**

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

Выполнил студент: Васькевич Юрий Андреевич

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

Проверил преподаватель: Гусятинер Леонид Борисович

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

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

Королев, 2020

Оглавление

[**Раздел 1. Техника решения задач с использованием структурного и объектно-ориентированного программирования.** 4](#_Toc73394135)

[**Практическое занятие №2.** 4](#_Toc73394136)

[**Практическое занятие №4.** 5](#_Toc73394137)

[Задачи на Begin. 5](#_Toc73394138)

[Задачи на Boolean. 5](#_Toc73394139)

[Задачи на Case. 6](#_Toc73394140)

[Задачи на If. 7](#_Toc73394141)

[**Блок-схемы для задач к практическому занятию №4** 9](#_Toc73394142)

[Блок-схема для задачи boolean1: 9](#_Toc73394143)

[Блок-схема для задачи boolean2: 9](#_Toc73394144)

[**Практическое занятие №5.** 10](#_Toc73394145)

[Задачи на For. 10](#_Toc73394146)

[Задачи на While. 11](#_Toc73394147)

[Блок-схема для задачи while5: 13](#_Toc73394148)

[**Практическое занятие №6.** 14](#_Toc73394149)

[**Практическое занятие №7** 16](#_Toc73394150)

[**Практическое занятие №8** 18](#_Toc73394151)

[**Практическое занятие №9** 20](#_Toc73394152)

[**Практическое занятие №10.** 22](#_Toc73394153)

[**Практическое занятие №11.** 26](#_Toc73394154)

[**Практическое занятие №12.** 27](#_Toc73394155)

[**Практическое занятие №13.** 28](#_Toc73394156)

[**Практическое занятие №14.** 29](#_Toc73394157)

[**Практическое занятие №15.** 30](#_Toc73394158)

[**Практическое занятие №16.** 30](#_Toc73394159)

[**Раздел 2. Техника решения задач с использованием библиотек** 33](#_Toc73394160)

[2.1 Установка и настройка среды JetBrains PyCharm 33](#_Toc73394161)

[2.2 Техника работы с базами данных 39](#_Toc73394162)

[2.3 Техника работы с библиотекой tkinter 43](#_Toc73394163)

[2.4 Техника работы с библиотекой NumPy 48](#_Toc73394164)

[2.5 Техника работы с библиотекой Matplotlib 50](#_Toc73394165)

[2.6 Элементы работы с библиотекой PyQt 52](#_Toc73394166)

[2.7 Элементы работы с библиотекой PyGame 64](#_Toc73394167)

[**Раздел 3. Разработка проекта с графическим интерфейсом** 67](#_Toc73394168)

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

# **Практическое занятие №2.**

Установка Python будет производиться в PyCharm Community Edition 2020

1. Открываем сайт https://www.python.org/downloads/

2. Нажимаем на кнопку "Download Python 3.9.0".

3. Ждём окончания загрузки.

4. Открывай установщик Python 3.9.0

5. Указываем путь установки Python library и ставим галочку на “Add Python 3.9 to PATH”

6. Ждём окончания загрузки.

7.Теперь мы можем начинать кодить в любом из удобных вам интегрированных средах разработки.

https://drive.google.com/file/d/1CdA0eJAUHnbn1lGqWnxPOlF6J8u1232Z/view

# **Практическое занятие №4.**

## Задачи на Begin.

**Задача 1: ( begin1.py )**

#Выполнил работу Васькевич Юрий Андреевич

#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!')

**Задача 2: ( begin2.py )**

#Выполнил работу Васькевич Юрий Андреевич

#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.

**Задача 1: ( boolean1.py )**

#Выполнил работу Васькевич Юрий Андреевич

#Дано целое число A.

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

x = int(input("Введите число A: "))

res = x > 0

print(res)

**Задача 2: ( boolean2.py )**

#Выполнил работу Васькевич Юрий Андреевич

#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 Case.

**Задача 1: ( case1.py )**

#Выполнил работу Васькевич Юрий Андреевич

#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!") if 0 < x < 8:

**Задача 2: ( case2.py )**

#Выполнил работу Васькевич Юрий Андреевич

#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")

**Задача 3: ( case3.py )**

#Выполнил работу Васькевич Юрий Андреевич

#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)

## Задачи на If.

**Задача 1: ( if1.py )**

#Выполнил работу Васькевич Юрий Андреевич

#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)

**Задача 2: ( if2.py )**#Выполнил работу Васькевич Юрий Андреевич

#Задача:

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

a = int(input())

b = int(input())

if a < b:

print(a)

else:

print(b)

**Задача 3: (if3.py)**

#Выполнил работу Васькевич Юрий Андреевич

#Задача:

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

a = int(input())

b = int(input())

if a > b:

print(a)

else:

print(b) x = x + 1

print("Число положительное.", x)

elif x < 0:

x = x - 2

print("Число отрицательное.", x)

else:

x = 10

print("Введенное число - ноль.", x)

# **Блок-схемы для задач к практическому занятию №4**

## Блок-схема для задачи boolean1:

****

Рис. 1.boolean1

Блок-схема для задачи boolean2:  


Рис. 2.boolean2

# **Практическое занятие №5.**

**‘Техника работы с циклическими программами, циклы while, for’**

## Задачи на For.

**Задача 1: (for1.py)**

#Выполнил работу Васькевич Юрий Андреевич

#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])

**Задача 2: (for2.py)**

#Выполнил работу Васькевич Юрий Андреевич

#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)

**Задача 3: (for3.py)**

#Выполнил работу Васькевич Юрий Андреевич

#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)

**Задача 4: (for4.py)**

#Выполнил работу Васькевич Юрий Андреевич

#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 = ' ')

**Задача 5: (for5.py)**

#Выполнил работу Васькевич Юрий Андреевич

#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 карточкиn = int(input())m = 0for i in range(1, n+1):

m += i

for i in range(n-1):

m -= int(input())

print(m)

## Задачи на While.

**Задача 1: (while1.py)**

#Выполнил работу Васькевич Юрий Андреевич

#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)

