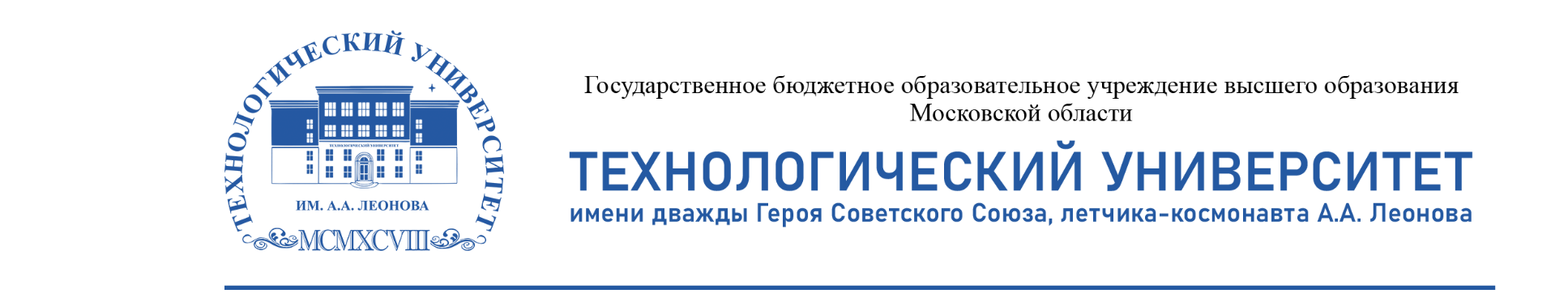
******

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:

Щепкин Михаил Викторович

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

Проверил преподаватель: Гусятинер Леонид Борисович

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

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

Королев, 2021

Оглавление

[**Раздел 1. Техника решения задач с использованием структурного и объектно-ориентированного программирования.** 3](#_Toc74870708)

[K2. Установка интерпретатора Python 3 и настройка окружения. 3](#_Toc74870709)

[K3. Техника работы в командной строке и среде IDLE. 4](#_Toc74870710)

[K4. Техника работы с линейными и разветвляющимися программами. 9](#_Toc74870711)

[K5. Техника работы с циклическими программами, цикл while 15](#_Toc74870712)

[K6. Техника работы с числами. 21](#_Toc74870713)

[K7. Техника работы со строками 31](#_Toc74870714)

[K8. Техника работы со списками. 38](#_Toc74870715)

[K9. Техника работы с циклом for и генераторами списков. 45](#_Toc74870716)

[K10. Техника работы с функциями. 50](#_Toc74870717)

[K11. Техника работы со словарями. 56](#_Toc74870718)

[K12. Техника работы с множествами. 60](#_Toc74870719)

[K13. Техника работы с кортежами. 65](#_Toc74870720)

[K14. Техника работы с файлами. 67](#_Toc74870721)

[K15.Техника работы с модулями. 72](#_Toc74870722)

[K16. Техника работы с классами. 82](#_Toc74870723)

[**Раздел 2. Техника решения задач с использованием библиотек** 86](#_Toc74870724)

[K17. Установка и настройка среды JetBrains PyCharm 86](#_Toc74870725)

[K18. Техника работы с базами данных 94](#_Toc74870726)

[K19. Техника работы с библиотекой tkinter 99](#_Toc74870727)

[K20. Техника работы с библиотекой NumPy 101](#_Toc74870728)

[K21. Техника работы с библиотекой Matplotlib 103](#_Toc74870729)

[K22. Элементы работы с библиотекой PyQt 105](#_Toc74870730)

[K23. Элементы работы с библиотекой PyGame 107](#_Toc74870731)

[**Раздел 3. Разработка проекта с графическим интерфейсом** 111](#_Toc74870732)

[3.1 K24.Изучение входной и выходной документации 111](#_Toc74870733)

[3.2 K25.Разработка требований к проекту. Построение диаграммы использования. 113](#_Toc74870734)

[3.3 K26.Разработка сценария проекта 115](#_Toc74870735)

[3.4 K27.Построение диаграммы классов 116](#_Toc74870736)

[3.5 K28.Разработка базы данных 116](#_Toc74870737)

[3.6 K29.Разработка главного модуля 119](#_Toc74870738)

[3.7 K30.Разработка входящих модулей 120](#_Toc74870739)

[3.8 K31.Тестирование и отладка 125](#_Toc74870740)

[3.9 K32.Дневник 126](#_Toc74870741)

[**Приложения** 129](#_Toc74870742)

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

# K2. Установка интерпретатора Python 3 и настройка окружения.

Материал подготовили студенты группы П2-18:

Щепкин Михаил Викторович

Инструкция по установке python представлена в виде презентации



(Приложение 1. Инструкция по установке python)

# K3. Техника работы в командной строке и среде IDLE.

Шаг 1.

1 Вариант. Открываем список команд с помощью комбинации клавишь:

Windows + R, далее вводим туда cmd.

2 Вариант. Открыть поисковую строку и ввести туда cmd.

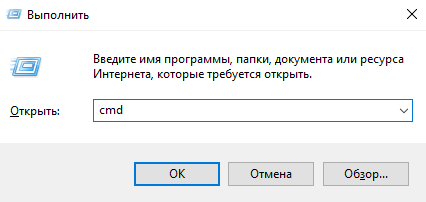


Рис. 1.

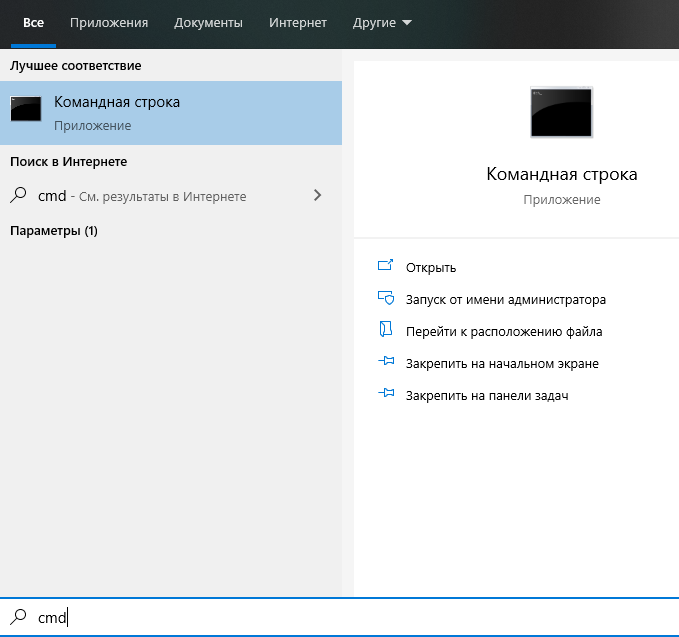


Рис. 1.1.

Шаг 2.

После открытия командной строки, создадим файл для этого мы используем команду "cd", которая позволит перемещаться по директориям.

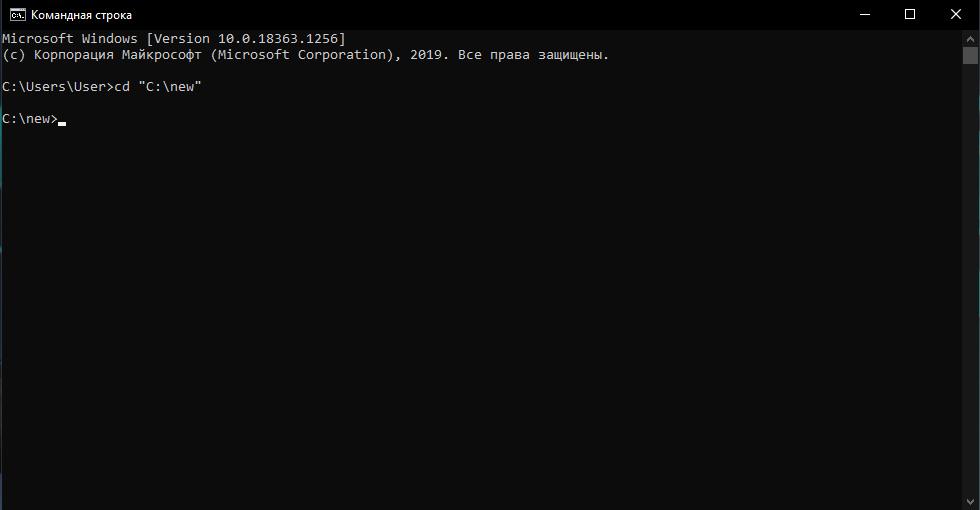


Рис. 2.

Шаг 3.

Теперь мы можем создать файл с кодом для этого вводим следующую команду copy con. И так мы создали пустой файл с расширением Python

далее вводим код print("Hello Word !")



Рис. 3.

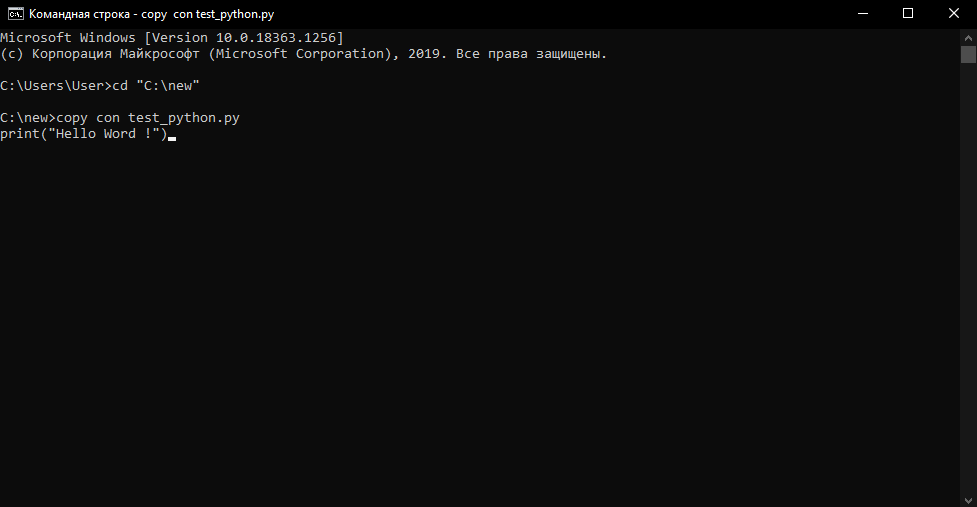


Рис. 3.1.

Шаг 4.

После того как мы ввели ("Hello Word !") нажимаем клавишу F6 и после Enter далее на выведет: Скопировано файлов:

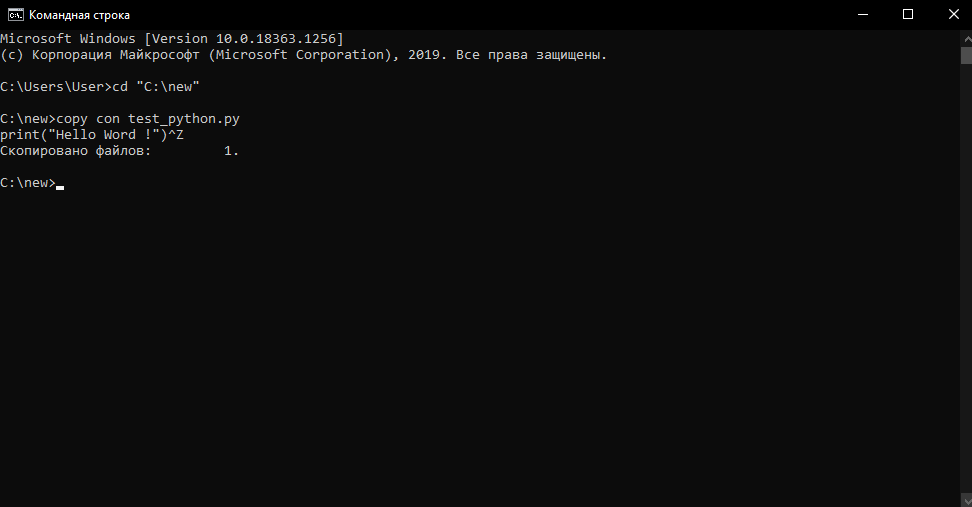


Рис. 4.

Шаг 5.

Запустим нашу программу пишем в командную строку название нашей программы и нажимаем Enter и мы видем что она вывела наш код.

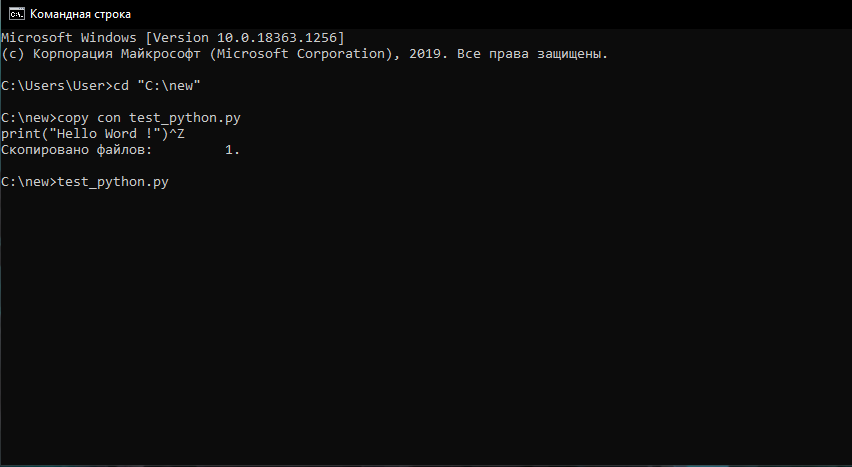


Рис. 5.

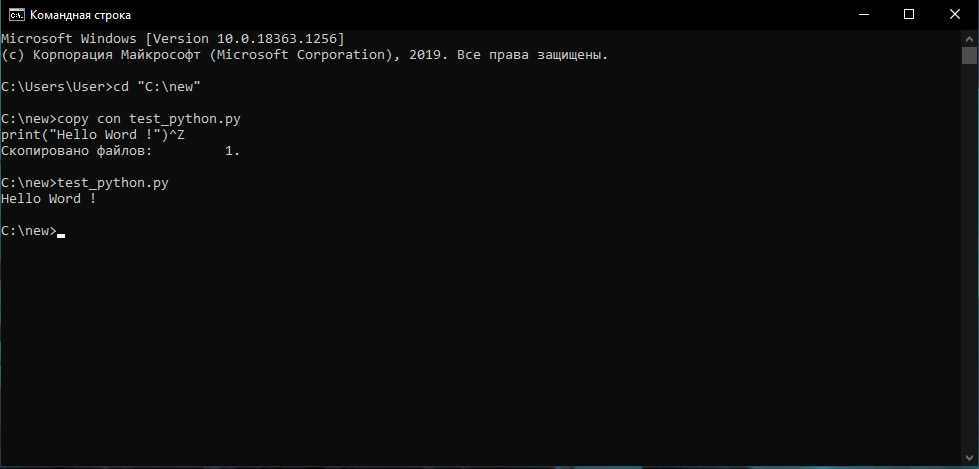


Рис. 5.1.

# K4. Техника работы с линейными и разветвляющимися программами.

Задачи на Begin.

Задание 1.

Листинг 1: 4\_1\_1.py

#Programming Taskbook

#ЮФУ Электронный задачник по программированию

#М. Э. Абрамян (Южный федеральный университет), 1998–2020

#Begin8°. Даны два числа a и b. Найти их среднее арифметическое: (a + b)/2.

a = int(input())

b = int(input())

print((a + b)/2)

Задание 2.

Листинг 2: 4\_1\_2.py

"""Programming Taskbook

ЮФУ Электронный задачник по программированию

© М. Э. Абрамян (Южный федеральный университет), 1998–2020

begin3 Даны стороны прямоугольника a и b.

Найти его площадь S = a·b и периметр P = 2·(a + b)."""

a = int(input())

b = int(input())

S = a \* b

P = 2 \* (a + b)

print(S)

print(P)

Задачи на Boolean.

Задание 1.

Листинг 3: 4\_2\_1.py

#Programming Taskbook

#ЮФУ Электронный задачник по программированию

#М. Э. Абрамян (Южный федеральный университет), 1998–2020

#Boolean4°. Даны два целых числа: A, B. Проверить

#истинность высказывания: «Справедливы неравенства A > 2 и B ≤ 3».

a = int(input())

b = int(input())

print(a > 2, b >= 3)

Задание 2.

Листинг 4: 4\_2\_2.py

#Programming Taskbook

#ЮФУ Электронный задачник по программированию

#© М. Э. Абрамян (Южный федеральный университет), 1998–2020

#Дано целое число A.

#Boolean2 Проверить истинность высказывания: «Число A является нечетным».

print(int(input()) / 2 )

Задачи на Case.

Задание 1.

Листинг 5: 4\_3\_1.py

#Programming Taskbook

#ЮФУ Электронный задачник по программированию

#М. Э. Абрамян (Южный федеральный университет), 1998–2020

#Case1. Дано целое число в диапазоне 1–7. Вывести строку —

#название дня недели, соответствующее данному числу

#1 — «понедельник», 2 — «вторник» и т. д.).

day = ["понедельник","вторник","среда","четверг","пятница","суббота","воскресенье"]

a = int(input("Введитечисло 1-7:"))-1

print(day[a])

Задание 2.

Листинг 6: 4\_3\_2.py

"""Programming Taskbook

ЮФУ Электронный задачник по программированию

© М. Э. Абрамян (Южный федеральный университет), 1998–2020

Дан номер месяца — целое число в диапазоне 1–12(1 — январь, 2 — февраль и т. д.).

Вывести название соответствующего времени года («зима», «весна», «лето», «осень»)."""

month = ["Январь, Зима","Февраль, Зима","Март, Весна","Апрель, Весна","Май, Весна","Июнь, Лето","Июль, Лето","Август, Лето","Сентябрь, Осень","Октябрь, Осень","Ноябрь, Осень","Декабрь, Зима"]

a = int(input("Введите число от 1-12"))-1

print(month[a])

Задание 3.

Листинг 7: 4\_3\_3.py

'''

Выполнил: Щепкин М.В. П2-18

Case2°. 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».

'''

ochenka = input('Введите оценку ').strip()

print({

'1': 'Плохо',

'2': 'Неудовлетворительно',

'3': 'удовлетворительно',

'4': 'Хорошо',

'5': 'Отлично'

}.get(ochenka, 'Ошибка, такой оценки нет'))

Задачи на If.

Задание 1.

Листинг 8: 4\_4\_1.py

#Вася приводит примеры возрастающих последовательностей из трех чисел,

#однако кое-где он допустил ошибки: в некоторых примерах

#второе и/или третье число не больше предыдущего. Он попросил вас написать

#программу, которая выводит "2 <= 1", если второе число не больше первого,

#"3 <= 2", если третье число не больше второго,

#и "Все в порядке" в остальных случаях. Все числа целые.

a = int(input())

b = int(input())

c = int(input())

if a >= b:

print("2 <= 1")

if b >= c:

print("3 <= 2")

if a < b < c:

print("Все в порядке")

Задание 2.

