

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

Выполнили студенты:

Джабраилов Т. А.

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

Завадский М. А.

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

Толоконников А. М.

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

Проверил преподаватель:

Гусятинер Леонид Борисович

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

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

Королев, 2021

**Оглавление**

1. **Техника решения задач с использованием структурного и объектно- ориентированного программирования**
   1. [Установка Интерпретаторов Python 3 и настройка окружения](#п11)
   2. [Техника работы в командной строке и среде IDLE](#п12)
   3. [Техника работы с линейными и разветвляющимися программами](#п13)
   4. [Техника работы с циклическими программами, цикл while](#п14)
   5. [Техника работы с числами](#п15)
   6. [Техника работы со строками](#п16)
   7. [Техника работы со списками](#п17)
   8. [Техника работы с циклом for и генераторами списков](#п18)
   9. [Техника работы с функциями](#п19)
   10. [Техника работы со словарями](#п110)
   11. [Техника работы с множествами](#п111)
   12. [Техника работы с кортежами](#п112)
   13. [Техника работы с файлами](#п113)
   14. [Техника работы с модулями](#п114)
   15. [Техника работы с классами](#п115)
2. [**Техника решения задач с использованием библиотек**](#п2)
   1. [Установка и настройка среды JetBrains PyCharm](#п21)
   2. [Техника работы с базами данных](#п22)
   3. [Техника работы с библиотекой tkinter](#п23)
   4. [Техника работы с библиотекой NumPy](#п24)
   5. [Техника работы с базами данных Matplotlib](#п25)
   6. [Техника работы с библиотекой PyQt](#п26)
   7. [Техника работы с библиотекой Pygame](#п27)
3. [**Разработка проекта с графическим интерфейсом**](#п3)
   1. [Изучение входной и выходной документации](#п31)
   2. [Дневник](#п32)

# **1.1** Установка Интерпретаторов Python 3 и настройка окружения

Обучающий ролик по установке питона: https://yadi.sk/i/L4ApJ8CK3ns7Ow

Материал из ролика:

ссылка на скачивание: <https://www.python.org/downloads/windows/>

Текстовый вариант:

1. Заходим на сайт https://www.python.org/downloads/.

2. Нажимаем на кнопку “Download Python 3.9.0”.

3. Ожидаем скачивания программы.

4. Запускаем, ставим галочку “Add Python 3.9 to PATH” и начинаем установку.

5. Дожидаемся установки и начинаем работать с Python.

## **1.2** Техника работы в командной строке и среде IDLE

## Для начала работы откроем командную строку

Что-бы открыть командную строку нужно зажать комбинацию клавиш: Win+ R

После чего нужно ввести слово (cmd) для открытия командной строки

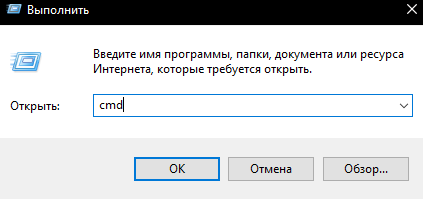


рис.1 Открытие командной строки

После выполнения команды появляется пустая командная строка рис.2

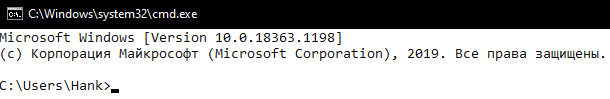


рис.2 Пустая командная строка

Изначально командная строка открывает папку с пользователем, что бы перейти в корневую папку диска нужно ввести данную команду:



рис.3 Папка с пользователем

Но часто командная строка не даёт как-либо взаимодействовать с корневой папкой диска, поэтому создаем новую папку



рис.4 Создание новой папки

Мы создали новую папку

После чего нам нужно перейти в неё



Рис.5 Переход в новую папку

Теперь мы можем создать файл с кодом

Для этого введем следующую команду



Рис.6 Создание файла с кодом

Мы создали новый пустой файл с расширением python

Далее мы можем ввести код

После того как мы закончим с написанием кода, нажмем на клавишу Enter, после чего нажмем на клавишу f6 и Enter

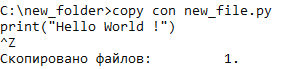


Рис.7 Написание кода

По итогу мы видим, что был создан (скопирован) данный файл

Давайте запустим его

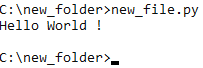


Рис.8 Созданный файл

В итоге мы видим, что программа запустилась и вывела код

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

Листинг 1 **Задача №1(Begin)**

# Выполнил Завадский Михаил П2-18.

# Дана сторона квадрата a. Найти его периметр P = 4\*a.

a = int(input())

a = a \* 4

print(a)

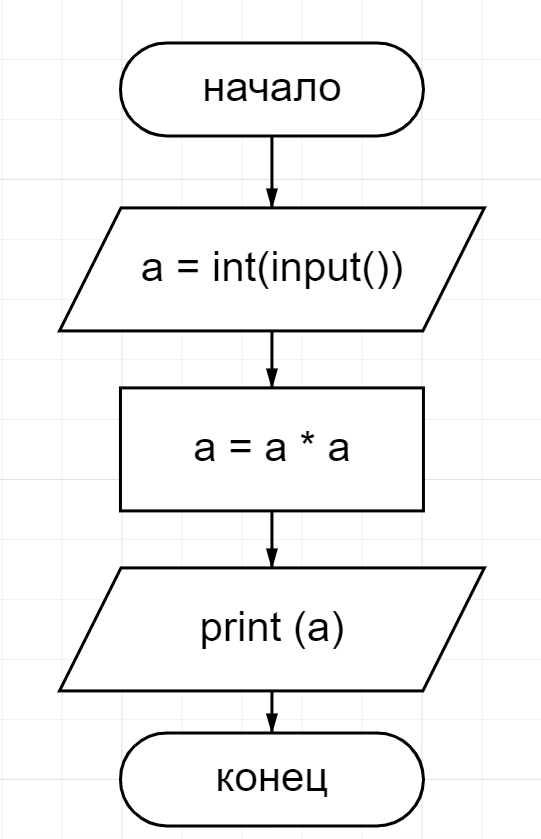


Рис.9 блок –схема 1

Листинг 2 **Задача №2(Begin)**

# Выполнил Завадский Михаил П2-18.

# Дана сторона квадрата a. Найти его площадь S = a^2.

a = int(input())

a = a \* a

print(a)

Листинг 3 **Задача №1(Bool)**

# Выполнил Завадский Михаил П2-18.

# Дано целое число A.

# Проверить истинность высказывания: «Число A является

# положительным».

x = int(input())

x = x > 0

print(x)

Листинг 4 **Задача №2(Bool)**

# Выполнил Завадский Михаил П2-18.

# Дано целое число A.

# Проверить истинность высказывания: «Число A является

# нечетным».

a = int(input())

a = a % 1 == 0

print(a)

Листинг 5 **Задача №1(if)**

# Сделал Толоконников А. М. П2-18.

# Дано целое число. Если оно является положительным, то

# прибавить к нему 1;

# в противном случае не изменять его. Вывести полученное число.

x = int(input())

if x > 0:

x = x + 1

print(x)

else:

print(x)

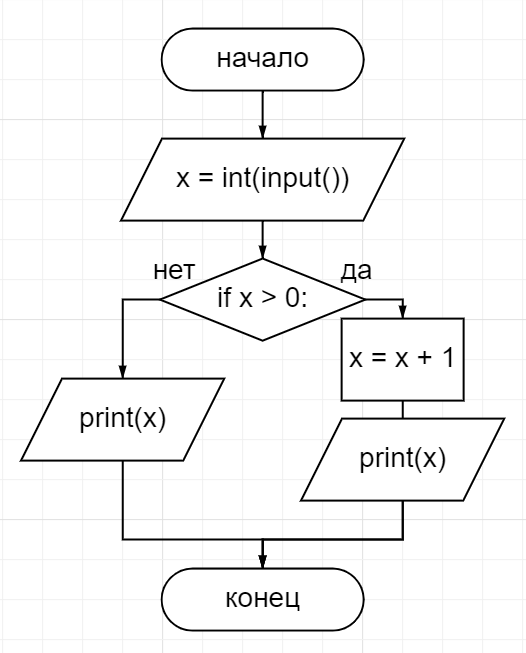


Рис.10 блок –схема 2

Листинг 6 **Задача №2(if)**

# Сделал Толоконников А. М. П2-18.

# Дано целое число. Если оно является положительным, то

# прибавить к нему 1;

# в противном случае вычесть из него 2. Вывести полученное число.

x = int(input())

if x > 0:

x = x + 1

print(x)

else:

x = x - 2

print(x)