**Задача 2: (while2.py)**

#Выполнил работу Васькевич Юрий Андреевич

#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)

**Задача 3: (while3.py)**

#Выполнил работу Васькевич Юрий Андреевич

#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())

**Задача 4: (while4.py)**

#Выполнил работу Васькевич Юрий Андреевич

#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])

**Задача 5: (while5.py)**

#Выполнил работу Васькевич Юрий Андреевич

#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

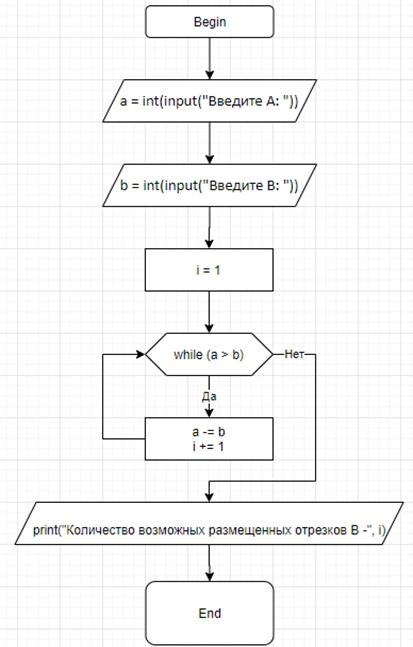
while (a > b):

a -= b

i += 1

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

## Блок-схема для задачи while5:



(Рис. 1)

# **Практическое занятие №6.**

**’Техника работы с числами’**

**Задача 1: (6zadanie 1 zada4a.py)**

#Выполнил задание Васькевич Юрий Андревич

#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')

**Задача 2: (6zadanie 2 zada4a.py)**

#Выполнил задание Васькевич Юрий Андреевич

#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))

**Задача 3: (6zadanie 3 zada4a.py)**

#Выполнил задание Васькевич Юрий Андреевич

import math

a = int(input())

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

print(s)

**Задача 4: (6zadanie 4 zada4a.py)**

#Выполнил задание Васькевич Юрий Андреевич

#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')

**Задача 5: (6zadanie 5 zada4a.py)**

#Выполнил задание Васькевич Юрий Андреевич

#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))

# **Практическое занятие №7**

**‘Техника работы со строками’**

**Задача 1: (7zadanie 1zada4a.py)**

stroka = ' hello' #создаем переменную со строкой

stroka = stroka.strip() #удаляем начальные и конечные проблемы

stroka = stroka.upper() #переводим буквы в верхний регистр

stroka = stroka[1:3] #выводим буквы с первого по третий регистр

print(stroka) #выводим переменную

**Задача 2: (7zadanie 2zada4a.py)**

stro = '+ otdixatb' #первая строка

stro2 = 'leshatb' #вторая строка

stro2 = stro2.upper() #перевод строк в верхний регистр

stro = stro.upper()

stro2 = stro2[0:6]+stro2[-1].lower() + stro[0:9]+stro2[-1].lower() #создаем срезы строк и применяем к 'B' перевод в нижний регистр,складываем строки

stro2 = ' '.join(stro2) #разделяем буквы двумя пробелами

print(stro2) #выводим конечный продукт (:

**Задача 3: (7zadanie 3zada4a.py)**

x = 'VSEM PRIVET, ETO ZASHITA OT CAPSLOKE' #строка в верхнем регистре

if x.isupper() == True: #проверка на верхний регистр

x = x.capitalize() #переводит в верхний регистр первую букву только самого первого слова строки

print(x) #вывод ст

**Задача 4: (7zadanie 4zada4a.py)**

x1 = 'Как же я устал от этого дерьма' #строка с ругательским словом

x1 = x1.replace('дерь','♡♡♡♡') #замена слова на другие символы

print(x1) #вывод строки

**Задача 5: (7zadanie 5zada4a.py)**

'''У тебя есть текст (ниче так текстик ) тебе нужно разбить его на стихотворение по 6 слов при этом автор текста не очен умен и вместо уроков русского он ходит в качалку и поэтому ча-ща и жи-ши не знает , и еще он забывает делать первые буквы по правилу оформления стихов заглавными, помоги ему сделать его стихи нормальными'''

a = 'когда мне встречяется в людях дурное, то долгое время я верить стараюсь, что это скорее всего напускное, ах, Что это случяйность. И я ошыбаюсь.' #текст который мы будем исправлять(строка)

e='' #задаем переменные

m=0

e1 = ''

for i in a: #заметим,что слов всегда на одно больше чем пробелов,

if i ==' ': #поэтому используем это для того,что бы отсчитать 6

m+=1 #слов,после чего сделать перенос строки

if i !=' ' or m % 6!=0:

e+=i

else:

e+='\n'

e = e.split('\n') #перевод строки в список по '\n'

for i in range(len(e)): #перебираем i в диапозоне длины e

e1 += e[i].capitalize() + '\n' #преобразовываем и переносим элементы списка,ссылаясь к их индексам

e1 = e1.replace('чя','ча')

e1 = e1.replace('шы','ши') #фильтр русского языка

print(e1)

# **Практическое занятие №8**

**‘Техника работы со списками’**

**Задача 1: (8zadanie.py)**

'''Задача: АнтиЧСВ: На вход подается предложение или даже текст в котором присутствует слово "я",

твоя цель удалить все местоимения 'я' кроме первой и посчитать колличество удалениний, и потом на основе соотношенения колво букв я / колво слов всего, выдать категорию ЧСВ'''

a = input() #цель задания:определить кол-во букв 'я',удалить все кроме первой,и вывести измененную строку,выдать категорию ЧСВ

a = a.lower()

verdikt = a

a = a.split(' ') #переводим строку в список про пробелам

ya = a.count('я') + a.count('я.') + a.count('я,') #считаем количество букв 'я' в списке

a.reverse() #переворачиваем список в обратном порядке для удобства применения метода .remove

ia = 0

while ia < ya - 1: #цикл удаления букв "я" из списка

if 'я.' in a:

a.remove('я.')

