Государственное бюджетное образовательное учреждение высшего образования
Московской области
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
имени дважды Героя Советского Союза, летчика-космонавта А.А. Леонова

Колледж космического машиностроения и технологий

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:
Митюшин П. А. и Слепов А. Д.
Группа: П1-18
Преподаватель:
Гусятинер Л. Б.

Королев 2021

Содержание отчёта

Раздел 1. Техника решения задач с использованием структурного программирования

1.1. Установка интерпретатора Python 3 и настройка окружения

Открываем браузер и переходим на страницу официального сайта python: https://www.python.org/ на главной странице будет кнопка download нажимаем на неё и под надписью Download the latest version for windows.

Запускаем установочный файл

После того, как вы выбрали установочный файл и загрузили его, просто запустите его двойным нажатием на загруженный файл. Затем открывается диалоговое окно, которое представлено на рис. 1:

Запускаем загрузочный файл и проходим процесс установки .

Рис. 1. Установщик Python

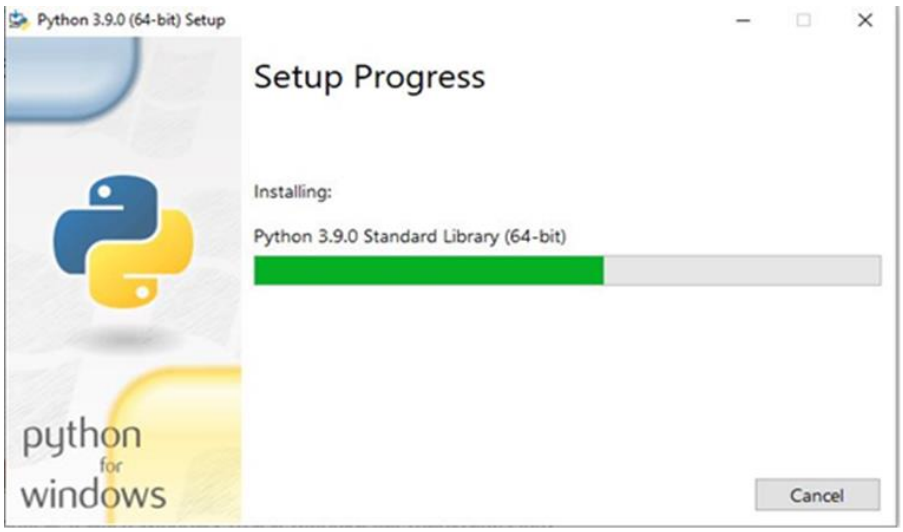


Рис. 2. Процесс установки

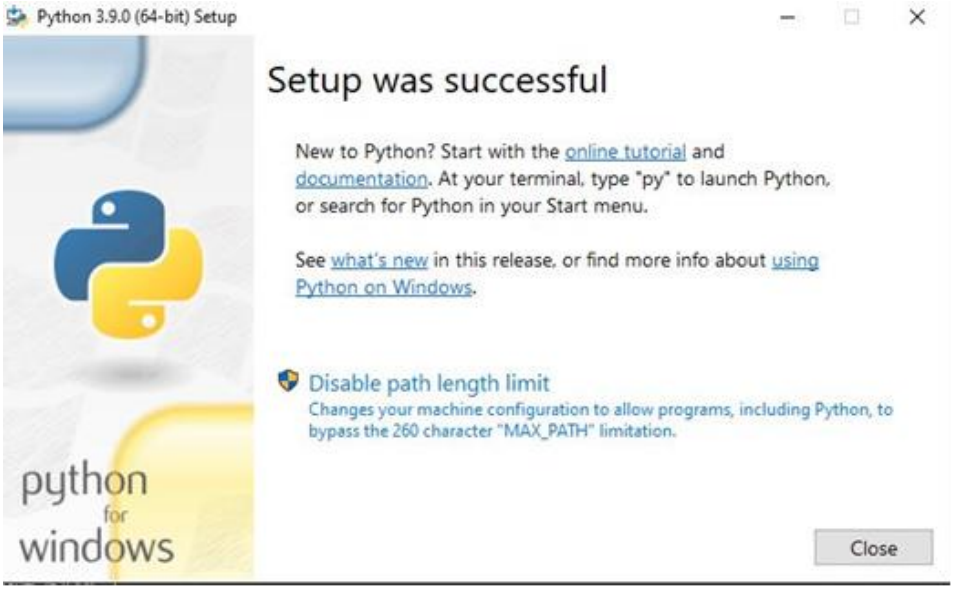


Рис. 3. Окно успешной установки

Процесс удаления описанный ниже нужен на случай, если была установлина 64 разрядная версия, а нужно было установить 32 разрядную версию python.