Листинг 7 **Задача №3(if)**

# Сделал Джабраилов Тимур П2-18.

# Дано целое число. Если оно является положительным, то

# прибавить к нему 1; если

# отрицательным, то вычесть из него 2; если нулевым, то заменить

# его на 10.

# Вывести полученное число

x = int(input())

if x > 0:

x = x + 1

print(x)

elif x < 0:

x = x - 2

print(x)

else:

x = 10

print(x)

Листинг 8 **Задача №4(if)**

# Сделал Толоконников А. М. П2-18

# Дано 10 целых чисел. Вычислите их сумму. Напишите программу,

# использующую наименьшее число переменных.

sum = 0

for i in range(10):

n = int(input())

sum+= n

print(sum)

Листинг 9 **Задача №5(if)**

# Сделал Толоконников А. М. П2-18

# По данному натуральному n вычислите сумму 1 \*\* 3 + 2 \*\* 3 + 3

# \*\* 3 + ... + n \*\* 3.

n = int(input())

sum = 0

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

sum += i \*\* 3

print(sum)

Листинг 10 **Задача №1(case)**

# Выполнил Завадский Михаил П2-18.

# 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())

if 0 < x < 8:

if x == 1:

print("Понедельник")

elif x == 2:

print("Вторник")

elif x == 3:

print("Среда")

elif x == 4:

print("Четверг")

elif x == 5:

print("Пятница")

elif x == 6:

print("Суббота")

elif x == 7:

print("Воскресенье")

else:

print("Неправильное число")

Листинг 11 **Задача №2(case)**

# Выполнил Джабраилов Тимур П2-18.

# 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())

if 0 < x < 6:

if x == 1:

print("Плохо")

elif x == 2:

print("Неудовлетворительно")

elif x == 3:

print("Удовлетворительно")

elif x == 4:

print("Хорошо")

elif x == 5:

print("Отлично")

else:

print("Ошибка")

Листинг 12 **Задача №3(case)**

# Сделал Толоконников А. М.П2-18.

# Дано целое число. Если оно является положительным, то

# прибавить к нему 1;если

# отрицательным, то вычесть из него 2;если нулевым, то заменить

# его на 10.

# Вывести полученное число

x = int(input())

if x > 0:

x = x + 1

print(x)

elif x < 0:

x = x - 2

print(x)

else:

x = 10

print(x)

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

Листинг 13 **Задача №1(for)**

# Сделал Толоконников А. М. П2-18

# Даны два целых числа A и B (при этом A ≤ B). Выведите все

# числа от A до B включительно.

a = int(input())

b = int(input())

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

print(i)

Листинг 14 **Задача №2(for)**

# Сделал Толоконников А. М. П2-18

# 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):

print(i)

else:

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

print(i)

Листинг 15 **Задача №3(for)**

# Сделал Толоконников А. М. П2-18

# Даны два целых числа A и В, A>B. Выведите все нечётные числа

# от

# A до B включительно, в порядке убывания.

# В этой задаче можно обойтись без инструкции if.

a = int(input())

b = int(input())

for i in range (a - (a + 1) % 2, b - b % 2, -2):

print(i, end=' ')

Листинг 16 **Задача №4(for)**

#Сделал Толоконников А. М. П2-18

#Дано 10 целых чисел. Вычислите их сумму. Напишите программу, использующую наименьшее число переменных.

sum = 0

for i in range(10):

n = int(input())

sum+= n

print(sum)

Листинг 17 **Задача №5(for)**

# Сделал Толоконников А. М. П2-18

# По данному натуральному n вычислите сумму 1 \*\* 3 + 2 \*\* 3 +

# 3 \*\* 3 + ... + n \*\* 3.

n = int(input())

sum = 0

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

sum += i \*\* 3

print(sum)

Листинг 18 **Задача №1(while)**

# Сделал Толоконников А. М. П2-18.

# По данному целому числу N распечатайте все квадраты

# натуральных чисел, не превосходящие N, в порядке возрастания.

n = int(input())

i = 1

while i \*\* 2 <= n:

print(i \*\* 2)

i += 1

Листинг 19 **Задача №2(while)**

# Сделал Толоконников А. М. П2-18.

# Дано целое число, не меньшее 2.

# Выведите его наименьший натуральный делитель,отличный от 1.

n = int(input())

i = 2

while n % i != 0:

i += 1

print(i)

Листинг 20 **Задача №3(while)**

# Сделал Толоконников А. М. П2-18.

# По данному натуральному числу N найдите наибольшую целую

# степень двойки, не превосходящую N.

# Выведите показатель степени и саму степень.

n=int(input())

a = 2

i = 1

while a <= n:

a \*= 2

i += 1

print(i - 1, a // 2)

Листинг 21 **Задача №4(while)**

# Выполнил Завадский Михаил П2-18.

# 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.

len = 0

while int(input()) != 0:

len += 1

print(len)

Листинг 22 **Задача №5(while)**

# Выполнил Завадский Михаил П2-18.

# Определите сумму всех элементов последовательности,

#завершающейся числом 0.

# В этой и во всех следующих задачах числа, следующие за первым нулем, учитывать не нужно.

sum = 0

element = int(input())

while element != 0:

sum += element

element = int(input())

print(sum)

**1.5** Техника работы с числами

Листинг 23 **Задача по модулю fraction**

# Завадский Михаил Задание разбор модуля Fraction

# Модуль fractions

from fractions import Fraction

print (Fraction())

# по умолчанию

(Fraction(0, 1))

print (Fraction(2, 4), Fraction(3, 9), Fraction(4, 16))

# если есть общий делитель то, числа будут сокращены

print (Fraction('1/3').\_\_round\_\_())

# округляет до ближайшего чётного числа

print (Fraction(1, 3) + Fraction(1, 4))

# также можно выполнять разные математические действия

print (Fraction(1, 3) - Fraction(1, 4))

print (Fraction(1, 3) \*\* Fraction(1, 4))

print (Fraction(1, 3) % Fraction(1, 4))

print (Fraction(1, 3) \* Fraction(1, 4))

Листинг 24 **Задача по модулю decimal**

# Сделал Толоконников Алексей Михайлович П2-18

# Decimal- вычисления с заданной точностью

from decimal import Decimal

# Округление чисел

# Объекты Decimal имеют метод quantize(),который

# позволяет округлять числа.

# В этот метод в качестве первого аргумента передается

# также объект Decimal,

# который указывает формат округления числа:

number = Decimal("0.444")

number = number.quantize(Decimal("1.00"))

print(number)

number = Decimal("0.555678")

print(number.quantize(Decimal("1.00")))

number = Decimal("0.999")

print(number.quantize(Decimal("1.00")))

# Как отделить рубли от копеек в вещественных числах?

# Decimal вместо float:

x = Decimal('100.25')

a = int(x)

b = int(100 \* (x - a))

print(a, b)

Листинг 25 **Задача по модулю complex**

# Сделали Толоконников Алексей Михайлович, Завадский Михаил

# Андреевич

# Модуль Complex

a = complex(10, 2)

print(a)

b = complex(6, 4)

print( b)

c = a + b

print(c)

print(a.imag) # Мнимая часть

print(b.real) # Действительная часть

print(a == b) # Проверка равенства

print(abs(a)) # Модуль комплексного числа

print(pow(b, 5))# Возведение в пятую степень комплексного числа

**1.6** Техника работы со строками

Листинг 26 **Задача №1(stroka)**

# Сделал Толоконников Алексей Михайлович

# Дана строка, состоящая из слов, разделенных пробелами.

# Определите, сколько в ней слов.

# Используйте для решения задачи метод count.

s = input()

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

Листинг 27 **Задача №2(stroka)**

# Сделал Толоконников Алексей Михайлович

# Дана строка. Разрежьте ее на две равные части (если длина

# строки — четная,

# а если длина строки нечетная, то длина первой части должна

# быть на один символ больше).

# Переставьте эти две части местами, результат запишите в новую

# строку и выведите на экран.

# При решении этой задачи не стоит пользоваться инструкцией if.

s = input()

l = len(s)//2+len(s)%2

a = s[l:]

b = s[:l]

print(a+b)

Листинг 28 **Задача №3(stroka)**

# Сделал Толоконников Алексей Михайлович

# Дана строка, состоящая ровно из двух слов, разделенных

# пробелом.

# Переставьте эти слова местами. Результат запишите в строку и

# выведите получившуюся строку.

# При решении этой задачи не стоит пользоваться циклами и

# инструкцией if.

s = input()

first\_word = s[:s.find(' ')]

second\_word = s[s.find(' ') + 1:]

print(second\_word + ' ' + first\_word)

Листинг 29 **Задача №4(stroka)**

# Сделал Толоконников Алексей Михайлович

# 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.

a = input()

print(a[2])

print(a[-2])

print(a[0:5])

print(a[0:-2])

print(a[::2])

print(a[1::2])

print(a[::-1])

print(a[::-2])

print(len(a))

