**

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 П2-18:   
Кузнецов Матвей Сергеевич

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

Проверил преподаватель: Гусятинер Леонид Борисович

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

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

Королев, 2020

Оглавление

[**Раздел 1.** 3](#_Toc72488429)

[Практическое занятие №2. 3](#_Toc72488430)

[Практическое занятие №3. 4](#_Toc72488431)

[Создание файла с кодом. 4](#_Toc72488432)

[Запуск. 4](#_Toc72488433)

[Import. 4](#_Toc72488434)

[Reload. 4](#_Toc72488435)

[Отработка ошибок. 4](#_Toc72488436)

[Практическое занятие №4. 6](#_Toc72488437)

[Задачи на Begin. 6](#_Toc72488438)

[Задачи на Boolean. 6](#_Toc72488439)

[Задачи на Case. 6](#_Toc72488440)

[Задачи на If. 6](#_Toc72488441)

[Блок-схемы для задач к практическому занятию №4 8](#_Toc72488442)

[Блок-схема для задачи boolean1: 8](#_Toc72488443)

[Блок-схема для задачи boolean2: 9](#_Toc72488444)

[Практическое занятие №5. 10](#_Toc72488445)

[Задачи на For. 10](#_Toc72488446)

[Задачи на While. 10](#_Toc72488447)

[Блок-схемы для задач к практическому занятию №5 11](#_Toc72488448)

[Блок-схема для задачи for1: 11](#_Toc72488449)

[Блок-схема для задачи while5: 12](#_Toc72488450)

[Практическое занятие №6. 13](#_Toc72488451)

[Практическое занятие №7 16](#_Toc72488452)

[Практическое занятие №8. 17](#_Toc72488453)

[Практическое занятие №9. 18](#_Toc72488454)

[Практическое занятие №10. 19](#_Toc72488455)

[Практическое занятие №11. 20](#_Toc72488456)

[Практическое занятие №12. 21](#_Toc72488457)

[Практическое занятие №13. 22](#_Toc72488458)

[Практическое занятие №14. 23](#_Toc72488459)

[Практическое занятие №15. 24](#_Toc72488460)

[Практическое занятие №16. 26](#_Toc72488461)

[**Раздел 2.** 28](#_Toc72488462)

[Практическое занятие №17. 28](#_Toc72488463)

[Практическое занятие №18. 34](#_Toc72488464)

[Практическое занятие №19. 39](#_Toc72488465)

[Практическое занятие №20. 42](#_Toc72488466)

[Практическое занятие №21. 45](#_Toc72488467)

[Практическое занятие №22. 47](#_Toc72488468)

[Практическое занятие №23. 49](#_Toc72488469)

[**Раздел 3.** 59](#_Toc72488470)

[**Разработка проекта с графическим интерфейсом** 59](#_Toc72488471)

[3.1 Изучение входной и выходной документации 59](#_Toc72488472)

[3.2 Разработка требований к проекту. Построение диаграммы использования. 61](#_Toc72488473)

[3.3 Разработка сценария проекта 63](#_Toc72488474)

[3.4 Построение диаграммы классов 63](#_Toc72488475)

[3.5 Разработка базы данных 64](#_Toc72488476)

[3.6 Разработка главного модуля 68](#_Toc72488477)

[3.7 Разработка входящих модулей 69](#_Toc72488478)

[3.8 Тестирование и отладка 75](#_Toc72488479)

[3.9 Дневник практики. 76](#_Toc72488480)

# **Раздел 1.**

# Практическое занятие №2.

Материал подготовил студент группы П2-18:

Кузнецов Матвей Сергеевич

Установка Python будет производиться в Visual Studio 2019.

1. Запускаем Visual Studio 2019.

2. Нажимаем на кнопку "Создание проекта";

3. В появившемся/открывшемся окне, прокручиваем до конца списка; (Будет надпись “Не нашли то, что искали?”)

4. Нажимаем на кнопку "Установка других средств и компонентов";

5. В открывшемся установщике выбираем раздел "Рабочие нагрузки";

6. После выбора раздела "Рабочие нагрузки", ищем в "Веб-разработки и облако" (Разработка на Python) и выбираем его;

7. После выбора (Разработка на Python), жмем на кнопку "Изменить"; (Ждем окончание загрузки)

8. После завершение установке, автоматически открывается Visual Studio, в котором можно запустить среду разработки Python.



# Практическое занятие №3.

Материал подготовил студент группы П2-18:

Кузнецов Матвей Сергеевич

## Создание файла с кодом.

Код исходной программы:  
print('Hello, world!') **(sample.py)**

## Запуск.

Для того, чтобы открыть существующий файл с кодом, нужно:  
1. В папке с нужным файлом нажать сочетание клавиш Shift + ПКМ.

2. В появившемся диалоговом окне нажать “Открыть окно PowerShell здесь”. (см. приложение 1)

3. Ввести команду “python название-файла”.

4. Получить результат выполнения программы (см. приложение 2).

## Import.

Чтобы подключить определенную библиотеку, нужно:  
1. В папке с нужным файлом нажать сочетание клавиш Shift + ПКМ.

2. В появившемся диалоговом окне нажать “Открыть окно PowerShell здесь”. (см. приложение 1)

3. Ввести в консоль команду “python”.

4. Ввести в консоль команду “import название-библиотеки”. (см. приложение 3)

## Reload.

Чтобы перезагрузить определенное нечто, требуется:  
1. В папке с нужным файлом нажать сочетание клавиш Shift + ПКМ.

2. В появившемся диалоговом окне нажать “Открыть окно PowerShell здесь”. (см. приложение 1)

3. Ввести в консоль команду “python”.  
4. Импортировать нужный файл. (см. приложение 4)  
5. Перезаписать исходный файл.   
Код измененной программы:

print('Hello!!!!!') **(sample.py)**

6. Написать команду “reload sample”.

## Отработка ошибок.

Примеры часто встречающихся ошибок:



Рисунок 1. Пример ошибки.

Данная ошибка говорит о не прописанной команде “python”.



Рисунок 2. Пример ошибки.

Данная ошибка говорит о том, что интерпретатор не может найти переменную с указанным именем.

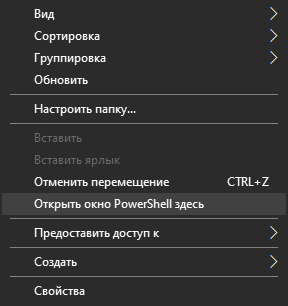


Рисунок 3. Способ исправления ошибки.



Рисунок 4. Способ исправления ошибки.



Рисунок 5. Способ исправления ошибки.

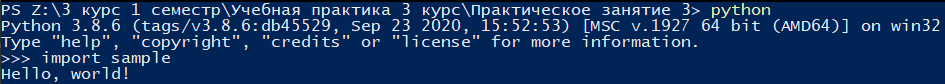


Рисунок 6. Исправление ошибки.

# Практическое занятие №4.

## Задачи на Begin.

**Задача 1: ( begin1.py )**

# Выполнил Кузнецов М. С. П2-18.

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

x = int(input("Введите сторону квадрата: "))

a = x \* 4

print("Периметр квадрата равен", a)

**Задача 2: ( begin2.py )**

# Выполнил Кузнецов М. С. П2-18.

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

x = int(input("Введите сторону квадрата: "))

a = x \* x

print("Площадь квадрата равна", a)

## Задачи на Boolean.

**Задача 1: ( boolean1.py )**

# Выполнил Кузнецов М. С. П2-18.

# Дано целое число A.

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

x = int(input("Введите число A: "))

res = x > 0

print(res)

**Задача 2: ( boolean2.py )**

# Выполнил Кузнецов М. С. П2-18.

# Дано целое число A.

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

x = int(input("Введите число A: "))

res = x % 2 == 0

print(res)

## Задачи на Case.

**Задача 1: ( case1.py )**

# Выполнил Кузнецов М. С. П2-18.

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

# недели,

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

# д.).

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

if 0 < x < 8:

if x == 1:

print("Это понедельник!")

elif x == 2:

print("Это вторник!")

elif x == 3:

print("Это среда!")

elif x == 4:

print("Это четверг!")

elif x == 5:

print("Это пятница!")

elif x == 6:

print("Это суббота!")

elif x == 7:

print("Это воскресенье!")

else:

print("Число не в диапазоне 1-7!")

**Задача 2: ( case2.py )**

# Выполнил Кузнецов М. С. П2-18.

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

# числу K

# (1 — «плохо», 2 — «неудовлетворительно», 3 — «удовлетворительно», 4

# — «хорошо», 5 — «отлично»).

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

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

if 0 < x < 6:

if x == 1:

print("Плохо")

elif x == 2:

print("Неудовлетворительно")

elif x == 3:

print("Удовретворительно")

elif x == 4:

print("Хорошо")

elif x == 5:

print("Отлично")

else:

print("Ошибка")

**Задача 3: ( case3.py )**

# Выполнил Кузнецов М. С. П2-18.

# Элементы окружности пронумерованы следующим образом:

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

# круга S = π·R2.

# Дан номер одного из этих элементов и его значение.

# Вывести значения остальных элементов данной окружности (в том же

# порядке).

# В качестве значения π использовать 3.14.

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

Pi = float(3.14)

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

if a == 1:

print("Радиус =", R)

elif a == 2:

D = 2 \* R

print("Диаметр =", D)

elif a == 3:

L = 2 \* Pi \* R

print("Длина окружности =", L)

elif a == 4:

S = Pi \* (R \* 2)

print("Площадь круга =", S)

## Задачи на If.

**Задача 1: ( if1.py )**

# Выполнил Кузнецов М. С. П2-18.

# Дано целое число. Если оно является положительным, то прибавить к

# нему 1;

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

x = int(input("Введите число: "))

if x > 0:

x = x + 1

print("Число положительное.", x)

else:

print("Число отрицательное или ноль.", x)

**Задача 2: ( if2.py )**# Выполнил Кузнецов М.С. П2-18.

# Дано целое число. Если оно является положительным, то прибавить к

# нему 1;

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

x = int(input("Введите число: "))

if x > 0:

x = x + 1

print("Число положительное.", x)

else:

x = x - 2

print("Число отрицательное или ноль.", x)

**Задача 3: ( if3.py )**

# Выполнил Кузнецов М. С. П2-18.

# Дано целое число. Если оно является положительным, то прибавить к

# нему 1;

# если отрицательным, то вычесть из него 2;

# если нулевым, то заменить его на 10. Вывести полученное число

x = int(input("Введите число: "))

if x > 0:

x = x + 1

print("Число положительное.", x)

elif x < 0:

x = x - 2

print("Число отрицательное.", x)

else:

x = 10

print("Введенное число - ноль.", x)

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

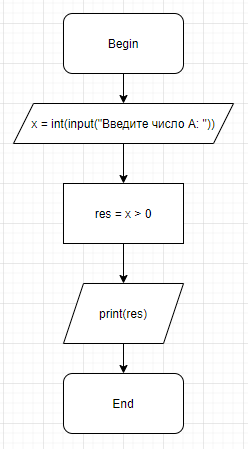
****

Рисунок 7. Блок-схема для задачи boolean1.

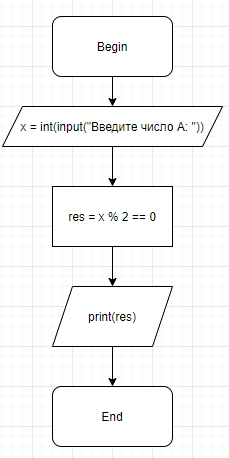


Рисунок 8. Блок-схема для задачи boolean2.

# Практическое занятие №5.

## Задачи на For.

**Задача 1: ( for1.py )**

# Выполнил Кузнецов М. С. П2-18.

# Дано вещественное число — цена 1 кг конфет.

# Вывести стоимость 1, 2, …, 10 кг конфет.

x = int(input("Введите цену 1кг конфет: "))

n = 0

for kg in range(1, 11):

n = x + n

print('>', n, 'стоит', kg, 'кг конфет')

kg += 1

**Задача 2: ( for2.py )**

# Выполнил Кузнецов М. С. П2-18.

# Дано вещественное число A и целое число N (> 0).

# Найти A в степени N:

A = int(input("Введите A: "))

N = int(input("Введите степень, в которую должно быть возведено число A: "))

n = int(1)

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

x = A \*\* n

print(A, "в степени", n, "будет", x)

n += 1

**Задача 3: ( for3.py )**

# Выполнил Кузнецов М. С. П2-18.

# Даны два целых числа A и B (A < B).

# Найти сумму всех целых чисел от A до B включительно.

A = int(input("Введите A: "))

B = int(input("Введите B: "))

x = int(0)

if A < B:

for i in range(A, B + 1):

x = x + i

print("Сумма всех целых чисел от", A, "до", B, "будет", x)

else:

print("A !< B. Exit program...")

**Задача 4: ( for4.py )**

#Выполнил Кузнецов М. С. П2-18.

# Даны два целых числа A и B (A < B).

# Найти произведение всех целых чисел от A до B включительно.

A = int(input("Введите A: "))

B = int(input("Введите B: "))

x = int(1)

if A < B:

for i in range(A, B + 1):

x = x \* i

print("Произведение всех целых чисел от", A, "до", B, "будет", x)

else:

print("A !< B. Exit program...")