Листинг 9: 4\_4\_2.py#Stepik.Вам необходимо проанализировать введенное число

#0 <= n < 1000, и вывести"Число однозначное", "Число двузначное" или

#"Число трехзначное" в зависимости от длины числа.

n = int(input())

if n < 10:

print ("Число однозначное")

elif n < 100:

print ("Число двузначное")

elif n < 1000:

print ("Число трехзначное")

Задание 3.

Листинг 10: 4\_4\_3.py

"""Прямоугольная (или декартова) система координат широко применяется

в математике. Напишите программу, которая принимает два целых,

неравных нулю числа — координату точки по оси OX и по оси OY,

и выведите, в какой координатной четверти находится точка."""

a = int(input())

b = int(input())

if a > 0 and b > 0:

print("1")

elif a < 0 and b < 0:

print("3")

elif a > 0 and b < 0:

print("4")

else:

print("2")

Блок-схемы для задач к практическому занятию №4.

Блок-схема для задачи 4\_1\_1.py:

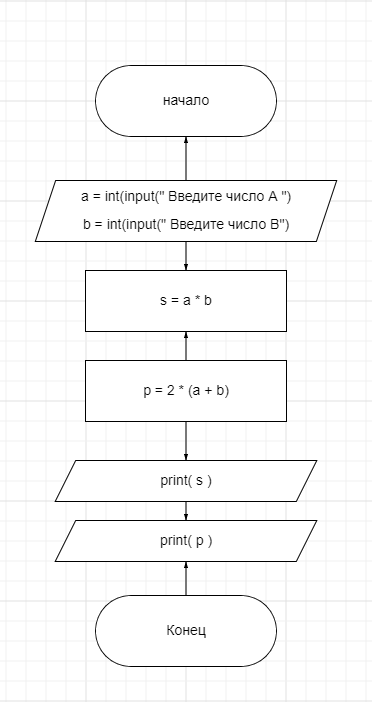


Рис. 1.(блок-схема для задачи 4\_1\_1)

Блок схема для задачи 4\_4\_1.py:

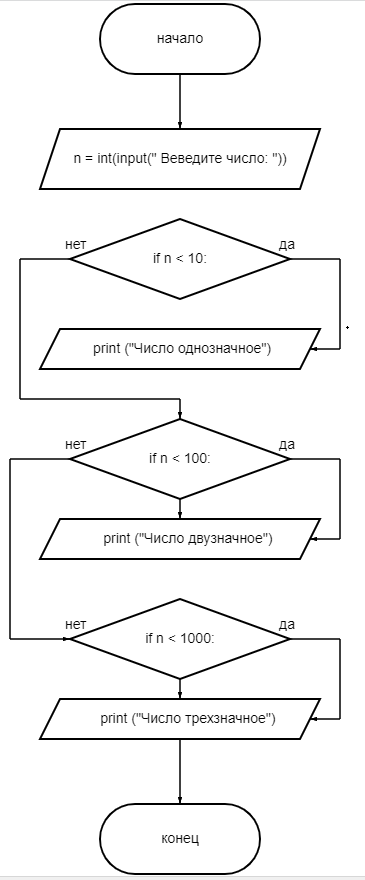


Рис. 2.(блок-схема для задачи 4\_4\_1)

# K5. Техника работы с циклическими программами, цикл while

Задачи на For.

Задание 1.

Листинг 11: 5\_1\_1.py

#Задача:

#С клавиатуры вводятся целые числа a > b.

#Выведите убывающую последовательность чисел по одному числу в строке.

a = int(input())

b = int(input())

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

print(i)

Задание 2.

Листинг 12: 5\_1\_2.py

#Задача:

#Напишите программу, которая считывает с клавиатуры два числа aa и bb,

#считает и выводит на консоль среднее

#арифметическое всех чисел из отрезка [a; b][a;b], которые кратны числу 33.

#В приведенном ниже примере среднее арифметическое считается для чисел на

#отрезке [-5; 12][−5;12]. Всего чисел, делящихся на 33, на этом отрезке 66:

#-3, 0, 3, 6, 9, 12−3,0,3,6,9,12. Их среднее арифметическое равно 4.54.5

#На вход программе подаются интервалы, внутри которых всегда есть хотя бы

#одно число, которое делится на 33.﻿

a = int(input())

b = int(input())

s = 0

c = 0

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

if j%3 == 0:

s = s+j #42

c = c+1

print(s/c)

Задание 3.

Листинг 13: 5\_1\_3.py

#https://pythontutor.ru/lessons/for\_loop/problems/series\_1/

#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)

Задание 4.

Листинг 14: 5\_1\_4.py

#https://pythontutor.ru/lessons/for\_loop/problems/series\_2/

#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,b = int(input()),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)

Задание 5.

Листинг 15: 5\_1\_5.py

#https://pythontutor.ru/lessons/for\_loop/problems/sum\_of\_ten\_numbers/

#Дано 10 целых чисел. Вычислите их сумму.

#Напишите программу, использующую наименьшее число переменных.

sum = 0

for i in range(10):

number = int(input())

sum += number

print(sum)

Задачи на While.

Задание 1

Листинг 16: 5\_2\_1.py

#Stepik.Напишите программу, которая считывает целые числа с консоли по одному числу в строке.

#Для каждого введённого числа проверить:

#если число меньше 10, то пропускаем это число;

#если число больше 100, то прекращаем считывать числа;

#в остальных случаях вывести это число обратно на консоль в отдельной строке.

a = 0

while a <= 100:

a = int(input())

if 10 <= a <= 100:

print(a)

Задание 2.

Листинг 17: 5\_2\_2.py

#https://pythontutor.ru/lessons/while/problems/list\_of\_squares/

#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

Задание 3.

Листинг 18: 5\_2\_3.py

#https://pythontutor.ru/lessons/while/problems/minimal\_divisor/

#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)

Задание 4.

Листинг 19: 5\_2\_4.py

#https://pythontutor.ru/lessons/while/problems/seq\_sum/

#Определите сумму всех элементов последовательности,

#завершающейся числом 0. В этой и во всех следующих задачах числа,

#следующие за первым нулем, учитывать не нужно.

el = int(input())

sum = 0

while el != 0:

sum += el

el = int(input())

print (sum)

Задание 5.

Листинг 20: 5\_2\_5.py.

#https://pythontutor.ru/lessons/while/problems/kth\_fibonacci/

#Последовательность Фибоначчи определяется так:

#φ0 = 0, φ1 = 1, φn = φn−1 + φn−2.

#По данному числу n определите n-е число Фибоначчи φn.

n = int(input())

if n == 0:

print(0)

else:

a, b = 0, 1

for i in range(2, n + 1):

a, b = b, a + b

print(b)

Блок-схемы для задач к практическому занятию №5.

Блок-схема для задачи (5\_1\_1.py):

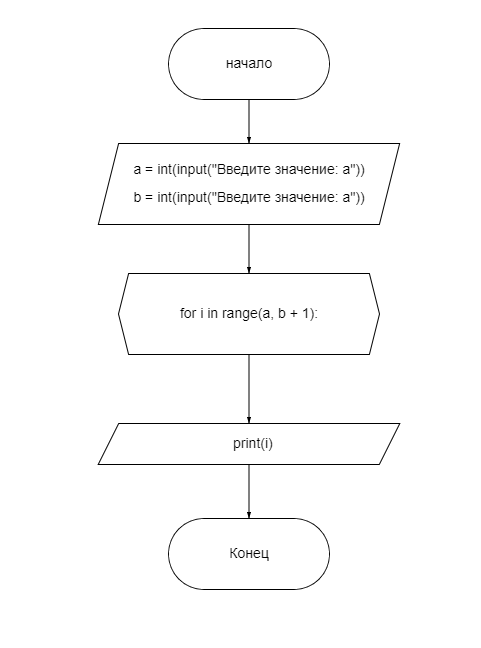


Рис. 3.(блок-схема для задачи 5\_1\_1)

Блок-схема для задачи (5\_2\_4.py):

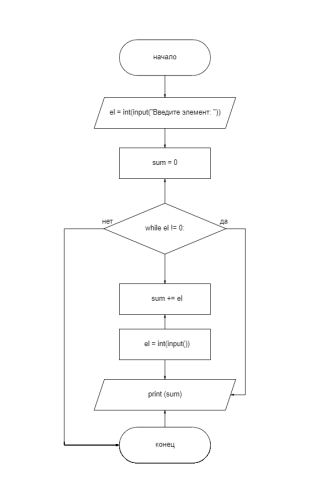


Рис. 4.(блок-схема для 5\_2\_4)

# K6. Техника работы с числами.

Листинг 21: 6\_1\_1.py

Модуль Fraction.

# Fractions1

# Выполнили: Щепкин М.В. П2-18

"""

На вход подаются две строки с числами, содержашие пару числитель-знаменатель через пробел. Вывести сумму этих чисел

"""

from fractions import Fraction

s = input().split()

s\_1 = input().split()

print(Fraction(int(s[0]), int(s[1])) + Fraction(int(s\_1[0]), int(s\_1[1])))

Листинг 22: 6\_1\_2.py

# Fractions2

# Выполнили: Щепкин М.В. П2-18

"""

На вход в строке подаётся три числа - чимлитель, знаменатель,

и степень в которую необходимо возвести число. Вывести результат

"""

from fractions import Fraction as f

s = input().split()

print(f(int(s[0]), int(s[1]))\*\*int(s[2]))

Листинг 23: 6\_1\_3.py

# Fractions3

# Выполнили: Щепкин М.В. П2-18

"""

На вход подаются две строки с числами, содержашие пару числитель-знаменатель

через пробел. Вывести Наименьшее из этих чисел

"""

from fractions import Fraction as f

s = input().split()

s\_1 = input().split()

if f(int(s[0]), int(s[1])) < f(int(s\_1[0]), int(s\_1[1])):

print("Наименьшее число", f(int(s[0]), int(s[1])))

else:

print("Наименьшее число", f(int(s\_1[0]), int(s\_1[1])))

Листинг 24: 6\_1\_4.py

Модуль Decimal.

# Decimal 1

# Выполнили: Щепкин М.В. П2-18

"""

На вход подаётся число, окуглить его до двух знаков после запятой и вывести рельзутат

"""

from decimal import Decimal

num = Decimal(input())

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

Листинг 25: 6\_1\_5.py

# Decimal 2

# Выполнили: Щепкин М.В. П2-18

"""

На вход подаются два числа с новой строки, окуглить перовое число до 4х знаков,

после запятой, а второе до трёх знаков после запятой

, сложить их и вывести результат

"""

from decimal import Decimal

num = Decimal(input())

num\_2 = Decimal(input())

print(num.quantize(Decimal("1.0000")) + num\_2.quantize(Decimal("1.0000")))

Листинг 26: 6\_1\_6.py

# Decimal 3

# Выполнили: Щепкин М.В. П2-18

"""

На вход подаются два числа с новой строки, вычесть из большего числа меньшее

и вывести результат

"""

from decimal import Decimal

num = Decimal(input())

num\_2 = Decimal(input())

if num > num\_2:

print(num - num\_2)

else:

print(num\_2 - num)

Задание 2. (6\_1\_1.py)

Модуль Fraction.

Fraction позволяет выполнить арифметические действия над рациональными числами. Fraction можно создать из пары чисел (числитель , знаменатель), из другого рационального числа, числа с плавающей точкой, числа типа decimal. Decimal, и из строки.

В начале мы импортируем модуль Fraction:

import Fraction

# Fractions1

# Выполнили: Щепкин М.В. П2-18

"""

На вход подаются две строки с числами, содержашие пару числитель-знаменатель через пробел. Вывести сумму этих чисел

"""

from fractions import Fraction

s = input().split()

s\_1 = input().split()

print(Fraction(int(s[0]), int(s[1])) + Fraction(int(s\_1[0]), int(s\_1[1])))

В приведенной программе мы складываем два дробных числа. s - числитель и s\_1 - знаменатель,

print(Fraction(int(s[0]), int(s[1])) + Fraction(int(s\_1[0]), int(s\_1[1])))

Здесь мы берем из строки первый элемент числитель и второй знаменатель, и складываем эти числа.

input:

s = 3 4

s\_1 = 5 6

output: 19/12

Модуль Decimal. (decimal2.py)

Модуль Decimal - используется для вычисления с заданной точностью.

Чтобы использовать модуль decimal нам нужно его импортировать как показано ниже:

import decimal

Для чего нужен decimal?

С помощью модуля Decimal мы можем создавать десятичные числа и обеспечив поддержку правильного округления десятичной арифметики с плавающей точкой. к примеру:

from decimal import Decimal

number1 = Decimal("0.1")

number2 = Decimal("0.7")

print(number1 + number2)

0.8

Decimal, в отличие от float, имеет ряд преимуществ:

Десятичные числа представлены точно (в отличие от float, где такие числа как 1.1 и 5.12 не имеют точного представления);

Точность десятичного модуля Decimal можно изменять (с помощью getcontext().prec);

Рассмотрим данную задачу с использованием модуля decimal:

Для начала мы импортируем модуль Decimal как показано ниже:

from decimal import Decimal

После мы присваеваем 2 значения num и num\_2, модулю decimal

num = Decimal(input())

num\_2 = Decimal(input())

Далее мы округляем числа с использованием модуля quantize (он предназначен для округления ) и выводим сумму округлённых чисел.

print(num.quantize(Decimal("1.0000")) + num\_2.quantize(Decimal("1.0000")))

input: 5.314 + 9.41235

output: 14.7264

Модуль math (math1.py)

Библиотека Math в Python обеспечивает доступ к некоторым популярным математическим функциям и константам, которые можно использовать в коде для более сложных математических вычислений.

Специальные константы библиотеки math

В библиотеке Math в Python есть две важные математические константы.

Число Пи из библиотеки math

Для начала использования модуля math его нужно импортировать используя команду import

import math

Затем можно получить доступ к константе, вызывая число pi:

print (math.pi)

И программа выведет число pi

3.141592653589793

Данную константу можно использовать для вычисления площади или длины окружности. Далее представлен пример простого кода, с помощью которого это можно сделать:

import math

radius = 2

print('Площадь окружности с радиусом 2 равна:', math.pi \* (radius \*\* 2))

Вывод

Площадь окружности с радиусом 2 равна: 12.566370614359172

Мы возвели радиус во вторую степень и умножили значение на число Пи, как и следовало сделать в соответствии с формулой πr2.(6\_2\_1.py)

Модуль math (6\_2\_2.py)

Число Эйлера из библиотеки math

Число Эйлера (е) является основанием натурального логарифма. Оно также является частью библиотеки Math в Python. Получить доступ к числу можно следующим образом:

Так же делаем импортируем math

import math

И далее чтоб получить доступ к числу Эйлера вводим

print (math.e)

Вывод

2.718281828459045

В следующем примере представлено, как можно использовать вышеуказанную константу:

import math

print((math.e + 6 / 2) \* 4.32)

Вывод

24.702977498943074

Модуль cmath предоставляет доступ к функциям, которые могут выполнять математические действия над комплексными числами. Помимо комплексных чисел, аргументами данных функций могут быть как целые так и вещественные (с плавающей точкой) числа, а так же любые объекты языка Python с методами complex() или float() необходимые для преобразования комплексных чисел в вещественные и наоборот.

В отличии от модуля math модуль cmath в свою очередь, всегда возвращает комплексные числа, даже если для вывода достаточно целого или вещественного типа данных, просто множитель перед мнимой единицей будет равен нулю

Степенные и логарифмические функции cmath.exp(x),cmath.log и cmath.log10 вычисляет экспоненту комплексного числа.

1.cmath.exp(6\_2\_3.py)

import cmath

print(cmath.exp(0+0j))

Вывод

(1+0j)

2.cmath.log

import cmath

print(cmath.log(1 + 0j))

Вывод

0j

3.cmath.log10 (6\_2\_4.py)

import cmath

x = 10\*\*(2 + 0.5j)

print(x)

Вывод

(40.73073101539468+91.32911666577952j)

Блок-схемы для задач к практическому занятию №6.

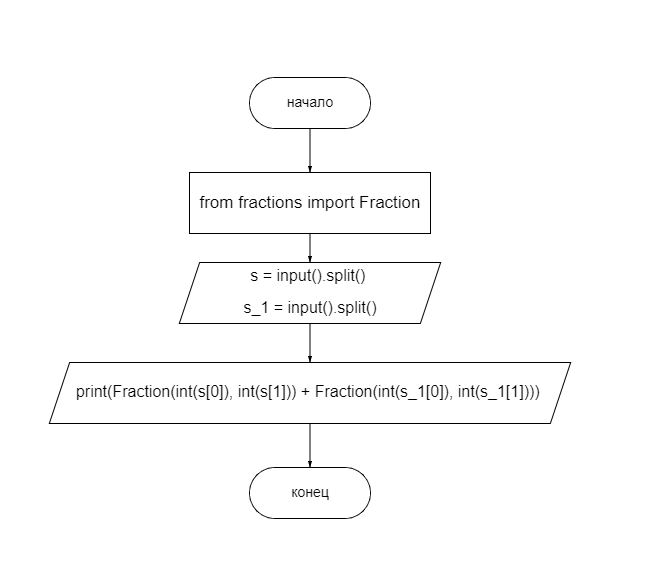
Блок-схема для листинг 21 (Модуль Fraction) : 

Рис. 5.(блок-схем для литинг 21 модуль Fraction)

Блок-схема для листинг 25 (Модуль Decimal) :

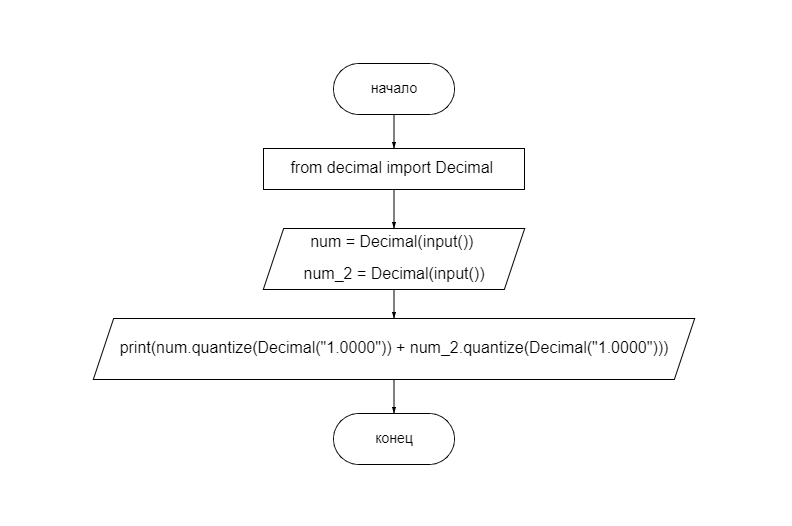


Рис. 6.(блок-схема для листинг 25 модуль Decimal)

# K7. Техника работы со строками

Задание 1.

Листинг 27: 7\_1\_1.py

#Алексей Зотов.Введение в Python (7-8 классы).

#https://stepik.org/lesson/201702/step/5?unit=175778

#Выполнили: Воронцов А.А. Бурлаев З.С. Щепкин М.В. П2-18

"""

С клавиатуры вводятся строки, последовательность заканчивается точкой.

Выведите буквы введенных слов в верхнем регистре, разделяя их пробелами.