**1.7** Техника работы со списками

Листинг 30 **Задача №1(список)**

# выполник Завадский Михаил

# Дан список чисел. Выведите значение наибольшего элемента в

# списке,

# а затем индекс этого элемента в списке.

# Если наибольших элементов несколько, выведите индекс первого

# из них.

a = input().split()

n = 0

for i in range(len(a)):

a[i]=int(a[i])

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

if a[i]>a[i-1] and a[i]>a[i+1]:

n+=1

print(n)

Листинг 31 **Задача №2(список)**

# Выполнил Толоконников Алексей Михайлович

# Дан список чисел. Посчитайте, сколько в нем пар элементов,

# равных друг другу. Считается, что любые два элемента,

# равные друг другу образуют одну пару, которую необходимо

# посчитать.

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

m = min(a)

M = max(a)

n1, n2, n3, y = 0,0,0,0

for i in range(m, M+1):

x = a.count(i)

for l in range(0, x):

n2 = n3

n1 = l

n3 = n1 + n2

y += n3

n1, n2, n3 = 0,0,0

print(y)

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

Листинг 32

# Выполнил Толоконников Алексей Михайлович П2-18.

# Дана квадратная матрица порядка M. Обнулить элементы матрицы,

# лежащие ниже главной диагонали. Условный оператор не

# использовать.

import random as rnd

m = rnd.randrange(2, 10, 2)

n = m

z = [rnd.randrange(0, 100) for i in range(m)]

print('Матрица размером:', m, 'на', m)

x = [z] \* n

print(\*x, sep='\n')

# **1.9** Техника работы с функциями

Листинг 33

# Выполнил Толоконников Алексей Михайлович П2-18.

# Даны четыре действительных числа: x1, y1, x2, y2.

# Напишите функцию distance(x1, y1, x2, y2), вычисляющая

# расстояние между точкой (x1,y1) и (x2,y2).

# Считайте четыре действительных числа и выведите результат работы

# этой функции.

from math import sqrt

def distance(x1, y1, x2, y2):

return sqrt((x1 - x2) \*\* 2 + (y1 - y2) \*\* 2)

x1 = float(input())

x2 = float(input())

y1 = float(input())

y2 = float(input())

print(distance(x1, x2, y1, y2))

# **1.10** Техника работы со словарями

Листинг 34

# Выполнил Толоконников Алексей Михайлович П2-18

# 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 или большим числом пробелов или символами # конца строки.

counter = {}

for word in input().split():

counter[word] = counter.get(word, 0) + 1

print(counter[word] - 1, end=' ')

# **1.11** Техника работы с множествами

Листинг 35

# Выполнил Толоконников Алексей Михайлович П2-18.

# Даны два списка чисел. Посчитайте, сколько чисел содержится

# одновременно

# как в первом списке, так и во втором.

a={}

a=set(input().split())

b={}

b=set(input().split())

c=a.intersection(b)

print(len(c))

# **1.12** Техника работы с кортежами

Листинг 36

**Задача №1(кортеж)**

# Выполнил Завадский Михаил П2-18.

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

a= int(input())

b=a

if b%2==1:

b=b+1

print(tuple(range(b,a\*10+1,2)))

Листинг 37

**Задача №2(кортеж)**

# Выполнил Завадский Михаил П2-18.

# С клавиатуры вводятся целые числа a > b.

# Выведите убывающую последовательность чисел по одному числу в

# строке.

a=int(input())

b=int(input())

c=tuple(range(a,b,-1))

for i in c:

print(i)

# **1.13** Техника работы с файлами

Листинг 38

# Выполнил Завадский Михаил Андреевич П2-18.

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

v = open('text.txt','a')

v.write('fwefwefwef')

v.close()

# **1.14** Техника работы с модулями

Листинг 40

# Выполнил Толоконников Алексей Михайлович П2-18.

# Контейнерные типы данных модуля collections.

# Класс Counter() модуля collections в Python.

# 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

# >>> cnt = Counter(['red', 'blue', 'red', 'green', 'blue',

# 'blue'])

# >>> dict(cnt)

# {'blue': 3, 'red': 2, 'green': 1}

# Синтаксис:

# import collections

# cnt = collections.Counter([iterable-or-mapping])

# Параметры:

# iterable-or-mapping - итерируемая последовательность или

# словарь.

# Пример работы с ним:

import collections

c = (['g', 'b', 'c', 'd', 'g', 'f', 'f', 'f', 'g', 'c', 'd'])

print('Оно помогает выводить количество неизменяемых элементов.\nПоследовательность: ', c)

print('Сколько в ней элементов:')

print(collections.Counter(c))

# **1.15** Техника работы с классами

Листинг 41

# Подготовлено Завадским Михаил Андреевичем П2-18

from tkinter import \*

# импортируем библиотеку для работы с окнами

class Window:

# self говорит о том, что переменная принадлежит данному классу

def \_\_init\_\_(self, width, height, title="MyProblem", resizable=(False, False), icon=None):

# отвечает какими параметрами изначально владеет наше окно

self.root = Tk()

# корневая переменная хранит Tk

self.root.title(title)

# название нашего окна

self.root.geometry(f"{width}x{height}+700+500")

# геометрия нашего окна(начальное расположение и рзарешение)

self.root.resizable(resizable[0], resizable[1])

# возможность изменения окна по координатам (x, y)

if icon:

self.root.iconbitmap(icon)

# Иконка

self.label = Label(self.root, text="Интересный текст")

# виджет label. Нужен для работы с текстом в окне

def run(self):

self.draw\_wigets()

self.root.mainloop()

def draw\_wigets(self):

self.label.pack()

# настройка виджетов

# пользуемся классом Window :

from window import Window

# импортируем класс window

window = Window(400, 360)

# настраиваем параметры окна

window.run()

# запускаем окно

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

2.1 Установка и настройка среды JetBrains PyCharm

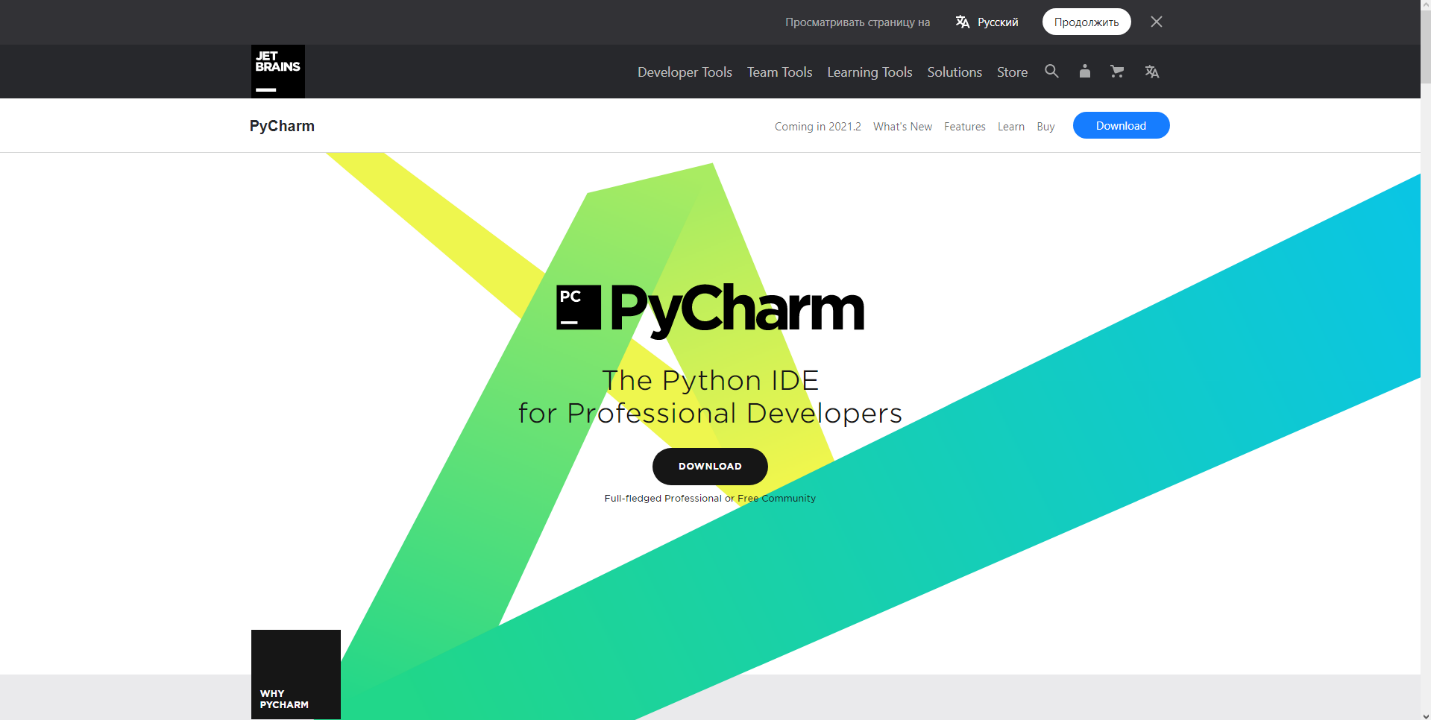


Рис.11. Для установки PyCharm нужно перейти на официальный сайт.