**Задача 5: ( for5.py )**

# Выполнил Кузнецов М. С. П2-18.

# Даны два целых числа A и B (A < B).

# Найти сумму квадратов всех целых чисел от A до B включительно.

A = int(input("Введите A: "))

B = int(input("Введите B: "))

x = int(1)

if A < B:

for i in range(A, B + 1):

x = x + (i \*\* 2)

x -= 1

print("Сумма квадратов всех целых чисел от", A, "до", B, "будет", x)

else:

print("A !< B. Exit program...")

## Задачи на While.

**Задача 1: ( while1.py )**

# Выполнил Кузнецов М. С. П2-18.

# По данному целому числу N распечатайте все квадраты натуральных

# чисел,

# не превосходящие N, в порядке возрастания.

n = int(input("Введите N:"))

i = 1

while i \*\* 2 <= n:

print(i \*\* 2)

i += 1

**Задача 2: ( while2.py )**

# Выполнил Кузнецов М. С. П2-18.

# Дано целое число, не меньшее 2.

# Выведите его наименьший натуральный делитель, отличный от 1.

n = int(input("Введите целое число: "))

i = 2

while n % i != 0:

i += 1

print(i)

**Задача 3: ( while3.py )**

# Выполнил Кузнецов М. С. П2-18.

# По данному натуральному числу N найдите наибольшую целую степень

# двойки, не превосходящую N.

# Выведите показатель степени и саму степень.

# Операцией возведения в степень пользоваться нельзя!

n = int(input("Введите натуральное число N:"))

x = 2

y = 1

while x <= n:

x \*= 2

y += 1

print(y - 1, x // 2)

**Задача 4: ( while4.py )**

# Выполнил Кузнецов М. С. П2-18.

# Дано целое число N (> 0). Если оно является степенью числа 3,

# то вывести true, если не является — вывести false.

n = int(input("Введите целое число N( >0 ):"))

s = 3

r = 1

while (r \* s <= n):

r \*= s

if r == n:

print("True")

else:

print("False")

**Задача 5: ( while5.py )**

# Выполнил Кузнецов М. С. П2-18.

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

# На отрезке длины A размещено максимально возможное количество

# отрезков длины B (без наложений).

# Не используя операции умножения и деления, найти количество отрезков

# B, размещенных на отрезке A.

a = int(input("Введите А: "))

b = int(input("Введите B: "))

i = 1

while (a > b):

a -= b

i += 1

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

# Блок-схемы для задач к практическому занятию №5

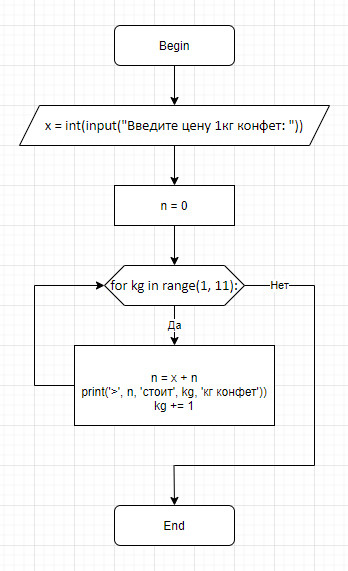


Рисунок 9. Блок-схема для задачи for1.

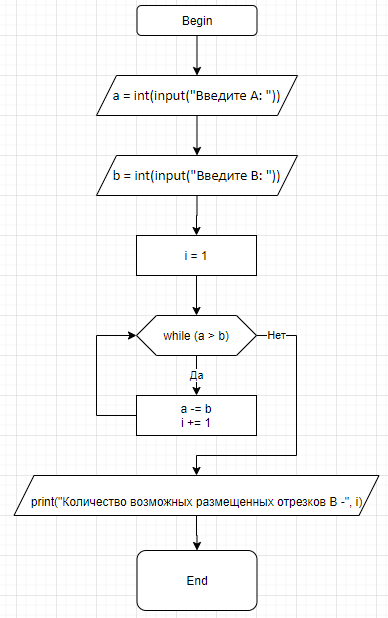


Рисунок 10. Блок-схема для задачи while5.

# Практическое занятие №6.

Complex.

# Выполнил Кузнецов М. С. П2-18.

# Операции с комплексными числами

a = complex(3, 4) # Первое число - действительное, второе - мнимое.

print('Обычное комплексное число a:', a)

b = complex(6, 4)

print('Обычное комплексное число b:', b)

c = a + b

print('Результат сложения этих комплексных чисел:', c)

print('Мнимая часть комплексного числа а:', a.imag)

print('Действительная часть комплексного числа b:', b.real)

print('Проверим на равенство комплексные числа a и b:', a == b)

print('Модуль комплексного числа a:', abs(a))

print('Возведение в четвертую степень комплексного числа b', pow(b, 4))  
**(complex.py)**

Decimal.

# Выполнил: Короленко Иван Романович

# Группа: П2-18

# Decimal- вычисления с заданной точностью

3.3 + 4.1 # Ответом будет 7,33333, в нашем же случае 7.4

from decimal import Decimal

a = Decimal('3.3') + Decimal('4.1')

print(a)

3.3 + 4.1 == 7.4 # должно быть True, но...

# Но с числами типа Decimal все верно:

c = Decimal('3.3') + Decimal('4.1')

print(c)

# С помощью дополнительных знаков мы можем определить, сколько будет

# символов в дробной части числа:

number = Decimal("0.10")

number = 3 \* number

print(number) # 0.30

# Однако нельзя смешивать в операциях дробные числа float и Decimal:

number = Decimal("0.1")

number = number + 0.1 # здесь возникнет ошибка

# Округление чисел

# Объекты Decimal имеют метод quantize(),который позволяет округлять

# числа.

# В этот метод в качестве первого аргумента передается также объект

# Decimal,

# который указывает формат округления числа:

number = Decimal("0.444")

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

print(number) # 0.44

number = Decimal("0.555678")

print(number.quantize(Decimal("1.00"))) # 0.56

number = Decimal("0.999")

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

# Как отделить рубли от копеек в вещественных числах?

# Decimal вместо float:

x = Decimal('40.80')

a = int(x)

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

print(a, b) # => 40 80

**(decimal.py)**

Fraction.

# Cлесарев А.М. Задание разбор модуля Fraction

# Модуль fractions

from fractions import Fraction

from decimal import Decimal

import math

# Fraction.limit\_denominator(max\_denominator=1000000)- ближайшее

# рациональное число со знаменателем не больше данного.

a = Fraction(3.1415) # Целое и вещественное число, так же можно  
# преобразовать в обыкновенную дробь

print (a)

print (a.limit\_denominator())

# по умолчанию numerator=0, denominator=1

print (Fraction())

# равносильно Fraction(1, 2)

print (Fraction(numerator=1, denominator=2))

# Если указанные числитель и знаменатель имеют общие делители,

# то перед созданием рационального числа они будут сокращены

print (Fraction(8, 16), Fraction(15, 30))

# Если указанные числитель и знаменатель имеют общие делители,

# то перед созданием рационального числа они будут сокращены

print (Fraction(3, Fraction(1, 2)))

# Округляет до ближайшего четного числа.

print (Fraction('1/2').\_\_round\_\_())

# создает обыкновенную дробь, которая является точным представлением

# десятичной дроби указанной в dec, где dec – это экземпляр класса

# decimal.Decimal

print (Fraction.from\_decimal(Decimal('0.7')))

# принимает flt – число типа float и возвращает обыкновенную дробь

# отношение числителя

# к знаменателю которой максимально приближается к значению flt.

print (Fraction.from\_float(0.5))

# fractions.gcd(a, b) - наибольший общий делитель чисел a и b.

print (math.gcd(1000, 3))

print (math.gcd(4, 6))  
# Использовался сайт https://pyprog.pro/python/st\_lib/fractions.html

**(fraction.py)**

# Практическое занятие №7

**Задание 1. (1.py)**

# Выполнил: Короленко И.Р.

# Группы: П2-18

# С клавиатуры вводятся строки, последовательность заканчивается

# точкой.

# Выведите буквы введенных слов в верхнем регистре, разделяя их

# пробелами.

# Примечание: узнайте, как работает строковый метод upper().

word = input()

mass = []

while word != '.':

mass.append(word)

word = input()

for item in mass:

output = ''

for letter in item:

output +=letter.upper()

print(\*output)

**Задание 2. (2.py)**

# Выполнил: Короленко И.Р.

# Группы: П2-18

words = input()

Wrd = words.split()

Wrd1= '\_'.join(words)

print(Wrd1)

# Второй вариант, с решения степика)

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

**Задание 3. (3.py)**

# Выполнил: Короленко И.Р.

# Группы: П2-18

s = input()

s = s.replace("."," ")

s1 = s.split()

s1 = [int(x) for x in s1]

print(s)

print(sum(s1))

**Задание 4. (4.py)**

# Слесарев А.М. 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.

err = list(input())

while err[0] != "." and a :

a = len(err)

if err[0] and err[1] and err[a-1] and err[a-2] == "@":

print('ошибка')

elif err[0] and err[1] and err[a-1] and err[a-2] == "!":

print('предупреждение')

elif err[0] and err[1] and err[a-1] and err[a-2] == "/":

print('информация')

elif err[0] and err[1] and err[a-1] and err[a-2] == "\*":

print('подробное сообщение')

err = list(input())

**7-2. (7-2.py)**

# Выполнил: Короленко Иван Романович

# Группа: П2-18

# Задание 1. Подготовить сравнительную инструкцию по использованию

# форматирования строк

# 1 Форматирование строк “По старинке” (оператор %)

name = input()

print('1) Hello, %s' % name)

# 2 Форматирование строк “По новому” (str.format)

print('2) Hello, {}'.format(name))

# 3 Интерполяция строк / f-Строки (Python 3.6+)

print(f'3) Hello, {name}!')

def greet(name, question):

return f"Hello, {name}! How's it {question}?"

print(greet(name, 'going'))

# 4 Шаблонные строки (Стандартная библиотека Template Strings)

from string import Template

t = Template('4) Hey, $name!')

print(t.substitute(name=name))

# Если для подстановки требуется только один аргумент,

# то значение - сам аргумент:

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

# А если несколько, то значениями будут

# являться все аргументы со строками подстановки (обычных или

# именованных):

print('{0}, {1}, {2}'.format('a', 'b', 'c'))

print('Coordinates: {latitude}, {longitude}'.format(latitude='37.24N', longitude='-115.81W'))

# В общем случае, аргументы могут быть как именованными, так b

# позиционными:

s = '{x}; {0}; {y}; {1}'

print(s.format('A', 'B', x = 1, y = 2))

# Здесь нужно обратить внимание на два нюанса: первый – позиционные

# аргументы, должны следовать перед именованными;

# второй – аргументами могут быть данные любого типа:

s = '{int}; {float}; {complex}'

print(s.format(int = 2, float = 2e-5, complex = 2+0.2j))

a = list(range(3))

b = dict([[1,'a'],[2,'b']])

c = set('aabbcc')

s = '{list}; {dict}; {set}'

print(s.format(list = a, dict = b, set = c))

# Мало того, что мы можем подставлять списки и словари, мы еще и можем

# обращаться к элементам, которые в них расположены.

# Практическое занятие №8.

**Задание 1 (8-1.z1.py)**

# Выполнил Кузнецов М. С. 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.

x = [int(a) for a in input().split()] # Сканим все числа и загоняем в список.

o = 0

for i in range(1, len(x) - 1): # Отбрасываем два крайних элемента, т.к у них по одному соседу.

if x[i] > x[i - 1] and x[i] > x[i + 1]: # Проверяем, i-ый элемент больше ли своих соседей.

o += 1

print(o)

**Задание 2 (8-1.z2.py)**

# Выполнил Кузнецов М. С. 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.

x = [int(a) for a in input().split()] # Сканим все числа и загоняем в список.

o = 0

for i in range(len(x)): # цикл в пределах количества чисел в списке.

t = 0

while t + i + 1 < len(x): # Счет значений.

if x[i] == x[t + i + 1]: # Проверка элементов.

o += 1

t += 1

print(o)

**Задание 3 (8-1.z3.py)**

# Выполнил Кузнецов М. С. П2-18.

# В списке все элементы различны. Поменяйте местами минимальный и

# максимальный элемент этого списка.

x = [int(i) for i in input().split()]

minn = 0

maxx = 0

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

if x[i] < x[minn]:

minn = i

if x[i] > x[maxx]:

maxx = i

x[minn], x[maxx] = x[maxx], x[minn]

print(' '.join([str(i) for i in x]))

**Задание 4 (8-2.z1.py)**# Выполнил: Короленко Иван Романович

# Студент: ККМТ П2-18.

'''

Задание 1. Array112. 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.

'''

from random import randint

arr = []

N = int(input())

for i in range(N):

arr.append(randint(1, 99))

print('Начальный массив\n',arr)

for i in range(1,N) :

print(arr)

for j in range(0,N-i) :

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

arr[j], arr[j+1] = arr[j+1], arr[j]

print('\nКонечный массив\n',arr)

**Задание 5 (8-2.z2.py)**

# Выполнил: Короленко Иван Романович

# Студент: ККМТ П2-18

'''

Задание 2. Array113. 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  
'''

from random import randint

arr = []

N = int(input())

for i in range(N):

arr.append(randint(1, 99))

print('Начальный массив\n',arr)

for i in range(0,N):

print(arr)

max\_val = max(arr[:N-i])

MaxI = arr.index(max\_val)

arr[MaxI], arr[N-i-1] = arr[N-i-1], arr[MaxI]

print('\nКонечный массив\n',arr)

**Задание 6 (8-2.z3.py)**

# Выполнил: Короленко Иван Романович

# Студент: ККМТ П2-18

'''

Задание 3. Array114. 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).