"""

s = input()

while s != ".":

print(' '.join(list(s.upper())))

s = input()

Задание 2.

Листинг 28: 7\_1\_2.py

#Алексей Зотов.Введение в Python (7-8 классы).

#https://stepik.org/lesson/201702/step/8?unit=175778

#Выполнили: Воронцов А.А. Бурлаев З.С. Щепкин М.В. П2-18

"""

Известно, что для логина часто не разрешается использовать строки содержащие пробелы.

Но пользователю нашего сервиса особенно понравилась какая-то строка.

Замените пробелы в строке на символы нижнего подчеркивания, чтобы строка могла сгодиться для логина.

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

"""

s = input()

x = s.replace(" ", "\_")

print(x)

Задание 3.

Листинг 29: 7\_1\_3.py

#Алексей Зотов.Введение в Python (7-8 классы).

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

#Выполнили: Воронцов А.А. Бурлаев З.С. Щепкин М.В. П2-18

"""

   Уберите точки из введенного IP-адреса.

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

"""

lst = []

summ = 0

s = input()

s = s.split(".")

for i in s:

lst.append(int(i))

s = " ".join(s)

print(s)

print(sum(lst))

Задание 4.

Листинг 30: 7\_1\_4.py

#Алексей Зотов.Введение в Python (7-8 классы).

#https://stepik.org/lesson/201702/step/14?unit=175778

#Выполнили: Воронцов А.А. Бурлаев З.С Щепкин М.В П2-18

"""

Программист логирует программу, чтобы хорошо знать, как она себя ведет (эта весьма распространенная и важная практика). Он использует разные типы сообщений для вывода ошибок (error), предупреждений (warning), информации (info) или подробного описания (verbose). Сообщения отличаются по внешнему виду. Назовем модификаторами такие символы, которые отличают сообщения друг от друга, позволяя программисту понять, к какому из типов относится сообщения. Модификаторы состоят из двух одинаковых символов и записываются по разу в начале и в конце строки.

@@ обозначает ошибку

!! обозначает предупреждение

// обозначает информационное сообщение

\*\* обозначает подробное сообщение

Напишите программу, которая принимает строки до точки и выводит, какого типа это сообщение. Если сообщение не содержит модификаторов, проигнорируйте его.

"""

st = input()

whilest != ".":

if "!!" in st:

print("предупреждение")

elif "@@" in st:

print("ошибка")

elif "//" in st:

print("информация")

elif "\*\*" in st:

print("подробноесообщение")

st = input()

Задание 1.

Листинг 31: 7\_2\_1.py

Форматированием строк "По старинке" (оператор %).

Строки в Python содержат уникальную встроенную операцию, доступ к которой можно получить через оператор %.

name = input() #Zachar

print ('Hollo, %s' % nane)

#Hollo, Zachar

Определитель формата %s в данном случае, чтобы сказать Python, где именно заменить значение имени, представленного в виде строки.

Задание 2.

Листинг 32: 7\_2\_2.py

Форматирование строки "По новому" (str.format).

Новый стиль форматирования строк избавляется от специального синтаксиса оператора % и делает синтаксис для форматирования строк более регулярным. Теперь форматирование обрабатывается вызовом .format() в объекте строки.

name = input() #Zachar

print ('Hello, {}' .format(name))

#Hello, Zachar

Задание 3.

Листинг 33: 7\_2\_3.py

Интерполяция строк / f-Строки.

Добавил новый подход форматирования строк под названием форматированные строчные литералы, или “f-строки”. Этот новый способ форматирования строк позволяет вам использовать встроенные выражения Python внутрь строковых констант. Вот простой, наглядный пример:

name = 'Zachar'

print (f'Hello, {name}!')

#Hello, Zachar!

Задание 4.

Листинг 34: 7\_2\_4.py

Шаблонные строки (Стандартная библиотека Template Strings).

Рассмотрим еще один инструмент для форматирования строк в Python: template strings. Сначала нам нужно импортировать класс Template из встроенного модуля Python.

from string import Template

t = Template('Hey, $name!')

name = input() #Zachar

print(t.substitute(name=name))

#Hey, Zachar!

Блок-схемы для задач к практическому занятию №7.

Блок-схема для листинг 30: 7\_1\_4.py

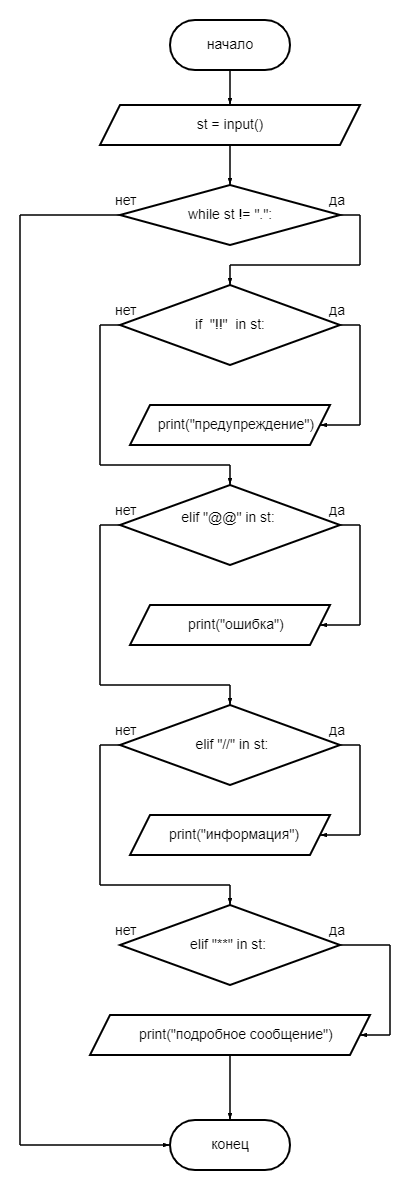


Рис. 7.(блок-схема для листинг 30, 7\_1\_4)

Блок-схема для листинг 27: 7\_1\_1.py

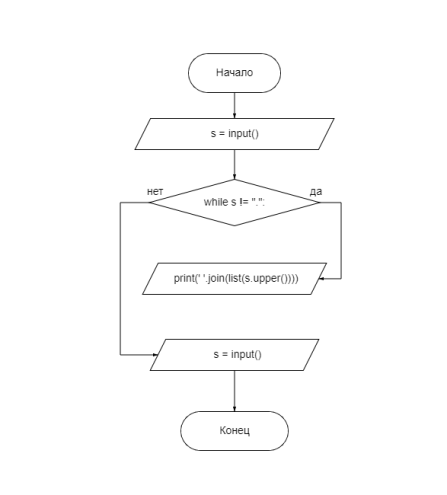


Рис. 8.(блок-схема для листинг 27, 7\_1\_1)

# K8. Техника работы со списками.

Задание 1.

Листинг 35: 8\_1\_1.py

#Pythontutor

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

#Выполнили: Щепкин М.В. П2-18

"""

Дан список чисел. Определите, сколько в этом списке элементов, которые больше двух своих соседей, и выведите количество таких элементов.

Крайние элементы списка никогда не учитываются, поскольку у них недостаточно соседей.

"""

cnt = 0

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

counter = 0

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

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

cnt += 1

print(cnt)

Задание 2.

Листинг 36: 8\_1\_2.py

#Pythontutor

#https://pythontutor.ru/lessons/lists/problems/num\_equal\_pairs/

#Выполнили: Щепкин М.В. П2-18

"""

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.

"""

cnt = 0

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

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

for j in range(i+1, len(a)):

if a[i] == a[j]:

cnt += 1

print(cnt)

Задание 3.

Листинг 37: 8\_1\_3.py

#Гусятинер Л.Б.

#Выполнили: Щепкин М.В. П2-18

"""

Дано N списков целых чисел (N вводится с клавиатуры, сами списки заполняются

случайным образом). Требуется сформировать

- список, содержащий уникальные значения, попадающие в каждый из N списков

- список, содержащий уникальные значения, попадающие хотя бы в один из N списков

Решение без использования set - дополнительный бонус

"""

import random

uni\_1 = [] # список, содержащий уникальные значения, попадающие в каждый из N списков

uni\_2 = [] # список, содержащий уникальные значения, попадающие хотя бы в один из N списков

n = int(input())

sp = []

for k in range(n):

vr\_sp = []

lensp = random.randint(1, 5)

for l in range(lensp):

vr\_sp.append(random.randint(0, 10))

sp.append(vr\_sp)

print(sp)

for i in range(len(sp)):

for j in range(0, len(sp[i])):

uni = sp[i][j]

cnt = 0

for z in range(0, n):

if uni in sp[z]:

cnt += 1

if cnt == (n) and uni not in uni\_1:

uni\_1.append(uni)

print(uni\_1)

for i in range(len(sp)):

for j in range(len(sp[i])):

uni = sp[i][j]

if uni not in uni\_2:

uni\_2.append(uni)

print(uni\_2)

Задание 4.

Листинг 38: 8\_1\_4.py

#Гусятинер Л.Б.

#Array112

# Выполнили: Щепкин М.В. П2-18

"""

Дан массив A размера N.

Упорядочить его по возрастанию методом сортировки

простым обменом («пузырьковой» сортировкой):

просматривать массив, сравнивая его соседние элементы

(A0 и A1, A1 и A2 и т. д.) и меняя их местами,

если левый элемент пары больше правого; повторить описанные

действия N 1 раз. Для контроля за выполняемыми действиями

выводить содержимое массива после каждого просмотра.

Учесть, что при каждом просмотре количество анализируемых

пар можно уменьшить на 1.

"""

n = int(input())

a = [int(input()) for k in range(n)]

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

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

if a[j] > a[j+1]:

a[j], a[j+1] = a[j+1], a[j]

print(a)

Задание 5.

Листинг 39: 8\_1\_5.py

#Гусятинер Л.Б.

#Array113

# Выполнили: Щепкин М.В. П2-18

"""

Дан массив A размера N.

Упорядочить его по возрастанию методом сортировки простым

выбором: найти максимальный элемент массива и поменять его

местами с последним (N-1 м) элементом; выполнить описанные

действия N 1 раз, каждый раз уменьшая на 1 количество

анализируемых элементов и выводя содержимое массива.

"""

n = int(input())

a = [int(input()) for num in range(n)]

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

m = i

j = i + 1

while j < len(a):

if a[j] < a[m]:

m = j

j = j + 1

a[i], a[m] = a[m], a[i]

print(a)

Задание 6.

Листинг 40: 8\_1\_6.py

#Гусятинер Л.Б.

#Array114

# Выполнили: Щепкин М.В. П2-18

"""

Дан массив A размера N. Упорядочить

его по возрастанию методом сортировки простыми вставками:

сравнить элементы A0 и A1 и, при необходимости меняя их

местами, добиться того, чтобы они оказались упорядоченными

по возрастанию; затем обратиться к элементу A2 и

переместить его в левую (уже упорядоченную) часть массива,

сохранив ее упорядоченность; повторить этот процесс для

остальных элементов, выводя содержимое массива после

обработки каждого элемента (от 1-го до N-1 го).

"""

n = int(input())

a = [int(input()) for num in range(n)]

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

m = a[i]

j = i-1

while j >=0 and m < a[j] :

a[j+1] = a[j]

j -= 1

a[j+1] = m

print(a)

Блок-схемы для задач к практическому занятию №8.

Блок-схема для листинг 31: 7\_2\_1.py

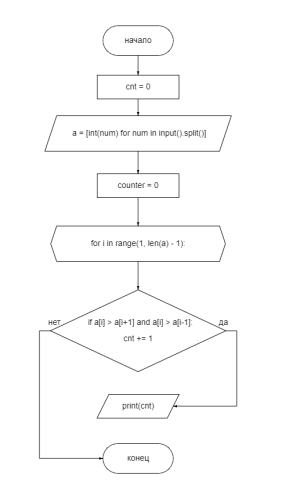


Рис. 9.(блок-схема для листинг 31, 7\_2\_1)

Блок-схема для листинг 32: 7\_2\_2.py

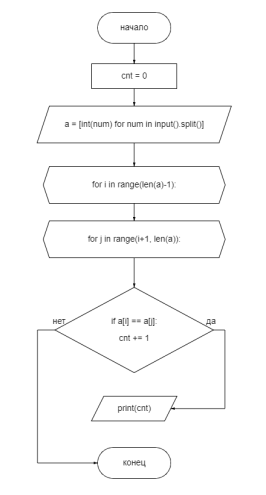


Рис. 10.(блок-схема для листинг 32, 7\_2\_2)

# K9. Техника работы с циклом for и генераторами списков.

Задание 1.

Листинг 41: 9\_2\_1.py

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

#Array 55

#Выполнили:. Щепкин М.В. П2-18

"""

Дан целочисленный массив A размера N (<= 15). Переписать в новый целочисленный

массив B все элементы с нечетными порядковыми номерами (1, 3, ...) и вывести размер

полученного массива B и его содержимое. Условный оператор не использовать.

"""

n = int(input())

a = [int(input()) for i in range(n)]

b = a[0::2]

print(b, len(b))

Задание 2.

Листинг 42: 9\_1\_2.py

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

#Array 57

#Выполнили: Щепкин М.В. П2-18

"""

Задание 2. Array57. Дан целочисленный массив A размера N. Переписать в новый целочисленный массив B

того же размера вначале все элементы исходного массива с четными номерами,

азатем — снечетными:

A[0], A[2], A[4], A[6], ..., A[1], A[3], A[5], ... .

Условный оператор не использовать.

"""

n = int(input())

a = [int(input()) for i in range(n)]

b = a[::2]

b.extend(a[1::2])

print(b)

Задание 3.

Листинг 43: 9\_1\_3.py

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

#Array 58

#Выполнили: Щепкин М.В. П2-18

"""

Дан массив A размера N.

Сформировать новый массив B того же размера по

следующему правилу: элемент B[K] равен сумме элементов

массива A с номерами от 0 до K.

"""

n = int(input())

a = [int(input()) for i in range(n)]

b = []

for i in range(n):

num = sum(range(a[0], a[i]+1))

b.append(num)

print(b)

Задание 4.

Листинг 44: 9\_1\_4.py

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

#Matrix3

#Выполнили: Щепкин М.В. П2-18

"""

Даны целые положительные числа M, N и набор из M чисел.

Сформировать матрицу размера M x N, у которой в каждом столбце

содержатся все числа из исходного набора (в том же порядке).

"""

m = int(input('число строк '))

n = int(input('число столбцов '))

a = [int(input()) for i in range(m)]

b = [[a[j]]\*n for j in range(m)]

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

Задание 5.

Листинг 45: 9\_1\_5.py

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

#Matrix56

#Выполнили: Щепкин М.В. П2-18

"""

Дана матрица размера M x N (N — четное число). Поменять местами

левую и правую половины матрицы.

"""

import random

m = int(input())

n = int(input())

a = [[random.randint(0, 10)for j in range(n)] for i in range(m)]

temp = []

print(\*a, sep = "\n")

for i in range(0, m):

for j in range(0, n//2):

a[i][j], a[i][n//2+j] = a[i][n//2+j], a[i][j]

print("\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/")

print(\*a, sep = "\n")

Задание 6.

Листинг 46: 9\_1\_6.py

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

#Matrix88

#Выполнили: Щепкин М.В. П2-18

"""

Дана квадратная матрица порядка M. Обнулить элементы матрицы,

лежащие ниже главной диагонали. Условный оператор не использовать.

"""

import random

m = int(input())

a = [[random.randint(0, 9)for j in range(m)] for i in range(m)]

print(\*a, sep = "\n")

for i in range(0, m):

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

a[j][i] = 0

print("\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/")

print(\*a, sep = "\n")

Блок-схемы для задач к практическому занятию №9.

Блок-схема для задачи листинг 43: 9\_1\_3.py



Рис. 11.(блок-схема для задачи, 9\_1\_3)

Блок-схема для задачи листинг 44: 9\_1\_4.py

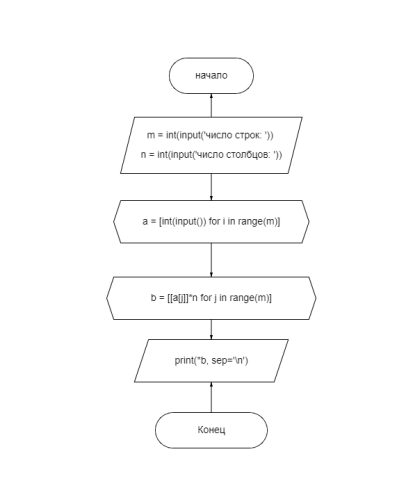


Рис. 12.(блок-схема для задачи, 9\_1\_4)

# K10. Техника работы с функциями.

Задание 1.

Листинг 47: 10\_1\_1.py

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

#Func6

#Выполнили: Щепкин М.В. П2-18

"""

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

чисел от A до B включительно (A и B — целые). Если A > B, то функция возвращает 0.

С помощью этой функции найти суммы чисел от A до B и от B до C, если даны числа A, B, C.

"""

def SumRange(a, b):

res = sum(range(a, b+1))

if a > b:

return 0

else:

return res

a = int(input())

b = int(input())

c = int(input())

print(SumRange(a, b))

print(SumRange(b, c))

Задание 2.

Листинг 48: 10\_1\_2.py

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

#Func10

#Выполнили: Щепкин М.В. П2-18

"""

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

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

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

положительных чисел.

"""

def IsSquare(K):

for i in range(1, K):

if i\*\*2 == K:

return True

break

return False

a = [int(input()) for x in range(10)]

for num in a:

print(IsSquare(num))

Задание 3.

Листинг 49: 10\_1\_3.py

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

#Func33

#Выполнили: Щепкин М.В. П2-18

"""

Описать функцию SortInc3(X), меняющую содержимое списка X из трех

вещественных элементов таким образом, чтобы их значения оказались упорядоченными по

возрастанию (функция возвращает None). С помощью этой функции упорядочить по

возрастанию два данных списка X и Y.

"""

def SortInc3(X):

return sorted(X)

x = [int(input()) for i in range(3)]

y = [int(input()) for j in range(3)]

print(SortInc3(x))

print(SortInc3(y))

Часть 2

Задание 1.

Листинг 50: 10\_2\_1.py

## Алексей Зотов.Введение в Python (7-8 классы).

# https://stepik.org/lesson/201702/step/13?unit=175778

# Выполнили: Щепкин М.В. П2-18, Гусятинер Л.Б.

"""

Использовать map, lambda

Квадраты в обратном порядке. Числа вводятся до точки. Через пробел выведите эти числа в

обратном порядке, возводя их в квадрат.

"""

lst = []

lst\_2 = []

s = input()

while s != ".":

lst.append(int(s))

s = input()

lst\_2 = list(map(lambda x: x\*\*2, lst[::-1]))

print(\*lst\_2)

Задание 2.

Листинг 51: 10\_2\_2.py

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

# array55

# Выполнили: Щепкин М.В. П2-18

"""

Дан целочисленный массив A размера N (<= 15). Переписать в новый целочисленный

массив B все элементы с нечетными порядковыми номерами (1, 3, ...) и вывести размер

полученного массива B и его содержимое. Условный оператор не использовать.

"""

n = int(input())

a = [int(input()) for y in range(n)]

b = list(filter(lambda x: x%2, a))

print(b, len(b))

Задание 6.