ia += 1

elif 'я,' in a:

a.remove('я,')

ia += 1

elif 'я' in a:

a.remove('я')

ia += 1

a.reverse() #обратно переворачиваем список

a = ' '.join(a) #конвертация в строку

a = a.capitalize() #переводим первую букву строки в верхний регистр

verdikt = verdikt.split(' ') #начало алгоритма выдачи категории на основе частоты встречающихся букв "я"

e = len(verdikt) / ya

print(a)

if e <= 22 and e > 10:

print('ЧСВ')

elif e <= 10:

print('К сожалению, вы Артем')

else:

print('Вы человек с хорошим характером')

print(ya)

# **Практическое занятие №9**

**‘Техника работы с циклом for и генераторами списков’**

**Задача 1**: (**9zadanie 1 zada4a.py)**

from random import randint #импорт функции генератора случайных чисел от первого аргумента до второго из модуля random

x1 = randint(2, 10) #генерируем случайный размер списка

spis = [] #создаем пустой список

for i in range(x1): #цикл который генерирует числа до нашего рандомного значения

randomik1 = randint(1,100) #генерируем случайное число от 1 до 100

spis.append(randomik1) #загоняем случайное число в конец списка

print(spis) #выводим наш случайно сгенерируемый список

.....................................................................................................................................

**Задача 2:** (**9zadanie 2 zada4a.py)**

''' Условие: Ты препод, который любит поразвлечься над студентами, поэтому

ты решил написать программу которая выбирает случайные(пронумерованные по сложности задания ) и создает 7 билетов в которых по 5 заданий'''

from random import \* #импорт модуля как набор стандартных функций

Sada4i = ['a1', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7', 'a8', 'a9', 'a10', 'b11', 'b12', 'b13', 'b14', 'b15', 'b16', 'c17',

'c18', 'c19', 'c20'] #список задач по сложности

billet = []

globalbillet = []

for i in range(7): #генерация 7 билетов

for x in range(5): #генерация 5 заданий

e = randint(0,19) #генератор индексов заданий

billet.append(Sada4i[e]) #добавление заданий в билет

globalbillet.append(billet) #добавление билетов в глобальную базу

billet = [] #обнуление билета

print(globalbillet) #вывод глобального билета

.....................................................................................................................................

**Задача 3**: (**9zadanie 3 zada4a.py)**

from random import \* #импорт модуля как набор стандартных функций

e = randint(1,20) #генератор случайного число

data = [i for i in range(e)] #генератор списка по диапозону случайного числа

print(data) #вывод списка

.....................................................................................................................................

**Задача 4**: (**9zadanie 4zada4a.py)**

from random import \*#импорт модуля как набор стандартных функций

e = randint(0, 150) #генератор случайного число

data = [i\*\*2 for i in range(e) if i % 3 != 0] #генарация списка с доп.условием(если i делится на три без остатка,то пропускает его)

print(data) #вывод списка

.....................................................................................................................................

**Задача 5: (9zadanie 5 zada4a.py)**

from random import \* #импорт модуля как набор стандартных функций

data = [i for i in 'strokakaynibud' if i == 'k' or i == 'a'] #генерация списка путем разбивания строки и добавление с учётом того,что i равняется 'k' или 'a'

print(data) #вывод списка

# **Практическое занятие №10.**

**‘Техника работы с функциями’**

**Задача 1: (10zadanie.py)**

import random

import math

'''

Цель Функционального программирования: Допустим ты разработчик и у тебя есть уже готовый продукт состоящий из 1000

строк кода, казалось бы, такую программу можно написать в обычном стиле, который не привязан к функциям(ФП) или

Объектам(ООП) но внезапно обнаруживается баг а ведь ты сидя над исправлением багов месяц думал что этого не могло

случиться, и если ты не писал код с помощью функция то скорее всего ты будешь смотреть много кода, и об игре в кс можешь

забыть, но в ФП ты бы понял с легкость какая из твоих функций косячит и накуканил бы ее на верный код, к тому же

функции можно использовать многоразово и вообще импортировать

'''

#######################################################

**Задача 2: (10zadanie.py)**

def sum(a, b): # Тупая изи функция Детерминированная

return a + b

print(sum(3, 6))

print(sum(3, 6))

#######################################################

**Задача 3:** (**10zadanie.py)**

print(random.random()) # Тупая не Детерминированная Функция (При одних и тех же локальных условиях

# значения могут быть разные)

#######################################################

**Задача 4:** (**10zadanie.py)**

sos = [32, 3, 50, 2, 29, 43]

def sort\_by\_sort(z): # Функция с побочкой она грязная, тк она изменяет начальный список и печатает его в консоль

z.sort()

print(z)

sort\_by\_sort(sos)

#####################################

**Задача 5: (10zadanie.py)**

def smart\_sort\_by\_sorted(spiska): # Функция без побочки

return sorted(spiska)

# мы с помощью принта печатаем функцию и тем самым мы печатаем ЗНАЧЕНИЕ КОТОРОЕ ЭТА ФУНКЦИЯ ВОЗВРАЩАЕТ(return)

e = smart\_sort\_by\_sorted(sos)

print(e)

#######################################################

**Задача 6: (10zadanie.py)**

def ФукцияОбычная(num): # Функция высшего порядка

return num \*\* 2

print(ФукцияОбычная(4))

#####################################

**Задача 7: (10zadanie.py)**

def ФункцияСложная(fun, num):

return fun(num) + fun(num)

print(ФункцияСложная(ФукцияОбычная, 4))

#######################################################

**Задача 8: (10zadanie.py)**

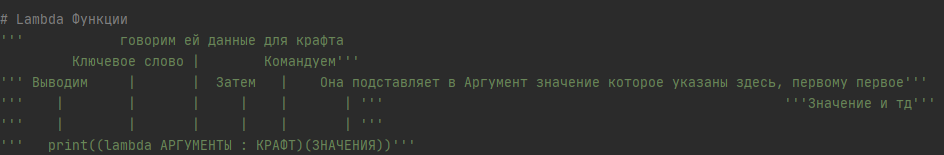


Рис.1.Объяснение lambda.

