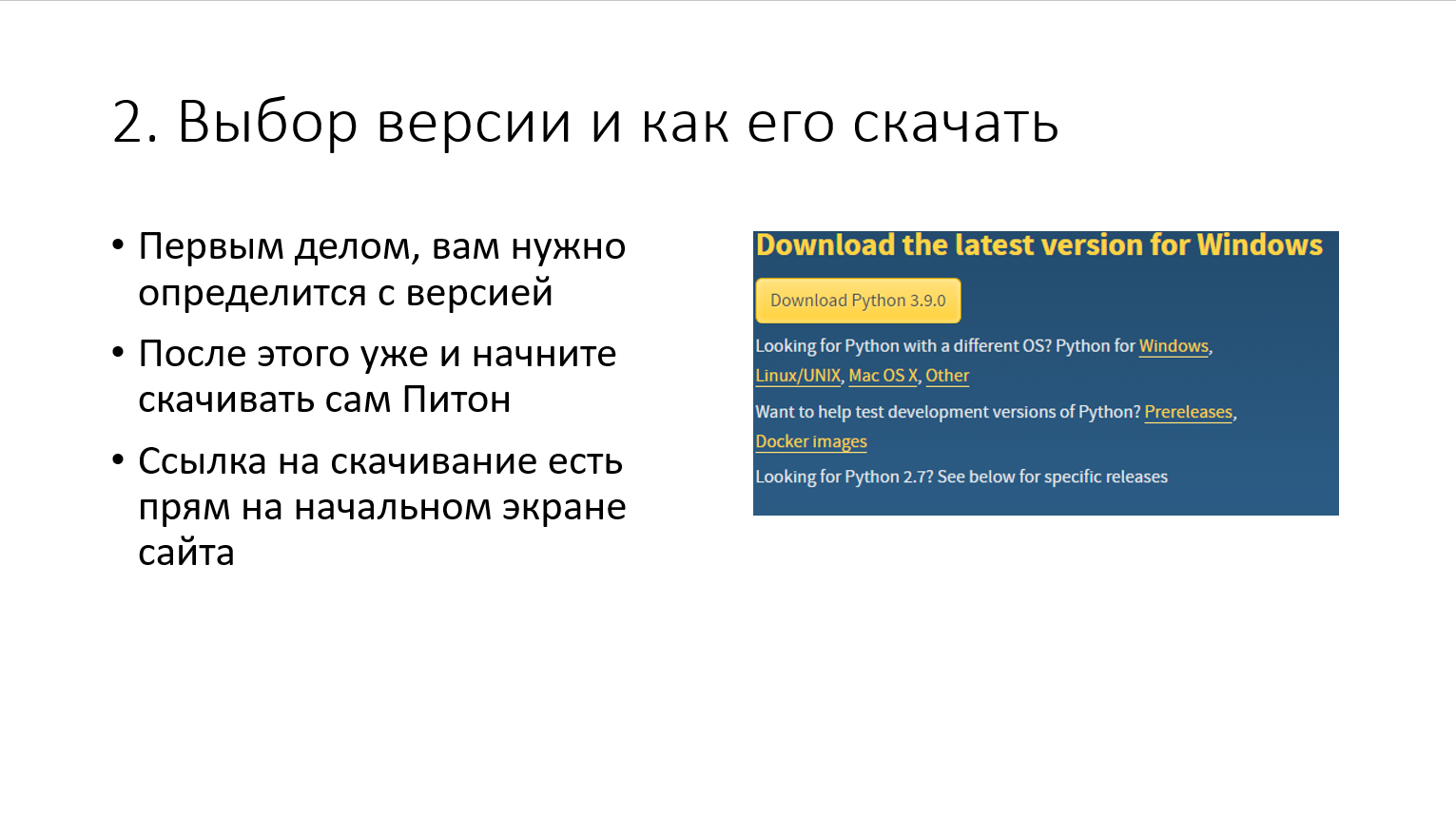


Рисунок 2. Второй этап

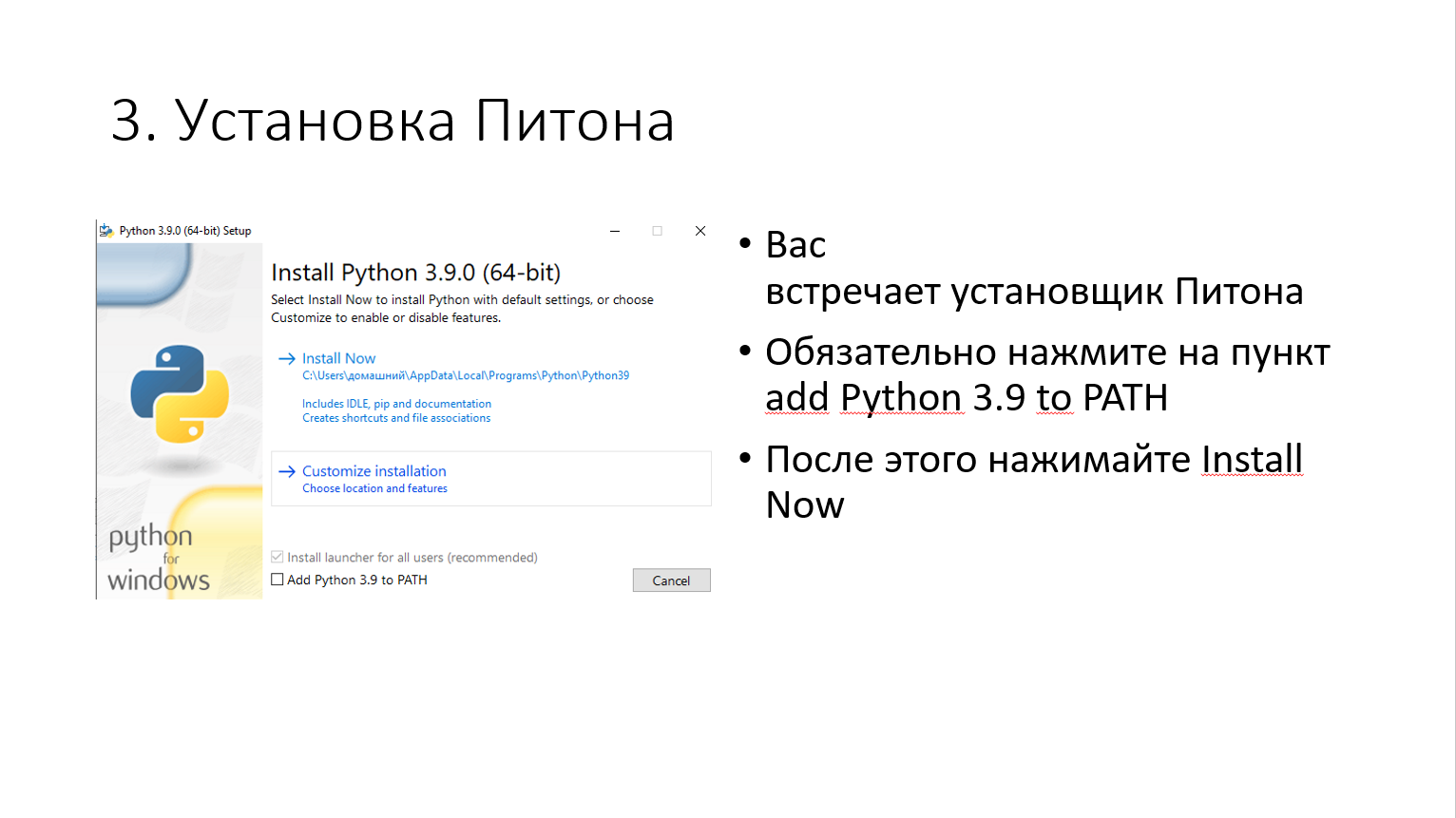


Рисунок 3. Третий этап.

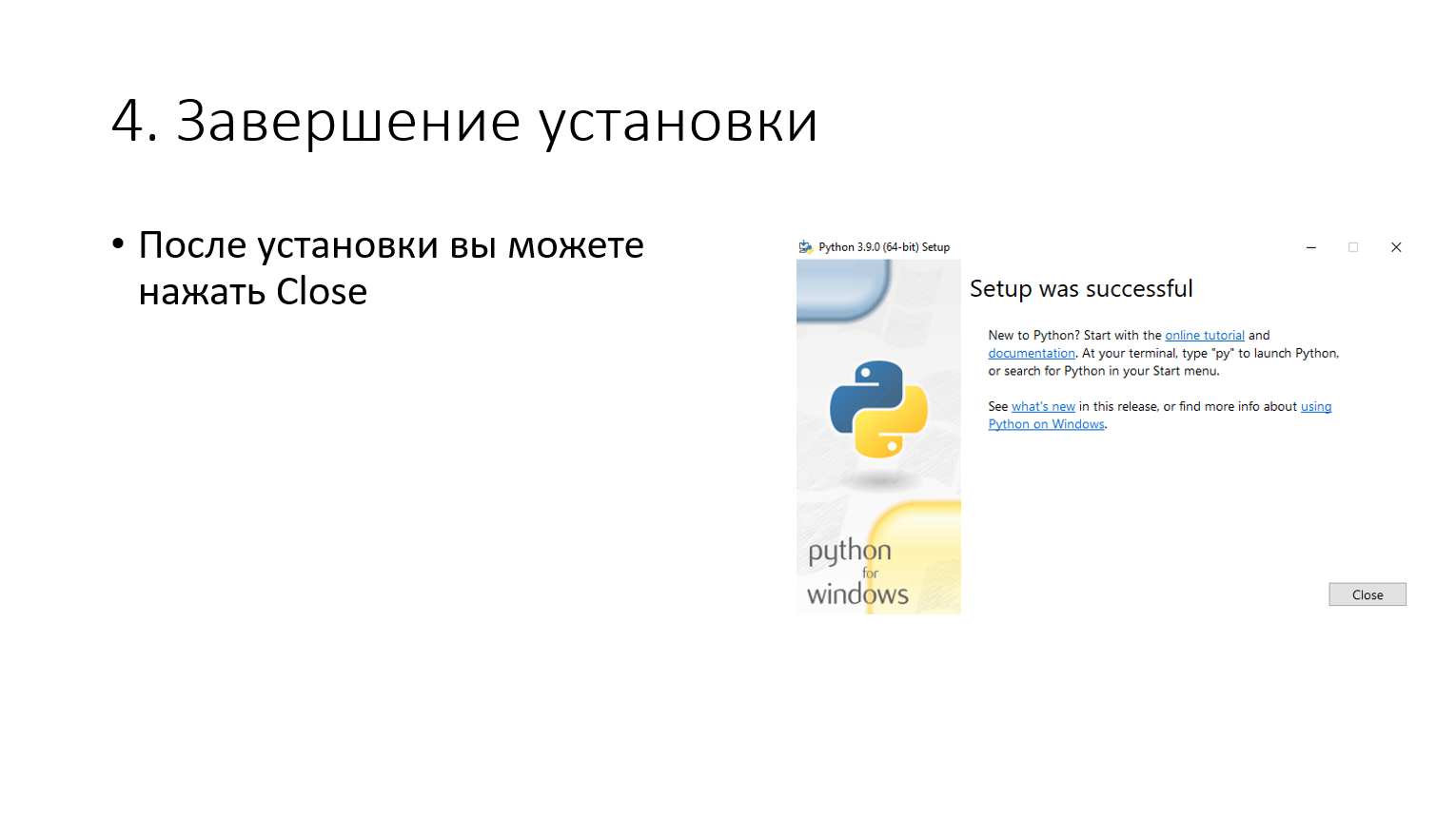


Рисунок 4. Четвертый этап.