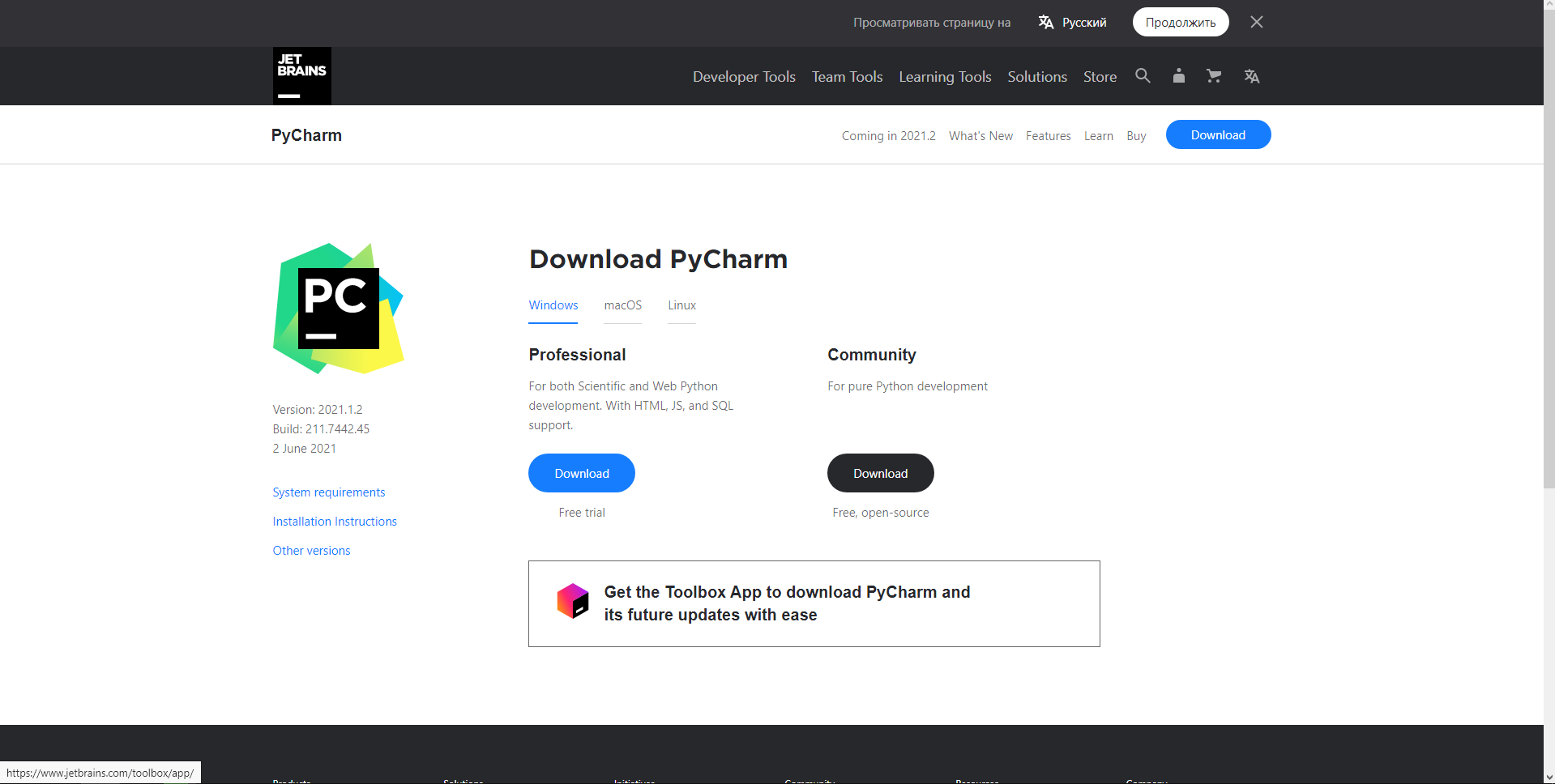


Рис.12. Выбираем версию “Community”, нажимаем на кнопку загрузки.

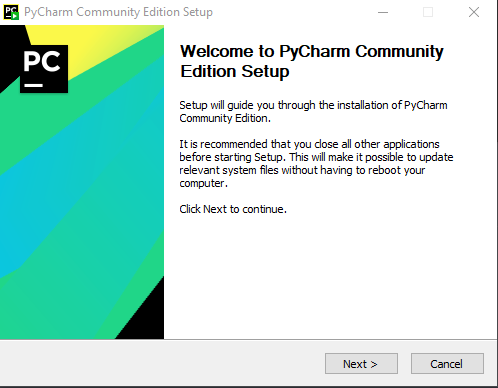


Рис.13. После открытия файла, наc встретит загрузчик, продолжим.

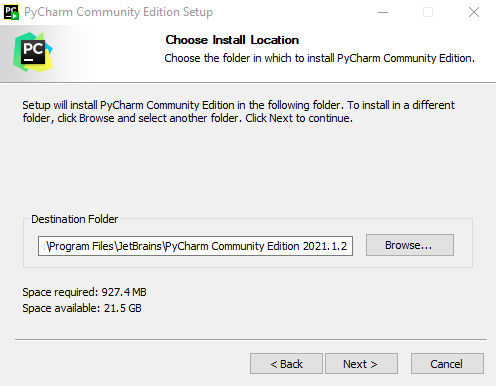


Рис.14. Выбираем папку установки и продолжаем.

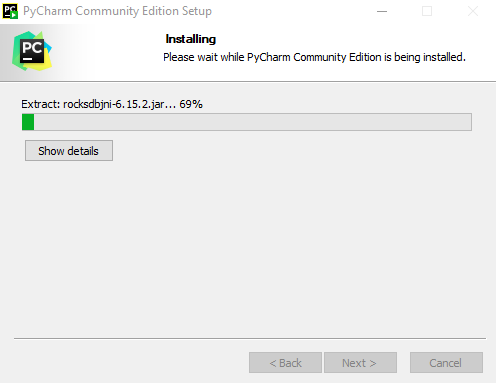


Рис.15. После чего начнется установка.

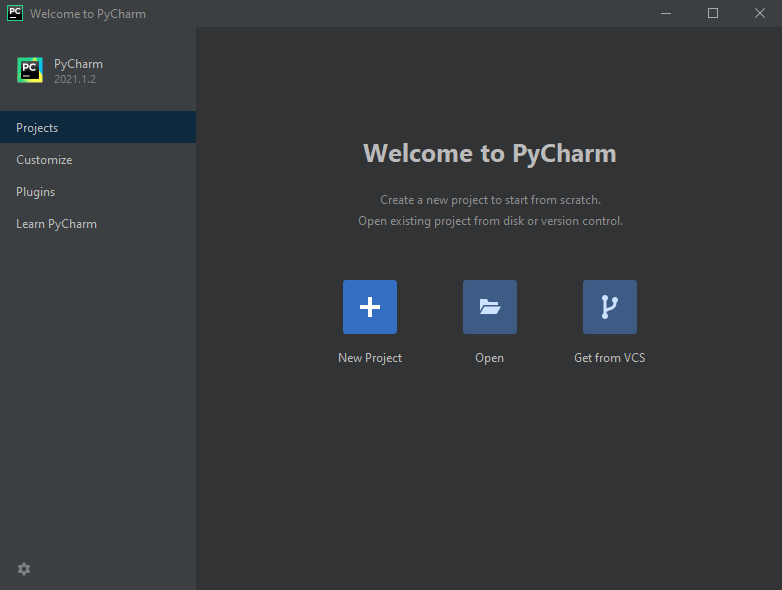


Рис.16. Нас встретит главное меню PyCharm, для создания проекта нажимаем на кнопку “New Project”.