'''

from random import randint

arr = []

N = int(input())

for i in range(N):

arr.append(randint(1, 99))

print('Начальный массив\n',arr)

for i in range(1,N) :

flag = False

x = arr[i]

k = i - 1

while k >= 0 :

if x >= arr[k] :

break

else :

arr[k+1] = arr[k]

flag = True

k -= 1

if flag and k >= -1:

arr[k+1] = x

print(arr)

print('\nКонечный массив\n',arr)

# Практическое занятие №9.

**Задание 1 (9-2.z1.py)**

# Выполнил Кузнецов М. С. 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.

import random

n = random.randrange(2, 15)

a = [random.randrange(0, 100) for i in range(n)]

print('Массив a, состоящий из', n, 'элементов получится:')

print(a)

print('Новый массив b, состоящий из нечетных порядковых номеров массива a:')

b = a[1::2]

print(\*b)

print('Его размер:', str(len(b)))

**Задание 2 (9-2.z2.py)**

# Выполнил Кузнецов М. С. П2-18.

# Дан целочисленный массив A размера N. Переписать в новый

# целочисленный массив B

# того же размера вначале все элементы исходного массива с четными

# номерами,

# а затем — с нечетными:

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

# Условный оператор не использовать.

import random

n = random.randrange(5, 15)

a = [random.randrange(0, 100) for i in range(n)]

b = a[::2] + a[1::2]

print('Массив a, состоящий из', n, 'элементов получится:')

print(\*a)

print('Массив b того же размера вначале все элементы исходного массива с четными номерами, а затем — с нечетными:')

print(\*b)

**Задание 3 (9-2.z3.py)**

# Выполнил Кузнецов М. С. 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.

import random

n = random.randrange(5, 15)

a = [random.randrange(0, 100) for i in range(n)]

b = []

x = 0

for i in range(n):

x += a[i]

b.insert(0, x)

b.reverse()

print('Массив a, состоящий из', n, 'элементов получится:')

print(\*a)

print('Массив b того же размера по правилу: элемент B[K] равен сумме элементов массива A с номерами от 0 до K.')

print(\*b)

**Задание 4 (9-2.z4.py)**

# Выполнил Кузнецов М. С. 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).

import random as rnd

m = rnd.randrange(2, 10)

n = rnd.randrange(2, 10)

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

print('m =', m, 'n =', n)

print(\*a, sep=', ')

b = [a] \* n

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

**Задание 5 (9-2.z5.py)**

# Выполнил Кузнецов М. С. П2-18.

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

# левую и правую половины матрицы.

import random as rnd

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

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

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

print('m =', m, 'n =', n)

half = len(a) // 2

b = [a[half:] + a[:half]] \* n

print(a)

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

# Практическое занятие №10.

**Задание 1 (10-1.z1.py)**

# Выполнил Кузнецов М. С. 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(x, y):

if x > y:

return 0

else:

return sum(range(x, y + 1))

A = int(input('Введите первое число: '))

B = int(input('Введите второе число: '))

C = int(input('Введите третье число. Если не хотите его использовать - введите нуль: '))

if (A and B > 0) and (C == 0):

print("A =", A)

print("B =", B)

print('Сумма чисел от', A, 'до', B, 'равна', SumRange(A, B))

elif A and B and C > 0:

print("A =", A)

print("B =", B)

print("C =", C)

print('Сумма чисел от', A, 'до', B, 'равна', SumRange(A, B))

print('Сумма чисел от', B, 'до', C, 'равна', SumRange(B, C))

**Задание 2 (10-1.z2.py)**

# Выполнил Кузнецов М. С. 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.

import math

import random as rnd

def IsSquare(k):

a = int(math.sqrt(k))

if k == a \* a:

return True

else:

return False

x = 0

for i in range(0, 10):

a = rnd.randrange(1, 100)

print(a, end=" ")

x += int(IsSquare(a))

print('\nКоличество квадратов чисел:', x)

**Задание 3 (10-1.z3.py)**

# Выполнил Кузнецов М. С. 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.

import random as rnd

def SortInc3(x):

x.sort()

A = [rnd.uniform(0, 100) for i in range(3)]

B = [rnd.uniform(0, 100) for i in range(3)]

SortInc3(A)

SortInc3(B)

print(A)

print(B)

**Задание 4 (10-2.z1.py)**

# Выполнил Кузнецов М. С. П2-18.

# Использовать map, lambda

# Квадраты в обратном порядке. Числа вводятся до точки. Через пробел

# выведите эти числа в

# обратном порядке, возводя их в квадрат.

# Sample Input:

# 5

# 16

# 20

# 1

# 9

# .

#

# Sample Output:

# 81 1 400 256 25

v = [int(x) for x in iter(input, '.')]

b = list(map(lambda x: print(x \* x, end=' '), v[::-1]))

**Задание 5 (10-2.z2.py)**

# Выполнил Кузнецов М. С. П2-18.

# Использовать lambda, filter.

# Array55. Дан целочисленный массив A размера N (<= 15). Переписать в

# новый целочисленный

# массив B все нечетные элементы массива a и вывести размер

# полученного массива B и его содержимое. Условный оператор не

# использовать.

import random

n = random.randrange(2, 15)

a = [random.randrange(0, 100) for i in range(n)]

print('Массив a, состоящий из', n, 'элементов получится:')

print(a)

print('Новый массив b, состоящий из нечетных элементов массива a:')

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

print(b)

print('Его размер:', str(len(b)))

**Задание 6 (10-2.z3.py)**

# Выполнил Кузнецов М. С. П2-18.

# Использовать lambda, map.

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

# 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())

x = []

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

x.append(i)

b = list(map(lambda x: print(x \*\* c, end=' '), x))

# Практическое занятие №11.

**Задание 1 (11-1.z1.py)**

# Выполнил: Короленко Иван Романович

# Студент: П2-18

'''

Задача «Номер появления слова»

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

подсчитайте, сколько раз оно встречалось в этом тексте ранее.

Словом считается последовательность непробельных символов идущих подряд, слова разделены

одним или большим числом пробелов или символами конца строки.

'''

words = {}

for word in input().split():

words[word] = words.get(word, 1) + 1

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

**Задание 2 (11-1.z2.py)**

# Выполнил: Короленко Иван Романович

# Студент: П2-18

'''

Условие. 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

'''

files = {}

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

name, \*operations = input().split()

files[name] = operations

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

operation, name = input().split()

if operation == 'read':

if 'R' in files[name]:

print('OK')

else:

print('Access denied')

elif operation == 'write':

if 'W' in files[name]:

print('OK')

else:

print('Access denied')

elif operation == 'execute':

if 'X' in files[name]:

print('OK')

else:

print('Access denied')

**Задание 3 (11-1.z3.py)**

# Выполнил: Короленко Иван Романович

# Студент: П2-18

'''

Задача «Самое частое слово»

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

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

выведите то, которое меньше в лексикографическом порядке.

'''

word = {}

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

line = input().split()

for words in line:

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

count = max(word.values())

most\_frequent = [k for k, v in word.items() if v == count]

print(min(most\_frequent))

**Задание 4 (11-2.z1.py)**

# 11\_2\_1 Слесарев А.М.

# Коля устал запоминать телефонные номера и заказал у Вас

# программу, которая заменила бы ему телефонную книгу. Коля может

# послать программе

# два вида запросов: строку, содержащую имя контакта и его номер,

# разделенные пробелом,

# или просто имя контакта. В первом случае программа должна добавить в

#книгу новый номер,

# во втором – вывести номер контакта. Ввод происходит до символа

# точки. Если введенное

# имя уже содержится в списке контактов, необходимо перезаписать

# номер.

pb = dict()

data = input()

while data != '.':

data = data.split()

if len(data) == 2:

nm, nmbr = data

pb[nm] = nmbr

else:

nm = ''.join(data)

print(pb[nm])

data = input()

**Задание 5 (11-2.z2.py)**

# 11\_2\_2 Слесарев А.М.

# Коля понял, что у многих из его знакомых есть несколько телефонных

# номеров и нельзя хранить только один из них.

# Он попросил доработать Вашу программу так, чтобы можно было

# добавлять к существующему контакту новый номер или

# даже несколько номеров, которые передаются через запятую. По запросу

# телефонного номера должен выводиться весь

# список номеров в порядке добавления, номера должны разделяться

# запятой. Если у контакта нет телефонных номеров,

# должна выводиться строка "Не найдено".

pb = dict()

data = input()

while data != '.':

data = data.replace(',', '').split()

if len(data) == 1:

nm = ''.join(data)

if nm in pb:

print(', '.join(pb[nm]))

else:

print("Не найдено")

else:

nm, nmbr = data[0], data[1:]

pb[nm] = pb.get(nm, []) + nmbr

data = input()

**Задание 6 (11-2.z3.py)**

# 11\_2\_2 Слесарев А.М.

# Коля понял, что у многих из его знакомых есть несколько телефонных

# номеров и нельзя хранить только один из них.

# Он попросил доработать Вашу программу так, чтобы можно было

# добавлять к существующему контакту новый номер или

# даже несколько номеров, которые передаются через запятую. По запросу

# телефонного номера должен выводиться весь

# список номеров в порядке добавления, номера должны разделяться

# запятой. Если у контакта нет телефонных номеров,

# должна выводиться строка "Не найдено".

a = []

b = []

sch = 0

run = 1

while run:

for i in input().split(', '):

a.append([])

for j in i:

if j.isalpha() != True:

a[sch].append(j)

if i == '.':

run = 0

sch += 1

sch = -1

for i in a:

m = ''

for j in i:

if i[0] == j == ' ':

b.append([])

sch += 1

if j.isdigit() or j == '+':

m = m + j

if m != '' or len(m) != 0:

b[sch].append(m)

a = []

# print(b)

for i in b:

for j in i:

# print(j)

if (j[0] == '8' and len(j) == 11):

a.append(j[1:])

elif (j[:2] == '+7' and len(j) == 12):

a.append(j[2:])

# print(a)

for i in a:

print('+7 ({0}) {1}-{2}-{3}'.format(i[0:3], i[3:6], i[6:8], i[8:10]))

# Практическое занятие №12.

**Задача 1(12-1.z1.py)**

# Выполнил Кузнецов М. С. П2-18.

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

# чисел.

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

**Задача 2(12-1.z2.py)**

# Выполнил Кузнецов М. С. П2-18.

# Даны два списка чисел. Посчитайте, сколько чисел содержится

# одновременно как в первом списке, так и во втором.

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

**Задача 3(12-1.z3.py)**

# Выполнил Кузнецов М. С. П2-18.

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

# первый,

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

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

b = [int(i) for i in input().split()]

x = set(a)

y = set(b)

z = x & y

r = list(z)

print(\*sorted(r))

**Задача 4(12-1.z4.py)**

# Выполнил Кузнецов М. С. 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.

a = set()

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

a.update(input().split())

print(len(a))

**Задача 5(12-1.z5.py)**

# Выполнил Кузнецов М. С. 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())

a = set()

b = set()

m = int(input())

for i in range(m):

s = input()

b.add(s)

a.add(s)

for i in range(n-1):

m = int(input())

d = []

for j in range(m):

s = input()

b.add(s)

d.append(s)

a = a.intersection(set(d))

print(len(a))

print(\*sorted(a))

print(len(b))

print(\*sorted(b))

**Задача 5(12-2.z1.py)**

# Выполнил Слесарев А. М. П2-18.

# Простейшая система проверки орфографии может быть основана на

# использовании списка известных слов.

# Если введённое слово не найдено в этом списке, оно помечается как

# "ошибка".

# Попробуем написать подобную систему.

# На вход программе первой строкой передаётся количество d известных

# нам слов, после чего

# на d строках указываются эти слова.

# Затем передаётся количество l строк текста для проверки, после чего

# l строк текста.

# Выведите уникальные "ошибки" в произвольном порядке. Работу

# производите без учёта регистра.

a = int(input())

b = []

for i in range(a):

x = input().lower()

if x not in b:

b.append(x)

d = int(input())

e = []

for j in range(d):

x = input().lower().split()

for i in x:

if i not in b and i not in e:

e.append(i)

print('\n'.join(e))

Практическое занятие №13.  
**Задание 1 (13-1.z1.py)**

# Выполнил Кузнецов М. С. П2-18.

# Вывести чётные

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

# Sample Input:

# 2

# Sample Output:

# (2, 4, 6, 8, 10, 12, 14, 16, 18, 20)

import random as r

a = r.randrange(1, 10)

b = []

for i in range(a, a \* 10 + 1):

if i % 2 == 0:

b.append(i)

print(tuple(b))

**Задание 2 (13-1.z2.py)**

# Выполнил: Короленко Иван Романович

# Студент: ККМТ П2-18

# Убывающий ряд.

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

# по одному числу в строке.

# Sample Input:

# -2

# -8

# Sample Output:

# -2

# -3

# -4

# -5

# -6

# -7

a = int(input())

b = int(input())

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

print(i+1)

**Задание 3 (13-2.py)**

# Выполнил: Короленко Иван Романович

# Студент: ККМТ П2-18

# Задание 1. Класс namedtuple() модуля collections в Python.

'''

Класс namedtuple() модуля collections возвращает новый подкласс кортежа с именем typename.