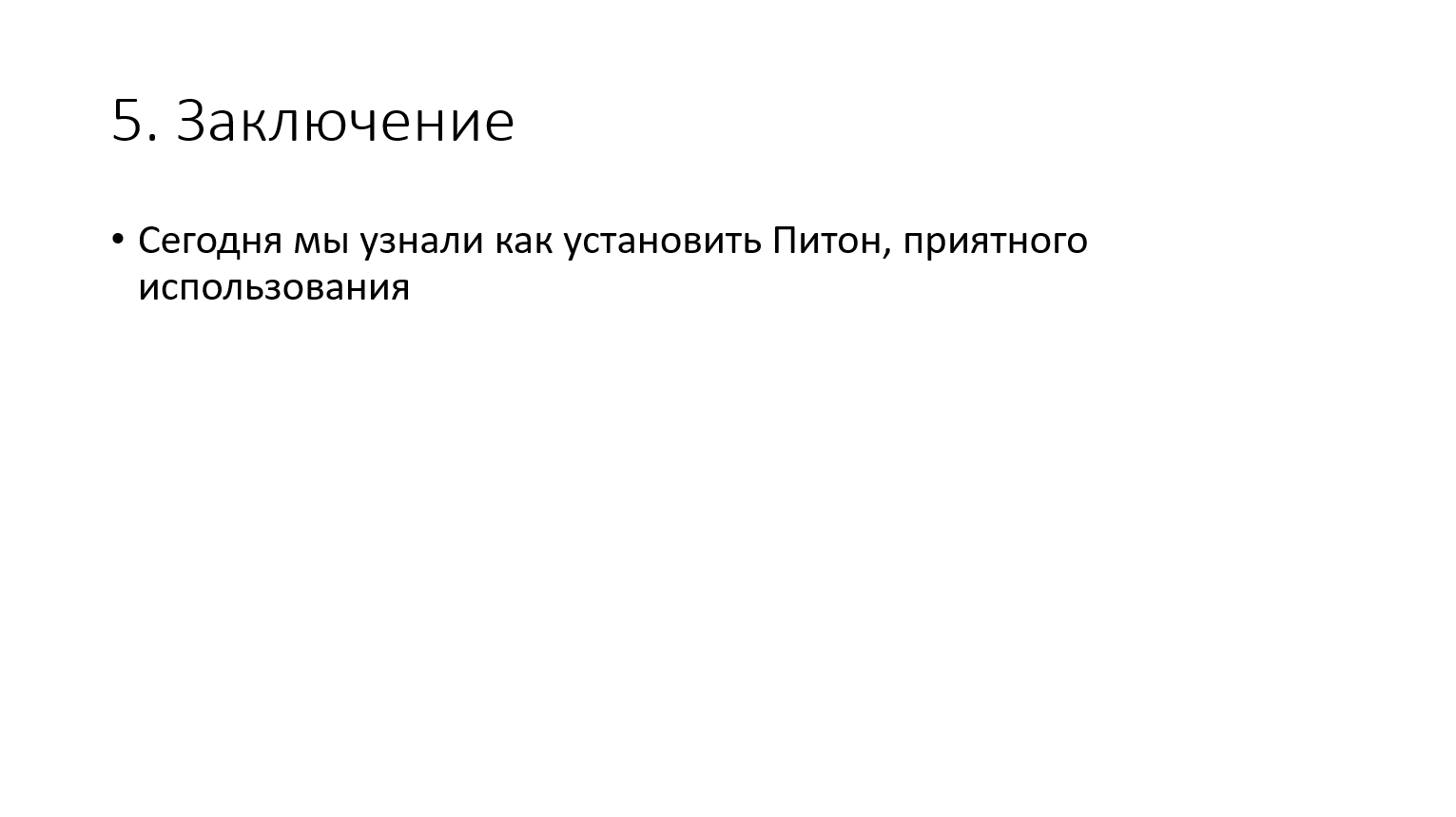


Рисунок 5. Пятый этап.

# Техника работы в командной строке и среде IDLE

Выполняя (запуская) команду “python” в вашем терминале, вы получаете интерактивную оболочку Python.



Рисунок 6. Интерактивная оболочка Python

Существует несколько способов закрыть оболочку Python:

>>> exit()

или же

>>> quit()

Кроме того, CTRL + D закроет оболочку и вернет вас в командную строку терминала.

[IDLE](https://docs.python.org/2/library/idle.html) - простой редактор для Python, который поставляется вместе с Python.

Откройте IDLE в вашей системе выбора.

В оболочке есть подсказка из трех прямоугольных скобок:

>>>

Теперь напишите в подсказке следующий код:

>>> print("Hello, World")

Нажмите Enter .

>>> print("Hello, World")

Hello, World



Рисунок 7. Первая программа

# Техника работы с линейными и разветвляющимися программами

Техника работы с циклическими программами, цикл while

**Задачи на Begin.**

**Приложения:** begin1.py, begin2.py, boolean1.py, boolean2.py, if1.py, if2.py, if3.py, case1.py, case2.py, case3.py

**Листинг 1:** (begin1.py)

#Задача:

#Известный книжный магазин "Питонист" обратился к вам как к специалисту за помощью. От вас требуется написать программу, которая принимает на вход две строки: имя автора и название книги (в такой последовательности), а затем выводит небольшое рекламное сообщение от магазина.

name = input()

book = input()

print( name, '- автор бестселлера', book, '- выпустил новую книгу! Спешите приобрести ее в магазине Питонист!')

**Листинг 2:** (begin2.py)

#Задача:

#Работник хотел написать программу, которая спрашивает у пользователя дату его рождения (день, месяц и год поочередно) и выводит в логи системы эту дату в формате YYYY - MM - DD. Помогите коллеге и исправьте ошибки.

day = input()

month = input()

year = input()

info = 'Дата рождения пользователя:'

print ( info, year, '-', month, '-', day )

**Задачи на Boolean.**

**Листинг 3:** (boolean1.py)

#Дано целое число A.

#Проверить истинность высказывания: «Число A является положительным».

x = int(input("Введите число A: "))

res = x > 0

print(res)

**Листинг 4:** (boolean2.py)

#Выполнил работу Яковлев Прокопий

#Дано целое число A.

#Проверить истинность высказывания: «Число A является нечетным».

x = int(input("Введите число A: "))

res = x % 2 == 0

print(res)

**Задачи на If**

**Листинг 5:** (if1.py)

#*Выполнил работу Яковлев Прокопий*

#Задача:

#Необходимо написать программу, выводящую кортеж, содержащий числа в полуинтервале [a; b), если a < b, и [b; a), если b < a. Числа a и b вводятся с клавиатуры.

a = int(input())

b = int(input())

c = ()

if(a>b):

a,b = b,a

c = tuple(range(a,b))

print(c)

**Листинг 6:** (if2.py)

#Выполнил работу Яковлев Прокопий

#Задача:

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

a = int(input())

b = int(input())

if a < b:

print(a)

else:

print(b)

**Листинг 7:** (if3.py)

#Выполнил работу Яковлев Прокопий

#Задача:

#Даны два целых числа. Выведите значение наибольшего из них.

a = int(input())

b = int(input())

if a > b:

print(a)

else:

print(b)

**Задачи на Case**

**Листинг 8:** (case1.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задание

#Дано целое число в диапазоне 1–7. Вывести строку — название дня недели,

#соответствующее данному числу (1 — «понедельник», 2 — «вторник» и т. д.).

x = int(input("Введите число в диапазоне 1-7: "))

if 0 < x < 8:

if x == 1:

print("Это понедельник!")

elif x == 2:

print("Это вторник!")

elif x == 3:

print("Это среда!")

elif x == 4:

print("Это четверг!")

elif x == 5:

print("Это пятница!")

elif x == 6:

print("Это суббота!")

elif x == 7:

print("Это воскресенье!")

else:

print("Число не в диапазоне 1-7!")

**Листинг 9:** (case2.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задание

#Дано целое число K. Вывести строку-описание оценки, соответствующей числу K

#(1 — «плохо», 2 — «неудовлетворительно», 3 — «удовлетворительно», 4 — «хорошо», 5 — «отлично»).

#Если K не лежит в диапазоне 1–5, то вывести строку «ошибка».

x = int(input("Введите K в диапазоне 1-5: "))

if 0 < x < 6:

if x == 1:

print("Плохо")

elif x == 2:

print("Неудовлетворительно")

elif x == 3:

print("Удовретворительно")

elif x == 4:

print("Хорошо")

elif x == 5:

print("Отлично")

else:

print("Ошибка")

**Листинг 10:** (case3.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задание

#Элементы окружности пронумерованы следующим образом:

#1 — радиус R, 2 — диаметр D = 2·R, 3 — длина L = 2·π·R, 4 — площадь круга S = π·R2.

#Дан номер одного из этих элементов и его значение.

#Вывести значения остальных элементов данной окружности (в том же порядке).

#В качестве значения π использовать 3.14.

R = float(input("Введите радиус окружности: "))

Pi = float(3.14)

a = int(input("Что хотите вывести?\n1 - Радиус(R)\n2 - Диаметр(D)\n3 - Длина окружности(L)\n4 - Площадь круга(S)\n> "))

if a == 1:

print("Радиус =", R)

elif a == 2:

D = 2 \* R

print("Диаметр =", D)

elif a == 3:

L = 2 \* Pi \* R

print("Длина окружности =", L)

elif a == 4:

S = Pi \* (R \* 2)

print("Площадь круга =", S)

# Техника работы с циклическими программами, цикл while

**Задачи на For.**

**Приложения:** for1.py, for2.py, for3.py, for4.py, for5.py, while1.py, while2.py, while3.py, while4.py, while5.py

**Листинг 11:** (for1.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задача:

#Дано несколько чисел. Вычислите их сумму.

#Сначала вводите количество чисел N, затем вводится ровно N целых чисел.

#Какое наименьшее число переменных нужно для решения этой задачи?

a = [0, int(input())]

for x in range(a[1]):

a[0] += int(input())

print(a[0])

**Листинг 12:** (for2.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задача:

#Дано N чисел: сначала вводится число N, затем вводится ровно N целых чисел.

#Подсчитайте количество нулей среди введенных чисел и выведите это количество.

#Вам нужно подсчитать количество чисел, равных нулю, а не количество цифр.

a = int(input())

b = []

c = 0

for i in range(a):

b.append(int(input()))

for i in range(a):

if b[i-1] == 0:

c += 1

print(c)

**Листинг 13:** (for3.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задача:

#Дано 10 целых чисел. Вычислите их сумму.

#Напишите программу, использующую наименьшее число переменных.

x = 0

for i in range(10):

x += int(input())

print(x)

**Листинг 14:** (for4.py)

#Выполнил работу Михайлов Данила Алексеевич

#Оформил задачу Пилипушко Андрей Сергеевич

#Задача:

#Даны два целых числа A и В.

#Выведите все числа от A до B включительно, в порядке возрастания,

#если A < B, или в порядке убывания в противном случае.

a = int(input())

b = int(input())

if a < b:

for i in range(a, b+1, 1):

print(i, end = ' ')

else:

for i in range(a, b-1, -1):

print(i, end = ' ')

**Листинг 15:** (for5.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задача:

# Для настольной игры используются карточки с номерами от 1 до N.

#Одна карточка потерялась. Найдите ее, зная номера оставшихся карточек.

#Дано число N, далее N − 1 номер оставшихся карточек (различные числа от 1 до N).

#Программа должна вывести номер потерянной карточки

n = int(input())

m = 0

for i in range(1, n+1):

m += i

for i in range(n-1):

m -= int(input())

print(m)

**Задачи на While**

**Листинг 16:** (while1.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задача:

#Определите сумму всех элементов последовательности, завершающейся числом 0.

#В этой и во всех следующих задачах числа, следующие за первым нулем,

#учитывать не нужно.

y = 0

while True:

x = int(input())

if x == 0:

break

y += x

print(y)

**Листинг 17:** (while2.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задача:

#В первый день спортсмен пробежал x километров,

#а затем он каждый день увеличивал пробег на 10% от предыдущего значения.

#По данному числу y определите номер дня,

#на который пробег спортсмена составит не менее y километров.

x, y = int(input()), int(input())

i = 0

while x < y:

x += x / 10

i += 1

print(i+1)

**Листинг 18:** (while3.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задача:

#Последовательность состоит из различных натуральных чисел и

#завершается числом 0. Определите значение второго по величине элемента в

#этой последовательности. Гарантируется, что в последовательности есть хотя бы

#два элемента.

a = []

while True:

i = int(input())

a.append(i)

if i == 0:

break

a.sort()

a.pop()

print(a.pop())

**Листинг 19:** (while4.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задание:

#Программа получает на вход последовательность целых неотрицательных чисел,

#каждое число записано в отдельной строке.

#Последовательность завершается числом 0,

#при считывании которого программа должна закончить свою работу и

#вывести количество членов последовательности (не считая завершающего числа 0).

#Числа, следующие за числом 0, считывать не нужно.

i = [1, -1]

while i[0] != 0:

i[0] = int(input())

i[1] += 1

print(i[1])

**Листинг 20:** (while5.py)

#Выполнил работу Яковлев Прокопий Максимович

#Задание

#Даны положительные числа A и B (A > B).

#На отрезке длины A размещено максимально возможное количество отрезков длины B (без наложений).