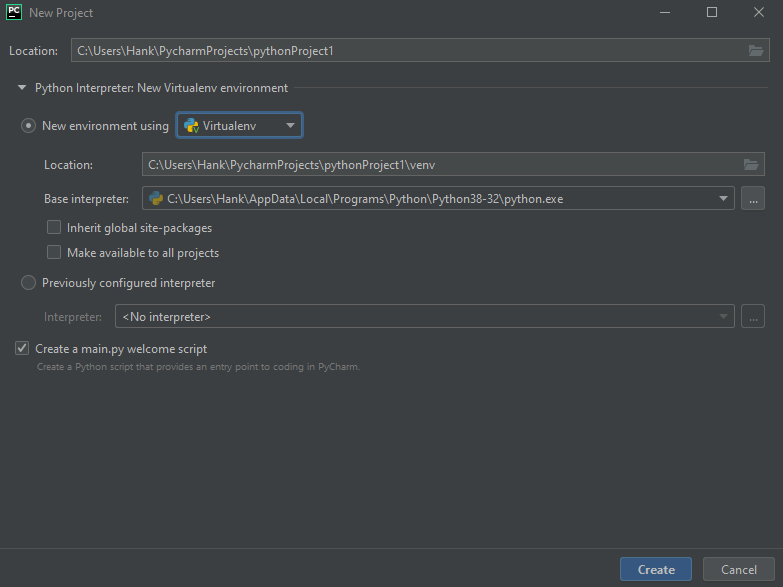


Рис.17. Перед созданием проекта можно изменить пару настроек, после чего нажимаем на кнопку “Create”.

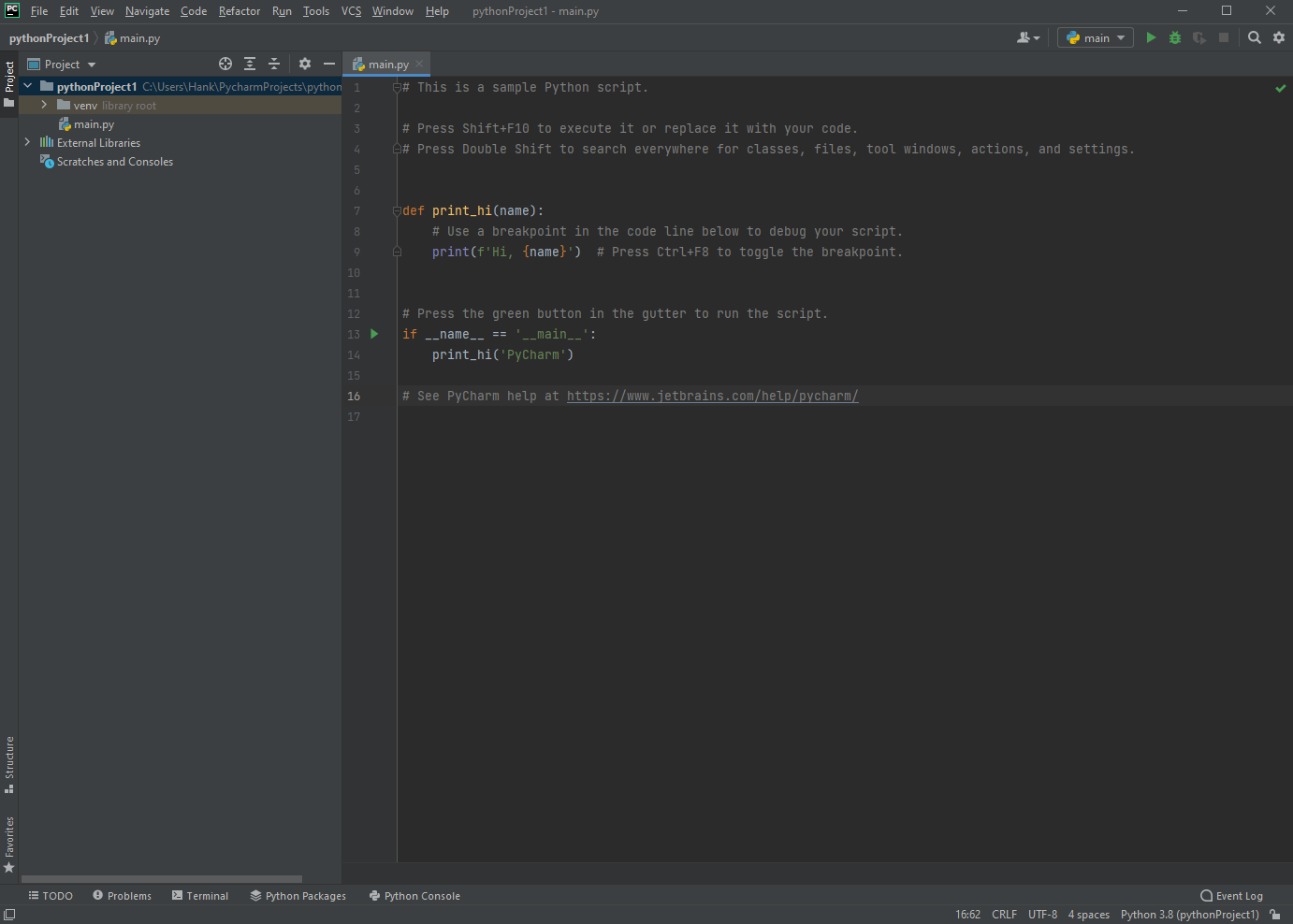


Рис.18. Нас встретит окно в котором мы можем начать писать код.

# 2.2 Техника работы с базами данных

Приложения: sql\_bd.py

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.

Листинг 42.

# Подготовлено Завадским М.А, Толоконниковым А.М

import sqlite3  
  
  
class Sqliter:  
  
 # Функция \_\_init\_\_ служит инициализацией БД  
 def \_\_init\_\_(self, name\_db):  
 self.connection = sqlite3.connect(name\_db)  
 self.cursor = self.connection.cursor()  
  
 # Функция create\_table создаёт таблицу в нашей БД  
 def create\_table(self, table):  
 with self.connection:  
 table = ''.join(chr for chr in table if chr.isalnum())  
 if table != '':  
 self.cursor.execute(f"CREATE TABLE IF NOT EXISTS `{table}` (\  
 id integer PRIMARY KEY,\  
 `fio` text, `group` text, `direction` text)")  
 self.save()  
  
 # Функция add\_student добавляет студента в БД, если его не существует  
 def add\_student(self, \*\*kwargs):  
 with self.connection:  
 data = kwargs  
 if kwargs.get('data') != None:  
 data = kwargs['data']  
 self.cursor.execute("INSERT INTO `students`\  
 (`fio`, `group`, `direction`) VALUES (?, ?, ?)",  
 (data['fio'], data['group'], data['direction']))  
 self.save()  
  
 # Функция get\_id возвращает id записи в таблице  
 def get\_id(self, \*\*kwargs):  
 with self.connection:  
 try:  
 data = kwargs  
 if kwargs.get('data') != None:  
 data = kwargs.get('data')  
 return self.cursor.execute("SELECT \* FROM `students` WHERE \  
 `fio` = ? AND `group` = ? AND `direction` = ?",  
 (data['fio'], data['group'], data['direction'])).fetchall()[0][0]  
 except:  
 return -1  
  
 # Функция save сохраняет изменения в БД  
 def save(self):  
 self.connection.commit()  
 print(f"{self.cursor.rowcount} отредактированно строк")  
  
 # Функция close закрывает БД  
 def close(self):  
 self.connection.close()  
  
# Функция input\_student возвращает данные о студенте в нужном для БД формате  
def input\_student(\*\*kwargs):  
 data = kwargs  
 if kwargs.get('data') != None:  
 data = kwargs.get('data')  
 student = {  
 'fio': data['fio'],  
 'group': data['group'],  
 'direction': data['direction']  
 }  
 return student  
  
# Создание БД  
bd = Sqliter("bd")  
# Создание таблицы students  
bd.create\_table('students')  
  
  
student = input\_student(fio="Cipkov Il'ya Vladimirovich",\  
 group="P1-18", direction="Programmer")  
# Если студента не находит в БД, то мы его добавляем в БД  
if bd.get\_id(data=student) == -1:  
 bd.add\_student(data=student)  
# Если такой студент существует, то в консоль выводит "Ne mogu("  
else:  
 print('Ne mogu(')  
  
# Прекращение работы с БД  
bd.close()

После завершения кода в той же папке где находится python-file появится файл с БД и в ней должна появиться запись «*1 Cipkov Il'ya Vladimirovich P1-18 Programmer*». Просто так его открыть и посмотреть нельзя.

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



Рис. 19. Установщик DB Browser

После установки, открываем DB Browser для работы с ним.