Новый подкласс используется для создания объектов, похожих на кортежи, которые имеют индексируемые и итерируемые поля, доступные для поиска по атрибутам.

Экземпляры подкласса также имеют полезную строку документации с typename и field\_names, а так же метод \_\_repr\_\_(), который перечисляет содержимое кортежа в формате name=value.

Имена полей field\_names представляют собой последовательность строк, таких как ['x', 'y'].

В качестве альтернативы, field\_names может быть одной строкой, в которой каждое имя поля разделено пробелами и/или запятыми, например, 'x y' или 'x, y'.

Для имен полей (элементов кортежа) может использоваться любой действительный идентификатор Python, за исключением имен, начинающихся с подчеркивания.

Допустимые идентификаторы состоят из букв, цифр и символов подчеркивания, но не начинаются с цифры или символа подчеркивания и не могут быть ключевыми словами, такими как class, for, return, global, pass и т. д.

Если аргумент rename=True, то недопустимые имена полей автоматически заменяются позиционными именами.

Например ['abc', 'def', 'ghi', 'abc'] преобразуется в ['abc', '\_1', 'ghi', '\_3'], исключая ключевое слово def и повторяющееся имя поля abc.

Значения аргумента defaults могут быть None или итерируемой последовательностью.

Поскольку аргумента со значением по умолчанию должны идти после любых обязательных аргументов, то значения по умолчанию будут применяются к самым правым параметрам. Например, если имена полей именованного кортежа это ['x', 'y', 'z'], а значения по умолчанию (1, 2), то тогда x будет обязательным аргументом, y по умолчанию будет 1, а z будет 2.

Если аргумент module определен, то атрибуту именованного кортежа \_\_module\_\_ присваивается значение module.

Экземпляры именованных кортежей не имеют словарей, поэтому они легковесны и требуют не больше памяти, чем обычные кортежи.

'''

# Параметры:

# typename - строка, имя именованного кортежа,

# field\_names - последовательность строк, имена элементов кортежа,

# rename - bool, авто-переименование повторяющихся имен элементов,

# defaults = None - итерируемая последовательность, значения по

# умолчанию имен кортежа,

# module = None - атрибут \_\_module\_\_ именованного кортежа.

# Простой пример

from collections import namedtuple

Point = namedtuple('Point', ['x', 'y'])

p = Point(11, y=22)

# можно обращаться по индексу

# как к обычному кортежу

p[0] + p[1]

# 33

# распаковать как обычный кортеж

x, y = p

x, y

# (11, 22)

# поля также доступны по названию

p.x + p.y

# 33

# человеко-читаемый \_\_repr\_\_

print(p)

# Point(x=11, y=22)

# Именованные кортежи особенно полезны для присвоения имен полей в

# результате кортежей, возвращаемых модулями csv или sqlite3:

from collections import namedtuple

EmployeeRecord = namedtuple('EmployeeRecord', 'name, age, \title, department, paygrade')

import csv

for emp in map(EmployeeRecord.\_make, csv.reader(open("employees.csv", "rb"))):

print(emp.name, emp.title)

import sqlite3

conn = sqlite3.connect('/companydata')

cursor = conn.cursor()

cursor.execute('SELECT name, age, title, department, paygrade FROM employees')

for emp in map(EmployeeRecord.\_make, cursor.fetchall()):

print(emp.name, emp.title)

# Поскольку именованный кортеж является обычным классом Python, его

# легко добавить или изменить с помощью подкласса.

# Вот как добавить вычисляемое поле и формат печати фиксированной

# ширины:

from collections import namedtuple

class Point(namedtuple('Point', ['x', 'y'])):

\_\_slots\_\_ = ()

@property

def hypot(self):

return (self.x \*\* 2 + self.y \*\* 2) \*\* 0.5

def \_\_str\_\_(self):

return 'Point: x=%6.3f y=%6.3f hypot=%6.3f' % (self.x, self.y, self.hypot)

for p in Point(3, 4), Point(14, 5/7):

print(p)

# Point: x=3.000 y=4.000 hypot=5.000

# Point: x=14.000 y=0.714 hypot=14.018

'''

Примеры

'''

from collections import Counter

counter\_one = Counter('superfluous')

# Counter({'u': 3, 's': 2, 'e': 1, 'l': 1, 'f': 1, 'o': 1, 'r': 1,

# 'p': 1})

print(counter\_one)

counter\_two = Counter('super')

counter\_one.subtract(counter\_two)

print(counter\_one)

# Counter({'u': 2, 'l': 1, 'f': 1, 'o': 1, 's': 1, 'e': 0, 'r': 0,

# 'p': 0})

sentence = "The red for jumped over the fence and ran to the zoo for food"

words = sentence.split(' ')

reg\_dict = {}

for word in words:

if word in reg\_dict:

reg\_dict[word] += 1

else:

reg\_dict[word] = 1

print(reg\_dict)

# Практическое занятие №14.

**Задание 1 (14-1.z1.py)**

# Выполнил Кузнецов М. С. П2-18.

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

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

b = input('Введите то, что хотите, чтоб оказалось в файле\n')

a.write(b)

a.close()

print('Готово?!')

**Задание 2 (14-1.z2.py)**

# Выполнил Слесарев А. М. П2-18.

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

# на строку S.

import os

import shutil

from os import path

s = input()

f1 = open('Text.txt', 'r')

f2 = open ('out.txt', 'w')

for line in f1:

if line == '\n':

f2.write(s + '\n')

else:

f2.write(line)

f2.close ()

f1.close ()

os.remove('Text.txt')

os.rename('out.txt','Text.txt')

**Задание 3 (14-1.z3.py)**

# Выполнил: Короленко Иван Романович

# Студент: ККМТ П2-18

# Задание 3

# Text20. Дан текстовый файл.

# Заменить в нем все подряд идущие пробелы на один пробел.

file1 = "text20.txt"

print("Read from:", file1)

file2 = "text20\_2.txt"

print("Write to:", file2)

flag = True

try:

with open(file1, 'r') as infile, open(file2, 'w') as outfile:

for line in infile:

line\_new = ""

print(line)

for c in line.strip("\n"):

if c != " ":

line\_new += c

flag = True

elif flag:

line\_new += c

flag = False

outfile.write(line\_new + "\n")

except IOError:

print('Open error: ', file1)

**Задание 4 (14-1.z4.py)**

# Выполнил Слесарев А. М. П2-18.

# Дан текстовый файл, каждая строка которого изображает целое число,

# дополненное слева и справа

# несколькими пробелами. Вывести количество этих чисел и их сумму.

summ = 0

i = 0

f1 = open('read.txt')

for line in f1:

line = line.strip()

if line != '':

summ += int(line)

i += 1

f1.close()

print('Summa=', summ, 'kol-vo', i)

**Задание 5 (14-1.z5.py)**

# Выполнил: Короленко Иван Романович

# Студент: ККМТ П2-18

# Задание 5.

# Text53. Дан текстовый файл. Создать символьный файл, содержащий все

# знаки препинания,

# встретившиеся в текстовом файле (в том же порядке).

with open('text53.txt') as inp, open('text53\_2.txt','w') as out:

res = []

for line in inp.readlines():

res += ''.join(list(line.rstrip()))

out.write(''.join(sorted(set(res),key = ord)))

print(res)

print(line)

# Практическое занятие №15.

**Задание 1 (15-1.z1.py)**

# Выполнил Кузнецов М. С. 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 (15-1.z2.py)**

# Выполнил Кузнецов М. С. 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 (15-2.py)**

# Слесарев А.М. П2-18.

from collections import defaultdict

a = [('Леша', 1), ('Ваня', 2), ('Мотя', 3), ('Леша', 4), ('Мотя', 1)]

b = {}

for k, v in a:

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

sorted(b.items())

print(a)

print(b)

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

def constant\_factory(value):

return lambda: value

d = defaultdict(constant\_factory('<missing>'))

d.update(name='John', action='ran')

print(d)

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

s = [('red', 1), ('blue', 2), ('red', 3), ('blue', 4), ('red', 1), ('blue', 4)]

d = defaultdict(set)

for k, v in s:

d[k].add(v)

sorted(d.items())

print(d)

**Задание 3 (15-3.py)**

# Выполнил: Короленко Иван Романович

# Студент: ККМТ 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.

'''

Существует много других значений и методов в модуле sys.

Теперь вы знаете, как выйти из программы Python,

как получить информацию о платформе, работать с аргументами, переданными командной строке, и многому другому.

'''

**Задание 4 (15-4.py)**

# Выполнили: Короленко Иван Романович и Кузнецов Матвей Сергеевич.

# Студенты: ККМТ П2-18

# Работа с файлами в Python с помощью модуля OS

# Задание 1. Вывод текущей директории

# Задание 2. Создание папки

# Задание 3. Изменение директории

# Задание 4. Создание вложенных папок

# Задание 5. Создание файлов

# Задание 6. Переименование файлов

# Задание 7. Перемещение файлов

# Задание 8. Список файлов и директорий

# Задание 9. Удаление файлов

# Задание 10. Удаление директорий

# Задание 11. Получение информации о файлах

import os # Библиотека для работы с файлами

import shutil

# Задание 1. Вывод текущей директории

print('Задание 1. Вывод текущей директории')

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

# Задание 2. Создание папки

print('Задание 2. Создание папки')

fileName = input('Введите имя файла: ')

# Ввод текста, которое будет далее использоваться как название

# создаваемого файла!

if not os.path.isdir(fileName):

# Проверка на схожесть имя файла

os.mkdir(fileName)

# Создает файл в случае если он прошёл проверку

# Задание 3. Изменение директории

print("Задание 3. Изменение директории")

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

os.chdir(changesName)

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

# Задание 4. Создание вложенных папок

print('Задание 4. Создание вложенных папок')

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

os.makedirs(nestedFile)

# Задание 5. Создание файлов

print('Задание 5. Создание файлов')

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

text\_file = open(createFileName, "w") # создать новый текстовый файл

createText = input('Введите текст или код, который будет в файле: ')

text\_file.write(createText) # запить текста в этот файл

# Задание 6. Переименование файлов

print('Задание 6. Переименование файлов')

os.rename("text.txt", "renamed-text.txt")

# Задание 7. Перемещение файлов

print('Задание 7. Перемещение файлов')

shutil.move("путь до текущего файла", "новый путь до файла")

# Задание 8. Список файлов и директорий

print('Задание 8. Список файлов и директорий')

os.listdir('путь до определенной папки')

# Задание 9. Удаление файлов

print('Задание 9. Удаление файлов')

os.remove('путь до файла')

# Задание 10. Удаление директорий

print('Задание 10. Удаление директорий')

os.rmdir('путь до файла')

# Задание 11. Получение информации о файлах

print('Задание 11. Получение информации о файлах')

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

# Практическое занятие №16.

Листинг 73. K16\_1.py

#Выполнил: Кузнецов М. С.

#Группа: П2-18

'''

К16\_1. Техника работы с классами.

Задание 1. Создание класса

Задание 2. Создание объекта

Задание 3. Функция init

Задание 4. Методы объектов

Задание 5. Параметр self

Задание 6. Изменение свойств объекта

Задание 7. Удалить свойства объекта

Задание 8. Удаление объектов

'''

#Создадим класс

class BruhClass:

x = 20

#Создадим объект

p1 = BruhClass()

print(p1.x)

#Функция init

class DataHero:

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

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

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

self.name = name

self.age = age

#Методы класса

def Welcome(self):

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

def DataHero(bruh):

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

print(f"Имя: {bruh.name}")

print(f"Возраст: {bruh.age}")

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

print()

hero1 = DataHero("Владимир", 20)

hero1.Welcome()

hero1.DataHero()

#Свойство объекта можно менять

hero1.age = 9999

hero1.DataHero()

#Свойства объектов можно удалять

del hero1.age

#Также можно удалять сам объект

del hero1

# **Раздел 2.**

# Практическое занятие №17.

PyCharm - это кросс-платформенная среда разработки.