#Не используя операции умножения и деления, найти количество отрезков B, размещенных на отрезке A.

a = int(input("Введите А: "))

b = int(input("Введите B: "))

i = 1

while (a > b):

a -= b

i += 1

print("Количество возможных размещенных отрезков B -", i)

**Блок-схема пз 4 и 5**

**Блок Схема к задаче While 5:**

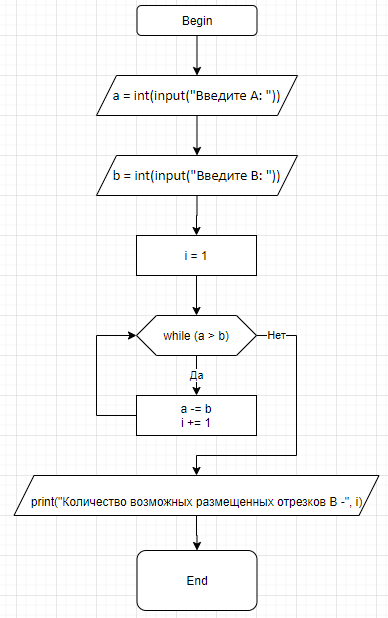


Рисунок 8. While 5

**Блок Схема к задаче While 2:**

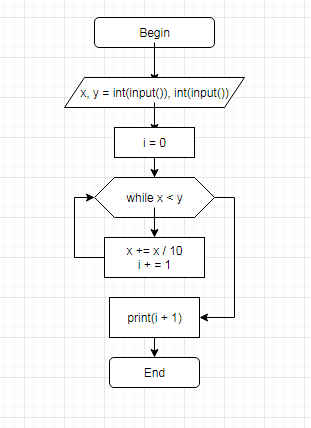


Рисунок 9. While 2

**Блок Схема к задаче boolean 1:**



Рисунок 10. Boolean 1

**Блок Схема к задаче boolean 2**:

****

Рисунок 11. Boolean 2

# Техника работы с числами

**Приложения:** Задача1.py, Задача2.py, Задача3.py, Задача4.py, Задача5.py

**Листинг 21**(Задача1.py):

#Выполнил задание Яковлев Прокопий Максимович

#Нахождение длинны отрезка по его концам

a = int(input())

b = int(input())

if a < b:

print(b - a)

elif a > b:

print(a - b)

else:

print('Длинна отрезка равняется 0')

**Листинг 22**(Задача2.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

#Нахождение произведения длин отрезков АС и ВС

a, b, c = int(input('Начало отрезка ')), int(input('Конец отрезка ')), int(input('Точка в отрезке '))

print((c - a) \* (b - c))

**Листинг 23**(Задача3.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

import math

a = int(input())

s = math.sqrt(3)/4\*a\*\*2

print(s)

**Листинг 24**(Задача4.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

#Нахождение корней квадратного уравнения

import math

a, b, c = int(input('A\*x + B\*x + C = 0\nВведите А ')), int(input('В ')), int(input('С '))

d = b\*\*2 - 4\*a\*c

if d > 0:

x = []

x.append((-b - math.sqrt(d)) / 2\*a)

x.append((-b + math.sqrt(d)) / 2\*a)

print('%.1f' % x[0], end = ' ')

print('%.1f' % x[1])

if d == 0:

x = -b / 2\*a

print(x)

if d < 0:

print('Корней нет')

**Листинг 25**(Задача5.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

#Нахождение гипотенузы и периметра прямоугольного треугольника

a, b = int(input('Длинна катета а ')), int(input('Длинна катета b '))

print(a + b + (a\*\*2 + b\*\*2)\*(1/2))

# Техника работы со строками

**Приложения:** Задача1.py, Задача2.py, Задача3.py, Задача4.py

**Листинг 26**(Задача1.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

#Заменить пробелы в строке на символы нижнего подчеркивания, чтобы строка

#могла сгодиться для логина.

#Если строка состоит из одного слова, менять ничего не нужно.

print(input().replace(' ', '\_'))

**Листинг 27**(Задача2.py)**:**

#Задание выполнил Яковлев Прокопий Максимович

#Дана строка, состоящая из слов, разделенных пробелами.

#Определите, сколько в ней слов.

print(input().count(' ')+1)

**Листинг 28**(Задача3.py)**:**

#Задание выполнил Яковлев Прокопий Максимович

#Дана строка.

#Сначала выведите третий символ этой строки.

#Во второй строке выведите предпоследний символ этой строки.

#В третьей строке выведите первые пять символов этой строки.

#В четвертой строке выведите всю строку, кроме последних двух символов.

#В пятой строке выведите все символы с четными индексами (считая, что индексация начинается с 0, поэтому символы выводятся начиная с первого).

#В шестой строке выведите все символы с нечетными индексами, то есть начиная со второго символа строки.

#В седьмой строке выведите все символы в обратном порядке.

#В восьмой строке выведите все символы строки через один в обратном порядке, начиная с последнего.

#В девятой строке выведите длину данной строки.

s = input()

print(s[2])

print(s[-2])

print(s[:5])

print(s[:-2])

print(s[::2])

print(s[1::2])

print(s[::-1])

print(s[::-2])

print(len(s))

**Листинг 29**(Задача4.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

#https://stepik.org/lesson/201702/step/9?unit=175778

#Уберите точки из введенного IP-адреса.

#Выведите сначала четыре числа через пробел, а затем сумму получившихся чисел.

a = input().split('.')

b = 0

for i in a:

print(int(i), end=' ')

b += int(i)

print('')

print(b)

# Техника работы со списками

**Приложения:** Задача1.py, Задача2.py, Задача3.py

**Листинг 30**(Задача1.py)**:**

#https://pythontutor.ru/lessons/lists/problems/even\_elements

#Выполнил задание Яковлев Прокопий Максимович

#Выведите все четные элементы списка.

#При этом используйте цикл for, перебирающий элементы списка, а не их индексы!

for num in input().split(' '):

if int(num) % 2 == 0:

print(int(num), end=' ')

**Листинг 31**(Задача2.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

#pythontutor.ru

#https://pythontutor.ru/lessons/lists/problems/more\_than\_neighbours/

#Дан список чисел. Определите, сколько в этом списке элементов,

#которые больше двух своих соседей, и выведите количество таких элементов.

#Крайние элементы списка никогда не учитываются, поскольку у них

#недостаточно соседей.

a = [int(n) for n in input().split(' ')]

sch = 0

for i in range(len(a)):

try:

if a[i] > a[i-1] and a[i] > a[i+1] and i != 0 and i != len(a):

sch += 1

except IndexError:

None

print(sch)

**Листинг 32**(Задача3.py)**:**

#https://pythontutor.ru/lessons/lists/problems/lineup/

#pythontutor.ru

#Выполнил задание Яковлев Прокопий Максимович

#Петя перешёл в другую школу.

#На уроке физкультуры ему понадобилось определить своё место в строю.

#Помогите ему это сделать.

#Программа получает на вход невозрастающую последовательность натуральных чисел,

#означающих рост каждого человека в строю.

#После этого вводится число X – рост Пети.

#Все числа во входных данных натуральные и не превышают 200.

#Выведите номер, под которым Петя должен встать в строй.

#Если в строю есть люди с одинаковым ростом, таким же, как у Пети,

#то он должен встать после них.

a = [int(n) for n in input().split(' ')]

p = int(input())

for i in range(len(a)):

if a[i] < p:

p = ''

print(i+1)

break

if p != '':

print(len(a) + 1)

# Техника работы с циклом for и генераторами списков

**Приложения:** Задача1.py, Задача2.py, Задача3.py, Задача4.py

**Листинг 33**(Задача1.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

from random import randint

x1 = randint(2, 10)

s = []

for i in range(x1):

r = randint(1,100)

s.append(r)

print(s)

**Листинг 34**(Задача2.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

#программа которая выбирает случайные(пронумерованные по сложности задания )

#и создает 7 билетов в которых по 5 заданий

from random import \*

s = ['a1', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7', 'a8', 'a9', 'a10', 'b11', 'b12', 'b13', 'b14', 'b15', 'b16', 'c17',

'c18', 'c19', 'c20']

b = []

gb = []

for i in range(7):

for x in range(5):

e = randint(0,19)

b.append(s[e])

gb.append(billet)

b = []

print(gb)

**Листинг 35**(Задача3.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

#Генерирует случайное числа в диапазоне

from random import \*

a = randint(1,20)

d = [i for i in range(a)]

print(d)

**Листинг 36**(Задача4.py)**:**

#Выполнил задание Яковлев Прокопий Максимович

#генерирует случайное число. Если i делится на 3 без остатка, то пропускает его

from random import \*

a = randint(0, 150)

d = [i\*\*2 for i in range(a) if i % 3 != 0]

print(d)

# Техника работы с функциями

**Приложения:** func1.py, func2.py

**Листинг 37**(func1.py)**:**

# http://ptaskbook.com/ru/tasks/func.php

# Выполнил задание Яковлев Прокопий Максимович

'''

Описать функцию SumRange(A, B) целого типа, находящую сумму всех целых

чисел от A до B включительно (A и B — целые).

Если A > B, то функция возвращает 0.

С помощью этой функции найти суммы чисел от A до B и от B до C,

если даны числа A, B, C.

'''

def SumRange(x, y):

s = 0

if x < y:

for i in range(x, y+1):

s += i

return s

elif x > y:

return 0

a, b, c = int(input()), int(input()), int(input())

print(SumRange(a, b))

print(SumRange(b, c))

**Листинг 38**(func2.py)**:**

# http://ptaskbook.com/ru/tasks/func.php

# Выполнил задание Яковлев Прокопий Максимович

'''

Описать функцию IsSquare(K) логического типа, возвращающую True,

если целый параметр K (> 0) является квадратом некоторого целого числа, и False

в противном случае. С ее помощью найти количество квадратов в наборе из 10 целых

положительных чисел.

'''

import math

def IsSquare(x):

if math.sqrt(x).is\_integer():

return True

else:

return False

a = [int(n) for n in input('Введите 10 чисел через пробел: ').split() if IsSquare(int(n))]

print(\*a)

* 1. Техника работы со словарями

**Приложения:** dict1.py, dict2.py, dict3.py

**Листинг 39**(dict1.py)**:**

#Есть имена учеников и номер их зачетной книжки, нужно занести все в словарь

n = ['Иван', 'Сергей']

num = [1,2]

a = {}

for i in range(len(n)):

a.update({num[i]:n[i]})

print(a)

**Листинг 40**(dict2.py)**:**

# Выполнил задание Яковлев Прокопий Максимович

'''

Условие. В единственной строке записан текст. Для каждого слова из данного текста

подсчитайте, сколько раз оно встречалось в этом тексте ранее.

Словом считается последовательность непробельных символов идущих подряд, слова разделены

одним пробелом или символами конца строки.

'''

a = {}

sch = 1

for i in input().split():

if i not in a:

a.update({i:sch})

else:

a.update({i:a[i] + 1})

print(a)

**Листинг 41**(dict3.py)**:**

# https://pythontutor.ru/lessons/sets/problems/number\_of\_words/

# Выполнил задание Яковлев Прокопий Максимович

'''

Условие. Дан текст: в первой строке задано число строк, далее идут сами строки.

Выведите слово, которое в этом тексте встречается чаще всего. Если таких слов несколько,

выведите все.

'''

a = {}

for i in range(int(input())):

st = input()

if st not in a:

a.update({st:1})

else:

a.update({st:a[st] + 1})

sch = max(a.values())

for k, v in a.items():

if v == sch:

print(k + ':', v)

# Техника работы с множествами

**Приложения:** задание1.py, задание2.py

**Листинг 42**(задание1.py)**:**

# https://pythontutor.ru/lessons/sets/problems/number\_of\_unique/

# Выполнил задание Яковлев Прокопий Максимович

'''

Задача «Количество различных чисел»

Условие. Дан список чисел. Определите,

сколько в нем встречается различных чисел.

'''

print(len(set(input().split())))

**Листинг 43**(задание2.py)**:**

# https://pythontutor.ru/lessons/sets/problems/number\_of\_coincidental/

# Выполнил задание Яковлев Прокопий Максимович

'''

Даны два списка чисел. Посчитайте, сколько чисел содержится одновременно как

в первом списке, так и во втором.

'''

print(len(set(input().split()) & set(input().split())))

# Техника работы с кортежами

**Приложения:** задание1.py, задание2.py

**Листинг 44**(задание1.py)**:**

# https://stepik.org/lesson/193753/step/4?unit=168148

# Выполнил задание Яковлев Прокопий Максимович

'''

Вывести чётные

Необходимо вывести все четные числа на отрезке [a; a \* 10].

'''

a = int(input())

[print(i) for i in range(a, a\*10+1) if i % 2 == 0]

**Листинг 45**(задание2.py)**:**

# https://stepik.org/lesson/193753/step/5?unit=168148

# Выполнил задание Яковлев Прокопий Максимович

'''

С клавиатуры вводятся целые числа a > b. Выведите убывающую последовательность чисел

по одному числу в строке.

'''

a, b = int(input()), int(input())

for i in range(b, a-1, -1):

print(i)

# Техника работы с файлами

**Приложения:** text1.py, text2.py, text3.py, text4.py, text5.py

**Листинг 46**(text1.py)**:**

# http://ptaskbook.com/ru/tasks/text.php

# Выполнил задание Яковлев Прокопий Максимович

'''

Дана строка S и текстовый файл. Добавить строку S в конец файла

'''

f = open("text1.txt", "r")

a = f.read()

a += 'S'

f.close()

f = open("text1.txt", "w")

f.write(a)

f.close()

**Листинг 47**(text2.py)**:**

# http://ptaskbook.com/ru/tasks/text.php

# Выполнил задание Яковлев Прокопий Максимович

'''

Дана строка S и текстовый файл. Заменить в файле все пустые строки на строку S.

'''

f = open("text2.txt", "r")

a = ''

for i in f.read():

for j in i:

if j == ' ':

j = 'S'

a += j

f.close()

f = open("text2.txt", "w")

f.write(a)

f.close()

**Листинг 48**(text3.py)**:**

# http://ptaskbook.com/ru/tasks/text.php

# Выполнил задание Яковлев Прокопий Максимович

'''

Дан текстовый файл. Заменить в нем все подряд идущие пробелы на один пробел.

'''

f = open("text3.txt", "r")

a = ''

sch = 0

for i in f.read():

for j in i:

if sch == 1 and j == ' ':

j = ''

elif j != ' ':

sch = 0

if j == ' ':

sch = 1

a += j

f.close()

f = open("text3.txt", "w")

f.write(a)

f.close()

**Листинг 49**(text4.py)**:**

# http://ptaskbook.com/ru/tasks/text.php

# Выполнил задание Яковлев Прокопий Максимович

'''

Дан текстовый файл, каждая строка которого изображает целое число, дополненное слева и справа

несколькими пробелами. Вывести количество этих чисел и их сумму.

'''

f = open("text4.txt", "r")

a = [n for n in f.read()]

b = [int(i) for i in a if i != ' ' and i != '\n']

print(len(b), sum(b))

f.close()

**Листинг 50**(text5.py)**:**

# http://ptaskbook.com/ru/tasks/text.php

# Выполнил задание Яковлев Прокопий Максимович

'''

Дан текстовый файл. Создать символьный файл, содержащий все знаки препинания,

встретившиеся в текстовом файле (в том же порядке).