ОкругленноеПИ = round(math.pi, 2) # 3,14

# Площадь круга

def area(radius):

return ОкругленноеПИ \* (radius \*\* 2) # PI\*R^2

print(area(5))

# Теперь запишем ввиде lambda функции

print((lambda home: ОкругленноеПИ \* (home \*\* 2))(5))

# Объем усеченного конуса :

print(

(lambda Ширина, РадиусБ, РадиусМ: (Ширина \* ОкругленноеПИ \* (РадиусБ \*\* 2 + РадиусБ \* РадиусМ + РадиусМ \*\* 2)) / 3)(

15, 10, 6))

#######################################################

**Задача 9: (10zadanie.py)**

Chisla = [43, 23, 56, 75, 12, 32]

print(sorted(Chisla))

Strochanki = ['zum6z', '0ybt0', 'h1uwq', '2k9f9', 'hin9h', 'b0p0m']

print(sorted(Strochanki))

# Если партия требует сложную сортировку:

bookses = [

{'title': 'Game of Thrones', 'published': '1996-08-01', 'pages': 694},

{'title': 'Clash of Kings', 'published': '1998-11-16', 'pages': 761},

{'title': 'Storm of Swords', 'published': '2000-08-08', 'pages': 973},

{'title': 'Feast for Crows', 'published': '2005-10-17', 'pages': 753},

{'title': 'Dance with Dragons', 'published': '2011-07-12', 'pages': 1016}

]

def title(book):

return book.get('title')

bookses.sort(key=title)

for i in bookses:

print(i)

def publish\_date(book):

return book.get('published')

bookses.sort(key=publish\_date)

for i in bookses:

print(i)

def pages(book):

return book.get('pages')

bookses.sort(key=pages)

for i in bookses:

print(i)

# Сортируем по датам

bookses.sort(key=publish\_date)

for i in bookses:

print(i)

# Сортируем по количеству страниц

bookses.sort(key=pages)

for i in bookses:

print(i)

# Ещё кое что:

# s.sort(key=lambda x: x[1])

'''s.sort(key = lambda x: x[1])'''

# **Практическое занятие №11.**

**“Техника работы со словарями”**

**Задача 1: (11zadanie.py**)

slovak = {'Dovakin':2013,'Ivan':2010,'Masha':2011}

'''создаем словарь'''

slovares = dict()

'''альтернативный способ создания словаря'''

slovak.get('Dovakin','Не найдено')

'''возвращаем значения ключа.Если его нет,печатает 'Не найдено'.'''

slovak['Kobolt'] = slovak.pop('Dovakin')

'''заменяем ключ в нашем словаре'''

slovares = slovak.popitem()

'''удаляем из одного словаря ключ со значением и загоняем в другой словарь'''

slovak.update(Jesus = 0)

'''способ добавления ключа со значением в словарь'''

slovak.setdefault('ivan',90)

'''альтернативный способ добавления'''

print(slovak,slovares)

# **Практическое занятие №12.**

**“ Техника работы с множествами”**

**Задача 1: (12zadanie.py**)

def normalisupem(mnoj, alt):

return len(mnoj) <= len(alt)

'''функция которая возвращает True,если 'other' больше или равно 'set'.'''

a = [1,2,3,4,5,6,7,8]

a = set(a)

'''перевод списка в множество'''

a.remove(2)

'''удаляем из множества цифру 2'''

b = [13,221,31,22,66,55,12,111,88]

b = set(b)

'''перевод списка в множество'''

if a.isdisjoint(b):

print(1)

else:

print(0)

'''проверка на отсутствие общих значений'''

'''a = a.union(b)'''

'''объединение двух множеств'''

dlina = normalisupem(a,b)

'''запуск функции которую мы создали'''

print(a,dlina)

# **Практическое занятие №13.**

**“Техника работы с кортежами”**

**Задача 1: (13zadanie.py**)

d = tuple('Hello')

'''Создание кортежа из слова 'Hello'''

lst = [2, "abc", 3.88]

'''#Создание кортежа из списка'''

e = tuple(lst)

'''Создать кортеж'''

f = tuple((3, 2, 0, -5))

'''Создание кортежа из другого кортежа'''

a = ('a', 'bc', 'def', 'ghij')

item = a[2]

'''Кортежи. Операция взятия индекса'''

b = (a, a[1], True)

'''Кортеж, содержащий другой кортеж и его элементы'''

c = (1, [2, 3, 4], "text")

'''Кортеж, содержащий список'''

z = c[1]

'''Вытянуть список из кортежа'''

c[1][1] = 8

'''Список в кортеже изменяемый (mutable), поэтому его можно изменить'''

# **Практическое занятие №14.**

**“Техника работы с файлами”**

**Задача 1: (14zadanie.py**)

'''

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()

#####################################################

**Задача 2: (14zadanie.py**)

'''

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()

# **Практическое занятие №15.**

**“ Техника работы с модулями”**

**Задача 1: (15zadanie.py)**

'''Есть 4 разных вида вызовов модулей:'''

'''import <пакет>(

import <модуль>

from <пакет> import <модуль или подпакет или объект>

from <модуль> import <объект>'''

'''Так же можно импортировать следующим способом:

from random import \*

Это нам позволяет импортировать весь модуль,и так же сократить вызов функции,к примеру вот так:

x1 = randint(2, 10)

если бы не звезда,то эта строка выглядила бы так:

x1 = random.randint(2, 10)

'''

'''Чтоб вызвать СВОЮ функцию,нам понадобится полное название нашего проекта, в котором есть собственно созданные функции.