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

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

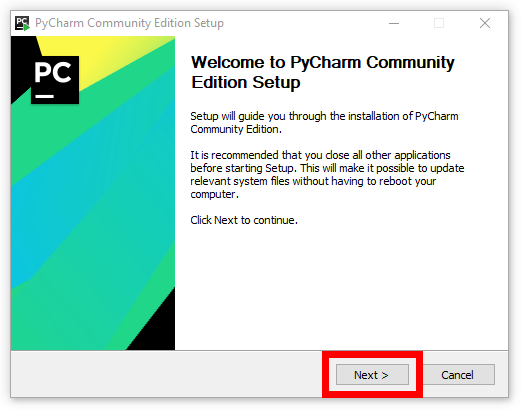


Рисунок 1. Установщик PyCharm

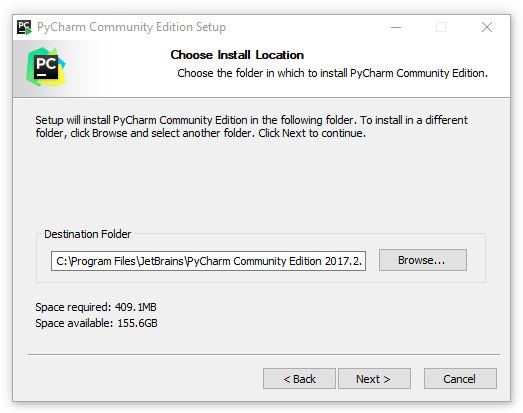


Рисунок 2. Место установки PyCharm

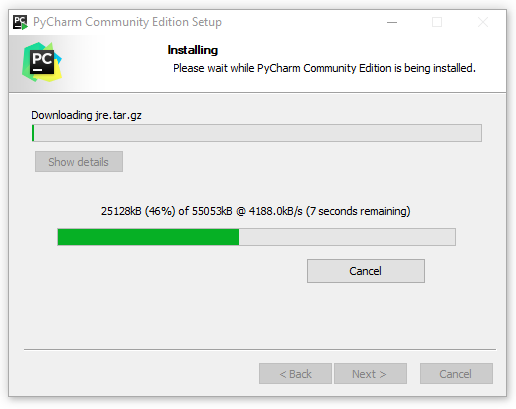


Рисунок 3. Процесс установки PyCharm

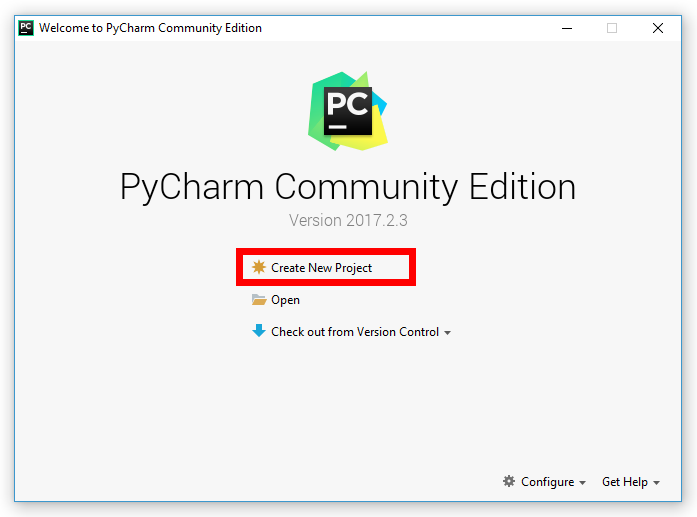


Рисунок 4. Стартовое окно PyCharm

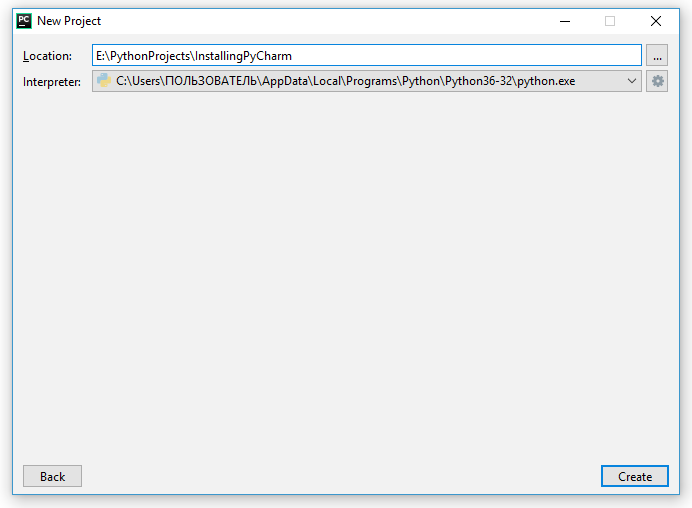
В следующем окне будет два поля. В первом укажите, где будет располагаться новый проект. Второе поле должно быть заполненным помолчанию. Там содержится путь к установленному ранее Python 

Рисунок 5. Выбор места для нового проекта и выбор интерпретатора

Дальше откроется окно самого редактора.

Теперь вы готовы начать писать программы на Python!

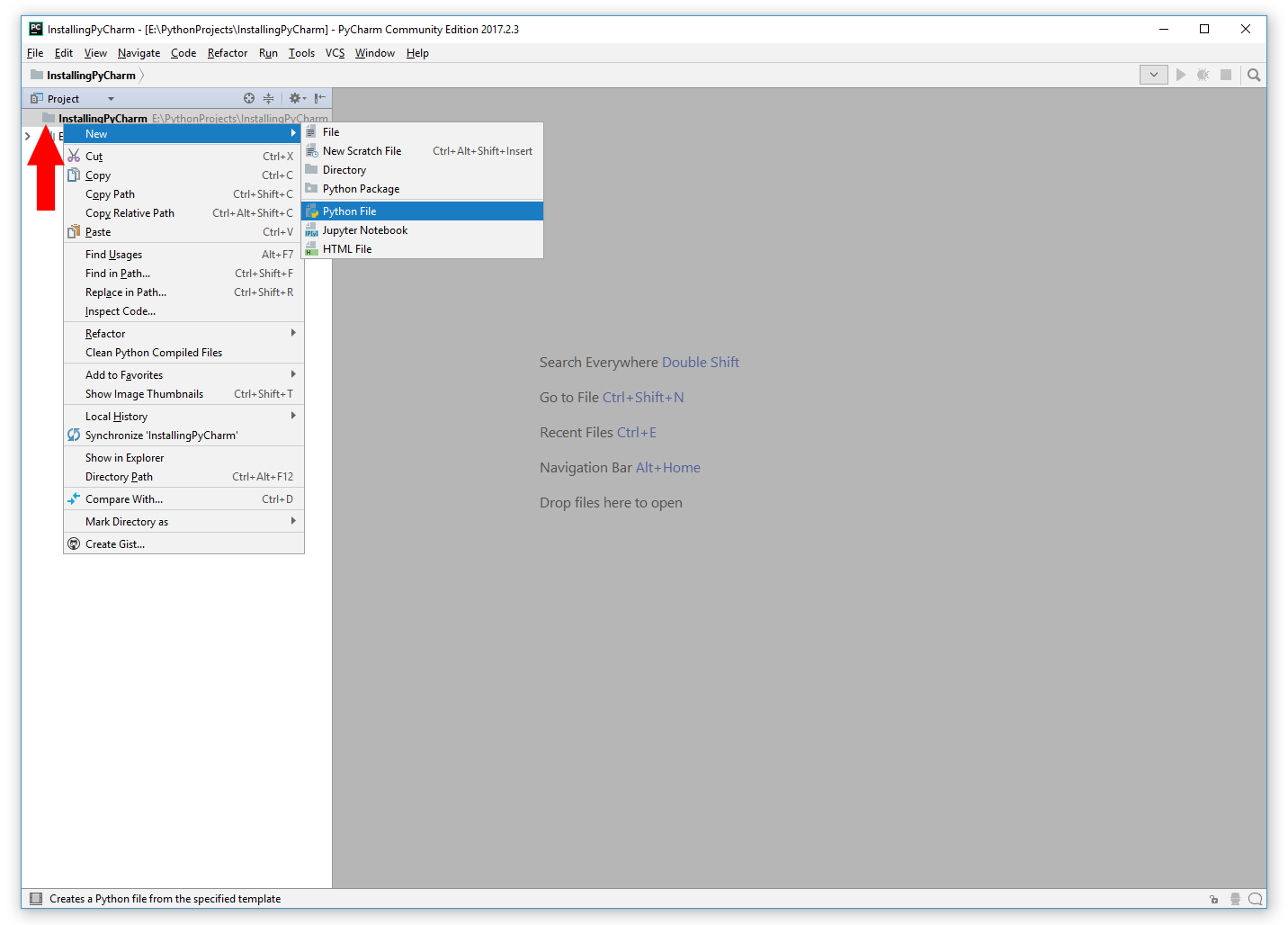
Кликните правой кнопкой на название вашего проекта и в раскрывающемся списке выберите пункт «New > Python file»: 

Рисунок 6. Создание нового Phython файла

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

Справа откроется сам файл. Пока он пустой. Пишем:

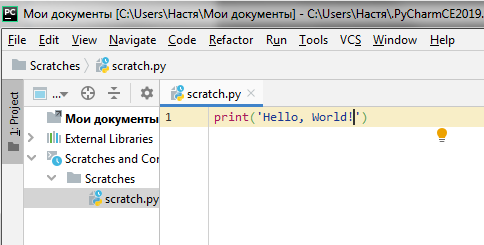


Рисунок 7. Написание программы

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

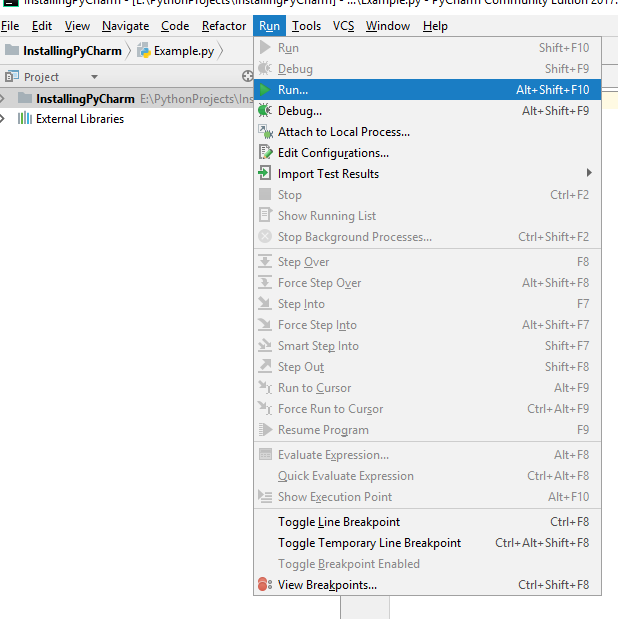


Рисунок 8. Запуск программы

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

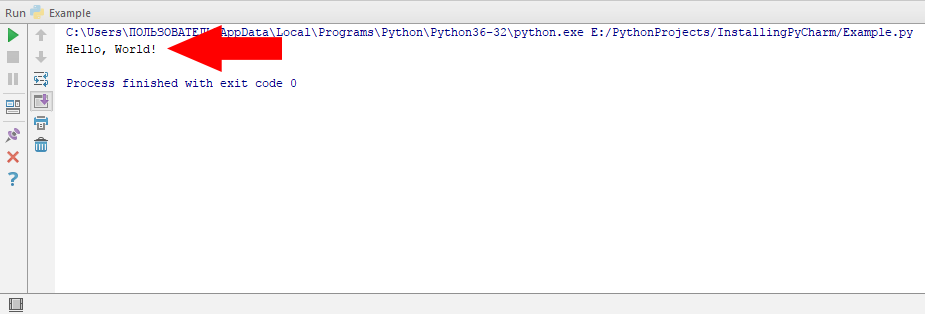


Рисунок 9. Выполнение программы

# Практическое занятие №18.

Приложения: sql\_bd.py

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

Листинг 77. sql\_bd.py

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

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

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

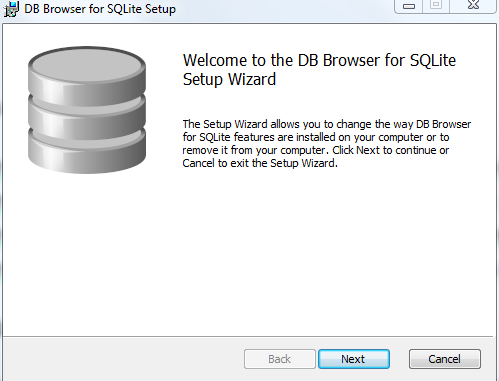


Рисунок 10. Установщик DB Browser

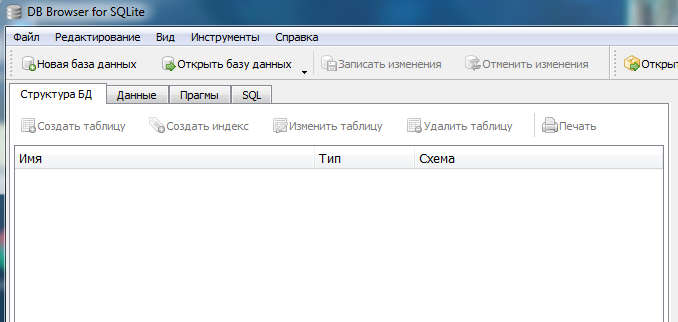
После установки, открываем DB Browser для работы с ним.

Рисунок 11. Окно DB Browser

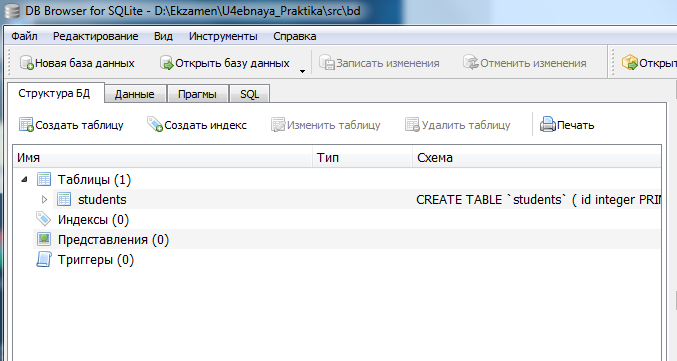
В открывшееся окно DB Browser’a перетаскиваем файл с БД.