'''

f = open("text5.txt", "r")

a = [[j for j in i if j.islower() == 0 and j.isdigit() == 0 and j.isalpha() == 0 and j.isupper() == 0 and j != ' ' and j != '\n'] for i in f.read()]

[print(\*i) for i in a if i]

f.close()

# Техника работы с модулями

**Приложения:** 15-1.1.py, 15-1.2.py, 15-2.py, 15-4.py

**Листинг 51(15-1.1.py):**

# Выполнил задание Яковлев Прокопий Максимович

# Контейнерные типы данных модуля collections.

# https://docs-python.ru/standart-library/modul-collections-python/

# Класс deque() модуля collections в Python.

# https://docs-python.ru/standart-library/modul-collections-python/klass-deque-modulja-collections/

# Двусторонняя очередь в Python.

# Класс collections.deque() это обобщение стеков и очередей и представляет собой двустороннюю очередь.

# Двусторонняя очередь deque() поддерживает поточно-ориентированные, эффективные по памяти операции добавления

# и извлечения элементов последовательности с любой стороны с примерно одинаковой производительностью O(1)

# в любом направлении.

#

# Списки поддерживают аналогичные операции, но они оптимизирован только для быстрых операций с последовательностями

# фиксированной длины и требуют затрат O(n) на перемещение памяти для операций pop(0) и insert(0, v), которые изменяют

# как размер, так и положение базового представления данных.

from collections import deque

d = deque(["One", "Two", "Three"])

print(d)

d.append("Four")# Добавим вправо

print(d)

d.appendleft("Zero")# Добавим влево

print(d)

d.pop()# Убираем справа

print(d)

d.popleft()# Убираем слева

print(d)

d.reverse()# Разворот

print(d)

d = deque([], maxlen=3)# Ограничение размера

d.append(1)

print(d)

d.append(12)

print(d)

d.append(123)

print(d)

d.append(1234)

print(d)

**Листинг 52(15-1.2.py):**

# Выполнил задание Яковлев Прокопий Максимович

# Контейнерные типы данных модуля collections.

# Класс Counter() модуля collections в Python.

# https://docs-python.ru/standart-library/modul-collections-python/klass-counter-modulja-collections/

# Подсчет количества повторений элементов в последовательности.

# класс collections.Counter() предназначен для удобных и быстрых подсчетов количества появлений неизменяемых

# элементов в последовательностях.

#

# from collections import Counter

# c = Counter(['a', 'b', 'a', 'c', 'b', 'a'])

# print(dict(c))

# Вывод => {'a': 3, 'b': 2, 'c': 1}

import collections

c = (['a', 'b', 'a', 'c', 'c', 'b', 'a', 'c', 'a', 'a', 'b'])

print(c)# Вывод списка

print(collections.Counter(c))# Вывод кол-ва каждых элементов в списке

**Листинг 53(15-2.py):**

# Выполнил задание Яковлев Прокопий Максимович

# Варианты использования defaultdict

from collections import defaultdict

a = [('a', 1),

('b', 2),

('c', 3),

('a', 4),

('c', 5)]

b = {}

for k, v in a:

b.setdefault(k, []).append(v)

sorted(b.items())

print(a)

print(b)

print()###

s = [('a', 1),

('b', 2),

('b', 3),

('a', 4),

('a', 5),

('b', 6)]

d = defaultdict(set)

for k, v in s:

d[k].add(v)

sorted(d.items())

print(d)

print()###

def c(a):

return lambda: a

d = defaultdict(c('<missing>'))

d.update(name='Micke', action='kick')

print(d)

**Листинг 54(15-4.py):**

# Выполнил задание Яковлев Прокопий Максимович

#

# Работа с файлами в Python с помощью модуля OS

#

# Задание 1. Вывод текущей директории

# Задание 2. Создание папки

# Задание 3. Изменение директории

# Задание 4. Создание вложенных папок

# Задание 5. Создание файлов

# Задание 6. Переименование файлов

# Задание 7. Перемещение файлов

# Задание 8. Список файлов и директорий

# Задание 9. Удаление файлов

# Задание 10. Удаление директорий

# Задание 11. Получение информации о файлах

import os # Библиотека для работы с файлами

import shutil

# Задание 1. Вывод текущей директории

print('Задание 1.')

print('Текущая деректория:', os.getcwd())

# Задание 2. Создание папки

print('Задание 2.')

fname = input('Введите имя файла: ')

'''

if not os.path.isdir(name):

os.mkdir(name)

'''

# Задание 3. Изменение директории

print('Задание 3.')

chname = input('Введите имя файла для изминения деректории: ')

os.chdir(chname)

print('Текущая директория изменилась на <', chname, '> :', os.getcwd())

# Задание 4. Создание вложенных папок

print('Задание 4.')

nfile = input('Введите имя конечного файла: ')

os.makedirs(nfile)

# Задание 5. Создание файлов

print('Задание 5.')

createFileName = input('Введите имя создоваемого файла с форматом: ')

textFile = open(createFileName, 'w') # Создать новый текстовый файл

createText = input('Введите содержание файла: ')

textFile.write(createText) # запить текста в этот файл

# Задание 6. Переименование файлов

print('Задание 6.')

os.rename('name.txt', 'rename.txt')

# Задание 7. Перемещение файлов

print('Задание 7.')

shutil.move('Путь до файла', 'Новый путь до файла')

# Задание 8. Список файлов и директорий

print('Задание 8.')

os.listdir('Путь до папки')

# Задание 9. Удаление файлов

print('Задание 9.')

os.remove('Путь до файла')

# Задание 10. Удаление директорий

print('Задание 10.')

os.rmdir('Путь до файла')

# Задание 11. Получение информации о файлах

print('Задание 11.')

print(os.stat('name.txt'))

# Техника работы с классами

**Приложения:** K16\_1.py, K16\_2.py

**Листинг 55** (K16\_1.py):

#Выполнил: Яковлев Прокопий Максимович

#Группа: П2-18

'''

К16\_1. Техника работы с классами.

Задание 1. Создание класса

Задание 2. Создание объекта

Задание 3. Функция init

Задание 4. Методы объектов

Задание 5. Параметр self

Задание 6. Изменение свойств объекта

Задание 7. Удалить свойства объекта

Задание 8. Удаление объектов

'''

#Создадим класс

class BruhClass:

x = 20

#Создадим объект

p1 = BruhClass()

print(p1.x)

#Функция init

class DataHero:

#Параметр self - ссылка на сам класс. Он должен быть первым.

#Вы можете его назвать по другому. Например bruh (метод DataHero).

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

#Методы класса

def Welcome(self):

print("Добро пожаловать, в долину силы, мой юный друг")

def DataHero(bruh):

print(f"-------------------")

print(f"Имя: {bruh.name}")

print(f"Возраст: {bruh.age}")

print(f"-------------------")

print()

hero1 = DataHero("Владимир", 20)

hero1.Welcome()

hero1.DataHero()

#Свойство объекта можно менять

hero1.age = 9999

hero1.DataHero()

#Свойства объектов можно удалять

del hero1.age

#Также можно удалять сам объект

del hero1

**Листинг 56** (K16\_2.py):

#Выполнил: Яковлев Прокопий Максимович

#Группа: П2-18

'''

К16\_2. Техника работы с классами.

Задание 1. Создание классов

Задание 2. Создание экземпляров класса

Задание 3. Доступ к атрибутам

Задание 4. Встроенные атрибуты класса

Задание 5. Уничтожение объектов (сбор мусора)

'''

#1. Создание классов

class DataHero:

"""Функции героя, статистика и другое о нем"""

obj\_count = 0

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

DataHero.obj\_count += 1

#Методы класса

#Количество объектов класса

def ActCount(self):

print(f"Количество объектов класса: {DataHero.obj\_count}")

#Приветствие

def Welcome(self):

print(f"Добро пожаловать, в долину силы, мой юный друг {self.name}")

#Данные героя

def DataHero(bruh):

print(f"-------------------")

print(f"Имя: {bruh.name}")

print(f"Возраст: {bruh.age}")

print(f"-------------------")

print()

#----------------------------------

#2. Создание экземпляров класса

hero1 = DataHero("Владимир", 20) #1ый объект класса DataHero

hero2 = DataHero("icefanik", 33) #2ой объект класса DataHero

#Документация класса

print(hero1.\_\_doc\_\_)

print(DataHero.\_\_doc\_\_)

#----------------------------------

#3. Доступ к атрибутам

hero1.Welcome()

hero1.DataHero()

print(f"Количество героев: {DataHero.obj\_count}")

#Свойство объекта можно менять

hero1.age = 9999

hero1.DataHero()

#Свойства объектов можно удалять

del hero2.age

#Также можно удалять сам объект

del hero2

print(hasattr(hero1, 'age')) #Возвращает True, если атрибут 'age' существует

print(getattr(hero1, 'age')) #Возвращает значение атрибута 'age'

delattr(hero1, 'age') #Удаляет атрибут 'age'

setattr(hero1, 'age', 8) #Устанавливает атрибут 'age' на 8

print()

#----------------------------------

#4. Встроенные атрибуты класса

#По объекту класса

print("#По объекту класса")

print(f"hero1.\_\_doc\_\_: {hero1.\_\_doc\_\_}") #Документация класса

print(f"hero1.\_\_dict\_\_: {hero1.\_\_dict\_\_}") #Словарь, содержащий пространство имен класса.

print()

#По классу

print("#По классу")

print(f"DataHero.\_\_doc\_\_: {DataHero.\_\_doc\_\_}") #Документация класса

print(f"DataHero.\_\_name\_\_: {DataHero.\_\_name\_\_}") #Наименование класса

print(f"DataHero.\_\_module\_\_: {DataHero.\_\_module\_\_}") #Имя модуля, в котором определяется класс. Этот атрибут \_\_main\_\_ в интерактивном режиме.

print(f"hero1.\_\_bases\_\_: {DataHero.\_\_bases\_\_}") #Могут быть пустые tuple, содержащие базовые классы, в порядке их появления в списке базового класса.

print(f"hero1.\_\_dict\_\_: {DataHero.\_\_dict\_\_}") #Словарь, содержащий пространство имен класса.

print()

#----------------------------------

#5. Уничтожение объектов (сбор мусора)

class Point:

def \_\_init\_\_(self, x=0, y=0):

self.x = x

self.y = y

def \_\_del\_\_(self):

class\_name = self.\_\_class\_\_.\_\_name\_\_

print(f"Класс {class\_name} уничтожен")

ptr1 = Point()

print(f"id\_ptr1 = {id(ptr1)}, ptr1 = {ptr1.x} {ptr1.y}")

ptr1 = Point(1, 2)

print(f"id\_ptr1 = {id(ptr1)}, ptr1 = {ptr1.x} {ptr1.y}")

ptr2 = ptr1

ptr3 = ptr1

print(id(ptr1), id(ptr2), id(ptr3))

del ptr1

del ptr2

del ptr3

# **Раздел 2. Техника решения задач с использованием библиотек**

## Установка и настройка среды JetBrains PyCharm

PyCharm - это кросс-платформенная среда разработки.

Переходим на страницу скачивания PyCharm <https://www.jetbrains.com/ru-ru/pycharm/>. Для скачивания доступно две версии: профессиональная и версия для сообщества. Версия для сообщества бесплатная. Ее и скачаем.

Запускаем скачанный .exe файл. В первом окне нас приветствует сам установщик. Смело кликаем «Next»:



Рисунок 12. Установщик PyCharm



Рисунок 13. Место установки PyCharm



Рисунок 14. Процесс установки PyCharm



Рисунок 15. Стартовое окно PyCharm

В следующем окне будет два поля. В первом укажите, где будет располагаться новый проект. Второе поле должно быть заполненным помолчанию. Там содержится путь к установленному ранее Python 

Рисунок 16. Выбор места для нового проекта и выбор интерпретатора

Дальше откроется окно самого редактора.

Теперь вы готовы начать писать программы на Python!

Кликните правой кнопкой на название вашего проекта и в раскрывающемся списке выберите пункт «New > Python file»: 

Рисунок 17. Создание нового Phython файла

Появится окно, в котором вы можете задать имя файлу. Задайте какое-нибудь имя и нажмите «OK».

Справа откроется сам файл. Пока он пустой. Пишем:



Рисунок 18. Написание программы

Теперь нужно запустить нашу небольшую (1 строка) программу. Для этого в верхнем меню перейдите в «Run > Run…».



Рисунок 19. Запуск программы

В появившемся окне щелкните по названию вашего файла. Теперь программа запустится. В нижней части редактора должна появиться консоль с результатом



Рисунок 20. Выполнение программы

# Техника работы с базами данных

**Приложения:** SQLite.py

Python SQLite поставляется вместе с любой установленной версией Python, поэтому нет необходимости устанавливать SQLite с помощью pip. Для возможности импортировать модуль SQLite3, нужно использовать Python3.