К примеру у нас есть проект под названием "Homework.py", и чтоб использовать функции из этого проэкта,нам понадобится вызвать это следующим образом:

import Homework

'''

# **Практическое занятие №16.**

**“Техника работы с классами”**

**Задача1: (16zadanie.py)**

# Объектно-ориентированное программирование: это парадигма программирования

# Структура создания ООП class 'name':

# \tАтрибуты и методы

# class Animals:

# class users:

class Car\_Ford:

# Атрибуты создаются так:

# Атрибуты определяют ключевые свойства класса его черты

model = 'Ford'

name = 'Fokus'

make = 2013

# Методы создаются так:

# Методами называются функции внутри класса

def start(self):

print("Заводим двигатель")

def stop(self):

print("Отключаем двигатель")

# Дальше для того, чтобы использовать классы нужно как и с функцией создать или призвать его

carA = Car\_Ford()

carB = Car\_Ford()

# Для вызова объектов внутри класса используется конструкция Экземпляк.Объект

# Обращаемся к методу

carB.start()

# Обращаемся к атрибуту

print(carB.model)

# Существуют локальные и глобальные атрибуты которые могут наследовать как все экземпляры класса

# так и быть уникальными для каждого

class Car\_New:

# Глобальный атрибут по умолчанию

car\_count = 0

# Метод в который помещены локальные настраиваемые атрибуты

def start(self, name, make, model):

print("Двигатель заведен")

self.name = name

self.make = make

self.model = model

# Не найдя car\_count внутри функции start Питон пойдет по правилу LEGB на уровель выше(E)

Car\_New.car\_count += 1

# self дает понять куда мы хатим засунуть атрибуты а именно себе то есть экземпляру

car\_A = Car\_New()

# Присваиваем индивидуальные значения с помощью локальных атрибутов

car\_A.start("Corrola", "Toyota", 2015)

print(car\_A.name)

print(car\_A.car\_count)

# Подробнее о методах класса

# Пример статичного метода

class Car\_New2:

@staticmethod

def get\_class\_details():

print("Это класс Car")

# При таком раскладе не нужно создавать экземпляр

Car\_New2.get\_class\_details()

# Конструктор — это специальный метод, который вызывается по умолчанию когда вы создаете объект класса.

# Для создания конструктора нужно создать метод с ключевым словом \_\_init\_\_

class Car:

# создание атрибутов класса

car\_count = 0

# создание методов класса

def \_\_init\_\_(self):

Car.car\_count += 1

print(Car.car\_count)

# Модификаторы доступа к классу

# публичный — public;

# приватный — private;

# защищенный — protected.

class CarAad:

def \_\_init\_\_(self):

print("Двигатель заведен")

self.name = "corolla"

self.\_\_make = "toyota"

self.\_model = 1999

# .\_\_Приватный

# .Публичный

# .\_Защищенный

car\_a = CarAad()

# Так как атрибут публичный все окей

print(car\_a.name)

# Так как атрибут приватный то выдаст ошибку

print(car\_a.make)

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

## Установка и настройка среды JetBrains PyCharm

PyCharm - это кросс-платформенная среда разработки.

Открываем 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. Нажимаем на кнопку «Next»:



Рисунок 108. Установщик PyCharm

В появившемся окне выберите путь для установки программы при помощи кнопки “Browse…” и нажмите кнопку “Next” для перехода к следующему этапу установки.



Рисунок 109. Место установки PyCharm

Программа начала установку PyCharm, нужно дождаться окончания.



Рисунок 110. Процесс установки PyCharm

После окончания установки, на рабочем столе компьютера появится ярлык с программой “PyCharm”

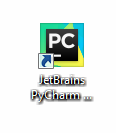


Рисунок 144. Ярлык на рабочем столе

Запускаем пограмму и в стартовом окне “PyCharm” создаем новые проект по кнопке “Create New Project”



Рисунок 111. Стартовое окно PyCharm

В следующем окне будет два поля. В первом укажите, где будет располагаться новый проект. Второе поле должно быть заполненным по умолчанию. Обычно в этом поле содержится путь к установленному ранее Python 

Рисунок 112. Выбор места для нового проекта и выбор интерпретатора

После нажатия кнопки “Create” откроется окно самого редактора.

Теперь вы можете начинать писать программы на Python!

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

Рисунок 113. Создание нового Phython файла

Появится окно, в котором вы можете задать имя файлу. Напишите любое понравившиеся имя и нажмите «OK».

В правом окне откроется созданный файл. Пока что он пустой. Пишем:



Рисунок 114. Написание программы

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



Рисунок 115. Запуск программы

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



Рисунок 116. 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 77. SQLite.py (Разработал Михайлов Д.А)

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/>.



Рис. 117. Установщик DB Browser

После установки, открываем DB Browser для работы с ним.

Рис. 118. Окно DB Browser

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

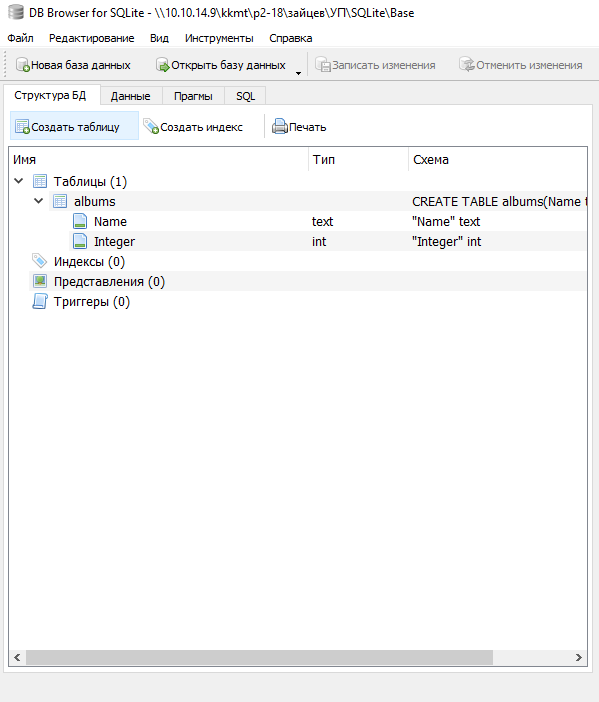


Рис. 119. Таблица Name и Integer в DB Browser

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

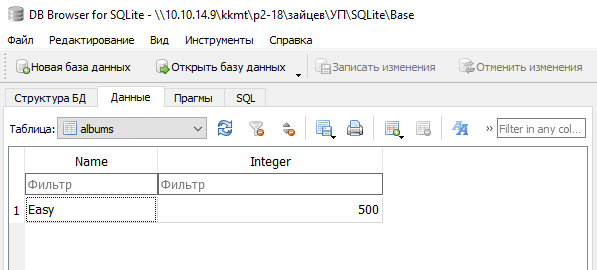


Рис. 120. Данные таблицы students

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 78: Tkinter.py (Разработал Зайцев Никита)