Рисунок 12. Таблица students в DB Browser

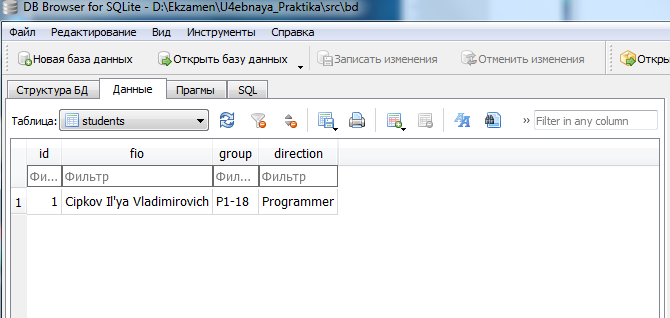
Чтобы просмотреть содержимое БД необходимо нажать на кнопку «Данные».

Рисунок 13. Данные таблицы students

Видно, что в таблицу занеслись корректные данные.

# Практическое занятие №19.

IntVar()), ("Pascal", IntVar()))

*# Заполнение и задание расположения CheckBox'ов* x, y = 2, 0  
 for name, var in parametrs:  
 Checkbutton(root, text=f"{name}", variable=var).grid(row=x, column=y)  
 x += 1  
  
 *# Задаём местоположение текстовой метке и полю для ввода* lb1 = Label(text="Имя:").grid(row=0, column=0)  
 en = Entry(width=30, textvariable=my\_var2).grid(row=0, column=1, columnspan=3)  
  
 *# Задаём местоположение текстовой метке и SpinBox'у(в котором находится список групп)* lb2 = Label(text="Группа:").grid(row=1, column=0, columnspan=1)  
 mas = ('П1-18', 'П2-18', 'П1-19', 'П2-19', 'П3-19')  
 sb = Spinbox(width=7, values=mas, textvariable=my\_var,\  
 command=group\_get).grid(row=1, column=1)

*# Задаём местоположение кнопкам Save и Delete* b = Button(root, text="Save", width=25, height=2, command=save).grid(row=9, column=1, columnspan=2)  
 b1 = Button(root, text="Delete", width=25, height=2,\  
 command=delete).grid(row=10, column=1, columnspan=2)  
  
 *# Создаём и задаём местоположение текстовой метке и RadioButton'ам* v = IntVar()  
 lb3 = Label(text="Пол:").grid(row=8, column=0)  
 rb1= Radiobutton(root, text="Мужской", value=1,\  
 variable=v).grid(row=8, column=1, columnspan=1)  
 rb2 = Radiobutton(root, text="Женский", value=2,\  
 variable=v).grid(row=8, column=2, columnspan=1)  
  
 *# Запуск цикла обработки событий* mainloop()  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Форма имеет вид:

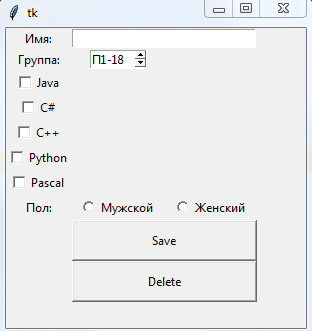


Рисунок 14. Вид формы

Заполним форму данными:

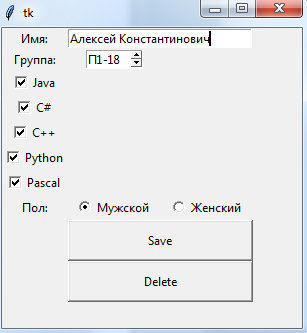


Рисунок 15. Заполненная форма

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

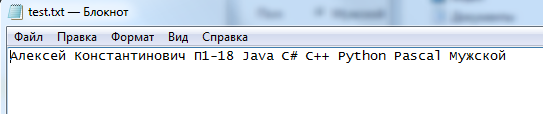


Рисунок 16. Вывод в файл

# Практическое занятие №20.

Приложения: primer.py

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

Установка:

• Через терминал командой: “pip install numpy”

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

Импорт модуля numpy:

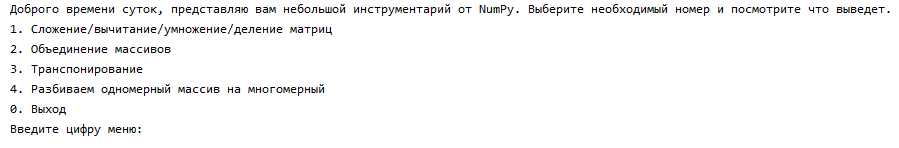
import numpy

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

Листинг 79: primer.py

import numpy  
  
  
def main():  
 *# Бесконечно выводим меню в консоль, пока пользователь не введёт 0* while(True):  
 menu()  
 print("Введите цифру меню: ")  
 num = int(input())  
 if(num == 1):  
 print("Введите функцию(-, +, \*, /):")  
 s = input()  
 function\_1(s)  
 print("\n")  
 elif(num == 2):  
 function\_2()  
 print("\n")  
 elif(num == 3):  
 function\_3()  
 print("\n")  
 elif(num == 4):  
 function\_4()  
 print("\n")  
 elif(num == 0):  
 *# Выход из программы* exit(0)  
 else:  
 print("Вы ввели не корректный номер функции. Попробуйте ещё раз\n")  
  
  
*# Функция menu обеспечивает вывод меню в консоль*def menu():  
 print("Доброго времени суток, представляю вам небольшой\  
 инструментарий от NumPy. Выберите необходимый номер и посмотрите что выведет.")  
 print("1. Сложение/вычитание/умножение/деление матриц")  
 print("2. Объединение массивов")  
 print("3. Транспонирование")  
 print("4. Разбиваем одномерный массив на многомерный")  
 print("0. Выход")  
  
*# Функция function\_1 выводит в консоль пример сложения/вычитания/деления/умножения матриц*def function\_1(s):  
 A = numpy.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
 B = A.copy()[::-1, ::-1]  
 if(s == "+"):  
 print("A\n", A)  
 print("B\n", B)  
 print("A + B\n", A + B)  
 elif(s == "-"):  
 print("A\n", A)  
 print("B\n", B)  
 print("A - B\n", A - B)  
 elif(s == "\*"):  
 print("A\n", A)  
 print("B\n", B)  
 print("A \* B\n", A \* B)  
 elif(s == "/"):  
 print("A\n", A)  
 print("B\n", B)  
 print("A / B\n", A // B)  
  
*# Функция function\_2 выводит в консоль пример объединения двух матриц в одну*def function\_2():  
 A = numpy.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
 B = A[::-1]  
 C = A[:, ::-1]  
 print("A\n", A)  
 print("\nB\n", B)  
 print("\nC\n", C)  
 print("\nstack(A, B, C)\n", numpy.stack((A, B, C)))  
  
*# Функция function\_3 выводит в консоль пример транспонирования матрицы*def function\_3():  
 A = numpy.array([[1, 2, 3], [4, 5, 6]])  
 print('A\n', A)  
  
 B = A.T  
 print('\nА транспонированный\n', B)  
  
*# Функция function\_4 выводит в консоль пример разбиения одномерного массива на многомерный*def function\_4():  
 A = numpy.arange(24)  
 B = A.reshape(4, 6)  
 C = A.reshape(4, 3, 2)  
 print("Массив размером 1х1\n", A)  
 print('\nМассив размером 4х6\n', B)  
 print('\nМассив размером 4х3х2\n', C)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':

main()

Рисунок 17. Вывод меню в консоль

# Практическое занятие №21.

Приложения: mat.py, mat2.py

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

Библиотеку matplotlib легко установить с помощью pip:

pip install matplotlib

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

Листинг 80: mat.py

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

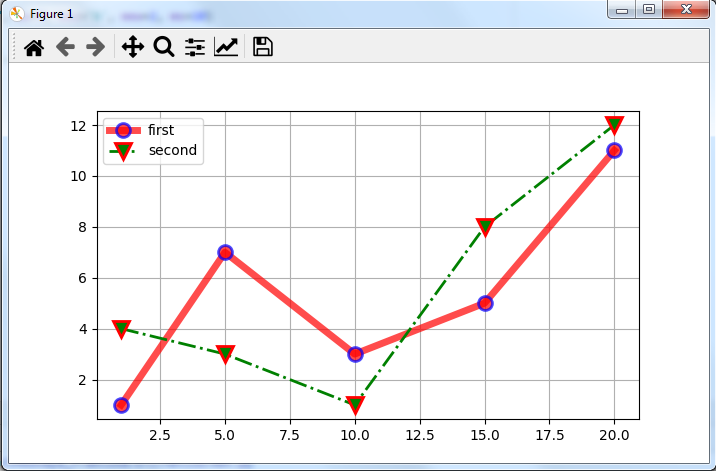


Рисунок 18. Вывод программы

# Практическое занятие №22.

Приложения: MyQt.py

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

Для установки библиотеки используется команда:

pip install PyQt5

Листинг 82: MyQt.py

from PyQt5.QtWidgets import \*  
from PyQt5.QtCore import QSize  
import random  
  
  
  
*# Наследуемся от QMainWindow*class MainWindow(QMainWindow):  
 *# Создаём диалоговое окно, которое открывается при нажатии на кнопку* def dialog(self, lineEdit, strList, secretList):  
 mbox = QMessageBox() *# Создаём диалоговое окно* mbox.setWindowTitle('Окно')  
 *# Условия вывода определённого текста в окне* if(lineEdit.text() in strList):  
 mbox.setText(lineEdit.text() + " krut!")  
 elif(lineEdit.text() in secretList):  
 mbox.setText(lineEdit.text() + "\nЭто правда")  
 else:  
 mbox.setText("prosto text")  
 mbox.setDetailedText("Наберите одно из этих слов\n'Python', 'PyQt5', 'Qt', 'Django', 'QML'")  
 mbox.exec()  
  
 *# Переопределяем конструктор класса* def \_\_init\_\_(self):  
 *# Обязательно нужно вызвать метод супер класса* QMainWindow.\_\_init\_\_(self)  
  
 self.setMinimumSize(QSize(480, 80)) *# Устанавливаем размеры* self.setWindowTitle("Проверка автодополнения") *# Устанавливаем заголовок окна* central\_widget = QWidget(self) *# Создаём центральный виджет* self.setCentralWidget(central\_widget) *# Устанавливаем центральный виджет* grid\_layout = QGridLayout() *# Создаём QGridLayout* central\_widget.setLayout(grid\_layout) *# Устанавливаем данное размещение в центральный виджет* grid\_layout.addWidget(QLabel("Проверка автодополнения", self), 0, 0)  
  
 *# Создаём поле ввода* lineEdit = QLineEdit(self)  
 strList = ['Python', 'PyQt5', 'Qt', 'Django', 'QML'] *# Создаём список слов* secretList = ['Хочу домой', 'Я сделал домашку', 'ККМТ - сила'] *# Создаём секретный список* rand = random.randint(1,10)  
 *# Создаём QCompleter, в который устанавливаем список, а также указатель на родителя  
 # С вероятностью 50% в QCompleter может добавиться секретный список* if(rand % 2 == 0):  
 completer = QCompleter(strList, lineEdit)  
 else:  
 completer = QCompleter(strList + secretList, lineEdit)  
 lineEdit.setCompleter(completer) *# Устанавливает QCompleter в поле ввода* grid\_layout.addWidget(lineEdit, 0, 1) *# Добавляем поле ввода в сетку* qb = QPushButton("Submit") *# Создаём кнопку  
 # Создаём действие для кнопки* qb.clicked.connect(lambda : self.dialog(lineEdit, strList, secretList))  
 grid\_layout.addWidget(qb, 0, 2)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import sys  
  
 app = QApplication(sys.argv)  
 mw = MainWindow()  
 mw.show()  
 sys.exit(app.exec())

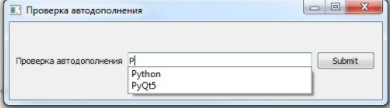


Рисунок 19. Окно проверки автодополнения

# Практическое занятие №23.

Приложения: main.py

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

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

python3 -m pip install -U pygame --user

или

sudo apt-get install python3-pygame

Для Windows:

Открыв командную строку (cmd) прописываем:

pip install pygame

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

Python

Import pygame

Листинг 83. main.py

import pygame

import random

class Hero:

"""Класс Hero. Здесь описана вся работа с героем"""

positions = list() # Координаты героеа

# Цвета для героев

colors = {

'RED': (237, 77, 12),

'PURPLE': (130, 12, 247),

'TURQUOISE': (22, 224, 214),

'YELLOW': (213, 247, 12),

'ORANGE': (240, 121, 40)

}

def \_\_init\_\_(self, size, type\_hero):

"""Создание героя"""

if type\_hero == 'circle' or type\_hero == 'square':

self.width = size[0]

self.height = size[0]

else:

self.width = size[0]

self.height = size[1]

self.type\_hero = type\_hero

self.bgc = (125, 125, 125)

self.x = 100

self.y = 100

self.speed = 1

self.target = {'y': '', 'x': ''}

self.\_\_class\_\_.positions.append([self.x, self.y])

self.index = len(self.\_\_class\_\_.positions)-1

#def \_\_repr\_\_(self):

# return f"{\_\_class\_\_.\_\_name\_\_ }(({self.width}x{self.height}), ({self.x}, {self.y}))"

def update\_positions(self):

"""Обновляем позиции героя в общеи списке позиций"""

self.positions[self.index][0] = self.x

self.positions[self.index][1] = self.y

def get\_all\_positions(self, pos):