**Листинг** **57** (SQLite.py):

import sqlite3

# Функция для таблицы в базе данных

def Create():

cursor.execute("""CREATE TABLE albums(Name text, Integer int)""")

conn.commit()

# Ввод значений

def Input(text):

cursor.executemany("INSERT INTO albums VALUES (?,?)", text)

conn.commit()

# Удаление по названиям

def Delete(n):

sql = "DELETE FROM albums WHERE Name = ?"

cursor.execute(sql, (n, ))

conn.commit()

# Создание базы данных

conn = sqlite3.connect('Base')

cursor = conn.cursor()

Create()

# Цикл выполнения запросов

while True:

text\_a = input().split()

# Команда для остановки цикла

if text\_a == ['stop']:

break

elif len(text\_a) == 1:

# Проверка на нахождение имени в базе данных

info = cursor.execute('SELECT \* FROM albums WHERE Name = ?', (text\_a[0], )).fetchall()

if len(info):

Delete(text\_a[0])

print(text\_a[0] + ' deleted')

continue

else:

print('Error')

continue

text\_a = [tuple(text\_a)]

Input(text\_a)

После завершения кода в той же папке где находится python-file появится файл с БД и в ней должна появиться запись которую вы напишете в консоль. Просто так открыть БД и посмотреть нельзя.

Для того чтобы увидеть БД, необходимо будет скачать DB Browser (SQLite). Открываем страницу скачивания программы <https://sqlitebrowser.org/dl/>.



Рисунок 21. Установщик DB Browser

После установки, открываем DB Browser для работы с ним.

Рисунок 22. Окно DB Browser

В открывшееся окно DB Browser’a перетаскиваем файл с БД.

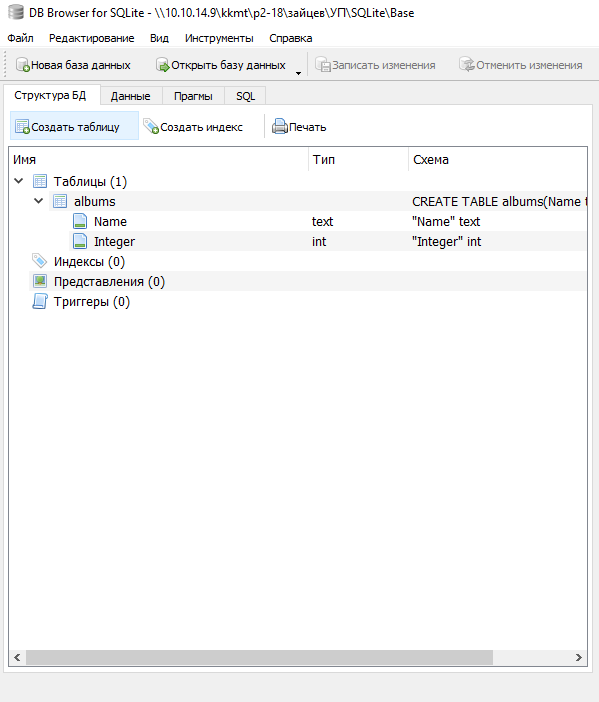


Рисунок 23. Таблица Name и Integer в DB Browser

Чтобы просмотреть содержимое БД необходимо нажать на кнопку «Данные».

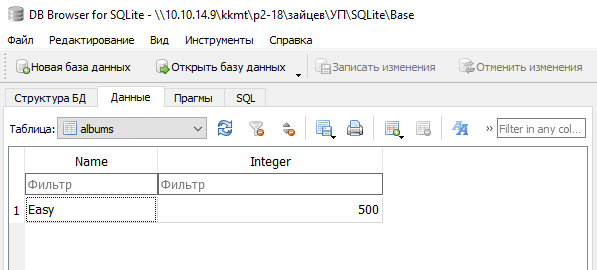


Рисунок 24. Данные таблицы students

Видно, что в таблицу занеслись корректные данные.

# Техника работы с библиотекой tkinter

**Приложения:** Tkinter.py

Библиотека Tkinter установлена в Python в качестве стандартного модуля, поэтому нам не нужно устанавливать что-либо для его использования. Чтобы импортировать его, пропишем import tkinter. В качестве примера для демонстрации tkinter будет приведена программа, которая принимает данные из формы и добавляет их в текстовый файл, также в программе есть возможность очищать весь файл полностью.

**Листинг 58 (**Tkinter.py):

from tkinter import \*

def main():

def mainwindow1():

def add1():

box1.insert(END, en1.get() + ' ' + en2.get())

en1.delete(0, END)

en2.delete(0, END)

def change1():

select = list(box1.curselection())

select.reverse()

for i in select:

box1.delete(i)

box1.insert(i, en1.get() + ' ' + en2.get())

en1.delete(0, END)

en2.delete(0, END)

def remove1():

select = list(box1.curselection())

select.reverse()

for i in select:

box1.delete(i)

def save1():

f = open('prof.txt', 'w')

f.writelines("\n".join(box1.get(0, END)))

f.close()

# Создание объекта окна верхнего уровня (на нём будут распологаться все элементы

root = Tk()

# Задание размера окна и его координат расположения

root.geometry("300x230+700+300")

root["bg"] = "gray22"

# Задаём местоположение текстовой метке и полю для ввода

lb1 = Label(root,text="Код должности:", background="gray22", foreground="#ccc")

lb1.place(x=2, y=16)

en1 = Entry(root, width=15)

en1.pack()

en1.place(x=100, y=18)

# Задаём местоположение текстовой метке и полю для ввода

lb2 = Label(root,text="Название:", background="gray22", foreground="#ccc")

lb2.place(x=15, y=48)

en2 = Entry(root, width=15)

en2.pack()

en2.place(x=100, y=50)

lb3 = Label(root,text="База данных:", background="gray22", foreground="#ccc")

lb3.place(x=15, y=130)

box1 = Listbox(root, selectmode=EXTENDED, width=15, height=8)

box1.place(x=100, y=80)

scroll1 = Scrollbar(root, command=box1.yview)

scroll1.pack(side=RIGHT, fill=Y)

box1.config(yscrollcommand=scroll1.set)

b1 = Button(root, text="Добавить",background="#555", foreground="#ccc", width=10, height=1, command=add1)

b1.place(x=200, y=87)

b2 = Button(root, text="Изменить",background="#555", foreground="#ccc", width=10, height=1, command=change1)

b2.place(x=200, y=117)

b3 = Button(root, text="Удалить",background="#555", foreground="#ccc", width=10, height=1, command=remove1)

b3.place(x=200, y=147)

b4 = Button(root, text="Сохранить",background="#555", foreground="#ccc", width=10, height=1, command=save1)

b4.place(x=200, y=177)

with open('prof.txt', 'r') as f:

lst = f.readlines()

for item in lst:

box1.insert(END, item)

# Создание объекта окна верхнего уровня (на нём будут распологаться все элементы

mainwindow = Tk()

# Задание размера окна и его координат расположения

mainwindow.geometry("300x230+700+300")

mainwindow["bg"] = "gray22"

mainmenu = Menu(mainwindow)

mainwindow.config(menu=mainmenu)

helpmenu = Menu(mainmenu, tearoff=0)

helpmenu2 = Menu(helpmenu, tearoff=0)

mainmenu.add\_cascade(label="О программе",menu=helpmenu)

helpmenu.add\_cascade(label="Справка",menu=helpmenu2)

helpmenu2.add\_command(label="Открыть сайт",)

helpmenu.add\_separator()

helpmenu.add\_command(label="Выход",)

lb1 = Label(mainwindow, text="Таблицы", background="gray22", foreground="#ccc")

lb1.place(x=128, y=10)

b1 = Button(mainwindow, text="Должности", background="#555", foreground="#ccc", width=15, height=1, command=mainwindow1)

b1.place(x=100, y=50)

# Запуск цикла обработки событий

mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

main()

Форма имеет вид:

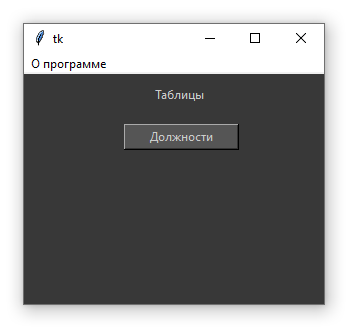


Рисунок 25. Вид формы

Жмем на кнопку “Должности” и заполняем форму данными:

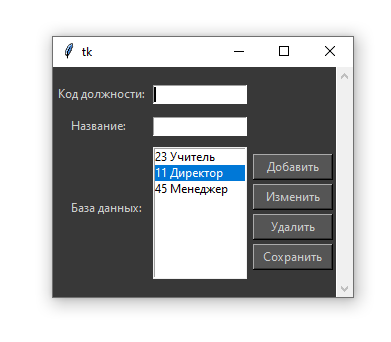


Рисунок 26. Заполненная форма

Нажмём кнопку “Сохранить”. В текущем каталоге с python-файлом появится текстовик, в котором и сохранятся наши данные.

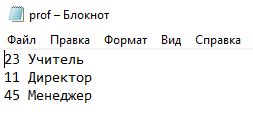


Рисунок 27. Вывод в файл

# Техника работы с библиотекой NumPy

**Приложения:** primer.py

NumPy это open-source модуль для python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций.

Установка:

• Через терминал командой: “pip install numpy”

• По инструкции на сайте <https://www.scipy.org/install.html>

Импорт модуля numpy:

import numpy

Главной особенностью numpy является объект array. Массивы схожи со списками в python, исключая тот факт, что элементы массива должны иметь одинаковый тип данных, как float и int. С массивами можно проводить числовые операции с большим объемом информации в разы быстрее и, главное, намного эффективнее чем со списками.

**Листинг 59** (Numpy.py):

import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(-5, 5, 100)

'''возвращает одномерный массив из указанного количества элементов.'''

def sigmoid(alpha):

return 1 / ( 1 + np.exp(- alpha \* x) )

'''Сигмоидная функция.'''

dpi = 80

fig = plt.figure(dpi = dpi, figsize = (512 / dpi, 384 / dpi) )

'''Функция указывающая размеры графика.'''

plt.plot(x, sigmoid(0.5), 'ro-')

plt.plot(x, sigmoid(1.0), 'go-')

plt.plot(x, sigmoid(2.0), 'bo-')

'''Переменная указывающася цвет.'''

plt.legend(['A = 0.5', 'A = 1.0', 'A = 2.0'], loc = 'upper left')

'''Легенда в верхнем правом углу.'''

fig.savefig('sigmoid.png')

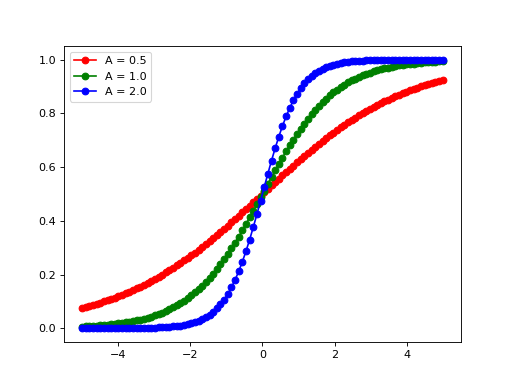
'''Сохранение PNG-картинки.'''

Рисунок 28. Вывод программы

# Техника работы с библиотекой Matplotlib

**Приложения:** mat.py, mat2.py

Библиотека matplotlib в Python помогает нам отображать данные на графиках в простейшем виде.

Библиотеку matplotlib легко установить с помощью pip:

pip install matplotlib

Теперь мы готовы создать несколько примеров, используя эту библиотеку визуализации данных.

**Листинг 60** (mat.py):

import matplotlib.pyplot as plt  
*# Координаты точек по x*x = [1, 5, 10, 15, 20]  
*# Координаты точек графика first по оси y*y1 = [1, 7, 3, 5, 11]  
*# Координаты точек графика second по оси y*y2 = [4, 3, 1, 8, 12]  
plt.figure(figsize=(7, 4)) *# Размер рисунка  
# Построение графика first*plt.plot(x, y1, 'o-r', alpha=0.7, label="first", lw=5, mec='b', mew=2, ms=10)  
*# Построение графика second*plt.plot(x, y2, 'v-.g', label="second", mec='r', lw=2, mew=2, ms=12)  
plt.legend() *# Отображение легенды*plt.grid(True) *# Отображение сетки*plt.show() *# Отображение графика*

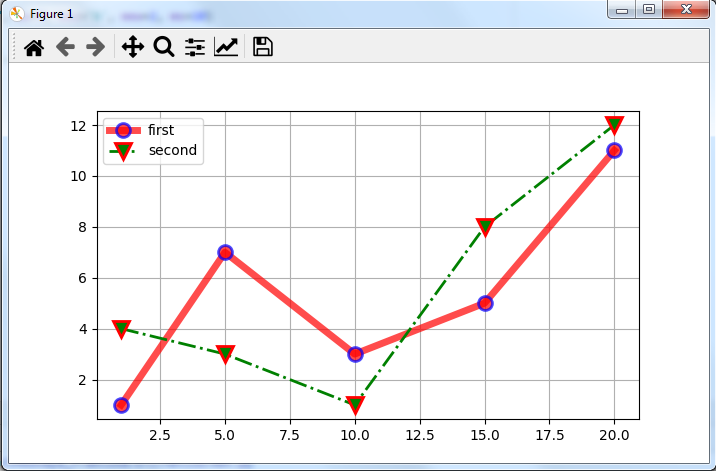


Рисунок 29. Вывод программы

**Листинг 61** (mat2.py):

import matplotlib.pyplot as plt  
import numpy as np  
*# Установка столбцов по значениям*p = [f"P{i}" for i in range(5)]  
*# Значения первой группы столбцов*g1 = [10, 21, 34, 12, 27]  
*# Значения второй группы столбцов*g2 = [17, 15, 25, 21, 26]  
width = 0.3 *# Ширина столбцов  
# [0, 1, 2, 3, 4]*x = np.arange(len(p))  
*# Размещение графиков рядом друг с другом*fig, ax = plt.subplots()  
rects1 = ax.bar(x - width/2, g1, width, label='g1')  
rects2 = ax.bar(x + width/2, g2, width, label='g2')  
ax.set\_title('Пример групповой диаграммы')  
*# Список местоположений тиков по оси x*ax.set\_xticks(x)  
*# Возвращает список текстовых экземпляров*ax.set\_xticklabels(p)  
*# Отображение легенды*ax.legend()  
plt.show()