Листинг 52: 10\_2\_3.py

# Алексей Зотов.Введение в Python (7-8 классы).

# https://stepik.org/lesson/239422/step/2?unit=211833

# Выполнили: Щепкин М.В. П2-18

"""

Быстрая инициализация. Программа получает на вход три числа через пробел — начало и конец

диапазона, а также степень, в которую нужно возвести каждое число из диапазона. Выведите

числа получившегося списка через пробел.

"""

a, b, c = map(int, input().split())

a = [i for i in range(a, b+1)]

print(\*list(map(lambda x: x\*\*c, a)))

Блок-схемы для задач к практическому занятию №10.

Блок-схема для задачи листинг 47: 10\_1\_1.py

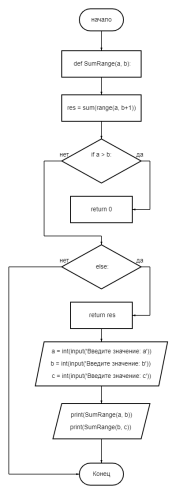


Рис. 13.(блок-схема для задачи, 10\_1\_1)

Блок-схема для задачи листинг 49: 10\_1\_3.py

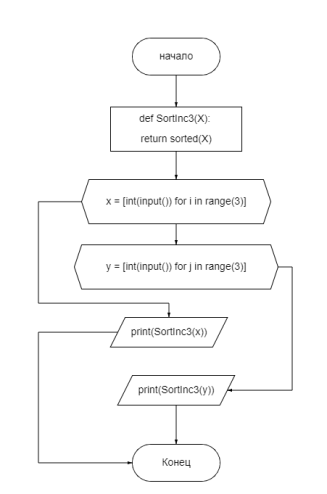


Рис. 14.(блок-схемы для задачи, 10\_1\_3)

# K11. Техника работы со словарями.

Задание 1.

Листинг 53: 11\_1\_1.py

# https://pythontutor.ru/lessons/dicts/problems/occurency\_index/

# Задача «Номер появления слова»

# Выполнили: Щепкин М.В. 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.

"""

d = dict()

for x in input().split():

if x in d.keys():

d[x] = d[x] + 1

else:

d[x] = 0

print(d[x], end = ' ')

Задание 2.

Листинг 54: 11\_1\_2.py

# https://pythontutor.ru/lessons/dicts/problems/permissions/

# Задача «Права доступа»

# Выполнили:0 Щепкин М.В. П2-18

"""

В файловую систему одного суперкомпьютера проник вирус, который сломал контроль

за правами доступа к файлам. Для каждого файла известно, с какими действиями можно к

нему обращаться:

запись W,

чтение R,

запуск X.

В первой строке содержится число N — количество файлов содержащихся в данной файловой

системе. В следующих N строчках содержатся имена файлов и допустимых с ними операций,

разделенные пробелами. Далее указано чиcло M — количество запросов к файлам. В последних

M строках указан запрос вида Операция Файл. К одному и тому же файлу может быть применено

любое колличество запросов.

Вам требуется восстановить контроль над правами доступа к файлам (ваша программа для

каждого запроса должна будет возвращать OK если над файлом выполняется допустимая

операция, или же Access denied, если операция недопустима.

"""

dst = {

'read': 'R',

'write': 'W',

'execute': 'X',

}

df = {}

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

file, \*per = input().split()

df[file] = set(per)

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

action, file = input().split()

if dst[action] in df[file]:

print('OK')

else:

print('Access denied')

Задание 3.

Листинг 55: 11\_1\_3.py

# https://pythontutor.ru/lessons/dicts/problems/most\_frequent\_word/

# Задача «Самое частое слово»

# Выполнили: Щепкин М.В. П2-18

"""

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

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

выведите то, которое меньше в лексикографическом порядке.

"""

d = {}

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

line = input().split()

for word in line:

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

maxi = max(d.values())

sl = [j for j, v in d.items() if maxi == v]

print(min(sl))

Блок-схемы для задач к практическому занятию №11.

Блок-схема для листинг 53: 11\_1\_1.py

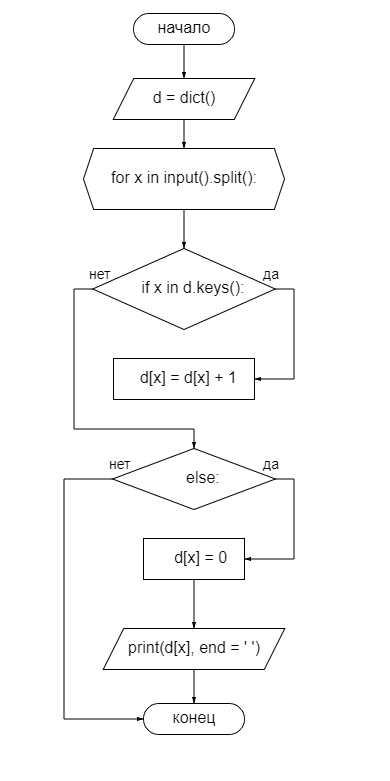


Рис. 15.(блок-схема для листинг 49, 11\_1\_1)

# K12. Техника работы с множествами.

Задание 1.

Листинг 56: 12\_1\_1.py

#Pythontutor

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

#Выполнили: Щепкин М.В. П2-18

"""

Дан список чисел. Определите, сколько в нем встречается различных чисел.

"""

a = input().split()

print (len(set(a)))

Задание 2.

Листинг 57: 12\_1\_2.py

#Pythontutor

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

#Выполнили: Щепкин М.В. П2-18

"""

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

в первом списке, так и во втором.

"""

a = input().split()

b = input().split()

c = a + b

print(len(c) - len(set(c)))

Задание 3.

Листинг 58: 12\_1\_3.py

#Pythontutor

#https://pythontutor.ru/lessons/sets/problems/sets\_intersection/

#Выполнили: Щепкин М.В. П2-18

"""

Даны два списка чисел. Найдите все числа, которые входят как в первый,

так и во второй список и выведите их в порядке возрастания.

"""

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

b = []

for i in a:

b.append(int(i))

b = sorted(b)

for i in b:

print(i)

Задание 4.

Листинг 59: 12\_1\_4.py

#Pythontutor

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

#Выполнили: Щепкин М.В. П2-18

"""

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

Определите, сколько различных слов содержится в этом тексте.

Словом считается последовательность непробельных символов идущих подряд, слова разделены

одним или большим числом пробелов или символами конца строки.

"""

t = set()

n = int(input())

for i in range (n):

t.update(input().split())

print (len(t))

Задание 5.

Листинг 60: 12\_1\_5.py

#Pythontutor

#https://pythontutor.ru/lessons/sets/problems/polyglotes/

#Выполнил: Щепкин М.В. П2-18

"""

Каждый из некоторого множества школьников некоторой школы знает некоторое

количество языков. Нужно определить сколько языков знают все школьники, и сколько языков

знает хотя бы один из школьников.

В первой строке задано количество школьников. Для каждого из школьников сперва записано

количество языков, которое он знает, а затем - названия языков, по одному в строке.

В первой строке выведите количество языков, которые знают все школьники. Начиная со

второй строки - список таких языков. Затем - количество языков, которые знает хотя бы

один школьник, на следующих строках - список таких языков. Языки нужно выводить в

лексикографическом порядке, по одному на строке.

"""

n = int(input())

b = set()

c = set()

for i in range(n):

f = int(input())

a = set()

for j in range(f):

a.add(input())

if i < 1:

c = a

b |= a

else:

b |= a

c &= a

print(len(c))

print("\n".join(sorted(c)))

print(len(b))

print("\n".join(sorted(b)))

Блок-схемы для задач к практическому занятию №12.

Блок-схема для листинг 57: 12\_1\_2.py

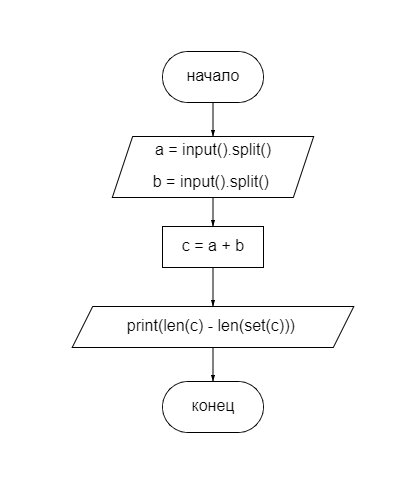


Рис. 16.(блок-схема для листинг 57, 12\_1\_2)

Блок-схема для листинг 59: 12\_1\_4.py

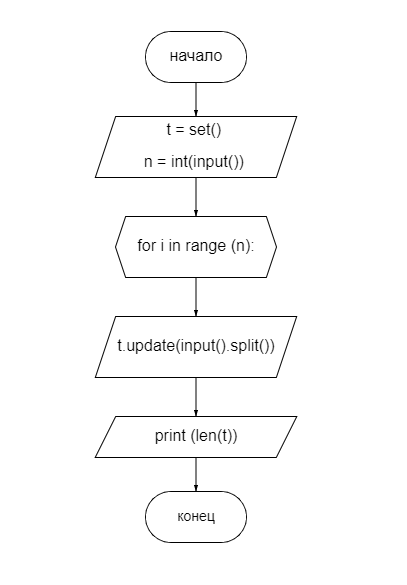


Рис. 17.(блок-схема для листинг 59, 12\_1\_4)

# K13. Техника работы с кортежами.

Задание 1.

Листинг 61: 13\_1\_1.py

#Алексей Зотов. Введение в Python (7-8 классы).

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

#Выполнили: Щепкин М.В П2-18

"""

Вывести чётные

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

"""

a = int(input())

b = tuple(x for x in range(a, a\*10+1) if x%2==0)

print(b)

Задание 2.

Листинг 61: 13\_1\_2.py

#Алексей Зотов. Введение в Python (7-8 классы).

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

#Выполнили: Щепкин М.В П2-18

"""

Убывающий ряд.

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

по одному числу в строке.

"""

a = int(input())

b = int(input())

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

print(i)

Задание 3.

Листинг 61: 13\_1\_3.py

#Гусятинер Л.Б.

#Зад 3

#Выполнили: Щепкин М.В П2-18

"""

В каждой строке файла хранится информация о пунктах и их координатах

относительно некоторого центра.

Требуется

1. Прочесть файл в список кортежей

2. Найти диаметр множества точек, то есть расстояние между наиболее удалёнными точками.

Указать наиболее удалённые пары

3. Сформировать список пар городов, имеющих одинаковое расстояние до центра

4. Отсортировать список одним из методов, реализованных в предыдущих работах

Результаты вывести на экран

"""

a = []

f = open('zad 3.txt', 'r')

for line in f:

lst = line.split()

cort = (lst[0], int(lst[1]), int(lst[2]))

a.append(cort)

print(a)

f.close()

# K14. Техника работы с файлами.

Задание 1.

Листинг 61: 14\_1\_1.py

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

#Text5

# Выполнили: Щепкин М.В. П2-18

"""

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

"""

f = open("zad 1.txt", "a")

s = input()

f.write(s)

f.close()

Задание 2.

Листинг 62: 14\_1\_2.py

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

#Text12

# Выполнили: Щепкин М.В. П2-18

"""

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

"""

s = input()

with open("zad 2.txt", "r") as f:

lines = f.readlines()

for i in range(0, len(lines)):

if lines[i] == "\n":

lines[i] = s+"\n"

with open("zad 2.txt", "w") as f:

f.writelines(lines)

Задание 3.

Листинг 63: 14\_1\_3.py

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

#Text20

#Выполнили: Щепкин М.В. П2-18

"""

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

"""

a = input()

with open('text.txt','a') as b:

b.write(' '.join(a.split()))

Задание 4.

Листинг 64: 14\_1\_4.py

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

#Text44

# Выполнили: Щепкин М.В. П2-18

"""

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

несколькими пробелами. Вывести количество этих чисел и их сумму.

"""

a = []

with open("zad 4.txt", "r") as f:

lines = f.readlines()

for i in range(0, len(lines)):

a.append(int(lines[i].strip()))

print(len(a), sum(a))

Задание 5.

Листинг 65: 14\_1\_4.py

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

#Text44

# Выполнили: Воронцов А.А. Бурлаев З.С. Щепкин М.В. П2-18

"""

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

встретившиеся в текстовом файле (в том же порядке).

"""

a = []

punctuation=['(', ')', '?', ':', ';', ',', '.', '!', '/', '"', "'"]

with open("zad 5 in.txt", "r") as f:

lines = f.readlines()

for i in range(0, len(lines)):

for j in range(0, len(lines[i])):

if lines[i][j] in punctuation:

a.append(lines[i][j])

with open("zad 5 out.txt", "w") as f\_o:

f\_o.writelines(a)

Блок-схемы для задач к практическому занятию №14.

Блок-схема для листинг 62: 14\_1\_2.py

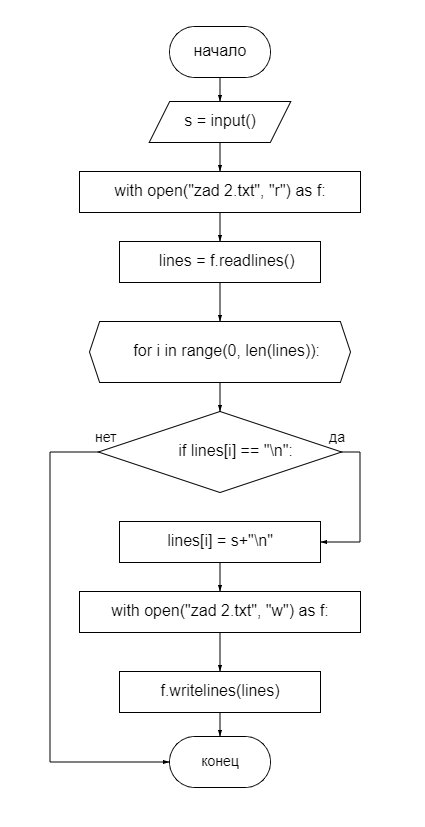


Рис. 18.(блок-схема для листинг 62, 14\_1\_2)

Блок-схема для листинг 64: 14\_1\_4.py

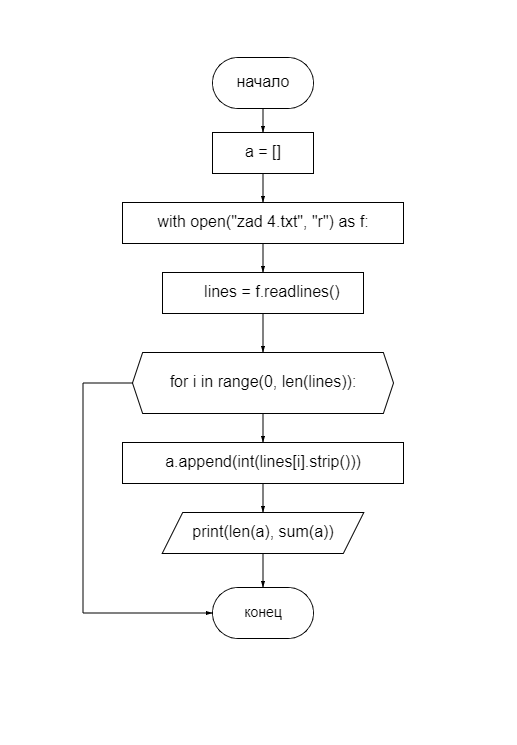


Рис. 19.(блок-схема для листинг 64, 14\_1\_4)

# K15.Техника работы с модулями.

Задание 1.

Листинг 66: 15\_1\_1.py

#Выполнили: Щепкин М.В П2-18

"""

Контейнерные типы данных модуля collections.

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

Класс deque() модуля collections в Python.

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

Двусторонняя очередь в Python.

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

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

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

в любом направлении.

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

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

как размер, так и положение базового представления данных.

Синтаксис:

import collections

dq = collections.deque([iterable[, maxlen]])

Параметры:

iterable - итерируемая последовательность,

maxlen - int, максимальное кол-во хранимых записей.

"""

# Примеры работы с ним:

from collections import deque

# Создаем

D = deque(["Mon", "Tue", "Wed"])

print(D)

# Добавим вправо

D.append("Thu")

print(D)

# Добавим влево

D.appendleft("Sun")

print(D)

# Убираем справа

D.pop()

print(D)

# Убираем слева

D.popleft()

print(D)

# Переворачиваем

D.reverse()

print(D)

# Можно ограничить его размер:

D = deque([], maxlen=3)

D.append(5)

print(D)

D.append(55)

print(D)

D.append(555)

print(D)

D.append(5555)

print(D)

# Когда будет превышен лимит, D[0] исчезнет.

Задание 2.

Листинг 67: 15\_1\_2.py

#Выполнили: Щепкин М.В П2-18

"""

Контейнерные типы данных модуля collections.

Класс Counter() модуля collections в Python.

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

Подсчет количества повторений элементов в последовательности.

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

элементов в последовательностях.

>>> from collections import Counter

>>> 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.

"""

# Примеры работы с ним:

import collections

c = (['a', 'b', 'c', 'd', 'e', 'f', 'f', 'f', 'a', 'c', 'd'])

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

print('А сколько в ней элементов:')

print(collections.Counter(c))

print('\n')

print('Так же с помощью ')

Задание 3.

Листинг 68: 15\_1\_3.py

#Выполнили: Щепкин М.В П2-18

"""

# Задание 1. Функция argv модуля sys в Python.

# Функция argv модуля sys в Python.

import sys

import os

print('Список аргументов')

print(sys.argv)

print('Исходные байты')

print([os.fsencode(arg) for arg in sys.argv])

# Задание 2. Имя используемой OS.

# Функция для Unix sys.platform,

# Функция для Windows sys.getwindowsversion(),

'''

sys.platform:

Функция sys.platform возвращает строку, которая содержит идентификатор платформы,

который можно использовать, например, для добавления компонентов,специфичных для платформы.

Для систем Unix, за исключением Linux и AIX, это имя ОС в нижнем регистре,

возвращаемое командой bash uname -s с добавлением первой части версии,

возвращаемой командой uname -r. Например 'sunos5' или 'freebsd8'.

Поэтому, рекомендуется использовать следующую идиому:

'''

# if sys.platform.startswith('freebsd'):

# elif sys.platform.startswith('linux'):

# elif sys.platform.startswith('aix'):

# Задание 3. Различные сведения о версии Python.

'''

Содержание:

Свойство sys.copyright,

Свойство sys.version,

Свойство sys.api\_version,

Свойство sys.version\_info,

Свойство sys.implementation,

Свойство sys.hexversion,

Свойство sys.winver.

sys.copyright:

Переменная sys.copyright возвращает строку, содержащая авторские права, относящиеся к интерпретатору Python.

sys.version:

Переменная sys.version возвращает строку, содержащую номер версии интерпретатора Python плюс дополнительную информацию

о номере сборки и используемом компиляторе.

Данная строка отображается при запуске интерактивного переводчика. Не извлекайте информацию из нее, лучше используйте

sys.version\_info и функции, предоставляемые модулем platform.

sys.api\_version:

Переменная sys.api\_version возвращает версию API языка C для этого интерпретатора Python. Программисты могут

использовать ее при отладке конфликтов версий между Python и модулями расширения.

sys.version\_info:

Переменная sys.api\_version возвращает кортеж, содержащий пять компонентов номера версии:

major,

minor,

micro,

releaselevel,

serial.