"""Вернет все позиции 'x' или 'y' """

if pos == 'x':

return [pos[0] for pos in self.positions]

else:

return [pos[1] for pos in self.positions]

def set\_map(self, x\_start, x\_end, y\_start, y\_end):

"""Создание карты для героя"""

# Устанавливаем границы

if self.type\_hero == 'circle':

x\_start = x\_start + self.width

x\_end = x\_end - self.width

y\_start = y\_start + self.width

y\_end = y\_end - self.width

elif self.type\_hero == 'rect' or self.type\_hero == 'square':

x\_end = x\_end - self.width

y\_end = y\_end - self.height

self.mapp = {

'x\_start':x\_start,

'x\_end': x\_end,

'y\_start': y\_start,

'y\_end': y\_end

}

def set\_color(self, \*\*kwargs):

"""Установление цвета герою"""

if kwargs.get('hex') != None:

self.bgc = kwargs['hex']

elif kwargs.get('color') != None:

self.bgc = self.colors.get(kwargs['color'], (125, 125, 125))

else:

self.bgc = random.choice( list(self.colors.values()) )

def set\_type(self, type\_hero):

"""Установление типа герою. (круг, квадрат, прямоугольник)"""

if type\_hero == 'circle' or type\_hero == 'square':

if self.width > self.height:

self.height = self.width

else:

self.width = self.height

self.type\_hero = type\_hero

self.set\_map(\

self.mapp['x\_start'],

self.mapp['x\_end'],

self.mapp['y\_start'],

self.mapp['y\_end'],

)

def set\_speed(self, speed):

"""Установление скорости герою"""

self.speed = speed

def add\_speed(self, speed):

"""Повышние скорости героя"""

if speed < 0:

self.sub\_speed(speed)

else:

self.speed += speed

print("speed: ", self.speed)

def sub\_speed(self, speed):

"""Понижение скорости героя"""

if speed < 0:

self.add\_speed(speed)

else:

self.speed -= speed

if self.speed < 0:

self.speed = 0

print("speed: ", self.speed)

def set\_pos(self, \*args):

"""Установление позиций героя"""

if args:

self.x = args[0]

self.y = args[1]

else:

self.x = random.randrange(self.mapp['x\_start'], self.mapp['x\_end'])

self.y = random.randrange(self.mapp['y\_start'], self.mapp['y\_end'])

self.update\_positions()

def draw(self, display):

"""Отрисовка героя"""

if self.type\_hero == 'circle':

pygame.draw.circle(display, self.bgc, (self.x, self.y), self.width)

elif self.type\_hero == 'rect':

pygame.draw.rect(display, self.bgc, (self.x, self.y, self.width, self.height))

elif self.type\_hero == 'square':

pygame.draw.rect(display, self.bgc, (self.x, self.y, self.width, self.width))

def set\_target(self, \*args):

"""Установление стороны в которую движется герой"""

if not args:

self.target['x'] = random.choice(['left', 'right'])

self.target['y'] = random.choice(['top', 'bottom'])

elif args[0] == 'left' or args[0] == 'right':

self.target['x'] = args[0]

elif args[1] == 'top' or args[1] == 'bottom':

self.target['y'] = args[1]

print(self.target)

def navigation(self):

"""Передвижение героя"""

if self.target['x'] == 'left':

self.x -= self.speed

elif self.target['x'] == 'right':

self.x += self.speed

if self.target['y'] == 'bottom':

self.y += self.speed

elif self.target['y'] == 'top':

self.y -= self.speed

# Уперся в другой объект

is\_altered = False

for pos in self.positions:

if (pos[0] == self.x) and (pos[1] == self.y):

is\_altered = True

if self.target['x'] != '':

if self.target['x'] == 'left':

self.target['x'] = 'right'

elif self.target['x'] == 'right':

self.target['x'] = 'left'

if self.target['y'] != '':

if self.target['y'] == 'top':

self.target['y'] = 'bottom'

elif self.target['y'] == 'bottom':

self.target['y'] = 'top'

# Уперся в границу

if not is\_altered:

if (self.x <= self.mapp['x\_start']):

self.target['x'] = 'right'

elif (self.x > self.mapp['x\_end']):

self.target['x'] = 'left'

if (self.y <= self.mapp['y\_start']):

self.target['y'] = 'bottom'

elif (self.y >= self.mapp['y\_end']):

self.target['y'] = 'top'

self.update\_positions()

# Функционал игры

def Update(display, map\_size, heroes):

global stop, mode

# Какая операция со скоростью.

# 1 - повысить скорость

# 0 - ничего не делать

# -1 - понизить скорость

operating\_speed = 0

# Какую кнопку нажали

for event in pygame.event.get():

# Выход

if event.type == pygame.QUIT:

pygame.quit()

quit()

# Нажатие кнопки = зажатию

elif mode == 'press':

if event.type == pygame.KEYDOWN:

# Приостановление игры

if event.key == pygame.K\_ESCAPE:

stop = not stop

print(stop)

if not stop:

# Смена мода

if event.key == pygame.K\_TAB:

mode = 'touch'

# Нажатие кнопки = нажатию

elif mode == 'touch':

if event.type == pygame.KEYDOWN:

# Приостановление игры

if event.key == pygame.K\_ESCAPE:

stop = not stop

print(stop)

if not stop:

# Создание нового героя

if event.key == pygame.K\_SPACE and not stop:

hero = Hero((25,), 'circle')

# hero = Hero((50,), 'square')

hero.set\_map( map\_size["x\_start"], map\_size["x\_end"], map\_size["y\_start"], map\_size["y\_end"] )

hero.set\_speed(1)

hero.set\_pos()

hero.set\_target()

# hero.set\_pos(50, 100)

# hero.set\_target('', '')

hero.set\_color()

heroes.append(hero)

del hero

# статус скорости = Повышение

elif event.key == pygame.K\_UP:

operating\_speed = 1

# # статус скорости = Понижение

elif event.key == pygame.K\_DOWN:

operating\_speed = -1

# Смена мода

elif event.key == pygame.K\_TAB:

mode = 'press'

# Отрисовка фона

display.fill(bgc)

# Если игра не пиостановлена

if not stop:

if mode == 'press':

keys = pygame.key.get\_pressed()

# Создание новых персонажей

if keys[pygame.K\_SPACE]:

hero = Hero((25,), 'circle')

# hero = Hero((50,), 'square')

hero.set\_map( map\_size["x\_start"], map\_size["x\_end"], map\_size["y\_start"], map\_size["y\_end"] )

hero.set\_speed(1)

hero.set\_pos()

hero.set\_target()

# hero.set\_pos(50, 100)

# hero.set\_target('', '')

hero.set\_color()

heroes.append(hero)

del hero

# статус скорости = Повышение

elif keys[pygame.K\_UP]:

operating\_speed = 1

# статус скорости = Понижение

elif keys[pygame.K\_DOWN]:

operating\_speed = -1

# Относиельно статуса скорости повышаем/понижаем скорость

for hero in heroes:

if operating\_speed == 1:

hero.add\_speed(1)

elif operating\_speed == -1:

hero.sub\_speed(1)

hero.navigation()

hero.draw(display)

# Возращаем нейтральный статус для скорости

if operating\_speed != 0:

operating\_speed = 0

# Обновляем окно

pygame.display.update()

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

# Инициализация окна

pygame.init()

# Данные для окна

display\_size = {"width": 800, "height": 600}

display = pygame.display.set\_mode((display\_size['width'], display\_size['height']))

pygame.display.set\_caption("Bruh")

# bgc = (255, 255, 255)

bgc = (40, 40, 40)

# Размер карты

map\_size = {

"x\_start": 0,

"x\_end": display\_size['width'],

"y\_start": 0,

"y\_end": display\_size['height'],

}

# Создание героя (шарика). Кладем его в спсок со всеми героями

heroes = list()

hero = Hero((25,), 'circle')

# hero = Hero((50,), 'square')

hero.set\_map( map\_size["x\_start"], map\_size["x\_end"], map\_size["y\_start"], map\_size["y\_end"] )

hero.set\_speed(1)

hero.set\_target()

hero.set\_pos()

hero.set\_color()

heroes.append(hero)

del hero

game = True # Запушена ли игра

stop = False # Приостановлена ли игра

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

# Запуск игры

while game:

Update(display, map\_size, heroes)

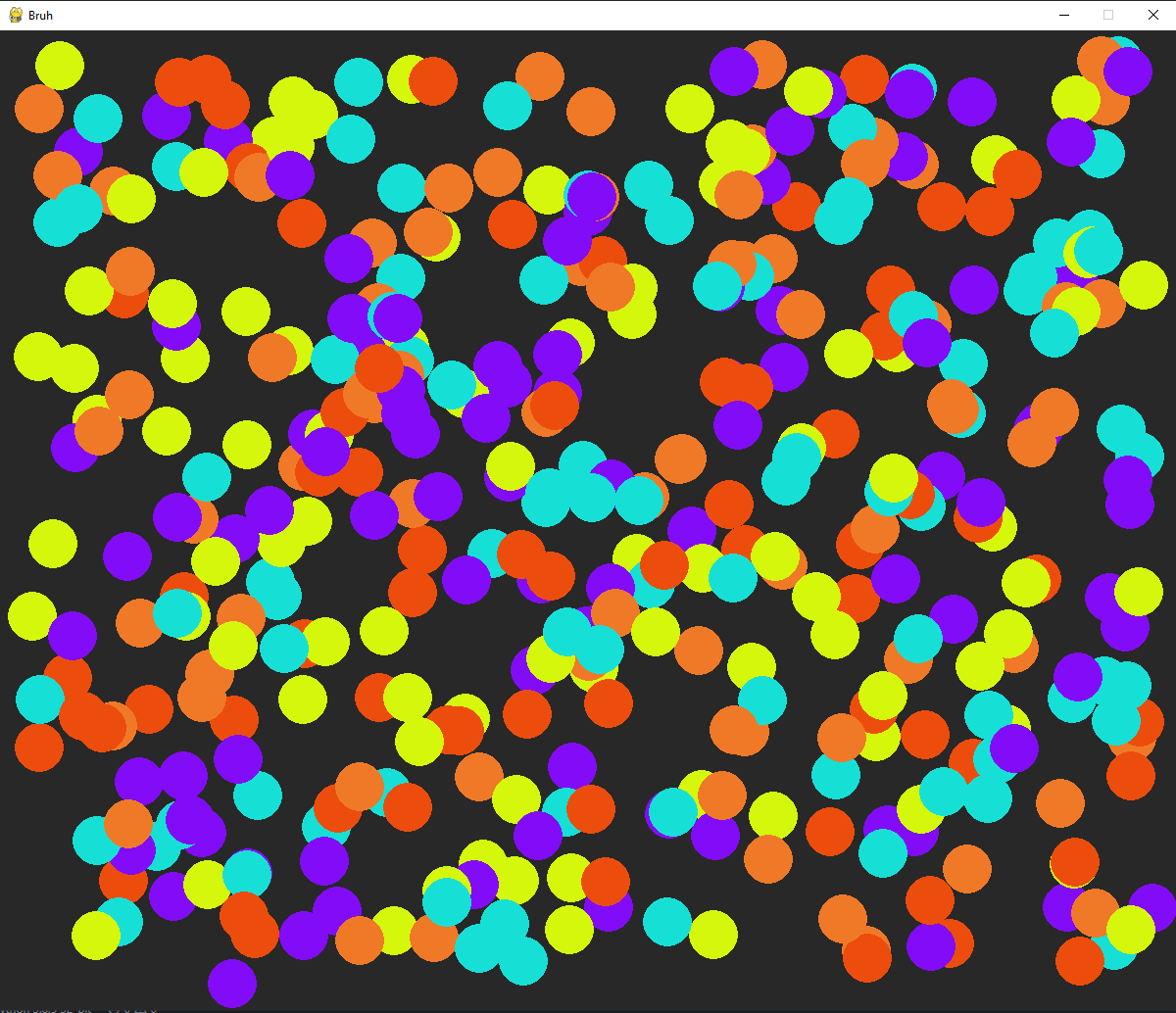


Рисунок 20. Скриншот игры

# **Раздел 3.**

# **Разработка проекта с графическим интерфейсом**

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

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

Чтобы импортировать его, пропишем

import tkinter

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

import sqlite3

3.1 Изучение входной и выходной документации

Входные данные:

1. Документ на оплату

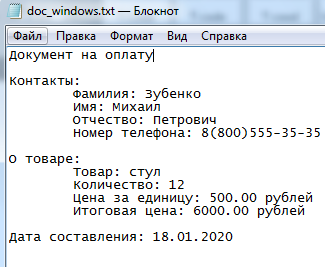


Рисунок 1. Документ на оплату

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

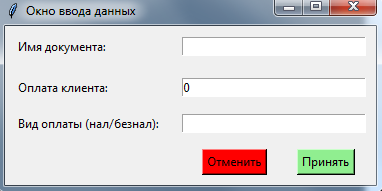
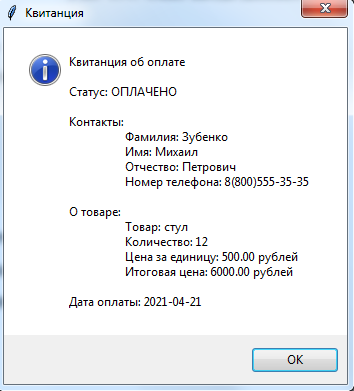


Рисунок 2. Окно ввода данных

Выходные данные

1. Квитанцию об оплате

  
Рисунок 3. Квитанция об оплате

1. Отчёт кассира

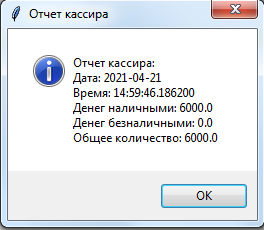


Рисунок 4. Отчёт кассира

1. База данных

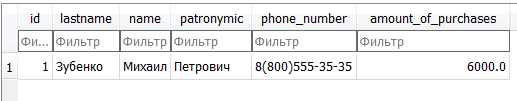


Рисунок 5. Таблица “clients”

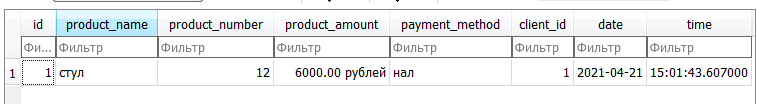


Рисунок 6. Таблица “registry”

3.2 Разработка требований к проекту. Построение диаграммы использования.