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:

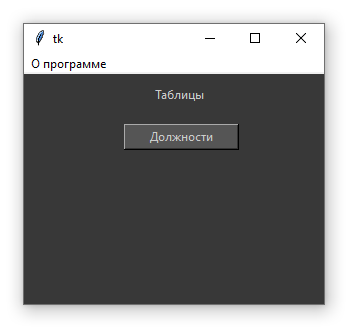


Рис. 121. Вид формы

Жмем на кнопку “Должности” и заполняем форму данными:

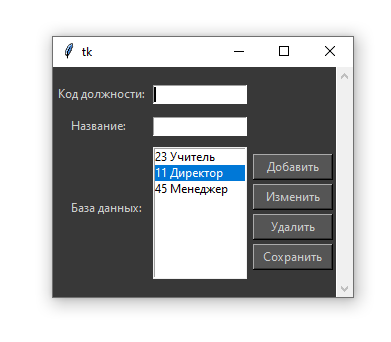


Рис. 122. Заполненная форма

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

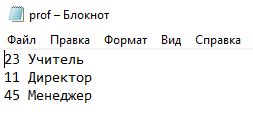


Рис. 123. Вывод в файл

# 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 79: Numpy.py (Разработал Пилипушко А.С)

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.'''

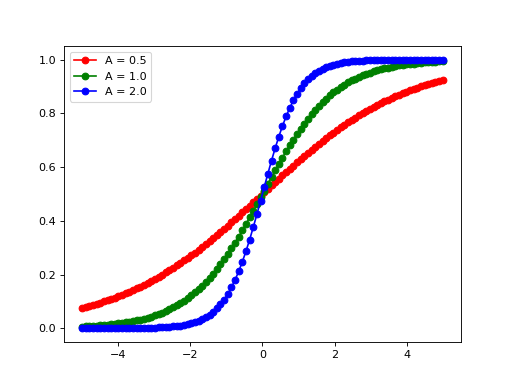
plt.legend(['A = 0.5', 'A = 1.0', 'A = 2.0'], loc = 'upper left')

'''Легенда в верхнем правом углу.'''

fig.savefig('sigmoid.png')

'''Сохранение PNG-картинки.'''

Рис. 124. 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 80: mat.py (Разработал Герасимов Д.А)

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*

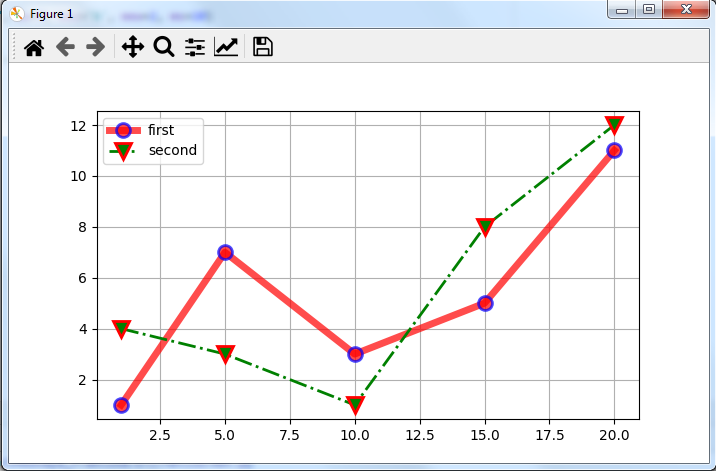


Рис. 125. Вывод программы

Листинг 81: mat2.py (Разработал Герасимов Д.А)

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()

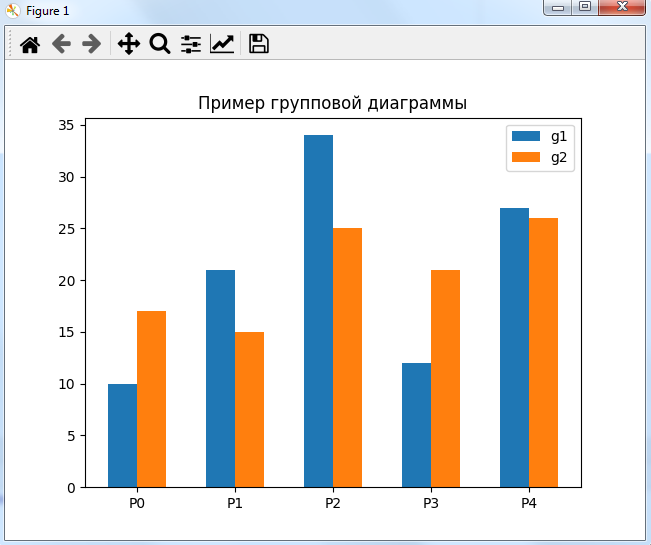


Рис. 126. Вывод программы

# 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 82: 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 83: 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", "+"))

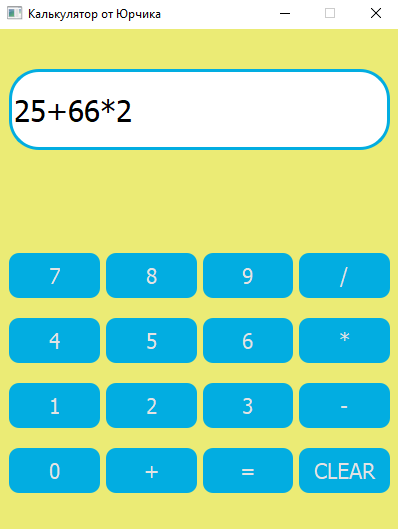


Рис. 127. Окно калькулятора

# 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 83. PyGame.py (Разработал Михайлов Д.А)

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 128. 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