Все значения, кроме releaselevel, являются целыми числами.

Значения releaselevel:

'alpha',

'beta',

'candidate'

'final'.

Компоненты также могут быть доступны по имени, sys.version\_info[0] что эквивалентно sys.version\_info.major и так далее.

sys.implementation:

Атрибут sys.implementation представляет собой объект, содержащий информацию о реализации текущего запущенного

интерпретатора Python.

Следующие атрибуты должны существовать во всех реализациях Python.

name - это идентификатор реализации, например "cpython". Фактическая строка определяется реализацией Python,

но она гарантированно будет строчной.

version - это именованный кортеж, в том же формате, что и sys.version\_info.

Он представляет собой версию реализации Python. Значение version отличается от конкретной версии языка Python,

которой соответствует текущий работающий интерпретатор. Например, для PyPy 1.8 sys.implementation.version может

быть sys.version\_info (1, 8, 0, 'final', 0),

тогда как sys.version\_info будет sys.version\_info (2, 7, 2, 'final', 0).

Для CPython они имеют одинаковое значение, так как это эталонная реализация.

hexversion - это версия реализации в шестнадцатеричном формате, например sys.hexversion.

cache\_tag - это тег, используемый механизмом импорта в именах файлов кэшируемых модулей.

По соглашению, это будет составная часть имени и версии реализации, например, "cpython-38".

Хотя реализация Python может использовать другое значение, если это уместно. Если для cache\_tag установлено значение

None, это означает, что кэширование модуля должно быть отключено.

Атрибут sys.implementation может содержать дополнительные атрибуты, специфичные для реализации Python.

Эти нестандартные атрибуты должны начинаться с подчеркивания и здесь не описаны. Независимо от содержимого

sys.implementation не

изменится ни во время выполнения интерпретатора, ни между версиями реализации. Однако, может измениться между версиями

языка Python.

sys.hexversion:

Значение sys.hexversion содержит номер версии, закодированный как одно целое число.

Оно гарантированно увеличивается с каждой версией, включая надлежащую поддержку выпусков.

Например, чтобы проверить, что интерпретатор Python имеет версию не ниже 3.5.2, используйте:

'''

# if sys.hexversion >= 0x030502F0:

# используйте дополнительные функции

# else:

# используйте альтернативную реализацию

# или предупредите пользователя

'''

Значение sys.hexversion это шестнадцатеричный номер версии и представляет собой результат передачи его встроенной

функции hex().

Для более удобного получения той же самой информации может использоваться именованный кортеж, возвращаемый

sys.version\_info.

sys.winver:

Номер версии, используемый для формирования ключей реестра на платформах Windows.

Значение хранится как строковый ресурс 1000 в DLL-библиотеке Python.

Обычно это первые три символа, возвращаемые sys.version. Значение sys.winver представлен в модуле sys в ознакомительных

целях, изменение этого значения не влияет на ключи реестра, используемые Python.

'''

# Задание 4. Каталоги и пути интерпретатора Python.

'''

Установка специфичных каталогов и путей для программы.

Следующие переменные модуля sys предоставляют информацию о каталогах и путях интерпретатора Python из кода.

Содержание:

Специфичный для программы каталог Python sys.prefix,

Каталог установки Python sys.base\_prefix,

Каталог для конкретной программы sys.exec\_prefix,

Каталог базовой установки Python sys.base\_exec\_prefix,

Путь исполняемого файла интерпретатора Python sys.executable,

Имя каталога библиотеки для конкретной платформы sys.platlibdir.

sys.prefix:

Атрибут sys.prefix это строка, предоставляет специфичный для площадки префикс каталога,

в котором установлены независимые от платформы файлы Python. По умолчанию, это строка '/usr/local'.

Значение sys.prefix устанавливается во время сборки Python с помощью аргумента --prefix для скрипта configure,

например $ ./configure --prefix=/opt/python-3.x.x/". Основная коллекция модулей библиотеки Python установлена в

каталоге prefix/lib/pythonX.Y, а независимые от платформы заголовочные файлы (все, кроме pyconfig.h) хранятся в

prefix/include/pythonX.Y, где XY - номер версии Python, например 3.7.

Примечание. Если запущена виртуальная среда, это значение будет изменено в site.py для указания на виртуальную среду.

Значение для установки Python по-прежнему будет доступно через base\_prefix.

sys.base\_prefix:

Атрибут sys.base\_prefix устанавливается при запуске Python перед запуском site.py то же значение, что и атрибут

sys.prefix.

Если не работает в виртуальной среде, значения останутся прежними. Если site.py обнаружит, что используется

виртуальная среда,

то значения sys.prefix и sys.exec\_prefix будут изменены на значения установки виртуальной среды, тогда как

sys.base\_prefix и

sys.base\_exec\_prefix останутся указывать на базовую установку Python (ту, из которой была создана виртуальная среда).

sys.exec\_prefix:

Атрибут sys.exec\_prefix это строка, задает префикс каталога для конкретной программы,

где установлены зависящие от платформы файлы Python. По умолчанию это '/usr/local'.

Значение sys.prefix устанавливается во время сборки Python с помощью аргумента --exec-prefix для скрипта configure.

В частности, все файлы конфигурации (например, заголовочный файл pyconfig.h) установлены в каталоге

exec\_prefix/lib/pythonX.Y/config,

а модули разделяемой библиотеки установлены в exec\_prefix/lib/pythonX.Y/lib-dynload, где XY номер версии Python,

например 3.7.

Примечание. Если запущена виртуальная среда, то значение sys.exec\_prefix будет изменено в site.py для указания на

виртуальную среду.

Значение базовой установки Python по-прежнему будет доступно через атрибут sys.base\_exec\_prefix.

sys.base\_exec\_prefix:

Атрибут sys.exec\_prefix устанавливает при запуске Python до запуска site.py то же значение, что и sys.exec\_prefix.

Если программа не работает в виртуальной среде, то значения останутся прежними. Если site.py обнаружит,

что используется виртуальная среда, то значения sys.prefix и sys.exec\_prefix будут изменены на значения установки

виртуальной среды,

тогда как sys.base\_prefix и sys.base\_exec\_prefix останутся указывать на базовую установку Python (ту, из которой

была создана

виртуальная среда).

sys.executable:

Атрибут sys.executable это строка, задающая абсолютный путь исполняемого двоичного файла для интерпретатора Python

в системах,

где это имеет смысл. Если Python не может получить реальный путь к своему исполняемому файлу, sys.executable будет

пустой строкой

или None.

sys.platlibdir:

Атрибут sys.exec\_prefix это имя каталога библиотеки для конкретной платформы. Он используется для построения пути к

стандартной библиотеке и путей установленных модулей расширения. (Новое в Python 3.9.)

На большинстве платформ он равен lib. В Fedora и SuSE на 64-битных платформах он равен lib64, что дает следующие

пути sys.path

(где X.Y - версия Python major.minor):

/usr/lib64/pythonX.Y/: Стандартная библиотека (например, os.py модуля os),

/usr/lib64/pythonX.Y/lib-dynload/: модули расширения языка C стандартной библиотеки (например, модуль errno, точное имя

файла зависит от платформы),

/usr/lib/pythonX.Y/site-packages/ (всегда используйте lib, а не sys.platlibdir): сторонние модули,

/usr/lib64/pythonX.Y/site-packages/: модули расширения языка C сторонних пакетов.

'''

# Задание 5. Объекты stdin, stdout, stderr модуля sys в Python.

os = sys.platform

if os == "win32":

# Работаем с реестром Windows

import \_winreg

elif os.startswith('linux'):

# Выполняем Linux комманду

import subprocess

subprocess.Popen(["ls, -l"])

# Задание 6. Функция exit() модуля sys в Python.

sys.exit([arg])

# Функция exit() модуля sys - выход из Python. Она реализуется путем вызова исключения SystemExit,

# поэтому выполняются действия по очистке, указанные в предложениях

# finally операторов try и можно перехватить попытку выхода на внешнем уровне.

# Параметры:

# arg - статус выхода, по умолчанию 0

# Возвращаемое значение:

# целое число, указывающее статус выхода.

# Необязательный аргумент arg может быть целым числом, указывающим статус выхода

# (по умолчанию равен нулю) или другим типом объекта. Если это целое число, ноль считается "успешным завершением",

# а любое ненулевое значение считается "ненормальным завершением" и т.п.

# Большинство систем требуют, чтобы arg находился в диапазоне 0 - 127,

# и в противном случае дают неопределенные результаты.

# Некоторые системы имеют соглашение о назначении определенных значений определенным кодам выхода, но они, как правило,

# недостаточно развиты. Программы Unix обычно используют 2 для ошибок

# синтаксиса командной строки и 1 для всех других видов ошибок.

# Если передается объект другого типа, то None эквивалентен передаче нуля, а любой другой объект выводится на sys.stderr

# и приводитк коду выхода 1.

# В частности, sys.exit() - это быстрый способ выйти из программы при возникновении ошибки.

"""

Задание 1.

Листинг 69: 15\_2\_1.py

#Выполнили: Щепкин М.В П2-18

# Задание 1. Вывод текущей директории

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

Задание 2.

Листинг 70: 15\_2\_2.py

#Выполнили: Щепкин М.В П2-18

# Задание 2. Создание папки

fileName = input('Введите имя файла: ')

if not os.path.isdir(fileName):

os.mkdir(fileName)

Задание 3.

Листинг 71: 15\_2\_3.py

#Выполнили: Щепкин М.В П2-18

# Задание 3. Изменение директории

changesName = input()

os.chdir(changesName)

print("Текущая директория изменилась на", "<", changesName, ">", ":", os.getcwd())

Задание 4.

Листинг 72: 15\_2\_4.py

#Выполнили: Щепкин М.В П2-18

# Задание 4. Создание вложенных папок

File = input('Введите имя файла для вложенной папки: ')

os.makedirs(File)

Задание 5.

Листинг 73: 15\_2\_5.py

#Выполнили: Щепкин М.В П2-18

# Задание 5. Создание файлов

Name = input()

file = open(Name, "w")

Text = input()

file.write(Text)

Задание 6.

Листинг 74: 15\_2\_6.py

#Выполнили: Щепкин М.В П2-18

# Задание 6. Переименование файлов

os.rename("text.txt", "text1.txt")

Задание 7.

Листинг 75: 15\_2\_8.py

#Выполнили: Щепкин М.В П2-18

# Задание 7. Перемещение файлов

shutil.move("откуда", "куда")

Задание 8.

Листинг 76: 15\_2\_9.py

#Выполнили: Щепкин М.В П2-18

# Задание 8. Список файлов и директорий

os.listdir('путь до папки')

Задание 9.

Листинг 77: 15\_2\_10.py

#Выполнили: Щепкин М.В П2-18

# Задание 9. Удаление файлов

os.remove('путь до файла')

Задание 10.

Листинг 78: 15\_2\_10.py

#Выполнили: Щепкин М.В П2-18

# Задание 10. Удаление директорий

os.rmdir('путь директорий до файла')

Задание 11.

Листинг 79: 15\_2\_11.py

#Выполнили: Щепкин М.В П2-18

# Задание 11. Получение информации о файлах

print(os.stat("1.txt"))

# K16. Техника работы с классами.

Задание 1.

Листинг 80: 16\_1\_1.py

#Выполнили: Щепкин М.В П2-18

# Задание 1. Создание класса

class Cat:

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

self.name = name

self.color = color

def myfunc(self):

print("Знакомьтесь, это " + self.name + ",", "он " + self.color)

p1 = Cat("Барсик", "рыжий")

p1.myfunc()

Задание 2.

Листинг 81: 16\_1\_2.py

#Выполнили: Щепкин М.В П2-18

# Задание 2. Создание объекта

class Cat:

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

self.name = name

self.color = color

def myfunc(self):

print("Знакомьтесь, это " + self.name + ",", "он " + self.color)

p1 = Cat("Барсик", "рыжий")

p1.myfunc()

#добавляем новый объект

p2 = Cat("Чёрныш", "чёрный")

#выполняем функцию для второго объекта

p2.myfunc()

Задание 3.

Листинг 82: 16\_1\_3.py

#Выполнили: Щепкин М.В П2-18

#Задание 3. Функция init

"""

У всех классов есть функция под названием \_\_init\_\_(),

которая всегда выполняется при создании объекта.

Используйте функцию \_\_init\_\_() для добавления значений

свойствам объекта или других операций,

которые необходимо выполнить, при создании объекта.

"""

class Pupil:

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

self.age = age

self.clas = clas

p1 = Pupil(13, 7)

print("Класс ученика - ", p1.clas)

Задание 4.

Листинг 83: 16\_1\_4.py

#Выполнили: Щепкин М.В П2-18

#Задание 4. Методы объектов

"""

Объекты также содержат методы.

Методы в объектах — это функции, принадлежащие объекту.

"""

class Cat:

def \_\_init\_\_(self, name, color, character):

self.name = name

self.color = color

self.character = character

#Метод

def myfunc(self):

print("Знакомьтесь, это " + self.name + ",", "он " + self.color + ",", "характер у него " + self.character)

p1 = Cat("Барсик", "рыжий", "ласковый")

p1.myfunc()

Задание 5.

Листинг 84: 16\_1\_5.py

#Выполнили: Щепкин М.В П2-18

#Задание 5. Параметр self

"""

Параметр self является ссылкой на сам класс и используется

для доступа к переменным принадлежащим классу.

Его не обязательно называть self, вы можете называть его как хотите,

но он должен быть первым параметром любой функции в классе

"""

#пример

class Table:

def \_\_init\_\_(self, legs, color, height):

self.legs = legs

self.color = color

self.height = height

p1 = Table(3, "black", 120)

print(p1.legs, p1.color, p1.height)

Задание 6.

Листинг 85: 16\_1\_6.py

#Выполнили: Щепкин М.В П2-18

#Задание 6. Изменение свойств объекта

class Cat:

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

self.name = name

self.color = color

def myfunc(self):

print("Знакомьтесь, это " + self.name + ",", "он " + self.color)

p1 = Cat("Барсик", "рыжий")

p1.myfunc()

#изменяем свойства объекта

p1.name = "Рыжик"

p1.myfunc()

Задание 7.

Листинг 86: 16\_1\_7.py

#Выполнили: М.В П2-18

#Задание 7. Удалить свойства объекта

#Выполнили: М.В П2-18

# Задание 1. Создание класса

class Cat:

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

self.name = name

self.color = color

def myfunc(self):

print("Знакомьтесь, это " + self.name + ",", "он " + self.color)

p1 = Cat("Барсик", "рыжий")

p1.myfunc()

#удаляем свойства объекта

del p1.color

Задание 8.

Листинг 87: 16\_1\_8.py

#Выполнили: Щепкин М.В П2-18

#Задание 8. Удаление объектов

class Cat:

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

self.name = name

self.color = color

def myfunc(self):

print("Знакомьтесь, это " + self.name + ",", "он " + self.color)

p1 = Cat("Барсик", "рыжий")

p1.myfunc()

#добавляем новый объект

p2 = Cat("Чёрныш", "чёрный")

#выполняем функцию для второго объекта

p2.myfunc()

#удаляем первый объект

del p10

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

# K17. Установка и настройка среды JetBrains PyCharm

PyCharm - это кросс-платформенная среда разработки.

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

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

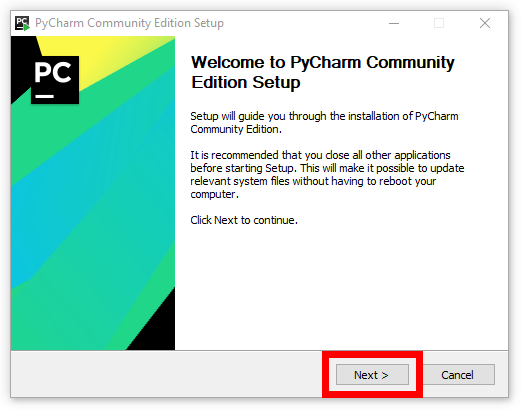


Рис. 20. (Установщик PyCharm)

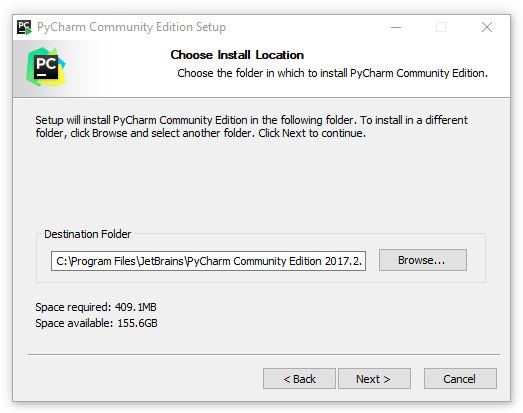


Рис. 21. (Место установки PyCharm)

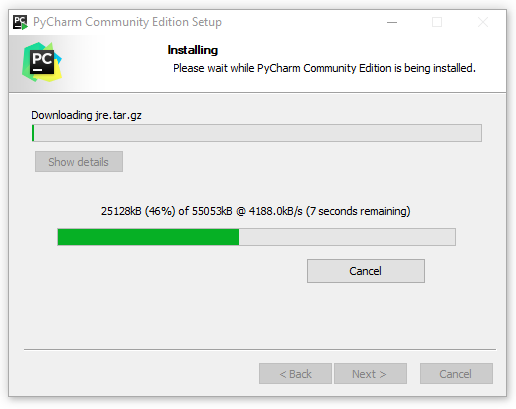


Рис. 22. (Процесс установки PyCharm)

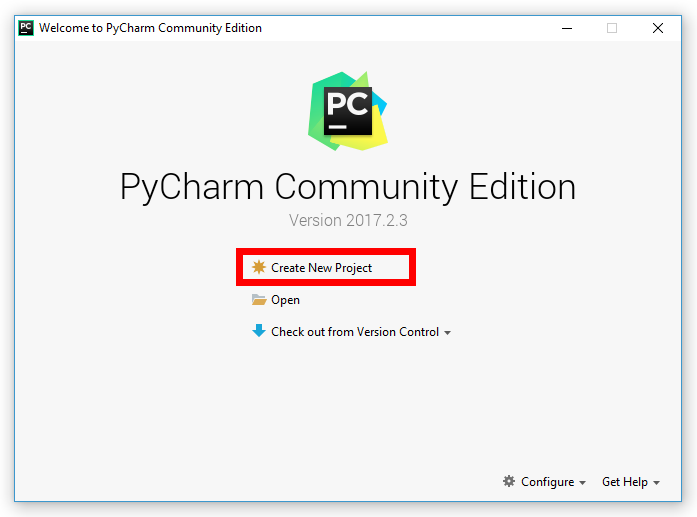


Рис. 23. (Стартовое окно PyCharm)

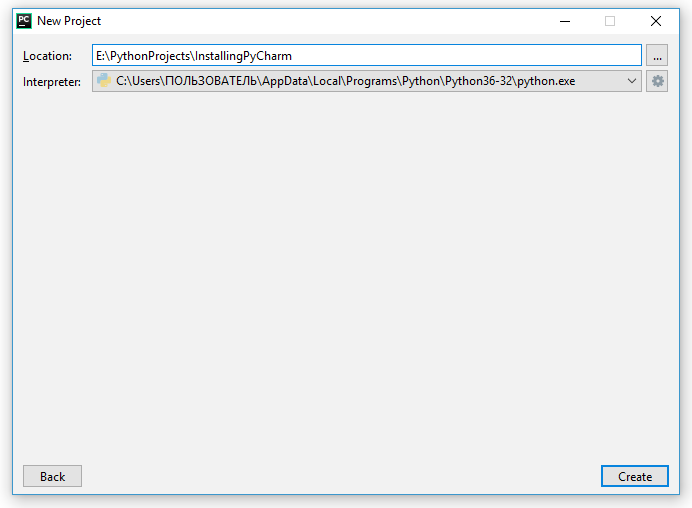
В следующем окне будет два поля. В первом укажите, где будет располагаться новый проект. Второе поле должно быть заполненным поумолчанию. Там содержится путь к установленному ранее Python 

Рис. 24. (Выбор места для нового проекта и выбор интерпретатора)

Дальше откроется окно самого редактора.