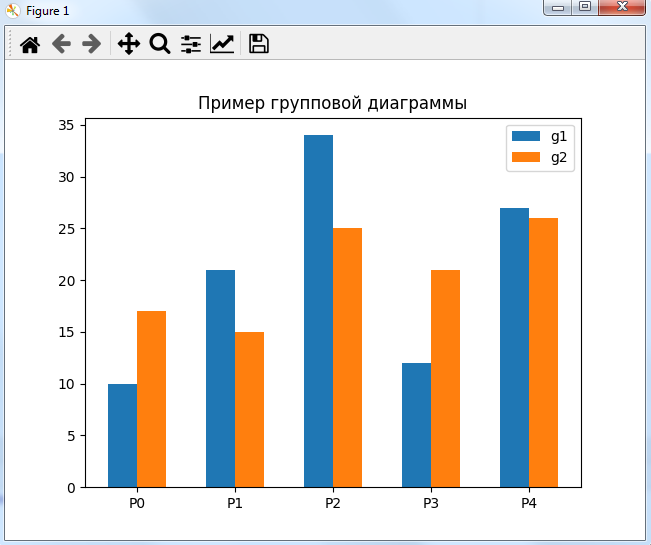


Рисунок 30. Вывод программы

# Элементы работы с библиотекой PyQt

**Приложения:** calculator.py, Design.py

PyQt — набор расширений [графического](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [фреймворка](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) [Qt](https://ru.wikipedia.org/wiki/Qt) для [языка программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [Python](https://ru.wikipedia.org/wiki/Python), выполненный в виде [расширения](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD) Python. Это мульти-платформенный инструментарий, который запускается на большинстве операционных систем, среди которых Unix, Windows и MacOS.

Для установки библиотеки используется команда:

pip install PyQt5

**Листинг 62 (**calculator.py):

import sys #этот модуль отвечает за определение среды запуска приложения

from Design import \* #импортируем наш разработанный дизайн

class MyWin(QtWidgets.QMainWindow): #создаем класс приложения,который наследуется от qMainWindow

def \_\_init\_\_(self): #создаем конструктор для экземпляра

QtWidgets.QWidget.\_\_init\_\_(self) #импортируем все методы библиотеки pyQt

self.ui = Ui\_MainWindow() #связываем дизайн с кнопками

self.ui.setupUi(self) #функция отрисовки главного окна

self.ui.pushButton\_3.clicked.connect(self.setZero) #назначаем функционал для кнопок...

self.ui.pushButton\_4.clicked.connect(self.setSix)

self.ui.pushButton\_8.clicked.connect(self.setFour)

self.ui.pushButton\_9.clicked.connect(self.setFive)

self.ui.pushButton\_10.clicked.connect(self.setEight)

self.ui.pushButton\_6.clicked.connect(self.setThree)

self.ui.pushButton\_11.clicked.connect(self.setSub)

self.ui.pushButton\_13.clicked.connect(self.setTwo)

self.ui.pushButton\_5.clicked.connect(self.setEnter)

self.ui.pushButton\_14.clicked.connect(self.setMult)

self.ui.pushButton\_18.clicked.connect(self.setOne)

self.ui.pushButton\_17.clicked.connect(self.setSeven)

self.ui.pushButton\_16.clicked.connect(self.setNine)

self.ui.pushButton\_15.clicked.connect(self.setDif)

self.ui.pushButton\_7.clicked.connect(self.setAdd)

self.ui.pushButton\_12.clicked.connect(self.setClear)

def setZero(self): #определяем функции...

a = self.ui.lineEdit.text() #создаем локальную переменную и сохраняем в нее строку из "lineEdit"

self.ui.lineEdit.setText(a + '0') #к "a" прибавляем '0' и сохраняем в "lineEdit"

def setOne(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '1')

def setTwo(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '2')

def setThree(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '3')

def setFour(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '4')

def setFive(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '5')

def setSix(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '6')

def setSeven(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '7')

def setEight(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '8')

def setNine(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '9')

def setSub(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '-')

def setMult(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '\*')

def setDif(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '/')

def setAdd(self):

a = self.ui.lineEdit.text()

self.ui.lineEdit.setText(a + '+')

def setClear(self):

self.ui.lineEdit.setText('')

def setEnter(self):

a = self.ui.lineEdit.text()

b = str(eval(a)) #функция "eval" преобразует строку в математическую операцию

self.ui.lineEdit.setText(b)

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication(sys.argv) #определение в среде виндовс

myapp = MyWin() #создание экземпляра окна

myapp.show() #отрисовка окна на экране

sys.exit(app.exec\_()) #устанавливает параметр завершения работы окна

**Листинг 63** (Design.py):

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(400, 500)

MainWindow.setMinimumSize(QtCore.QSize(400, 500))

MainWindow.setMaximumSize(QtCore.QSize(400, 500))

MainWindow.setStyleSheet("background-color: rgb(235, 235, 117);\n"

"")

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.gridLayoutWidget = QtWidgets.QWidget(self.centralwidget)

self.gridLayoutWidget.setGeometry(QtCore.QRect(10, 210, 381, 271))

self.gridLayoutWidget.setObjectName("gridLayoutWidget")

self.gridLayout = QtWidgets.QGridLayout(self.gridLayoutWidget)

self.gridLayout.setContentsMargins(0, 0, 0, 0)

self.gridLayout.setObjectName("gridLayout")

self.pushButton\_4 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_4.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_4.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_4.setFlat(False)

self.pushButton\_4.setObjectName("pushButton\_4")

self.gridLayout.addWidget(self.pushButton\_4, 1, 2, 1, 1)

self.pushButton\_8 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_8.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_8.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_8.setFlat(False)

self.pushButton\_8.setObjectName("pushButton\_8")

self.gridLayout.addWidget(self.pushButton\_8, 1, 0, 1, 1)

self.pushButton\_9 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_9.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_9.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_9.setFlat(False)

self.pushButton\_9.setObjectName("pushButton\_9")

self.gridLayout.addWidget(self.pushButton\_9, 1, 1, 1, 1)

self.pushButton\_10 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_10.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_10.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_10.setFlat(False)

self.pushButton\_10.setObjectName("pushButton\_10")

self.gridLayout.addWidget(self.pushButton\_10, 0, 1, 1, 1)

self.pushButton\_3 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_3.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_3.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_3.setFlat(False)

self.pushButton\_3.setObjectName("pushButton\_3")

self.gridLayout.addWidget(self.pushButton\_3, 3, 0, 1, 1)

self.pushButton\_6 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_6.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_6.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_6.setFlat(False)

self.pushButton\_6.setObjectName("pushButton\_6")

self.gridLayout.addWidget(self.pushButton\_6, 2, 2, 1, 1)

self.pushButton\_11 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_11.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_11.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_11.setFlat(False)

self.pushButton\_11.setObjectName("pushButton\_11")

self.gridLayout.addWidget(self.pushButton\_11, 2, 3, 1, 1)

self.pushButton\_12 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_12.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_12.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_12.setFlat(False)

self.pushButton\_12.setObjectName("pushButton\_12")

self.gridLayout.addWidget(self.pushButton\_12, 3, 3, 1, 1)

self.pushButton\_13 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_13.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_13.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_13.setFlat(False)

self.pushButton\_13.setObjectName("pushButton\_13")

self.gridLayout.addWidget(self.pushButton\_13, 2, 1, 1, 1)

self.pushButton\_5 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_5.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_5.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_5.setFlat(False)

self.pushButton\_5.setObjectName("pushButton\_5")

self.gridLayout.addWidget(self.pushButton\_5, 3, 2, 1, 1)

self.pushButton\_14 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_14.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_14.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_14.setFlat(False)

self.pushButton\_14.setObjectName("pushButton\_14")

self.gridLayout.addWidget(self.pushButton\_14, 1, 3, 1, 1)

self.pushButton\_18 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_18.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_18.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_18.setFlat(False)

self.pushButton\_18.setObjectName("pushButton\_18")

self.gridLayout.addWidget(self.pushButton\_18, 2, 0, 1, 1)

self.pushButton\_17 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_17.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_17.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_17.setFlat(False)

self.pushButton\_17.setObjectName("pushButton\_17")

self.gridLayout.addWidget(self.pushButton\_17, 0, 0, 1, 1)

self.pushButton\_16 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_16.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_16.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_16.setFlat(False)

self.pushButton\_16.setObjectName("pushButton\_16")

self.gridLayout.addWidget(self.pushButton\_16, 0, 2, 1, 1)

self.pushButton\_15 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_15.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_15.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_15.setFlat(False)

self.pushButton\_15.setObjectName("pushButton\_15")

self.gridLayout.addWidget(self.pushButton\_15, 0, 3, 1, 1)

self.pushButton\_7 = QtWidgets.QPushButton(self.gridLayoutWidget)

self.pushButton\_7.setMinimumSize(QtCore.QSize(0, 45))

self.pushButton\_7.setStyleSheet("QPushButton{\n"

"border:0px solid rgb(12, 40, 71);\n"

" background-color: rgb(2, 173, 225);\n"

"border-radius:10px;\n"

" color: rgb(227, 227, 227);\n"

" font: 16pt \"MS Shell Dlg 2\";\n"

"}\n"

"QPushButton:hover{\n"

"border:3px solid white;\n"

"\n"

"}\n"

"QPushButton:pressed{\n"

"background-color: rgb(229, 76, 229);\n"

"}\n"

"")

self.pushButton\_7.setFlat(False)

self.pushButton\_7.setObjectName("pushButton\_7")

self.gridLayout.addWidget(self.pushButton\_7, 3, 1, 1, 1)

self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)

self.lineEdit.setGeometry(QtCore.QRect(10, 40, 381, 81))

self.lineEdit.setStyleSheet("font: 30px \"MS Shell Dlg 2\";\n"

"background-color: rgb(255, 255, 255);\n"

"border-radius:30px;\n"

"border:3px solid rgb(2, 173, 225);\n"

"\n"

"")

self.lineEdit.setText("")

self.lineEdit.setObjectName("lineEdit")

MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

MainWindow.setWindowTitle(\_translate("MainWindow", "Калькулятор от Юрчика"))

self.pushButton\_4.setText(\_translate("MainWindow", "6"))

self.pushButton\_8.setText(\_translate("MainWindow", "4"))

self.pushButton\_9.setText(\_translate("MainWindow", "5"))

self.pushButton\_10.setText(\_translate("MainWindow", "8"))

self.pushButton\_3.setText(\_translate("MainWindow", "0"))

self.pushButton\_6.setText(\_translate("MainWindow", "3"))

self.pushButton\_11.setText(\_translate("MainWindow", "-"))

self.pushButton\_12.setText(\_translate("MainWindow", "CLEAR"))

self.pushButton\_13.setText(\_translate("MainWindow", "2"))

self.pushButton\_5.setText(\_translate("MainWindow", "="))

self.pushButton\_14.setText(\_translate("MainWindow", "\*"))

self.pushButton\_18.setText(\_translate("MainWindow", "1"))

self.pushButton\_17.setText(\_translate("MainWindow", "7"))

self.pushButton\_16.setText(\_translate("MainWindow", "9"))

self.pushButton\_15.setText(\_translate("MainWindow", "/"))

self.pushButton\_7.setText(\_translate("MainWindow", "+"))

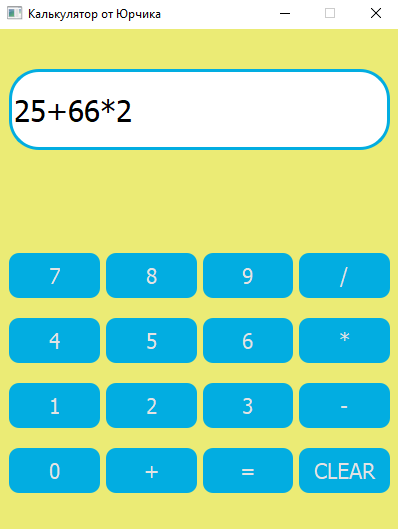


Рисунок 31. Окно калькулятора

# Элементы работы с библиотекой PyGame

**Приложения:** PyGame.py

Pygame – это библиотека модулей для языка Python, созданная для разработки 2D игр.