Требования к проекту:

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

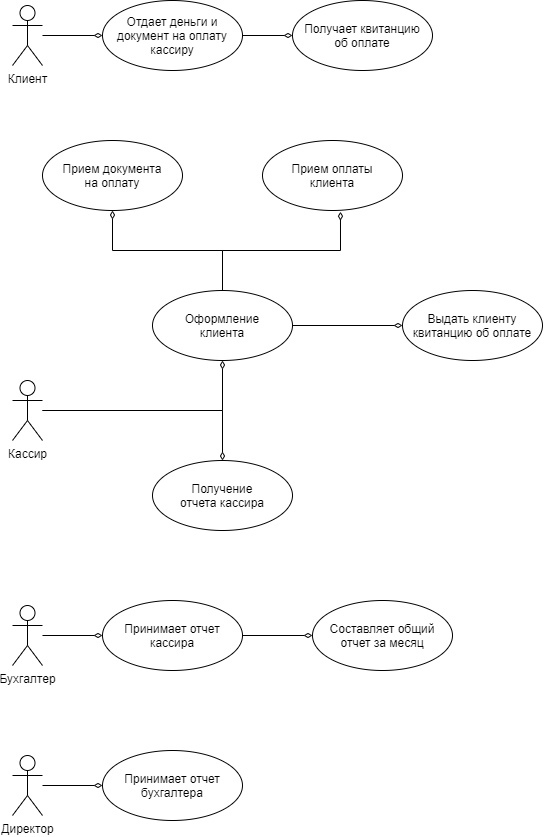


Рисунок 7. Диаграмма использования

* 1. Разработка сценария проекта

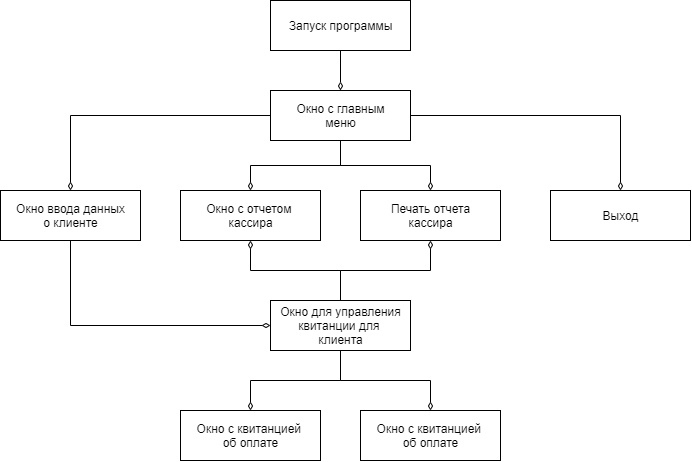
В данном разделе приведен сценарий использования программы пользователем.

Рисунок 8. Сценарий проекта

* 1. Построение диаграммы классов

В данном разделе находятся диаграммы классов приложения.

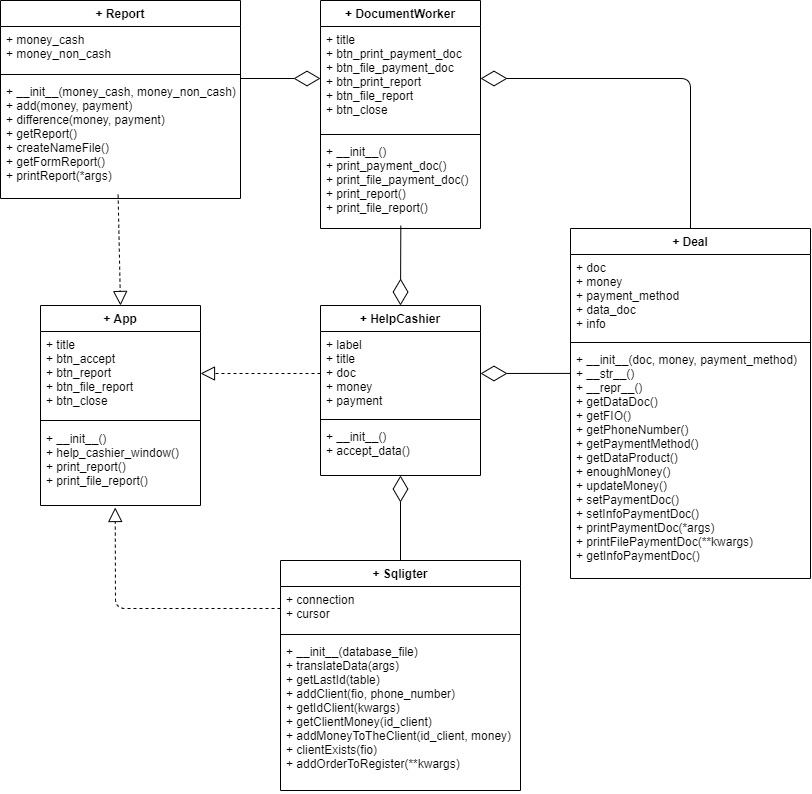


Рисунок 9. Диаграмма классов

* 1. Разработка базы данных

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

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

Для работы с бд мы решили написать отдельный класс SQLigter.

Листинг 84: [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. Разработка главного модуля

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

Листинг 85. 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. 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 86. 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")

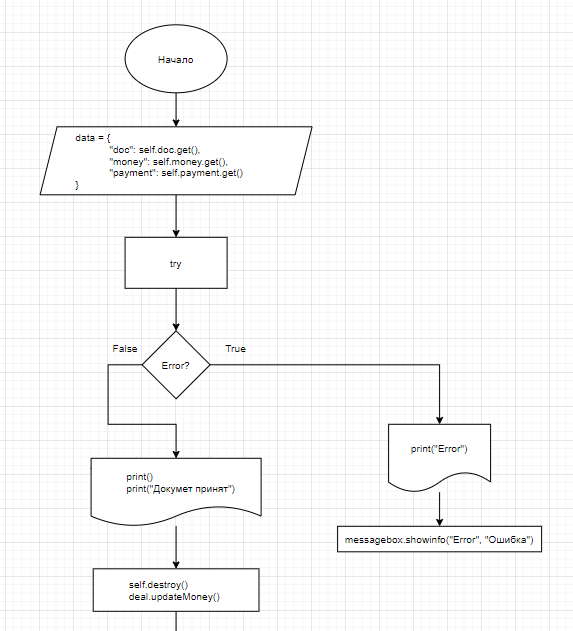


Рисунок 10. Блок-схема к функции accept\_data. Часть 1



Рисунок 11. Блок-схема к функции accept\_data. Часть 2

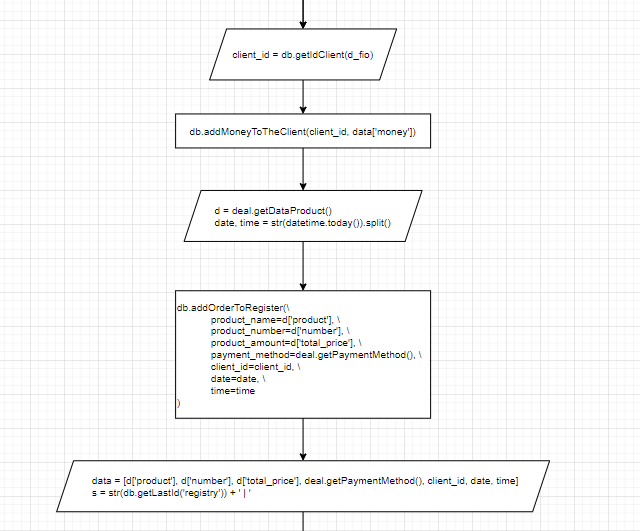


Рисунок 12. Блок-схема к функции accept\_data. Часть 3.

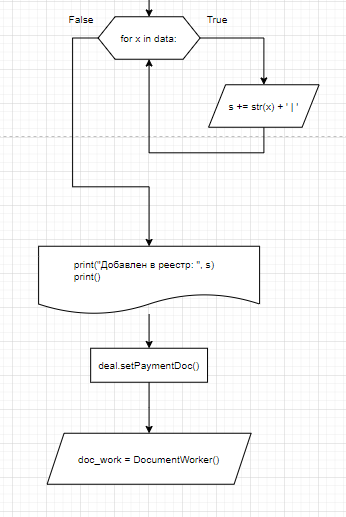


Рисунок 13. Блок-схема к функции accept\_data. Часть 4

**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. Тестирование и отладка

В ходе написания проекта при попытке запустить программу были получены ошибки:

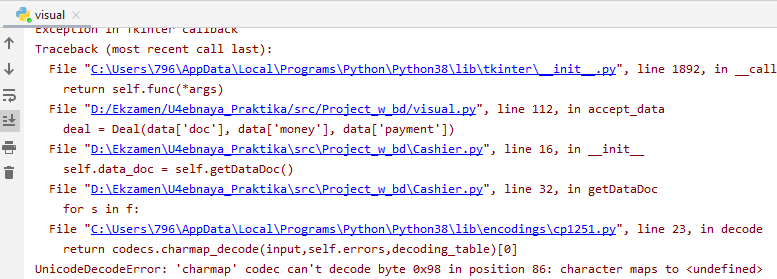


Рисунок 14. Ошибки

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



Рисунок 15. Успешная сборка

* 1. Дневник практики.

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работ | Отметка о выполнении |
| 24.11.2020 | Выдача заданий на практику. |  |
| 25.11.2020 | Установка интерпретатора. Настройка окружения. |  |
| 25.11.2020 | Техника работы в командной строке. Техника работы в среде IDLE. |  |
| 26.11.2020 | Техника работы с линейными программами. Техника работы с разветвляющимися программами. Техника работы с циклическими программами. |  |
| 27.11.2020 | Техника работы с циклическими программами. Техника работы с числами. |  |
| 28.11.2020 | Техника работы со строками. Техника работы со списками. |  |
| 30.11.2020 | Техника работы со списками. Техника работы с циклом for и генераторами списков. |  |
| 01.12.2020 | Техника работы с функциями. Техника работы со словарями. |  |
| 02.12.2020 | Техника работы со словарями. Техника работы с множествами. |  |
| 03.12.2020 | Техника работы с кортежами. Техника работы с файлами. |  |
| 04.12.2020 | Техника работы с файлами. Техника работы с модулями. |  |
| 05.12.2020 | Техника работы с модулями. Техника работы с классами. |  |
| 07.12.2020 | Техника работы с классами. |  |
| 27.04.2021 | Установка и настройка среды JetBrains PyCharm. Техника работы с базами данных. |  |
| 28.04.2021 | Техника работы с базами данных. Техника работы с библиотекой tkinter. |  |
| 30.04.2021 | Техника работы с библиотекой tkinter |  |
| 30.05.2021 | Техника работы с библиотекой tkinter. Техника работы с библиотекой NumPy. |  |
| 04.05.2021 | Техника работы с библиотекой NumPy. Техника работы с библиотекой Matplotlib. |  |
| 04.05.2021 | Техника работы с библиотекой Matplotlib. Элементы работы с библиотекой PyQt. |  |
| 05.05.2021 | Элементы работы с библиотекой PyQt. |  |
| 06.05.2021 | Элементы работы с библиотекой PyQt. Элементы работы с библиотекой PyGame. |  |
| 09.05.2021 | Элементы работы с библиотекой PyGame. Выполнение отчёта и презентации по использованию библиотек. |  |
| 11.05.2021 | Выполнение отчёта и презентации по использованию библиотек. |  |
| 12.05.2021 | Изучение входной и выходной документации. Разработка требований к проекту. Построение диаграммы использования |  |
| 12.05.2021 | Разработка требований к проекту. Построение диаграммы использования. Разработка сценария проекта. |  |
| 13.05.2021 | Разработка сценария проекта. Построение диаграммы классов. |  |
| 14.05.2021 | Построение диаграммы классов. Разработка базы данных. |  |
| 19.05.2021 | Разработка базы данных. Разработка главного модуля. |  |
| 20.05.2021 | Разработка главного модуля. Разработка входящих модулей. |  |
| 21.05.2021 | Разработка входящих модулей. |  |
| 22.05.2021 | Разработка входящих модулей. Тестирование и отладка. Разработка документации. |  |
| 22.05.2021 | Разработка документации. Защита проекта. |  |
| 24.05.2021 | Защита проекта. Сдача зачёта по практике. |  |