Теперь вы готовы начать писать программы на Python!

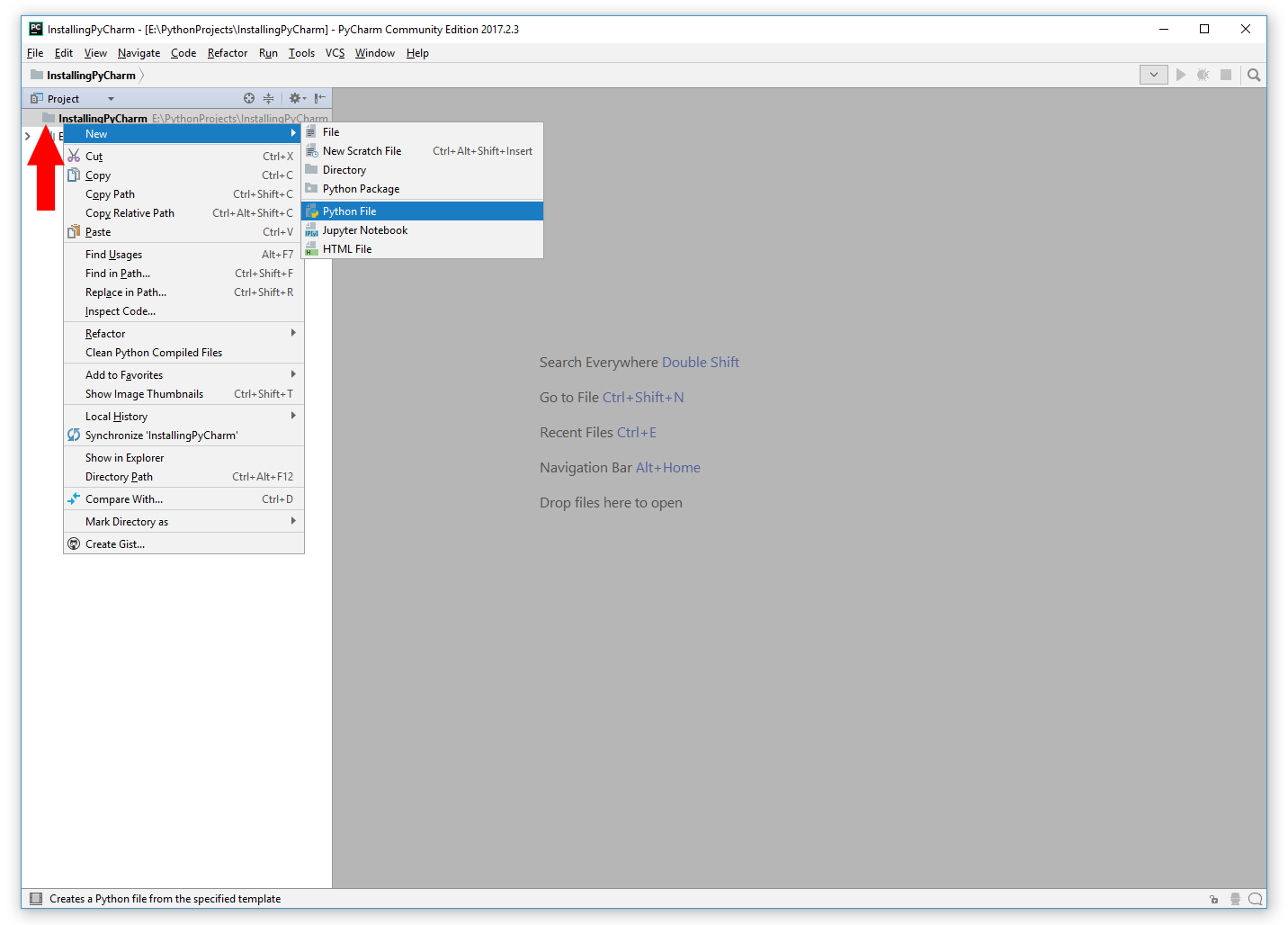
Кликните правой кнопкой на название вашего проекта и в раскрывающемся списке выберите пункт «New > Python file»: 

Рис. 25. (Создание нового Phython файла)

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

Справа откроется сам файл. Пока он пустой. Пишем:

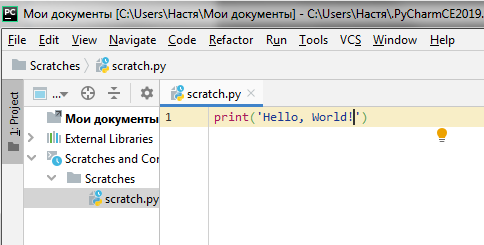


Рис. 26. (Написание программы)

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

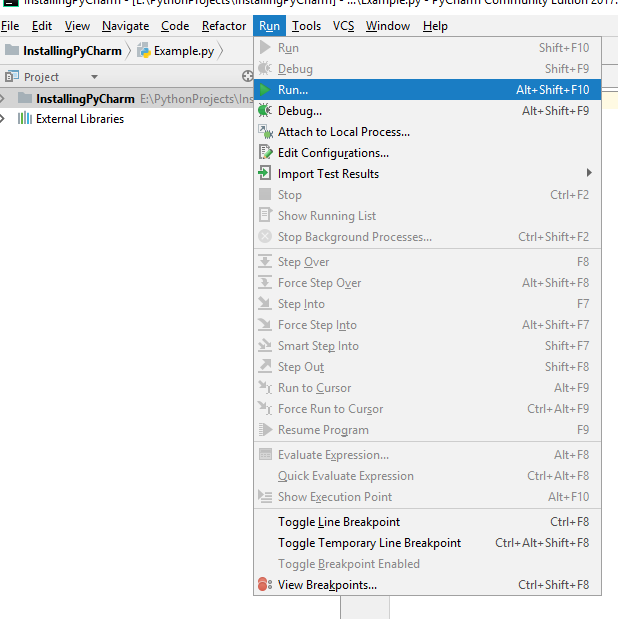


Рис. 27. (Запуск программы)

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

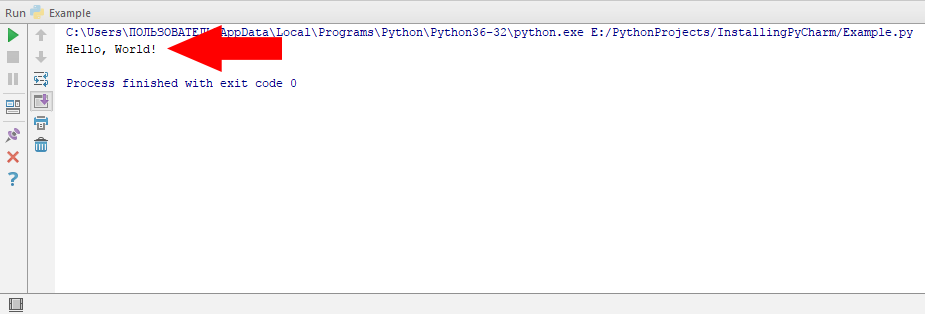


Рис. 28. (Выполнение программы)

# K18. Техника работы с базами данных

Приложения: table.py

Листинг 87: table.py

#Выполнил Щепкин Михаил П2-18

#Импортируем sqlite3

import sqlite3

#Создаём базу данных.

db = sqlite3.connect('TableBase.db')

sql = db.cursor()

#Создаём таблицу в базе данных.

sql.execute("""CREATE TABLE IF NOT EXISTS DataBase (Age INTEGER, Login REAL, Password REAL)""")

db.commit()

#Вводим: Возраст, Логин и Пароль.

DataBase\_Age = input('Age: ')

DataBase\_Login = input('Login: ')

DataBase\_Password = input('Password: ')

#Проверяем логин который был введен.

sql.execute(f"SELECT Login FROM DataBase WHERE Login = '{DataBase\_Login}'")

if sql.fetchone() is None:

sql.execute(f"INSERT INTO DataBase VALUES('{DataBase\_Age}','{DataBase\_Login}','{DataBase\_Password}')")

db.commit()

#Если логин не используется выводим.

print("Зарегестрировано!")

else:

#Если логин уже был использован выводим.

print("Такой логин уже существует!")

print("Список логинов которые уже были зарегистрированы: ")

for value in sql.execute("SELECT \* FROM DataBase"):

print(value[1])

db.close()

После завершения кода в той же папке где находится python-file появится файл с БД и в ней должна появиться записи: возраст, логин и пароль. 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/>.

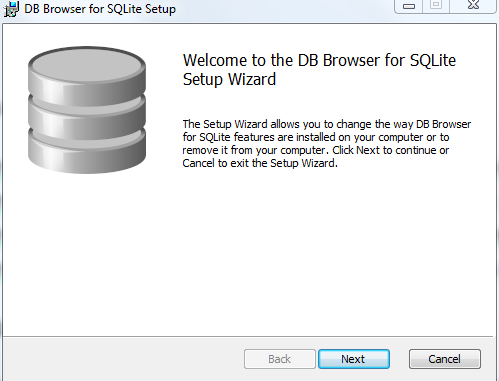


Рис. 29. (Установщик DB Browser)

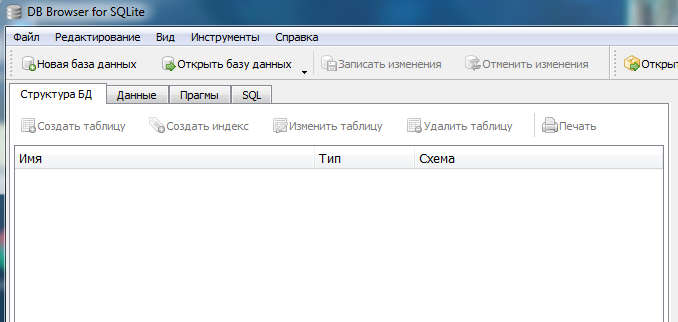
После установки, открываем DB Browser для работы с ним.

Рис. 30. (Окно DB Browser)

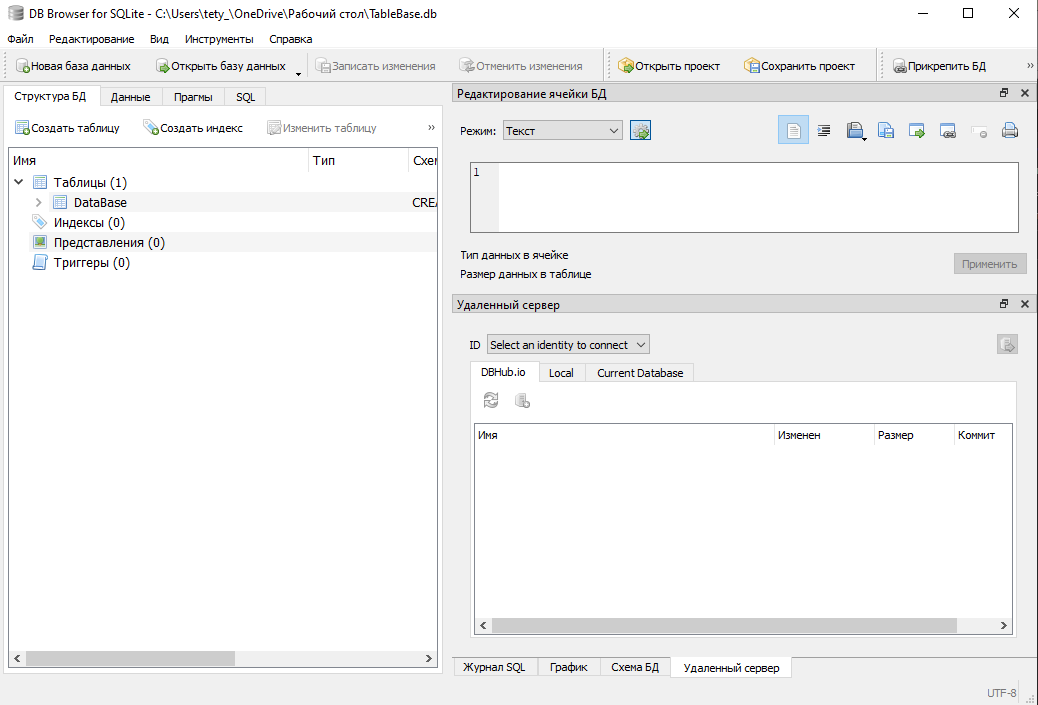
В открывшееся окно DB Browser’a перетаскиваем файл с БД.

Рис. 31. (Таблицы DataBase в DB Browser)

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

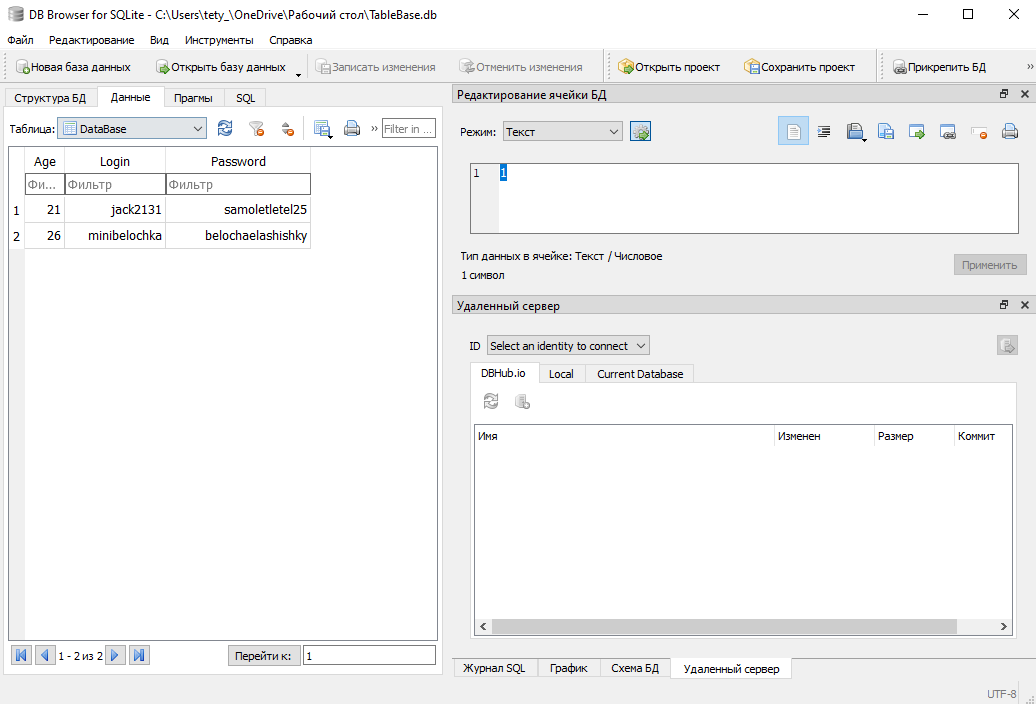


Рис. 32. (Данные таблицы DataBase)

Видно, что в таблицу занеслись корректные данные.

# K19. Техника работы с библиотекой tkinter

Приложения: K19.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 89. K19.py

#Выполнил Щепкин Михаил П2-18

#Импортирование библиотеки.

from tkinter import \*

from tkinter import messagebox

#Настройки окна

root = Tk()

root.geometry ('300x300')

root.title('window')

root ['bg'] = '#ccc'

#Создание параметра функции check, проверки ввода Логина и Пароля

def check( all ):

Lg = login.get()

Pg = password.get()

if Lg and Pg:

messagebox.showinfo('Success','Вы успешно вошли!')

if not Lg and Pg:

messagebox.showerror('Error 0x1', 'Введите логин!')

elif not Pg and Lg:

messagebox.showerror('Error 0x2', 'Введите пароль!')

if not Lg and not Pg:

messagebox.showerror('Error 0x0', 'Введите данные!')

#Создание текста логин

text\_login = Label(text = 'Login', font = ("Times New Roman", 14),

fg = '#000000',

bg = '#ccc')

#Создание логина

login = Entry(root, font = ("Times New Roman", 16), fg = '#eff5c9',

bg = '#48494f',

relief = 'solid',

justify = 'center')

#Создание текста пароль

text\_password = Label(text = 'Password', font = ("Times New Roman", 14),

fg = '#000000',

bg = '#ccc')

#Создание пароля

password = Entry(root, font = ("Times New Roman", 16),

fg = '#eff5c9',

bg = '#48494f',

relief = 'solid',

justify = 'center',

show = '\*')

#Создание флажка, Оставаться в системе!

check\_status = Checkbutton(text = 'Оставаться в системе!',

font = ("Times New Roman", 14),

bg = '#ccc',

fg = '#3d3d42',

activebackground = '#ccc',

activeforeground = '#3d3d42')

#Создание кнопки, Войти

enter = Button(text = 'Войти', font = ("Times New Roman", 16),

bg = '#48494f',

fg = '#000000',

relief = 'solid',

activeforeground = '#eff5c9',

activebackground = '#6e6f73',

width = '20')

#Packer

text\_login.pack()

login.pack()

text\_password.pack()

password.pack()

check\_status.pack()

enter.pack()

#Bind

enter.bind('<Button - 1>', check )

root.mainloop()

# K20. Техника работы с библиотекой NumPy

Приложения: K20.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 as np

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 90: K20.py

#Выполнил Щепкин П2-18

#Импортирование библиотек NumPy и Random

import numpy as np

import random

#print("Вы ввели не корректное действие!\nПовторите попытку!")

# Функция мейн позволяет использовать другие функции

def main():

menu()

while(True):

keyb = int(input("Выберите пункт в меню: "))

if (keyb == 1):

print("\nДоступные действия: (\*, /, +, -)")

L = input("Введите доступные действия: ")

fun1(L)

print("\n")

elif(keyb == 2):

fun2()

print("\n")

elif(keyb == 3):

print("\nДоступное кол-во эллементов в массиве: 4, 6 или 9.\nБудет создан 1 массив\n")

fun3()

print("\n")

elif(keyb == 0):

exit(0)

else:

print("Вы ввели не корректное действие!\nПовторите попытку!")