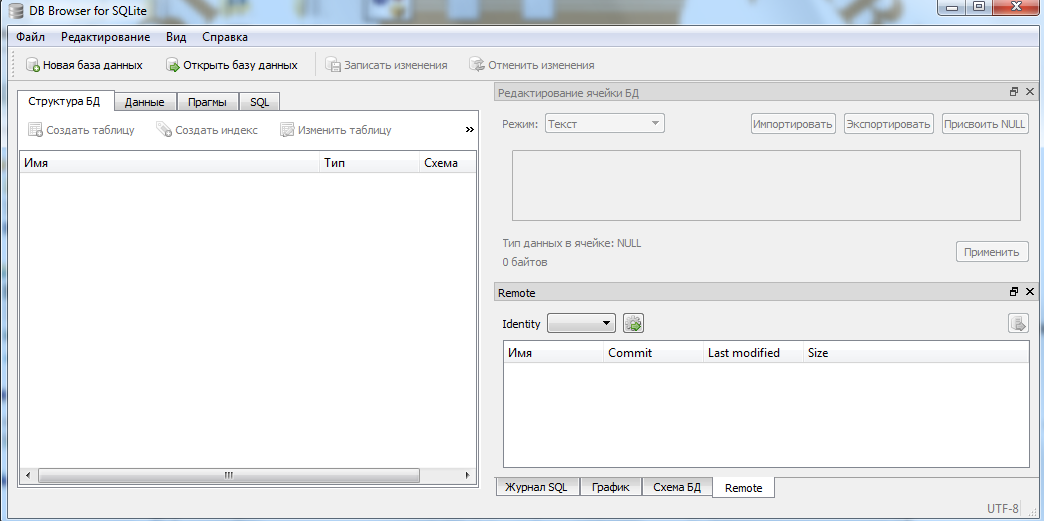


Рис. 20. Окно DB Browser

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

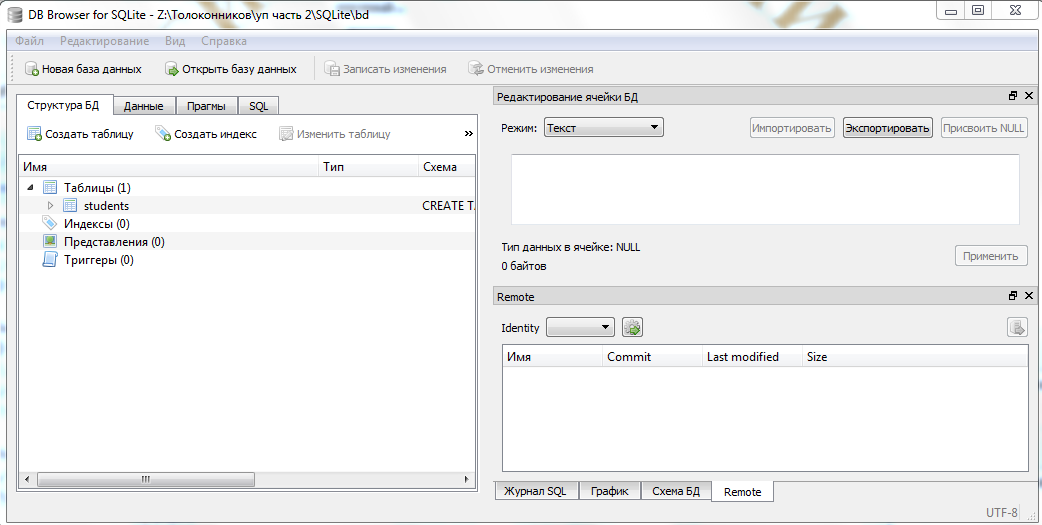


Рис. 21. Таблица students в DB Browser

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

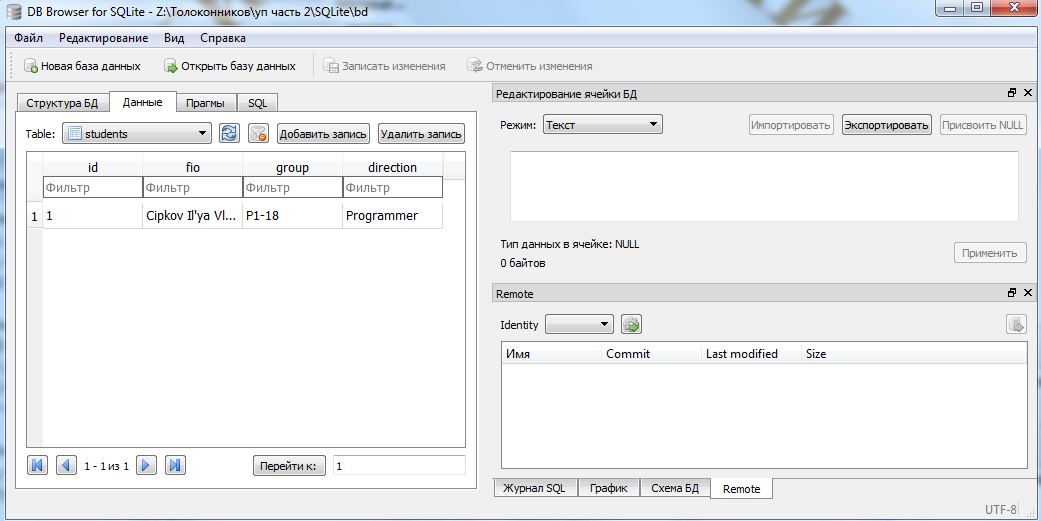


Рис. 22. Данные таблицы students

Видно, что в таблицу занеслись корректные данные.

# 2.3 Техника работы с библиотекой tkinter

Приложения: MyTable.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 43

# Подготовлено Толоконниковым А.М, Завадским М.А

'''

Туристский клуб

Туристский клуб организует одно – и многодневные пешеходные туры в группах с

руководителем по разным маршрутам и разной категории сложности.

Таблицы: Маршруты (Код маршрута, название, продолжительность, категория сложности),

Руководители (Код руководителя, ФИО, телефон), Группы (Код группы, название группы,

код маршрута, код руководителя, количество туристов, дата отправления).

Требуется:

- определить перечень групп на маршруте;

- построить сравнительную диаграмму количества туристов по уровню сложности маршрутов.

'''

from tkinter import \*

# Функция save сохраняет введённые данные в строку и записывает строчку в файл

def save():

str = get\_marsh() + ' ' + naz\_get() + ' ' + get\_tim() + ' ' + get\_dif() + '\n' + get\_ruk() + ' ' + get\_fio() + ' ' + get\_tel() + '\n' + gru\_get() + ' ' + get\_nazgru() + ' ' + get\_kodmarch() + ' ' + get\_kodruk() + ' ' + get\_koltur() + ' ' + get\_data()

with open("marshrut.txt", "a") as f:

f.write(str + '\n')

# Функция delete очищает текстовоый файл полностью

def delete():

f = open('marshrut.txt', 'w+')

f.seek(0)

f.close()

# Функция naz\_get возвращает имя введённое пользователем

def naz\_get():

return my\_var.get()

# Функция get\_marsh возвращает имя введённое пользователем

def get\_marsh():

return my\_var2.get()

# Функция get\_tim возвращает имя введённое пользователем

def get\_tim():

return my\_var3.get()

# Функция get\_dif возвращает имя введённое пользователем

def get\_dif():

return my\_var4.get()

# Функция get\_ruk возвращает имя введённое пользователем

def get\_ruk():

return my\_var5.get()

# Функция get\_fio возвращает имя введённое пользователем

def get\_fio():

return my\_var6.get()

# Функция get\_tel возвращает имя введённое пользователем

def get\_tel():

return my\_var7.get()

# Функция gru\_get возвращает имя введённое пользователем

def gru\_get():

return my\_var8.get()

# Функция get\_nazgru возвращает имя введённое пользователем

def get\_nazgru():

return my\_var9.get()

# Функция get\_kodmarch возвращает имя введённое пользователем

def get\_kodmarch():

return my\_var10.get()

# Функция get\_kodruk возвращает имя введённое пользователем

def get\_kodruk():

return my\_var11.get()

# Функция get\_koltur возвращает имя введённое пользователем

def get\_koltur():

return my\_var12.get()

# Функция get\_data возвращает имя введённое пользователем

def get\_data():

return my\_var13.get()

# Задаём размер окна и его координаты

root = Tk()

root.title('Туристский клуб')

root.geometry('450x450+740+300')

root.resizable(width=False, height=False)

# StringVar используется для легкого отслеживания изменений в переменных tkinter

my\_var = StringVar('')

my\_var2 = StringVar('')

my\_var3 = StringVar('')

my\_var4 = StringVar('')

my\_var5 = StringVar('')

my\_var6 = StringVar('')

my\_var7 = StringVar('')

my\_var8 = StringVar('')

my\_var9 = StringVar('')

my\_var10 = StringVar('')

my\_var11 = StringVar('')

my\_var12 = StringVar('')

my\_var13 = StringVar('')

canvas = Canvas(root, width=300, height=250)

canvas.pack()

frame = Frame(root)

frame.place(relwidth=1, relheight=1)

lable1 = Label(frame, text='заполните поля и сохраните', fg='white', bg='black').place(relx=.2, rely=.81)

lable2 = Label(frame, text='Маршрут', fg='white', bg='black').grid(row=0, column=1)

lable3 = Label(frame, text='Руководители', fg='white', bg='black').grid(row=5, column=1)

lable4 = Label(frame, text='Группы', fg='white', bg='black').grid(row=9, column=1)