Pygame не входит в стандартную библиотеку Python, то есть не поставляется с установочным пакетом, а требует отдельной установки. В Ubuntu и родственных дистрибутивах это можно сделать двумя способами – с помощью pip и apt-get:

python3 -m pip install -U pygame --user

или

sudo apt-get install python3-pygame

Для Windows:

Открыв командную строку (cmd) прописываем:

pip install pygame

Убедиться, что библиотека загружена, и что все установилось нормально, можно опять же в командной строке (cmd). Следует вписать следующие две строчки:

Python

Import pygame

**Листинг 64** (PyGame.py):

import pygame as pg

import random

# Создание квадрата по координатам

def grect(x, y):

return x, y, 100, 100

# Создание окна с разрешением

pg.init()

sc = pg.display.set\_mode((700, 500))

cl = pg.time.Clock()

# Создание переменных и списка

x, y = 100, 200

t = True

color = ''

colorAll = ['white', 'orange', 'blue', 'green', 'red', 'yellow']

# Запуск цикла

while 1:

# Задаем цвет фона

sc.fill(pg.Color("black"))

# Проверка на закрытие окна

[exit() for event in pg.event.get() if event.type == pg.QUIT]

# Переключение режимов

if x == 100:

t = False

color = random.choice(colorAll)

elif x == 200:

t = True

color = random.choice(colorAll)

# События режимов

if t == True:

x -= 1

else:

x += 1

pg.draw.rect(sc, pg.Color("white"), (100, 50, 100, 100))

# Создание статичных квадратов

pg.draw.rect(sc, pg.Color("white"), grect(x, y))

pg.draw.rect(sc, pg.Color(color), (100, 350, 100, 100))

# Обновление ока с заданным тиком

pg.display.flip()

cl.tick(60) hero.set\_speed(1)

hero.set\_target()

hero.set\_pos()

hero.set\_color()

heroes.append(hero)

del hero

game = True # Запушена ли игра

stop = False # Приостановлена ли игра

mode = "touch" # Мод. Либо 'touch', либо 'press'

# Запуск игры

while game:

Update(display, map\_size, heroes)



Рисунок 32. Скриншот игры

# **Раздел 3. Разработка проекта с графическим интерфейсом**

В данном проекте мы работаем в основе с двумя библиотеками “tkinter” и “sqlite3”.

Библиотека Tkinter установлена в Python в качестве стандартного модуля, поэтому нам не нужно устанавливать что-либо для его использования.

Чтобы импортировать его, пропишем

import tkinter

Python SQLite поставляется вместе с любой установленной версией Python, поэтому нет необходимости устанавливать его с помощью pip. Чтобы импортировать его, поскольку мы будем использовать Python3, мы импортируем модуль sqlite3.

import sqlite3

3.1 Изучение входной и выходной документации

Входные данные:

1. Содержание базы данных

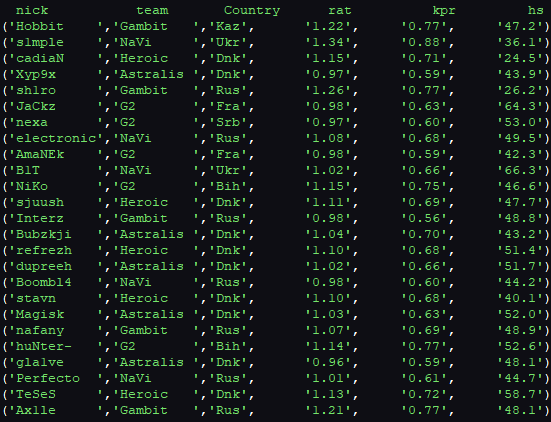
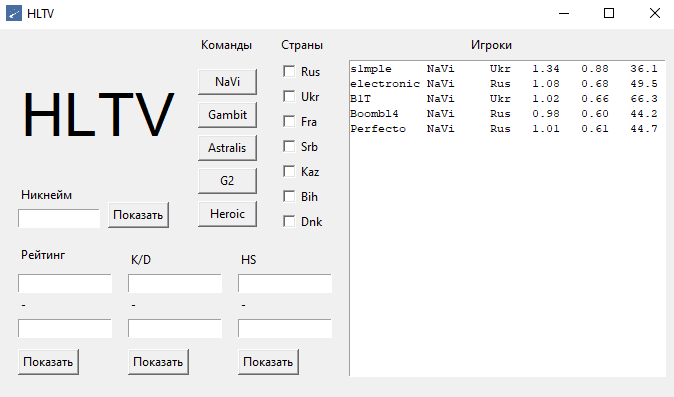


Рисунок 33. Документ на оплату

Выходные данныеь

1. Сортировка по команде «NaVi»

  
Рисунок 34. Сортировка по команде

1. Вывод по регионам: «Rus», «Ukr»

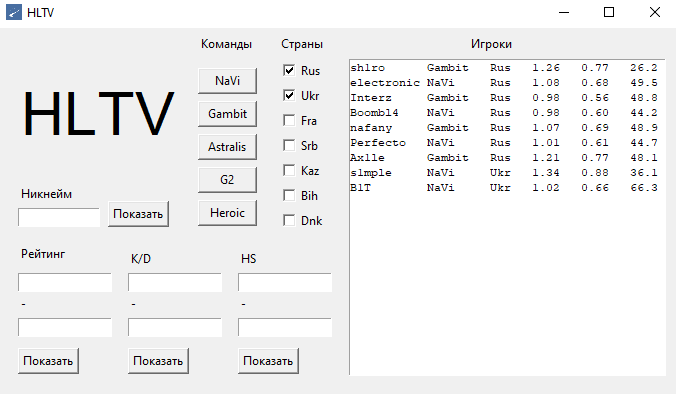


Рисунок 35. Сортировка по регионам

3.2 Разработка требований к проекту. Построение диаграммы использования.

Требования к проекту:

1. Быстрая сортировка данных
2. Удобный и понятный вывод

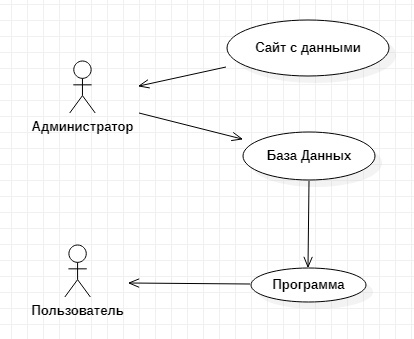


Рисунок 36. Диаграмма использования

* 1. Разработка сценария проекта

В данном разделе приведен сценарий использования программы пользователем (Рисунок 136).

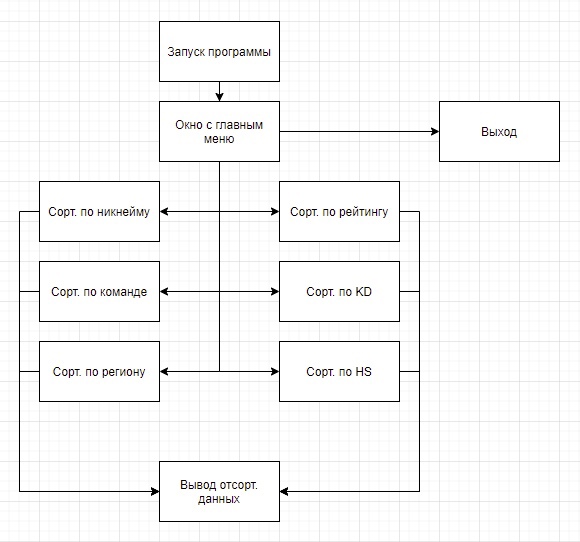


Рисунок 37. Сценарий проекта

* 1. Построение диаграммы классов

В данном разделе находятся диаграммы классов приложения.(Рисунок 137)

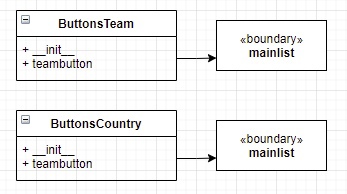


Рисунок 38. Диаграмма классов

* 1. Разработка базы данных

В проекте в качестве базы данных используется **SQLite**. Причины использования:

1. **Минимальные затраты ресурсов.** Для работы большинства систем управления базами данных необходим специальный процесс сервера базы данных. SQLite обходится без сервера: база данных SQLite представляет собой обычный файл. Когда БД не используется, она не расходует процессорное время.
2. **Надежность и быстрота.** Базы данных SQLite поддерживают транзакции баз данных. Кроме того, операции чтения и записи данных реализуются на оптимизированном коде С.

**Приложение:** players.py

**Листинг 65** ([players.py](https://github.com/prog-kkkmt/p1-18/blob/Gymrasimov/%D0%A3%D0%9F/project_w_bd/sqlighter.py)):

import sqlite3

# Создание списка таблици

''' nick team Country rat kpr hs '''

text = [('Hobbit ','Gambit ','Kaz', '1.22', '0.77', '47.2'),

('s1mple ','NaVi ','Ukr', '1.34', '0.88', '36.1'),

('cadiaN ','Heroic ','Dnk', '1.15', '0.71', '24.5'),

('Xyp9x ','Astralis ','Dnk', '0.97', '0.59', '43.9'),

('sh1ro ','Gambit ','Rus', '1.26', '0.77', '26.2'),

('JaCkz ','G2 ','Fra', '0.98', '0.63', '64.3'),

('nexa ','G2 ','Srb', '0.97', '0.60', '53.0'),

('electronic','NaVi ','Rus', '1.08', '0.68', '49.5'),

('AmaNEk ','G2 ','Fra', '0.98', '0.59', '42.3'),

('B1T ','NaVi ','Ukr', '1.02', '0.66', '66.3'),

('NiKo ','G2 ','Bih', '1.15', '0.75', '46.6'),

('sjuush ','Heroic ','Dnk', '1.11', '0.69', '47.7'),

('Interz ','Gambit ','Rus', '0.98', '0.56', '48.8'),

('Bubzkji ','Astralis ','Dnk', '1.04', '0.70', '43.2'),

('refrezh ','Heroic ','Dnk', '1.10', '0.68', '51.4'),

('dupreeh ','Astralis ','Dnk', '1.02', '0.66', '51.7'),

('Boombl4 ','NaVi ','Rus', '0.98', '0.60', '44.2'),

('stavn ','Heroic ','Dnk', '1.10', '0.68', '40.1'),

('Magisk ','Astralis ','Dnk', '1.03', '0.63', '52.0'),

('nafany ','Gambit ','Rus', '1.07', '0.69', '48.9'),

('huNter- ','G2 ','Bih', '1.14', '0.77', '52.6'),

('gla1ve ','Astralis ','Dnk', '0.96', '0.59', '48.1'),

('Perfecto ','NaVi ','Rus', '1.01', '0.61', '44.7'),

('TeSeS ','Heroic ','Dnk', '1.13', '0.72', '58.7'),

('Ax1le ','Gambit ','Rus', '1.21', '0.77', '48.1')]

# Создание и запись списка в таблицу

conn = sqlite3.connect('PlayersTable')

curs = conn.cursor()

curs.execute("""CREATE TABLE Players(

Name TEXT,

Team TEXT,

Country TEXT,

Rating TEXT,

Kill\_per\_round TEXT,

Headshot REAL)""")

curs.executemany("INSERT INTO Players VALUES (?, ?, ?, ?, ?, ?)", text)

conn.commit()

'''

a = ['Rus', '', '', '']

print(\*a)

g = curs.execute("""SELECT \*

FROM Players

WHERE Country IN (?, ?, ?, ?)""", a)

print(\*g, '----------------------------------------------')

a[1] = 'Ukr'

g = curs.execute("""SELECT \*

FROM Players

WHERE Country IN (?, ?, ?, ?)""", a)

print(\*g)

conn.commit()

'''

'''

SELECT \*

FROM Players

WHERE Team = "G2";

SELECT \*

FROM Players

WHERE Country IN ("Rus", "Ukr");

SELECT \*

FROM Players

WHERE Rating >= 1

ORDER BY Rating DESC;

SELECT \*

FROM Players

WHERE Name LIKE "S%";

'''

* 1. Разработка главного модуля

Главный модуль состоит из двух классов ButtonsTeam и ButtonsCountry. После нажатия на определенную кнопку, вызываются функции, которые наследуются из класса.

**Приложение:** PlayersTK.py

**Листинг 66** (PlayersTK.py):

from tkinter import \*

import sqlite3

# Создание классов для кнопок "Команды" и "Страны"

class ButtonsTeam:

def \_\_init\_\_(self, var, px, py):

self.var = var

self.teambuttons = Button(text=self.var, width=7,

command=self.teambutton).place(x=px, y=py)

# Сортировка и отображение

def teambutton(self):

conn = sqlite3.connect('PlayersTable')

curs = conn.cursor()

\*gridplayers, = curs.execute("""SELECT \*

FROM Players

WHERE Team LIKE ?""", (self.var + '%',))

mainlist.delete(0, END)

for i in gridplayers:

\*i, = i

rowtext.set(str(i[0]) + ' ' + str(i[1]) + '' + \

str(i[2]) + ' ' + str(i[3]) + ' ' + \

str(i[4]) + ' ' + str(i[5]))

mainlist.insert(END, rowtext.get())

class ButtonsCountry:

def \_\_init\_\_(self, var, px, py):

self.i = 0

self.var = var

self.turn = IntVar()

self.teambuttons = Checkbutton(text=self.var, command=self.teambutton,

variable=self.turn).place(x=px, y=py)

# Сортировка и отображение

def teambutton(self):

global grid

conn = sqlite3.connect('PlayersTable')

curs = conn.cursor()

if self.turn.get() == 1:

self.i = grid.index('')

grid[self.i] = self.var

else:

grid[self.i] = ''

\*gridplayers, = curs.execute("""SELECT \*

FROM Players

WHERE Country IN (?, ?, ?, ?, ?, ?, ?)