1. Пример числовых и текстовых значений из базы данных



Рис. 129. База данных

1. Окно таблицы без вывода Авто

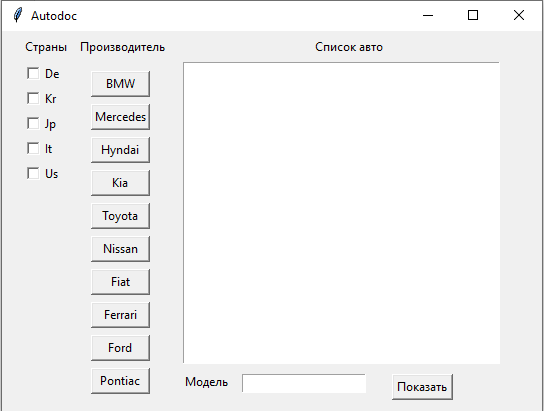


Рис. 130. Окно таблицы

1. Пример вывода в таблицу с сортировкой по выбраному производителю:

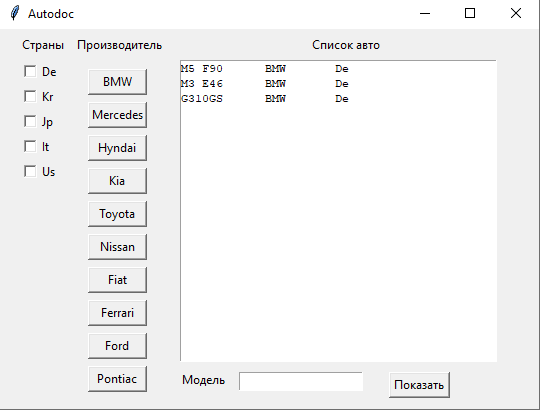


Рис. 131. Вывод по производителю “BMW”

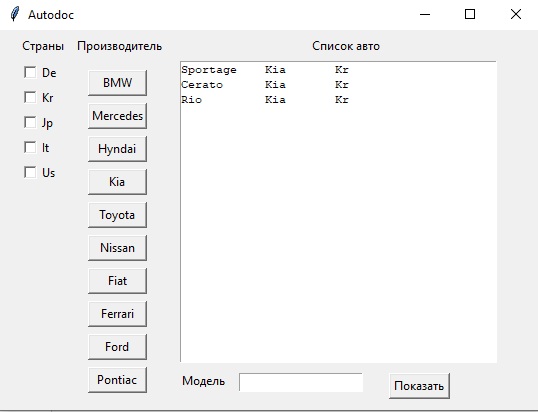


Рис.132. Сортировка по производителю “KIA”

Пример вывода в таблицу с сортировкой по разным странам

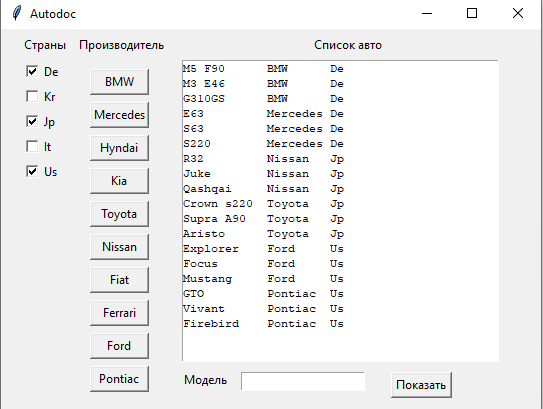


Рис.133. Сортировка по 3 странам

3.2 Разработка требований к проекту. Построение диаграммы использования.

Требования к проекту:

1. Корректное использования данных
2. Работа с функциями сортировки
3. Загрузка данных в визуальный интерфейс

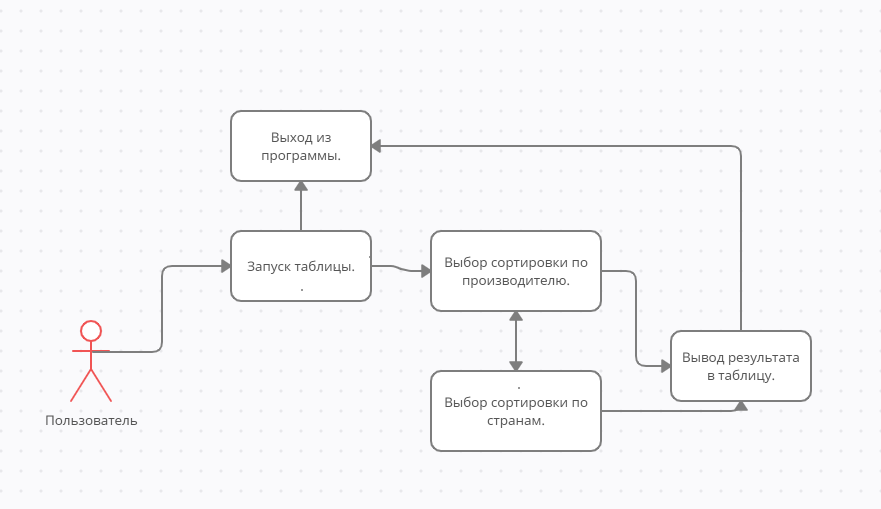


Рисунок 135. Диаграмма использования

* 1. Разработка сценария проекта

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

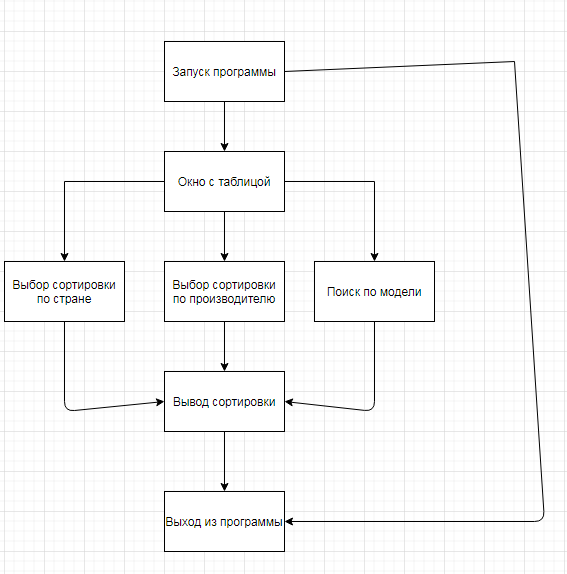


Рис. 136. Сценарий проекта

* 1. Построение диаграммы классов

1. В данном разделе находятся диаграммы классов приложения.