# Функция меню,выводим меню в консоль

def menu():

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

print("2. Объединение массивов")

print("3. Разбиение одномерного массива на многомерный")

print("0. Выход")

# Функция 1, выводит в консоль умножение, делени, вычитание, сложение матриц.

def fun1(s):

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

B = np.array([[7,5,1],[2,4,6],[3,9,1]])

print("Матрица A:\n", A, "\n","Матрица B:\n", B)

if(L == "+"):

print("Сложение матриц A и B: \n", A + B)

elif(L == "\*"):

print("Умножение матриц A и B: \n", A \* B)

elif(L == "/"):

print("Деление матриц A и B: \n", A // B)

elif(L == "-"):

print("Вычитание матриц A и B:\n", A - B)

#Функция 2, выводит в консоль объединение двух матриц в одну.

def fun2():

D = np.array([[3,1,5],[7,0,4],[6,2,9]])

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

F = D[:,::-1]

C = F[::-1]

print("\nМатрица A:\n",D)

print("\nМатрица B:\n",G)

print("\nОбъединение матриц A и B в одну:\n",np.stack((D,G,C)))

#Функция 3, выводит в консоль изменённую форму массива

def fun3():

n = int(input("Введите кол-во элементов в массиве: "))

A = random.sample(range(9),n)

print("Массив A:\n",A)

B = np.array([A])

i = input("Введите кол-во элементов которое вы выбрали: ")

if (i == "6"):

F = B.reshape(2,3)

print("\nМатрица массива А размером 2 на 3:\n",F)

elif (i == "4"):

G = B.reshape(2,2)

print("\nМатрица массива А размеров 2 на 2:\n",G)

elif(i == "9"):

D = B.reshape(3,3)

print("\nМатрица массива А размером 3 на 3:\n",D)

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

main()

# K21. Техника работы с библиотекой Matplotlib

Приложение: K21.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 90: K21.py

#Выполнил Щепкин П2-18

#Импортирование библиотеки Matplotlib

import matplotlib.pyplot as plt

# Координаты точек по x

x = [1, 12, 18, 24, 30]

# Координаты точек у первого графика по оси y

y1 = [0, 6, 3, 12, 1]

# Координаты точек у второго графика по оси y

y2 = [0, 2, 9, 7, 18]

#Размер окна

plt.figure(figsize = (12, 5))

# Построение первого графика

plt.plot(x, y1, 'o-r', alpha = 0.7, label="1. График", lw = 1, mec ='b',mew = 2, ms = 5)

# Построение второго графика

plt.plot(x, y2, 'v-.g', label = "2. График", mec ='r', lw = 2, mew = 2, ms = 9)

plt.legend()

#Отображение сетки

plt.grid(True)

#Отображение графиков

plt.show()

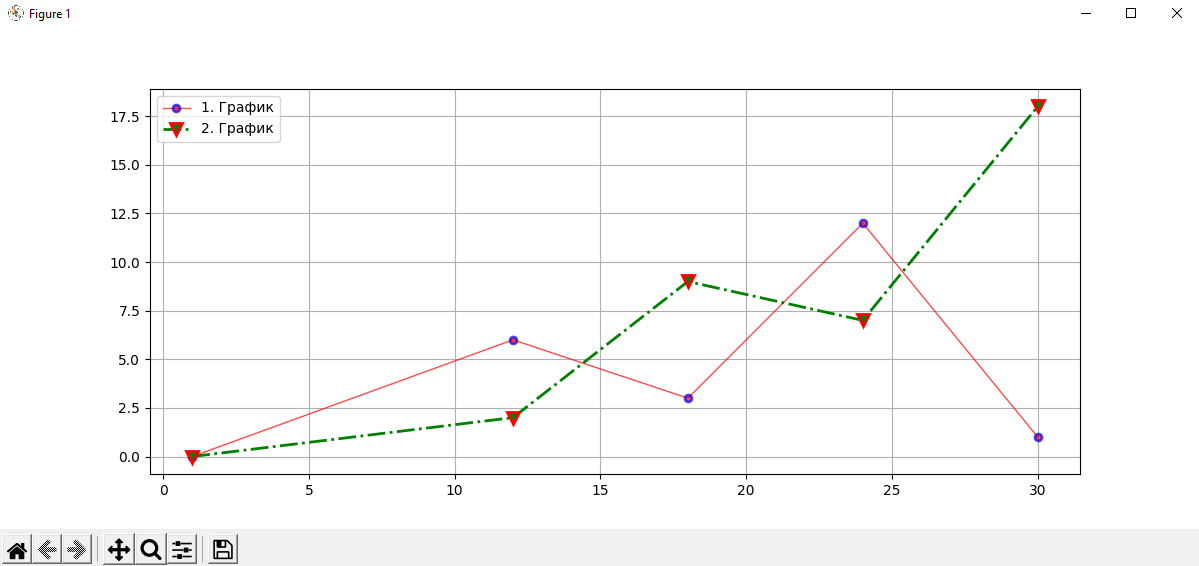


Рис. 33. (Вывод программы prak21.py)

# K22. Элементы работы с библиотекой PyQt

Приложения: PyQt.py, gui.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 91: PyQt.py

#Выполнил Щепкин П2-18

#Импорт библиотек

from PyQt5.QtGui import \*

from PyQt5.QtCore import \*

from PyQt5.QtWidgets import \*

import random

import string

import sys

import gui

class GenKey(QMainWindow, gui.WinGui):

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

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

self.Win2(self)

self.Pb.clicked.connect(self.button)

#Создание функциий, которые генерируют до 10 символов с использованием цифр и #букв верхнего регистра.

def Random(self, RmSize = 6, chars = string.ascii\_uppercase + string.digits):

return ''.join(random.choice(chars) for x in range(RmSize))

def button(self):

self.Text.append(self.Random(10))

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

AppMain = QApplication(sys.argv)

window = GenKey()

window.show()

AppMain.exec()

Листинг 92: gui.py

#Выполнил Щепкин П2-18

#Импортирование библиотек

from PyQt5 import QtCore, QtGui, QtWidgets

class WinGui(object):

def Win2(self, MainWindow):

MainWindow.setObjectName("MainWindow")

#Размер окна

MainWindow.resize(470, 290)

self.Widget = QtWidgets.QWidget(MainWindow)

self.Widget.setObjectName("Widget")

self.Pb = QtWidgets.QPushButton(self.Widget)

#Размер кнопки

self.Pb.setGeometry(QtCore.QRect(30, 90, 150, 50))

self.Pb.setObjectName("PushButton")

self.Text = QtWidgets.QTextEdit(self.Widget)

#Размер окна вывода

self.Text.setGeometry(QtCore.QRect(200, 40, 220, 180))

self.Text.setObjectName("TextEdit")

MainWindow.setCentralWidget(self.Widget)

self.WinText(MainWindow)

def WinText(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

#Название окна

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

#Название кнопки

self.Pb.setText(\_translate("MainWindow", "Сгенерировать ключ"))

# K23. Элементы работы с библиотекой PyGame

Приложения: arkanoid.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:

Листинг 93: arkanoid.py

# Выполнил Щепкин П2-18

# подключаем графическую библиотеку

from tkinter import \*

# подключаем модули, которые отвечают за время и случайные числа

import time

import random

tk = Tk()

tk.title('Game')

# запрещаем менять размеры окна.

tk.resizable(0, 0)

tk.wm\_attributes('-topmost', 1)

canvas = Canvas(tk, width=500, height=400, highlightthickness=0)

canvas.pack()

tk.update()

# Ball, который будет отвечать за шарик

class Ball:

# конструктор — он вызывается в момент создания нового объекта на основе этого класса

def \_\_init\_\_(self, canvas, paddle, score, color):

self.canvas = canvas

self.paddle = paddle

self.score = score

self.id = canvas.create\_oval(10,10, 25, 25, fill=color)

self.canvas.move(self.id, 245, 100)

# задаём список возможных направлений для старта

starts = [-2, -1, 1, 2]

random.shuffle(starts)

self.x = starts[0]

self.y = -2

self.canvas\_height = self.canvas.winfo\_height()

self.canvas\_width = self.canvas.winfo\_width()

self.hit\_bottom = False

# функция касание платформы

def hit\_paddle(self, pos):

paddle\_pos = self.canvas.coords(self.paddle.id)

if pos[2] >= paddle\_pos[0] and pos[0] <= paddle\_pos[2]:

if pos[3] >= paddle\_pos[1] and pos[3] <= paddle\_pos[3]:

# увеличиваем счёт (обработчик этого события будет описан ниже)

self.score.hit()

return True

return False

# функция, который отвечает за движение шарика

def draw(self):

# передвигаем шарик на заданный вектор x и y

self.canvas.move(self.id, self.x, self.y)

pos = self.canvas.coords(self.id)

# если шарик падает сверху

if pos[1] <= 0:

self.y = 2

if pos[3] >= self.canvas\_height:

self.hit\_bottom = True

# выводим сообщение и количество очков

canvas.create\_text(250, 120, text='Вы проиграли', font=('Courier', 30), fill='red')

# если было касание платформы

if self.hit\_paddle(pos) == True:

# шарик отскакивает наверх

self.y = -2

# левая стенка

if pos[0] <= 0:

# Право

self.x = 2

# Правая стенка

if pos[2] >= self.canvas\_width:

# Лево

self.x = -2

# Описываем класс Paddle, который отвечает за платформы

class Paddle:

# конструктор

def \_\_init\_\_(self, canvas, color):

self.canvas = canvas

# создаём прямоугольную платформу 10 на 100.

self.id = canvas.create\_rectangle(0, 0, 100, 10, fill=color)

# задаём список возможных стартовых положений платформы и перемешиваем их

start\_1 = [40, 60, 90, 120, 150, 180, 200]

random.shuffle(start\_1)

# выбираем первое из перемешанных

self.starting\_point\_x = start\_1[0]

self.canvas.move(self.id, self.starting\_point\_x, 300)

self.x = 0

self.canvas\_width = self.canvas.winfo\_width()

# Стрелка вправо метод turn\_right()

self.canvas.bind\_all('<KeyPress-Right>', self.turn\_right)

# Стрелка влево — turn\_left()

self.canvas.bind\_all('<KeyPress-Left>', self.turn\_left)

self.started = False

# Enter — всё стартует

self.canvas.bind\_all('<KeyPress-Return>', self.start\_game)

# движемся вправо

def turn\_right(self, event):

self.x = 2

# движемся влево

def turn\_left(self, event):

self.x = -2

# игра начинается

def start\_game(self, event):

self.started = True

# функция, которая отвечает за движение платформы

def draw(self):

self.canvas.move(self.id, self.x, 0)

pos = self.canvas.coords(self.id)

# если мы упёрлись в левую границу

if pos[0] <= 0:

# останавливаемся

self.x = 0

# если упёрлись в правую границу

elif pos[2] >= self.canvas\_width:

# останавливаемся

self.x = 0

# класс Score, который отвечает за отображение счета

class Score:

def \_\_init\_\_(self, canvas, color):

# в самом начале счёт равен нулю

self.score = 0

self.canvas = canvas

# создаём надпись, которая показывает текущий счёт

self.id = canvas.create\_text(450, 10, text=self.score, font=('Courier', 15), fill=color)

# функция касания платформы

def hit(self):

# увеличиваем счёт на единицу

self.score += 1

self.canvas.itemconfig(self.id, text=self.score)

score = Score(canvas, 'green')

paddle = Paddle(canvas, 'White')

ball = Ball(canvas, paddle, score, 'red')

while not ball.hit\_bottom:

# Игра началась

if paddle.started == True:

# двигаем шарик

ball.draw()

# двигаем платформу

paddle.draw()

tk.update\_idletasks()

tk.update()

time.sleep(0.01)

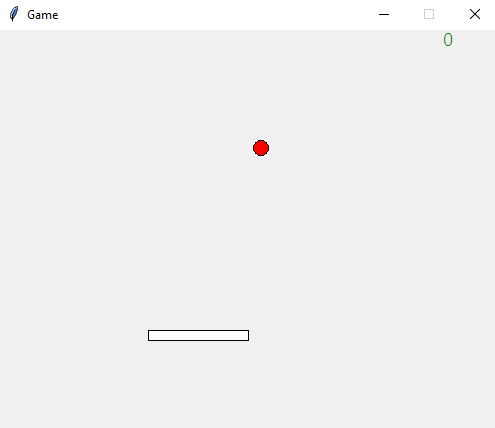


Рис 34. (Скриншот игры)

# **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 K24.Изучение входной и выходной документации

Входные данные:

1. Документ на оплату

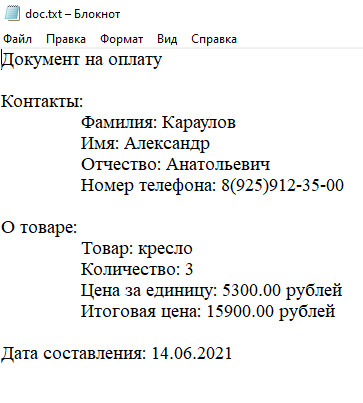


Рис. 35. (Документ на оплату)

1. Сумма денег клиента на оплату и вид оплаты (нал/безнал)

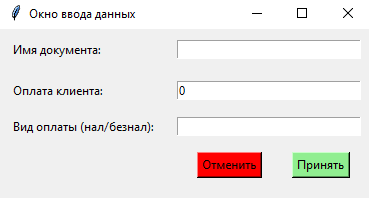
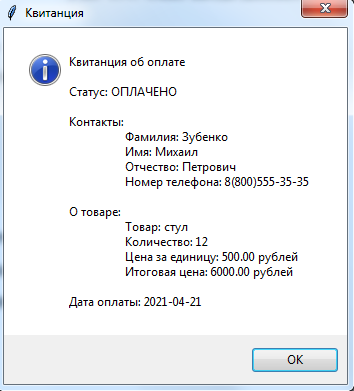


Рис. 36. (Окно ввода данных)

Выходные данные

1. Квитанцию об оплате

  
Рис. 37. (Квитанция об оплате)

1. Отчёт кассира

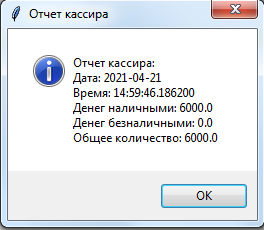


Рис. 38. (Отчёт кассира)

1. База данных

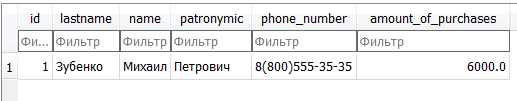


Рис. 39. (Таблица “clients”)

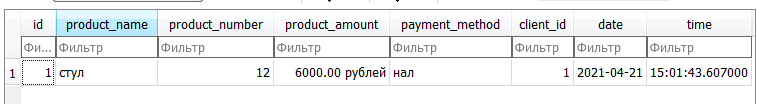


Рис. 40. (Таблица “registry”)

3.2 K25.Разработка требований к проекту. Построение диаграммы использования.

Требования к проекту:

1. Быстрое оформление документов
2. Правильность оформления документов
3. Печать документов

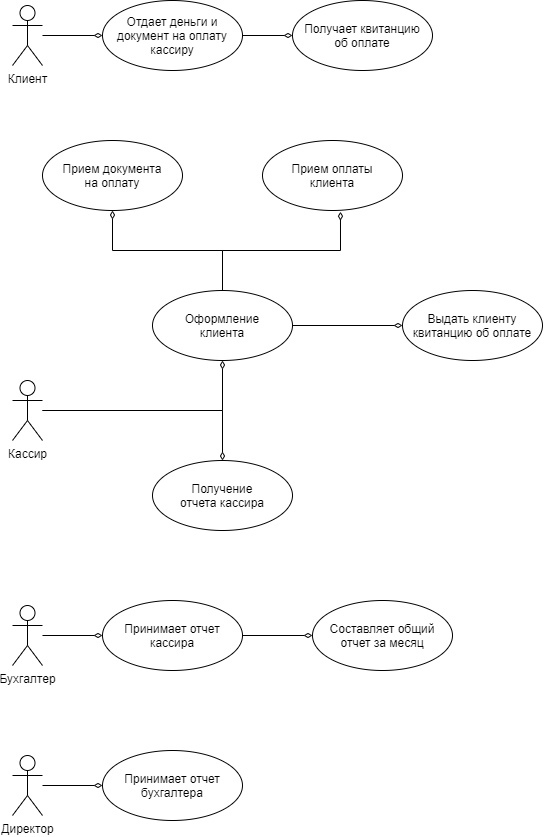


Рис. 41. (Диаграмма использования)

* 1. K26.Разработка сценария проекта

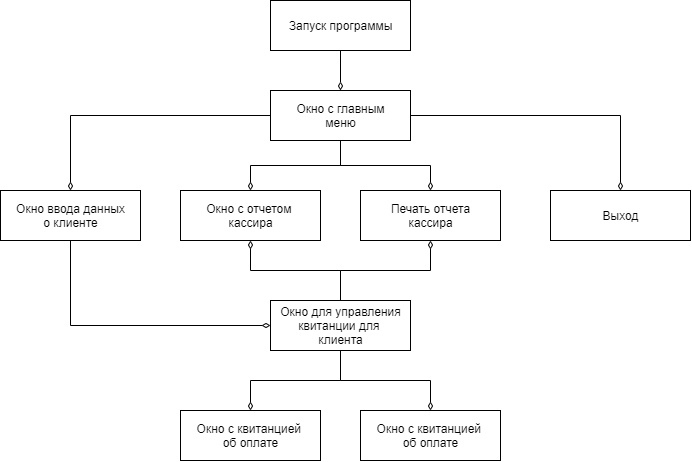
В данном разделе приведен сценарий использования программы пользователем (Рисунок 42).

Рис. 42. (Сценарий проекта)

* 1. K27.Построение диаграммы классов

В данном разделе находятся диаграммы классов приложения.(Рисунок 43)

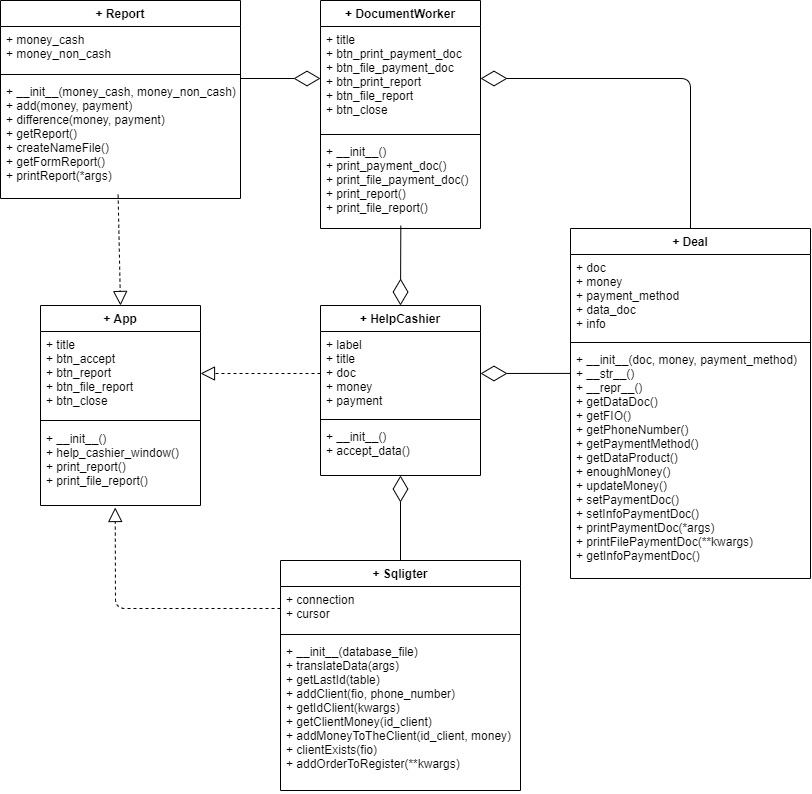


Рис. 43. (Диаграмма классов)

* 1. K28.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.