ORDER BY Country""", grid)

mainlist.delete(0, END)

for i in gridplayers:

\*i, = i

rowtext.set(str(i[0]) + ' ' + str(i[1]) + '' + \

str(i[2]) + ' ' + str(i[3]) + ' ' + \

str(i[4]) + ' ' + str(i[5]))

mainlist.insert(END, rowtext.get())

#----------------------------------------------------------

# Функци для кнопок "Показать"

def getr():

global rowtext, mainlist, ratingbox1, ratingbox2

conn = sqlite3.connect('PlayersTable')

curs = conn.cursor()

\*gridplayers, = curs.execute("""SELECT \*

FROM Players

WHERE Rating BETWEEN ? AND ?

ORDER BY Rating DESC""",\

(float(ratingbox1.get()), float(ratingbox2.get())))

mainlist.delete(0, END)

for i in gridplayers:

\*i, = i

rowtext.set(str(i[0]) + ' ' + str(i[1]) + ' ' + \

str(i[2]) + ' ' + str(i[3]) + ' ' + \

str(i[4]) + ' ' + str(i[5]))

mainlist.insert(END, rowtext.get())

def getk():

global rowtext, mainlist, killbox1, killbox2

conn = sqlite3.connect('PlayersTable')

curs = conn.cursor()

\*gridplayers, = curs.execute("""SELECT \*

FROM Players

WHERE Kill\_per\_round BETWEEN ? AND ?

ORDER BY Kill\_per\_round DESC""",\

(float(killbox1.get()), float(killbox2.get())))

mainlist.delete(0, END)

for i in gridplayers:

\*i, = i

rowtext.set(str(i[0]) + ' ' + str(i[1]) + '' + \

str(i[2]) + ' ' + str(i[3]) + ' ' + \

str(i[4]) + ' ' + str(i[5]))

mainlist.insert(END, rowtext.get())

def geth():

global rowtext, mainlist, hsbox1, hsbox2

conn = sqlite3.connect('PlayersTable')

curs = conn.cursor()

\*gridplayers, = curs.execute("""SELECT \*

FROM Players

WHERE Headshot BETWEEN ? AND ?

ORDER BY Headshot DESC""",\

(hsbox1.get(), hsbox2.get()))

mainlist.delete(0, END)

for i in gridplayers:

\*i, = i

rowtext.set(str(i[0]) + ' ' + str(i[1]) + '' + \

str(i[2]) + ' ' + str(i[3]) + ' ' + \

str(i[4]) + ' ' + str(i[5]))

mainlist.insert(END, rowtext.get())

def getplayer():

global rowtext, mainlist, namebox

conn = sqlite3.connect('PlayersTable')

curs = conn.cursor()

\*gridplayers, = curs.execute("""SELECT \*

FROM Players

WHERE Name LIKE ?

ORDER BY Name""",\

('%' + namebox.get() + '%',))

mainlist.delete(0, END)

for i in gridplayers:

\*i, = i

rowtext.set(str(i[0]) + ' ' + str(i[1]) + '' + \

str(i[2]) + ' ' + str(i[3]) + ' ' + \

str(i[4]) + ' ' + str(i[5]))

mainlist.insert(END, rowtext.get())

#----------------------------------------------------------

# Создание окна, иконки и переменных

tk = Tk()

tk.title('HLTV')

tk.geometry('680x370+700+300')

tk.call('wm', 'iconphoto', tk.\_w, PhotoImage(file='HLTV-logo.png'))

logo = Label(tk, text='HLTV', font='Arial 45').place(x=20, y=50)

rowtext = StringVar()

nameplayer = StringVar()

grid = ['', '', '', '', '', '', '']

#----------------------------------------------------------

# Создание названий разделов и кнопок

nametext = Label(tk, text='Никнейм').place(x=20, y=155)

teamtext = Label(tk, text='Команды').place(x=200, y=5)

countrytext = Label(tk, text='Страны').place(x=280, y=5)

reatingtext = Label(tk, text='Рейтинг').place(x=20, y=215)

KDtext = Label(tk, text='K/D').place(x=130, y=220)

HStext = Label(tk, text='HS').place(x=240, y=220)

ButtonsTeam('NaVi', 200, 40)

ButtonsTeam('Gambit', 200, 73)

ButtonsTeam('Astralis', 200, 106)

ButtonsTeam('G2', 200, 139)

ButtonsTeam('Heroic', 200, 172)

ButtonsCountry('Rus', 280, 30)

ButtonsCountry('Ukr', 280, 55)

ButtonsCountry('Fra', 280, 80)

ButtonsCountry('Srb', 280, 105)

ButtonsCountry('Kaz', 280, 130)

ButtonsCountry('Bih', 280, 155)

ButtonsCountry('Dnk', 280, 180)

#----------------------------------------------------------

# Создание окн ввода и кнопок "Показать"

namebox = Entry(tk, width=13, textvariable=nameplayer)

namebox.place(x=20, y=180)

ratingbox1 = Entry(tk, width=15)

ratingbox1.place(x=20, y=245)

ratingtext = Label(tk, text='-').place(x=20, y=265)

ratingbox2 = Entry(tk, width=15)

ratingbox2.place(x=20, y=290)

killbox1 = Entry(tk, width=15)

killbox1.place(x=130, y=245)

killtext = Label(tk, text='-').place(x=130, y=265)

killbox2 = Entry(tk, width=15)

killbox2.place(x=130, y=290)

hsbox1 = Entry(tk, width=15)

hsbox1.place(x=240, y=245)

hstext = Label(tk, text='-').place(x=240, y=265)

hsbox2 = Entry(tk, width=15)

hsbox2.place(x=240, y=290)

namebutton = Button(tk, text='Показать', command=getplayer).place(x=110, y=173)

ratingbutton = Button(tk, text='Показать', command=getr).place(x=20, y=320)

killbutton = Button(tk, text='Показать', command=getk).place(x=130, y=320)

hsbutton = Button(tk, text='Показать', command=geth).place(x=240, y=320)

#----------------------------------------------------------

# Создание окна для показа таблици

maintext = Label(tk, text = 'Игроки').place(x=470, y=5)

mainlist = Listbox(tk, width=45, height=21, font='Courier 8')

mainlist.place(x=350, y=30)

mainloop()

* 1. Тестирование и отладка

В ходе написания проекта при попытке запустить программу были получены ошибки (Рисунок 40):

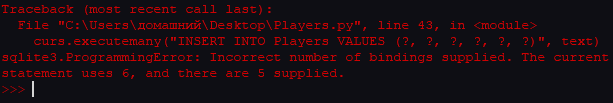


Рисунок 39. Ошибки

При проверке кода были исправлены найденные ошибки, в результате при запуске программы ошибок не было (Рисунок 41):



Рисунок 40. Успешная сборка

* 1. Дневник

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работ | Отметка о выполнении |
| 10.12.2020 | Техника работы с линейными программами. Техника работы с разветвляющимися программами. Техника работы с циклическими программами. |  |
| 11.12.2020 | Техника работы с циклическими программами. Техника работы с числами. |  |
| 12.12.2020 | Техника работы со строками. Техника работы со списками. |  |
| 14.12.2020 | Техника работы со списками. Техника работы с циклом for и генераторами списков. |  |
| 15.12.2020 | Техника работы с функциями. Техника работы со словарями. |  |
| 16.12.2020 | Техника работы со словарями. Техника работы с множествами. |  |
| 17.12.2020 | Техника работы с кортежами. Техника работы с файлами. |  |
| 18.12.2020 | Техника работы с файлами. Техника работы с модулями. |  |
| 19.12.2020 | Техника работы с модулями. Техника работы с классами. |  |
| 21.12.2020 | Техника работы с классами. |  |
| 30.03.2021 | Установка и настройка среды JetBrains PyCharm. Техника работы с базами данных. |  |
| 31.03.2021 | Техника работы с базами данных. Техника работы с библиотекой tkinter. |  |
| 01.04.2021 | Техника работы с библиотекой tkinter |  |
| 02.04.2021 | Техника работы с библиотекой tkinter. Техника работы с библиотекой NumPy. |  |
| 05.04.2021 | Техника работы с библиотекой NumPy. Техника работы с библиотекой Matplotlib. |  |
| 06.04.2021 | Техника работы с библиотекой Matplotlib. Элементы работы с библиотекой PyQt. |  |
| 07.04.2021 | Элементы работы с библиотекой PyQt. |  |
| 08.04.2021 | Элементы работы с библиотекой PyQt. Элементы работы с библиотекой PyGame. |  |
| 09.04.2021 | Элементы работы с библиотекой PyGame. Выполнение отчёта и презентации по использованию библиотек. |  |
| 12.04.2021 | Выполнение отчёта и презентации по использованию библиотек. |  |
| 13.04.2021 | Изучение входной и выходной документации. Разработка требований к проекту. Построение диаграммы использования |  |
| 14.04.2021 | Разработка требований к проекту. Построение диаграммы использования. Разработка сценария проекта. |  |
| 15.04.2021 | Разработка сценария проекта. Построение диаграммы классов. |  |
| 16.04.2021 | Построение диаграммы классов. Разработка базы данных. |  |
| 19.04.2021 | Разработка базы данных. Разработка главного модуля. |  |
| 20.04.2021 | Разработка главного модуля. Разработка входящих модулей. |  |
| 21.04.2021 | Разработка входящих модулей. |  |
| 22.04.2021 | Разработка входящих модулей. Тестирование и отладка. Разработка документации. |  |
| 23.04.2021 | Разработка документации. Защита проекта. |  |
| 26.04.2021 | Защита проекта. Сдача зачёта по практике. |  |

# **Приложения**

**Раздел 1. Техника решения задач с использованием структурного и объектно-ориентированного программирования**

**1.3** **Техника работы с линейными и разветвляющимися программами**

**Файлы:**

Листинг 1(begin1.py)

Листинг 2(begin2.py)

Листинг 3(boolean1.py)

Листинг 4(boolean2.py)

Листинг 5(if1.py)

Листинг 6(if2.py)

Листинг 7(if3.py)

Листинг 8(case1.py)

Листинг 9(case2.py)

Листинг 10(case3.py)

**1.4 Техника работы с циклическими программами, цикл while**

**Файлы:**

Листинг 11(for1.py)

Листинг 12(for2.py)

Листинг 13(for3.py)

Листинг 14(for4.py)

Листинг 15(for5.py)

Листинг 16(while1.py)

Листинг 17(while2.py)

Листинг 18(while3.py)

Листинг 19(while4.py)

Листинг 20(while5.py)

**1.5 Техника работы с числами**

**Файлы:**

Листинг 21(Задача1.py)

Листинг 22(Задача2.py)

Листинг 23(Задача3.py)

Листинг 24(Задача4.py)

Листинг 25(Задача5.py)

**1.6 Техника работы со строками**

**Файлы:**

Листинг 26(Задача1.py)

Листинг 27(Задача2.py)

Листинг 28(Задача3.py)

Листинг 29(Задача9.py)

**1.7 Техника работы со списками**

**Файлы:**

Листинг 30(Задача1.py)

Листинг 31(Задача2.py)

Листинг 32(Задача3.py)

**1.8 Техника работы с циклом for и генераторами списков**

**Файлы:**

Листинг 33(Задача1.py)

Листинг 34(Задача2.py)

Листинг 35(Задача3.py)

Листинг 36(Задача4.py)

**1.9 Техника работы с функциями**

**Файлы:**

Листинг 37(func1.py)

Листинг 38(func2.py)

**1.10 Техника работы со словарями**

**Файлы:**

Листинг 39(dict1.py)

Листинг 40(dict2.py)

Листинг 41(dict3.py)

**1.11 Техника работы с множествами**

**Файлы:**

Листинг 42(Задание1.py)

Листинг 43(Задание2.py)

**1.12 Техника работы с кортежами**

**Файлы:**

Листинг 44(Задание1.py)

Листинг 45(Задание2.py)

**1.13 Техника работы с файлами**

**Файлы:**

Листинг 46(text1.py)

Листинг 47(text2.py)

Листинг 48(text3.py)

Листинг 49(text4.py)

Листинг 50(text5.py)

**1.14 Техника работы с модулями**

**Файлы:**

Листинг 51(15-1.1.py)

Листинг 52(15-1.2.py)

Листинг 53(15-2.py)

Листинг 54(15-4.py)

**1.15 Техника работы с классами**

**Файлы:**

Листинг 55(K16\_1.py)

Листинг 56(K16\_2.py)

**Раздел 2. Техника решения задач с использованием библиотек**

**2.2 Техника работы с базами данных**

**Файлы:**

Листинг 57(SQLite.py)

**2.3 Техника работы с библиотекой tkinter**

**Файлы:**

Листинг 58(Tkinter.py)

**2.4 Техника работы с библиотекой NumPy**

**Файлы:**

Листинг 59(Numpy.py)

**2.5 Техника работы с библиотекой Matplotlib**

**Файлы:**

Листинг 60(mat.py)

Листинг 61(mat2.py)

**2.6 Элементы работы с библиотекой PyQt**

**Файлы:**

Листинг 62**(**calculator.py)

Листинг 63(Design.py)

**2.7 Элементы работы с библиотекой PyGame**

**Файлы:**

Листинг 64(PyGame.py)

**Раздел 3. Разработка проекта с графическим интерфейсом**

**Разработка базы данных**

**Файлы:**

Листинг 65([players.py](https://github.com/prog-kkkmt/p1-18/blob/Gymrasimov/%D0%A3%D0%9F/project_w_bd/sqlighter.py))

Листинг 66(PlayersTK.py)