# Первые данные

title = Label(frame, text='код маршрута').grid(row=1, column=0)

marshInput = Entry(frame, bg='white', textvariable=my\_var2).grid(row=1, column=1)

title = Label(frame, text='название маршрута').grid(row=2, column=0)

nazInput = Entry(frame, bg='white', textvariable=my\_var).grid(row=2, column=1)

title = Label(frame, text='продолжительность').grid(row=3, column=0)

timInput = Entry(frame, bg='white', textvariable=my\_var3).grid(row=3, column=1)

title = Label(frame, text='категория сложности').grid(row=4, column=0)

difInput = Entry(frame, bg='white', textvariable=my\_var4).grid(row=4, column=1)

# Вторые данные

title = Label(frame, text='код руководителя').grid(row=6, column=0)

rukInput = Entry(frame, bg='white', textvariable=my\_var5).grid(row=6, column=1)

title = Label(frame, text='ФИО').grid(row=7, column=0)

fioInput = Entry(frame, bg='white', textvariable=my\_var6).grid(row=7, column=1)

title = Label(frame, text='телефон').grid(row=8, column=0)

telInput = Entry(frame, bg='white', textvariable=my\_var7).grid(row=8, column=1)

# Третие данные

title = Label(frame, text='Код группы').grid(row=10, column=0)

gruInput = Entry(frame, bg='white', textvariable=my\_var8).grid(row=10, column=1)

title = Label(frame, text='Название группы').grid(row=11, column=0)

nazgruInput = Entry(frame, bg='white', textvariable=my\_var9).grid(row=11, column=1)

title = Label(frame, text='Код маршрута').grid(row=12, column=0)

kodmarchInput = Entry(frame, bg='white', textvariable=my\_var10).grid(row=12, column=1)

title = Label(frame, text='Код руководителя').grid(row=13, column=0)

kodrukInput = Entry(frame, bg='white', textvariable=my\_var11).grid(row=13, column=1)

title = Label(frame, text='Количество туристов').grid(row=14, column=0)

kolturInput = Entry(frame, bg='white', textvariable=my\_var12).grid(row=14, column=1)

title = Label(frame, text='Дата отправления').grid(row=15, column=0)

dataInput = Entry(frame, bg='white', textvariable=my\_var13).grid(row=15, column=1)

# Кнопки

btn = Button(frame, text='сохранить в txt', command=save).place(relx=.29, rely=.75)

btn2 = Button(frame, text='очищение txt', command=delete).place(relx=.8, rely=.938)

# Запуск цикла обработки событий

root.mainloop()

2.4 Техника работы с библиотекой NumPy

Листинг 44

#выполнил Завадский М.А, Джабраилов Т.А

import numpy

def main():

# Бесконечно выводим меню в консоль, пока пользователь не введёт 0

while(True):

menu()

print("Введите цифру меню: ")

num = int(input())

if(num == 1):

print("Введите функцию(-, +, \*, /):")

s = input()

function\_1(s)

print("\n")

elif(num == 2):

function\_2()

print("\n")

elif(num == 3):

function\_3()

print("\n")

elif(num == 4):

function\_4()

print("\n")

elif(num == 0):

# Выход из программы

exit(0)

else:

print("Вы ввели не корректный номер функции. Попробуйте ещё раз\n")

# Функция menu обеспечивает вывод меню в консоль

def menu():

print("Доброго времени суток, представляю вам небольшой\

инструментарий от NumPy. Выберите необходимый номер и посмотрите что выведет.")

print("1. Сложение/вычитание/умножение/деление матриц")

print("2. Объединение массивов")

print("3. Транспонирование")

print("4. Разбиваем одномерный массив на многомерный")

print("0. Выход")

# Функция function\_1 выводит в консоль пример сложения/вычитания/деления/умножения матриц

def function\_1(s):

A = numpy.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

B = A.copy()[::-1, ::-1]

if(s == "+"):

print("A\n", A)

print("B\n", B)

print("A + B\n", A + B)

elif(s == "-"):

print("A\n", A)

print("B\n", B)

print("A - B\n", A - B)

elif(s == "\*"):

print("A\n", A)

print("B\n", B)

print("A \* B\n", A \* B)

elif(s == "/"):

print("A\n", A)

print("B\n", B)

print("A / B\n", A // B)

# Функция function\_2 выводит в консоль пример объединения двух матриц в одну

def function\_2():

A = numpy.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

B = A[::-1]

C = A[:, ::-1]

print("A\n", A)

print("\nB\n", B)

print("\nC\n", C)

print("\nstack(A, B, C)\n", numpy.stack((A, B, C)))

# Функция function\_3 выводит в консоль пример транспонирования матрицы

def function\_3():

A = numpy.array([[1, 2, 3], [4, 5, 6]])

print('A\n', A)

B = A.T

print('\nА транспонированный\n', B)

# Функция function\_4 выводит в консоль пример разбиения одномерного массива на многомерный

def function\_4():

A = numpy.arange(24)

B = A.reshape(4, 6)

C = A.reshape(4, 3, 2)

print("Массив размером 1х1\n", A)

print('\nМассив размером 4х6\n', B)

print('\nМассив размером 4х3х2\n', C)

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

main()

2.5 Техника работы с базами данных Matplotlib

Листинг 45

# Выполнил Толоконников А.М.

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, 5, 3, 11]

# Координаты точек графика second по оси y

y2 = [4, 3, 1, 9, 12]

# Координаты точек графика third по оси y

y3 = [5, 6, 4, 7, 12]

plt.figure(figsize=(7, 4)) # Размер рисунка

# Построение графика first

plt.plot(x, y1, 'o-y', 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)

# Построение графика third

plt.plot(x, y3, 'o-.m', label="third", mec='r', lw=3, mew=4, ms=13)

plt.legend() # Отображение легенды

plt.grid(True) # Отображение сетки

plt.show() # Отображение графика

Листинг 46

# Выполнил Толоконников А.М.

import matplotlib.pyplot as plt

import numpy as np

fig, ax = plt.subplots()

x = np.array([-3, -2, -1, 0.1, 1, 2, 3])

y = np.array([9, 4, 1, 0.1, 1, 4, 9])

ax.plot(x, y)

ax.plot(x, np.sin(x), color=(1.0,0.2,0.3), linestyle='-', marker='o')

ax.plot(x, x +5, color='blue', linestyle='--')

ax.plot(x, x + 3, color='g', linestyle=':', marker='^')

ax.plot(x, np.cos(x), color='0.75', linestyle='-.')

ax.plot(x, x, color='#FFDD44', linestyle='--', marker='\*')

plt.show()

Листинг 47

# Выполнил Толоконников А.М.

import matplotlib.pyplot as plt

import numpy as np

from mpl\_toolkits.mplot3d import Axes3D

fig, ax = plt.subplots()

N = 50

x = np.random.rand(N)

y = np.random.rand(N)

colors = np.random.rand(N)

area = (30 \* np.random.rand(N))\*\*2

fig = plt.figure(figsize=(10, 6))

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(x, y, s=area, c=colors, alpha=0.5, marker='\*', cmap='viridis')

plt.show()

2.6 Техника работы с библиотекой PyQt

Листинг 48

#Выполнил Толоконников А.М., Джабраилов Т. А.

from PyQt5.QtWidgets import \*

from PyQt5.QtCore import QSize

import random

# Наследуемся от QMainWindow

class MainWindow(QMainWindow):

# Создаём диалоговое окно, которое открывается при нажатии на кнопку

def dialog(self, lineEdit, strList, secretList):

mbox = QMessageBox() # Создаём диалоговое окно

mbox.setWindowTitle('Окно')

# Условия вывода определённого текста в окне

if(lineEdit.text() in strList):

mbox.setText(lineEdit.text() + " krut!")

elif(lineEdit.text() in secretList):

mbox.setText(lineEdit.text() + "\nЭто правда")

else:

mbox.setText("prosto text")

mbox.setDetailedText("Наберите одно из этих слов\n'Python', 'PyQt5', 'Qt', 'Django', 'QML'")

mbox.exec()

# Переопределяем конструктор класса

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

# Обязательно нужно вызвать метод супер класса

QMainWindow.\_\_init\_\_(self)

self.setWindowTitle("Проверка автодополнения") # Устанавливаем заголо-вок окна

central\_widget = QWidget(self) # Создаём центральный виджет

self.setCentralWidget(central\_widget) # Устанавливаем центральный ви-джет

grid\_layout = QGridLayout() # Создаём QGridLayout

central\_widget.setLayout(grid\_layout) # Устанавливаем данное размещение в центральный виджет

grid\_layout.addWidget(QLabel("Проверка автодополнения", self), 0, 0)