Для работы с базой данных мы решили написать отдельный класс SQLigter.

Листинг 94: [sqlighter.py](https://github.com/prog-kkkmt/p1-18/blob/Gymrasimov/%D0%A3%D0%9F/project_w_bd/sqlighter.py)

import sqlite3

from sqlite3.dbapi2 import connect

class SQLigter:

def \_\_init\_\_(self, database\_file):

"""Подключаемся к БД и сохраняем курсор соединения"""

self.connection = sqlite3.connect(database\_file)

self.cursor = self.connection.cursor()

# ! Обобщенные функции (для всех таблиц)

def translateData(self, args):

"""Переводит данные из словаря или из списка в список значений"""

if (len(args) == 1) and (type(args[0]) == dict):

return args[0].values()

elif type(args) == dict:

return args.values()

return args

def getLastId(self, table):

"""Вернет последний индекс"""

with self.connection:

try:

#return self.cursor.execute(f"SELECT `id` FROM `{table}` ORDER BY `id` DESC LIMIT 1").fetchall()[0][0]

return self.cursor.execute("SELECT `id` FROM `" + table + "` ORDER BY `id` DESC LIMIT 1").fetchall()[0][0]

except Exception:

return 0

# ! Работа с таблицей данных клиента `clients`

def addClient(self, fio, phone\_number):

"""Добавляем нового пользователя в базу"""

with self.connection:

#lastname, name, patronymic = self.translateData(fio)

need\_id = self.getLastId('clients')+1

print(fio)

result = self.cursor.execute("INSERT INTO `clients` (`id`, `lastname`, `name`, `patronymic`, `phone\_number`, `amount\_of\_purchases`) VALUES (?,?,?,?,?,?)", (need\_id, fio['lastname'], fio['name'], fio['patronymic'], phone\_number, 0.0)).fetchall()

self.connection.commit()

return result

def getIdClient(self, kwargs):

"""Вернет id клиента"""

#lastname, name, patronymic = self.translateData(args)

with self.connection:

return self.cursor.execute(\

"SELECT \* FROM `clients` WHERE `lastname` = ? AND `name` = ? AND `patronymic` = ?", \

(kwargs['lastname'], kwargs['name'], kwargs['patronymic'])\

).fetchall()[0][0]

def getClientMoney(self, id\_client):

"""Вернет сумму денег клиента из таблицы `clients`"""

with self.connection:

return self.connection.execute("SELECT \* FROM `clients` WHERE `id` = ?", (id\_client,)).fetchall()[0][-1]

def addMoneyToTheClient(self, id\_client, money):

"""Добавляет в таблицу `clients` в позицую `money` деньги"""

with self.connection:

amount\_summ = self.getClientMoney(id\_client) + money

self.connection.execute("UPDATE `clients` SET `amount\_of\_purchases` = ?", (amount\_summ,))

return amount\_summ

def clientExists(self, fio):

"""Существует ли клиент"""

with self.connection:

try:

result = self.cursor.execute(\

"SELECT \* FROM `clients` WHERE `lastname` = ? AND `name` = ? AND `patronymic` = ?", \

(fio['lastname'], fio['name'], fio['patronymic'])\

).fetchall()

except:

result = ''

return bool(len(result))

# ! Работа с таблицей реестра заказов `registry`

def addOrderToRegister(self, \*\*kwargs):

"""Добавляем в реестр заказов новый заказ"""

with self.connection:

# product\_name, product\_number, product\_amount, payment\_method, client\_id, date, time

need\_id = self.getLastId('registry') + 1

# [data\_for\_add.append(x) for x in self.translateData(args)]

kwargs['id'] = need\_id

result = self.cursor.execute(\

"INSERT INTO `registry` (\

`id`, `product\_name`, `product\_number`, `product\_amount`, `payment\_method`, `client\_id`, `date`, `time`\

) VALUES (?,?,?,?,?,?,?,?)", \

(kwargs['id'], kwargs['product\_name'], kwargs['product\_number'], kwargs['product\_amount'], kwargs['payment\_method'], \

kwargs['client\_id'], kwargs['date'], kwargs['time'])\

).fetchall()

self.connection.commit()

return result

* 1. K29.Разработка главного модуля

Главный модуль состоит из класса App. Он выполняется сразу при запуске программы. Его также можно назвать главным меню.

Листинг 95. Главный модуль App

from tkinter import \*

from tkinter import messagebox

import tkinter as tk

from datetime import datetime

from Cashier import Report

cashier = Report(0, 0)

class App(tk.Tk):

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

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

self.title("Мини-кассир")

# self.geometry('450x500')

# Верхний отступ

tk.Label(self, text='').pack(pady=5)

# Текст окна

tk.Label(self, text='Ваш мини-кассир', font=(80), pady=20).pack(pady=7)

# Кнопки в окне

self.btn\_accept = tk.Button(self, text='Принять документ и деньги', command=self.help\_cashier\_window)

self.btn\_report = tk.Button(self, text='Отчет кассира', command=self.print\_report)

self.btn\_file\_report = tk.Button(self, text='Печатать отчета кассира', command=self.print\_file\_report)

self.btn\_close = tk.Button(self, text='Выход', command=self.destroy)

# Позиционирование кнопок

self.btn\_accept.pack(padx=30, ipadx=11)

self.btn\_report.pack(ipadx=54)

self.btn\_file\_report.pack(ipadx=20)

self.btn\_close.pack(ipadx=80)

# Нижний отступ

tk.Label(self, text='').pack(pady=50)

def help\_cashier\_window(self):

help\_cashier = HelpCashier(self)

help\_cashier.grab\_set()

def print\_report(self):

messagebox.showinfo("Отчет кассира", cashier.getFormReport())

cashier.printReport()

print()

def print\_file\_report(self):

name\_file = cashier.createNameFile()

messagebox.showinfo("Имя квитанции", name\_file)

print("Отчет в файле: ", name\_file)

print()

* 1. K30.Разработка входящих модулей

Описание спецификаций к модулям

Всего разработано 2 модуля:

* Окно ввода данных (HelpCashier)
* Меню для управления квитанцией клиента (DocumentWorker)

Взаимодействие между модулями можно наглядно увидеть по диаграммам классов из раздела 2.4.

Описание модулей:

**Окно ввода данных** – на данном окне находится форма для формирования квитанции клиенту. Для этого нужно заполнить следующие поля:

* Наименование документа
* Денежная сумма, которую дает клиент
* Метод оплаты клиента

Рассмотрим функцию обработки квитанции на оплату и её блок-схему:

Листинг 96. Функция обработки квитанции на оплату

def accept\_data(self):

data = {

"doc": self.doc.get(),

"money": self.money.get(),

"payment": self.payment.get()

}

try:

global deal

deal = Deal(data['doc'], data['money'], data['payment'])

# Если не хватило денег

if deal.enoughMoney() == False:

print()

print("Недостаточно")

messagebox.showinfo("Error", "Недостаточно средств")

# Денег хватает

else:

print()

print("Докумет принят")

self.destroy()

deal.updateMoney() # Обновляем сумму денег на стоимость товара

data['money'] = deal.money

cashier.add(data['money'], data['payment']) # Добавляем деньги в отчет

d\_fio = deal.getFIO() # Получаем ФИО клиента

# Если его нет в бд с клиентами, добавляем

if db.clientExists(d\_fio) == False:

db.addClient(d\_fio, deal.getPhoneNumber())

s = ''

for key in d\_fio:

s += d\_fio[key] + ' '

print("Добавлен новый клиент: ", s)

client\_id = db.getIdClient(d\_fio)

db.addMoneyToTheClient(client\_id, data['money']) # Добавляем деньги в общий список покупок клиента

# Добавляем в реестр заказов новый заказ

d = deal.getDataProduct()

date, time = str(datetime.today()).split()

db.addOrderToRegister(\

product\_name=d['product'], \

product\_number=d['number'], \

product\_amount=d['total\_price'], \

payment\_method=deal.getPaymentMethod(), \

client\_id=client\_id, \

date=date, \

time=time

)

data = [d['product'], d['number'], d['total\_price'], deal.getPaymentMethod(), client\_id, date, time]

s = str(db.getLastId('registry')) + ' | '

for x in data:

s += str(x) + ' | '

print("Добавлен в реестр: ", s)

print()

deal.setPaymentDoc()

doc\_work = DocumentWorker()

except:

print("Error")

messagebox.showinfo("Error", "Ошибка")

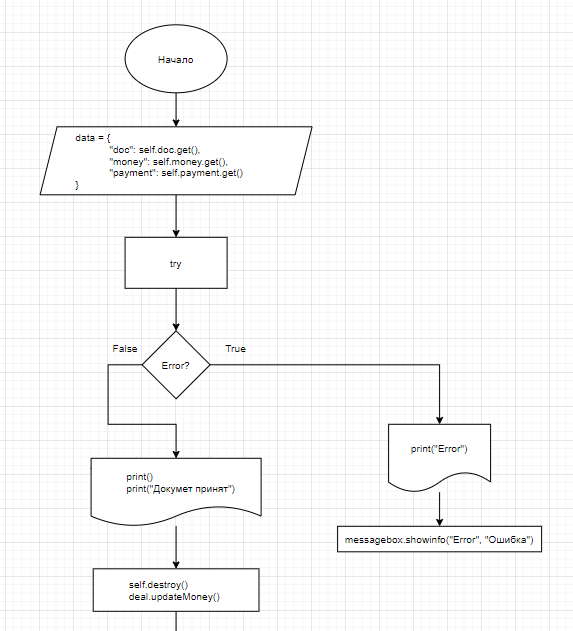


Рис. 44. (Блок-схема к функции accept\_data. Часть 1)



Рис. 45. (Блок-схема к функции accept\_data. Часть 2)

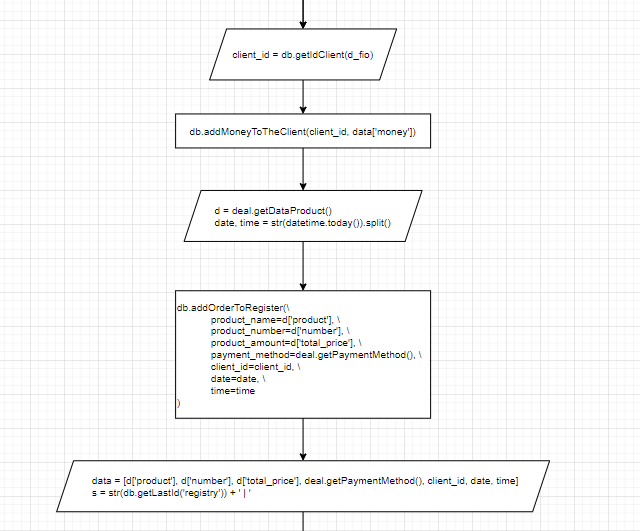


Рис. 46. (Блок-схема к функции accept\_data. Часть 3.)

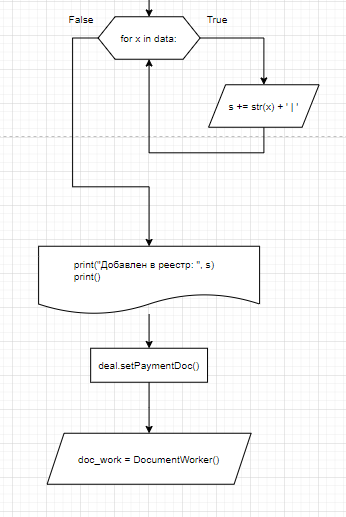


Рис. 47. (Блок-схема к функции accept\_data. Часть 4)

**Меню для управления квитанцией клиента** – на этом окне располагается меню для работы с квитанцией клиента об оплате. В этом меню есть следующие кнопки:

* Квитанция об оплате
* Печатать квитанции об оплате
* Отчет кассира
* Печатать отчета кассира
* Закрыть окно
  1. K31.Тестирование и отладка

В ходе написания проекта при попытке запустить программу были получены ошибки (Рисунок 48):

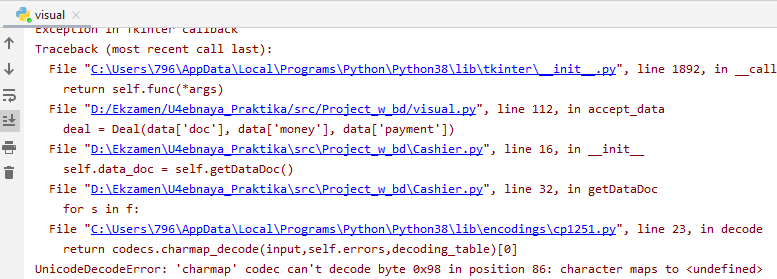


Рис. 48. (Ошибки)

При проверке кода были исправлены найденные ошибки, в результате при запуске программы ошибок не было (Рисунок 49):



Рис. 49. (Успешная сборка)

* 1. K32.Дневник

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

# **Приложения**

К4.Техника работы с линейными и разветвляющимися программами

**Файлы:**

**K4\K4\_1\4\_1\_1.py**

**K4\K4\_1\4\_1\_2.py**

**K4\K4\_2\4\_2\_1.py**

**K4\K4\_2\4\_2\_2.py**

**K4\K4\_3\4\_3\_1.py**

**K4\K4\_3\4\_3\_2.py**

**K4\K4\_3\4\_3\_3.py**

**K4\K4\_1\4\_4\_1.py**

**K4\K4\_1\4\_4\_2.py**

**K4\K4\_1\4\_4\_3.py**

К5.Техника работы с циклическими программами, цикл while

**Файлы:**

**K5\K5\_1\5\_1\_1.py**

**K5\K5\_1\5\_1\_2.py**

**K5\K5\_1\5\_1\_3.py**

**K5\K5\_1\5\_1\_4.py**

**K5\K5\_1\5\_1\_5.py**

**K5\K5\_2\5\_2\_1.py**

**K5\K5\_2\5\_2\_2.py**

**K5\K5\_2\5\_2\_3.py**

**K5\K5\_2\5\_2\_4.py**

**K5\K5\_2\5\_2\_5.py**

К6.Техника работы с числами

**Файлы:**

**K6\K6\_1\6\_1\_1.py**

**K6\K6\_1\6\_1\_2.py**

**K6\K6\_1\6\_1\_3.py**

**K6\K6\_1\6\_1\_4.py**

**K6\K6\_1\6\_1\_5.py**

**K6\K6\_1\6\_1\_6.py**

**K6\K6\_2\6\_2\_1.py**

**K6\K6\_2\6\_2\_2.py**

**K6\K6\_2\6\_2\_3.py**

**K6\K6\_2\6\_2\_4.py**

К7.Техника работы со строками

**Файлы:**

**K7\K7\_1\7\_1\_1.py**

**K7\K7\_1\7\_1\_2.py**

**K7\K7\_1\7\_1\_3.py**

**K7\K7\_1\7\_1\_4.py**

**K7\K7\_2\7\_2\_1.py**

**K7\K7\_2\7\_2\_2.py**

**K7\K7\_2\7\_2\_3.py**

**K7\K7\_2\7\_2\_4.py**

К8.Техника работы со списками

**Файлы:**

**K8\8\_1\_1.py**

**K8\8\_1\_2.py**

**K8\8\_1\_3.py**

**K8\8\_1\_4.py**

**K8\8\_1\_5.py**

**K8\8\_1\_6.py**

К9.Техника работы с циклом for и генераторами списков

**Файлы:**

**K9\9\_2\_1.py**

**K9\9\_2\_2.py**

**K9\9\_2\_3.py**

**K9\9\_2\_4.py**

**K9\9\_2\_5.py**

**K9\9\_2\_6.py**

К10.Техника работы с функциями

**Файлы:**

**K10\K10\_1\10\_1\_1.py**

**K10\K10\_1\10\_1\_2.py**

**K10\K10\_1\10\_1\_3.py**

**K10\K10\_2\10\_2\_1.py**

**K10\K10\_2\10\_2\_2.py**

**K10\K10\_2\10\_2\_3.py**

К11.Техника работы со словарями

**Файлы:**

**K11\11\_1\_1.py**

**K11\11\_1\_2.py**

**K11\11\_1\_3.py**

К12.Техника работы с множествами

**Файлы:**

**K12\12\_1\_1.py**

**K12\12\_1\_2.py**

**K12\12\_1\_3.py**

**K12\12\_1\_4.py**

**K12\12\_1\_5.py**

К13.Техника работы с кортежами

**Файлы:**

**K13\13\_1\_1.py**

**K13\13\_1\_2.py**

**K13\13\_1\_3.py**

К14.Техника работы с файлами

**Файлы:**

**K14\14\_1\_1.py**

**K14\14\_1\_2.py**

**K14\14\_1\_3.py**

**K14\14\_1\_4.py**

К15.Техника работы с модулями

**Файлы:**

**K15\K15\_1\15\_1\_1.py**

**K15\K15\_1\15\_1\_2.py**

**K15\K15\_1\15\_1\_3.py**

**K15\K15\_2\15\_2\_1.py**

**K15\K15\_2\15\_2\_2.py**

**K15\K15\_2\15\_2\_3.py**

**K15\K15\_2\15\_2\_4.py**

**K15\K15\_2\15\_2\_5.py**

**K15\K15\_2\15\_2\_6.py**

**K15\K15\_2\15\_2\_7.py**

**K15\K15\_2\15\_2\_8.py**

**K15\K15\_2\15\_2\_9.py**

**K15\K15\_2\15\_2\_10.py**

**K15\K15\_2\15\_2\_11.py**

К16.Техника работы с классами

**Файлы:**

**K16\16\_1\_1.py**

**K16\16\_1\_2.py**

**K16\16\_1\_3.py**

**K16\16\_1\_4.py**

**K16\16\_1\_5.py**

**K16\16\_1\_6.py**

**K16\16\_1\_7.py**

**K16\16\_1\_8.py**

K18**.** Техника работы с базами данных

**Файлы:**

**K18\table.py**

К19.Техника работы с библиотекой tkinter

**Файлы:**

**Tkinter\K19.py**

К20.Техника работы с библиотекой NumPy

**Файлы:**

**NumPy\K20.py**

К21.Техника работы с библиотекой Matplotlib

**Файлы:**

**Matpotlib\K21.py**

К22.Элементы работы с библиотекой PyQt

**Файлы:**

**PyQt5\PyQt.py**

К23.Элементы работы с библиотекой PyGame

**Файлы:**

**Pygame/arkanoid.py**

К29.Разработка базы данных

**Файлы:**

**Проект\sqlighter.py**

К30.Разработка главного модуля

**Файлы:**

**Проект\visual.py**