(Рисунок 137)

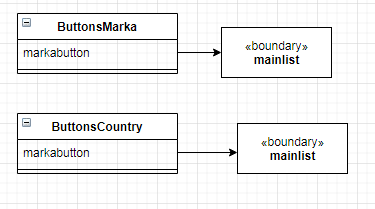


Рис. 137. Диаграмма классов

* 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.

Листинг 85. Марки.py

import sqlite3

# Создание списка таблици

''' Model Proiz Country'''

text = [('M5 F90 ','BMW ','De'),

('E63 ','Mercedes ','De'),

('Solaris ','Hyndai ','Kr'),

('Sportage ','Kia ','Kr'),

('S63 ','Mercedes ','De'),

('Crown s220 ','Toyota ','Jp'),

('M3 E46 ','BMW ','De'),

('Sonata ','Hyndai ','Kr'),

('R32 ','Nissan ','Jp'),

('Explorer ','Ford ','Us'),

('500 ','Fiat ','It'),

('S220 ','Mercedes ','De'),

('SF90 ','Ferrari ','It'),

('Cerato ','Kia ','Kr'),

('Supra A90 ','Toyota ','Jp'),

('GTO ','Pontiac ','Us'),

('200 ','Fiat ','It'),

('i20 ','Hyndai ','Kr'),

('Juke ','Nissan ','Jp'),

('Punto ','Fiat ','It'),

('Vivant ','Pontiac ','Us'),

('488 ','Ferrari ','It'),

('Rio ','Kia ','Kr'),

('G310GS ','BMW ','De'),

('Aristo ','Toyota ','Jp'),

('f40 ','Ferrari ','It'),

('Focus ','Ford ','Us'),

('Qashqai ','Nissan ','Jp'),

('Mustang ','Ford ','Us'),

('Firebird ','Pontiac ','Us')]

# Создание и запись списка в таблицу

conn = sqlite3.connect('MarkaTable')

curs = conn.cursor()

curs.execute("""CREATE TABLE Model(

Model TEXT,

Proiz TEXT,

Country TEXT)""")

curs.executemany("INSERT INTO Model 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()

'''

* 1. Разработка интерфейса в tkinter

Главный модуль состоит из двух классов ButtonsMarka и ButtonsCountry. Он выполняется после на определенные кнопки связанные с этими классами. Его также можно назвать главным меню.

Листинг 84:  [AutoTable.py](https://github.com/prog-kkkmt/p1-18/blob/Gymrasimov/%D0%A3%D0%9F/project_w_bd/sqlighter.py)

from tkinter import \*

import sqlite3

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

class ButtonsMarka:

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

self.var = var

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

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

def markabutton(self):

conn = sqlite3.connect('MarkaTable')

curs = conn.cursor()

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

FROM Model

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

mainlist.delete(0, END)

for i in gridmarka:

\*i, = i

rowtext.set(str(i[0]) + ' ' + str(i[1]) + ' ' + str(i[2]))

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

class ButtonsCountry:

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

self.i = 0

self.var = var

self.turn = IntVar()

self.markabuttons = Checkbutton(text=self.var, command=self.markabutton,

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

def markabutton(self):

global grid

conn = sqlite3.connect('MarkaTable')

curs = conn.cursor()

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

self.i = grid.index('')

grid[self.i] = self.var

else:

grid[self.i] = ''

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

FROM Model

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

ORDER BY Country, Proiz""", grid)

mainlist.delete(0, END)

for i in gridmarka:

\*i, = i

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

str(i[2]))

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

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

# Функции для кнопок "Показать"

def getmarka():

global rowtext, mainlist, namebox

conn = sqlite3.connect('MarkaTable')

curs = conn.cursor()

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

FROM Model

WHERE Model LIKE ?

ORDER BY Model""",\

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

mainlist.delete(0, END)

for i in gridmarka:

\*i, = i

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

str(i[2]))

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

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

# Создание окна и переменных

tk = Tk()

tk.title('Autodoc')

tk.geometry('540x380+700+300')

rowtext = StringVar()

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

namemodel = StringVar()

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

'''

minr = DoubleVar()

maxr = DoubleVar()

mink = DoubleVar()

maxk = DoubleVar()

minh = DoubleVar()

maxh = DoubleVar()

'''

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

# Создание названий разделов и кнопок

nametext = Label(tk, text='Модель').place(x=180, y=340)

teamtext = Label(tk, text='Производитель').place(x=75, y=5)

countrytext = Label(tk, text='Страны').place(x=20, y=5)

ButtonsMarka('BMW', 89, 40)

ButtonsMarka('Mercedes', 89, 73)

ButtonsMarka('Hyndai', 89, 106)

ButtonsMarka('Kia', 89, 139)

ButtonsMarka('Toyota', 89, 172)

ButtonsMarka('Nissan', 89, 205)

ButtonsMarka('Fiat', 89, 238)

ButtonsMarka('Ferrari', 89, 271)

ButtonsMarka('Ford', 89, 304)

ButtonsMarka('Pontiac', 89, 337)

ButtonsCountry('De', 20, 30)

ButtonsCountry('Kr', 20, 55)

ButtonsCountry('Jp', 20, 80)

ButtonsCountry('It', 20, 105)

ButtonsCountry('Us', 20, 130)

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

# Создание окон ввода и кнопок "Показать"

namebox = Entry(tk, width=20, textvariable=namemodel)

namebox.place(x=240, y=343)

namebutton = Button(tk, text='Показать', command=getmarka).place(x=390, y=343)

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

# Создание окна для показа таблици

maintext = Label(tk, text = 'Список авто').place(x=310, y=5)

mainlist = Listbox(tk, width=45, height=20, font='Courier 8')

mainlist.place(x=180, y=30)

mainloop()

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. |  |