# Создаём поле ввода

lineEdit = QLineEdit(self)

strList = ['Python', 'PyQt5', 'Qt', 'Django', 'QML'] # Создаём список слов

secretList = ['Хочу домой', 'Я сделал домашку', 'ККМТ - сила'] # Создаём секретный список

rand = random.randint(1,10)

# Создаём QCompleter, в который устанавливаем список, а также указатель на родителя

# С вероятностью 50% в QCompleter может добавиться секретный список

if(rand % 2 == 0):

completer = QCompleter(strList, lineEdit)

else:

completer = QCompleter(strList + secretList, lineEdit)

lineEdit.setCompleter(completer) # Устанавливает QCompleter в поле вво-да

grid\_layout.addWidget(lineEdit, 0, 1) # Добавляем поле ввода в сетку

qb = QPushButton("Submit") # Создаём кнопку

# Создаём действие для кнопки

qb.clicked.connect(lambda : self.dialog(lineEdit, strList, secretList))

grid\_layout.addWidget(qb, 0, 2)

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QApplication(sys.argv)

mw = MainWindow()

mw.show()

sys.exit(app.exec())

2.7 Техника работы с библиотекой Pygame

Приложения: main.py

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

Листинг 49

#подготовлено Завадским Михаилом П2-18

import pygame

# импортируем из модуля рандом модуль диапазона случайных чисел

from random import randrange as rnd

from os import path

WIDTH, HEIGHT = 1200, 800

fps = 60

# настройки платформы

paddle\_w = 330

paddle\_h = 35

paddle\_speed = 15

paddle = pygame.Rect(WIDTH // 2 - paddle\_w // 2, HEIGHT - paddle\_h - 10, paddle\_w, paddle\_h)

# настройки мячика

ball\_radius = 20

ball\_speed = 6

ball\_rect = int(ball\_radius \* 2 \*\* 0.5)

ball = pygame.Rect(rnd(ball\_rect, WIDTH - ball\_rect), HEIGHT // 2, ball\_rect, ball\_rect)

dx, dy = 1, -1

# настройка блоков(которые ломает шарик)

block\_list = [pygame.Rect(10 + 120 \* i, 10 + 70 \* j, 100, 50) for i in range(10) for j in range(4)] # блоки их расположения относительно разрешения

color\_list = [(rnd(30, 256), rnd(30, 256), rnd(30, 256)) for i in range(10) for j in range(4)] # разные цвета для блоков

pygame.init()

sc = pygame.display.set\_mode((WIDTH, HEIGHT))

clock = pygame.time.Clock()

# Фон

img = pygame.image.load('1.jpg').convert()

pygame.mixer.music.load('tgfcoder-FrozenJam-SeamlessLoop.wav')

pygame.mixer.music.play(-1)

pygame.mixer.music.set\_volume(0.0)

kn = pygame.mixer.Sound('Knock.wav')

def detect\_collision(dx, dy, ball, rect): # сложная математика

if dx > 0:

# когда dx положительный то шарик движется в право и сталкивается с левой # стороной блока для отрицательного случая всё будет противоположно

delta\_x=ball.right-rect.left

else:

delta\_x=rect.right-ball.left

# находим аналогично x

if dy > 0:

delta\_y=ball.bottom-rect.top

else:

delta\_y=rect.bottom-ball.top

if abs(delta\_x - delta\_y) < 10:

dx, dy =-dx, -dy

elif delta\_x > delta\_y:

dy =-dy

elif delta\_y > delta\_x:

dx =-dx

return dx, dy

while True:

for event in pygame.event.get():

if event.type== pygame.QUIT:

exit()

sc.blit(img, (0, 0))

# прорисовка мира

[pygame.draw.rect(sc, color\_list[color], block) for color, block in enumerate(block\_list)] # блоки

pygame.draw.rect(sc, pygame.Color('darkorange'), paddle)

# платформа для отскоков

pygame.draw.circle(sc, pygame.Color('white'), ball.center, ball\_radius)

# рисуем шарик

# движения мячика

ball.x += ball\_speed \* dx

ball.y += ball\_speed \* dy

# коллизии лево право

if ball.centerx < ball\_radius or ball.centerx > WIDTH - ball\_radius:

dx = -dx

# коллизия вверх

if ball.centery < ball\_radius:

dy = -dy

# коллизия платформы

if ball.colliderect(paddle) and dy > 0:

dx, dy = detect\_collision(dx, dy, ball, paddle)

# если мячик ударился мы удаляем из списка блок

hit\_index = ball.collidelist(block\_list)

if hit\_index != -1:

hit\_rect = block\_list.pop(hit\_index)

hit\_color = color\_list.pop(hit\_index)

dx, dy = detect\_collision(dx, dy, ball, hit\_rect)

kn.play()

# победа, поражение

if ball.bottom > HEIGHT:

print('GAME OVER!')

exit()

elif not len(block\_list):

print('WIN!!!')

exit()

# управление

key = pygame.key.get\_pressed()

if key[pygame.K\_LEFT] and paddle.left > 0:

# конпка передвежиная + запрет выходить за поле

paddle.left -= paddle\_speed

if key[pygame.K\_RIGHT] and paddle.right < WIDTH:

# конпка передвежиная + запрет выходить за поле

paddle.right += paddle\_speed

# обновление сцены

pygame.display.flip()

clock.tick(fps)

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

3.1 Изучение входной и выходной документации

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

Листинг 50

import tkinter as tk

from tkinter import ttk

import sqlite3

class Main(tk.Frame):

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

super().\_\_init\_\_(root)

self.init\_main()

self.db = db

self.view\_records()

def init\_main(self):

toolbar = tk.Frame(bg='#d7d8e0', bd=2)

toolbar.pack(side=tk.TOP, fill=tk.X)

btn\_open\_dialog = tk.Button(toolbar, text='Добавить позицию', command=self.open\_dialog, bg='#d7d8e0', bd=0,

compound=tk.TOP)

btn\_open\_dialog.pack(side=tk.LEFT)

self.tree = ttk.Treeview(self, columns=('ID', 'description', 'costs', 'total'), height=15, show='headings')

self.tree.column('ID', width=30, anchor=tk.CENTER)

self.tree.column('description', width=365, anchor=tk.CENTER)

self.tree.column('costs', width=150, anchor=tk.CENTER)

self.tree.column('total', width=100, anchor=tk.CENTER)

self.tree.heading('ID', text='ID')

self.tree.heading('description', text='Наименование')

self.tree.heading('costs', text='Статья дохода\расхода')

self.tree.heading('total', text='Сумма')

self.tree.pack()

def records(self, description, costs, total):

self.db.insert\_data(description, costs, total)

self.view\_records()

def view\_records(self):

self.db.c.execute('''SELECT \* FROM finance''')

[self.tree.delete(i) for i in self.tree.get\_children()]

[self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]

def open\_dialog(self):

Child()

class Child(tk.Toplevel):

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

super().\_\_init\_\_(root)

self.init\_child()

self.view = app

def init\_child(self):

self.title('Добавить доходы\расходы')

self.geometry('400x220+400+300')

self.resizable(False, False)

label\_description = tk.Label(self, text='Наименование:')

label\_description.place(x=50, y=50)

label\_select = tk.Label(self, text='Статья дохода\расхода:')

label\_select.place(x=50, y=80)

label\_sum = tk.Label(self, text='Сумма:')

label\_sum.place(x=50, y=110)

self.entry\_description = ttk.Entry(self)

self.entry\_description.place(x=200, y=50)

self.entry\_money = ttk.Entry(self)

self.entry\_money.place(x=200, y=110)

self.combobox = ttk.Combobox(self, values=[u'Доход', u'Расход'])

self.combobox.current(0)

self.combobox.place(x=200, y=80)

btn\_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)

btn\_cancel.place(x=300, y=170)

btn\_ok = ttk.Button(self, text='Добавить')

btn\_ok.place(x=220, y=170)

btn\_ok.bind('<Button-1>', lambda event: self.view.records(self.entry\_description.get(),

self.entry\_money.get(),

self.combobox.get()))

self.grab\_set()

self.focus\_set()

class DB:

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

self.conn = sqlite3.connect('finance.db')

self.c = self.conn.cursor()

self.c.execute(

'''CREATE TABLE IF NOT EXISTS finance (id integer primary key, description text, costs text, total real)''')

self.conn.commit()

def insert\_data(self, description, costs, total):

self.c.execute('''INSERT INTO finance(description, costs, total) VALUES (?, ?, ?)''',

(description, costs, total))

self.conn.commit()

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

db = DB()

app = Main(root)

app.pack()

root.title("Финансы")

root.geometry("650x450+300+200")

root.resizable(False, False)

root.mainloop()

3.2. 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. Разработка документации. |  |
| 23.04.2021 | Разработка документации. Защита проекта. |  |
| 26.04.2021 | Защита проекта. Сдача зачёта по практике. |  |