Рис. 4. Удаление python

Запускаем ранее использованный файл установки. Затем просто нажимаем на Uninstall и ждём завершения процесса удаления. На этом процесс удаления завершён.

Для Ubuntu установка проще. Просто открыв терминал вводим команду “sudo apt-get update”, а затем устанавливаем python “sudo apt-get install python”(Писать естественно без кавычек).

1.2. Техника работы в командной строке и среде IDLE

В PyCharm можно выбрать версию python когда вы в начале запускаете программу. При запуске файла в PyCharm, всплывает диалоговое окно как показанное на рис 1.

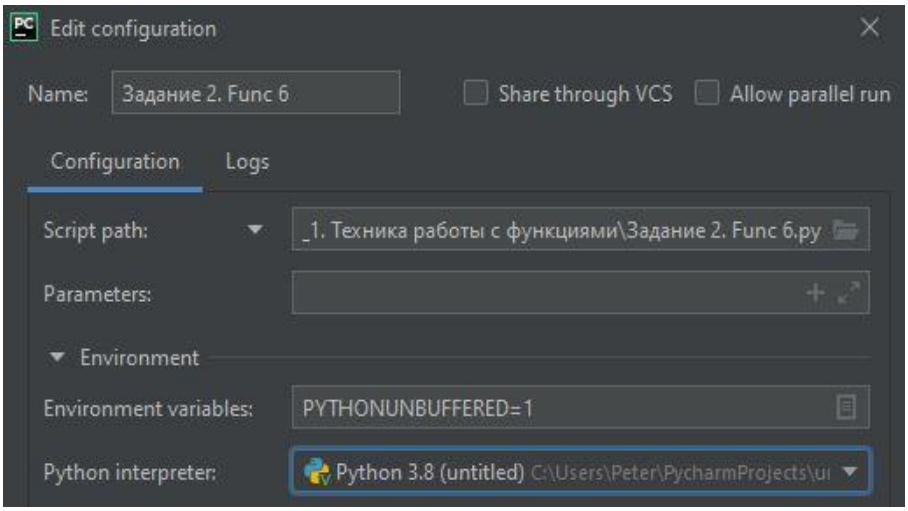


Рис. 1. Выбор интерпретатора

После того как выбрали версию языка как на рис. 1, затем нажимаем на внизу располагавшуюся кнопку run. После этого программа будет запушена на выбранной вами ранее версии python.

Выполняя команду “python” в вашем терминале, вы получаете интерактивную оболочку Python это показано на рис 2.

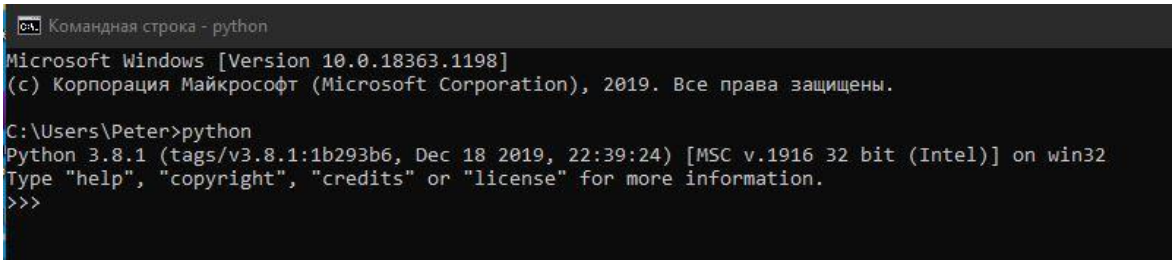


Рис. 2. Интерактивная оболочка python

IDLE - простой редактор для Python, который устанавливается вместе с Python. IDLE как правило находится в меню пуск. После того как вы нажмёте на IDEL в меню пуск то вы увидите окно показанное на рис 3.

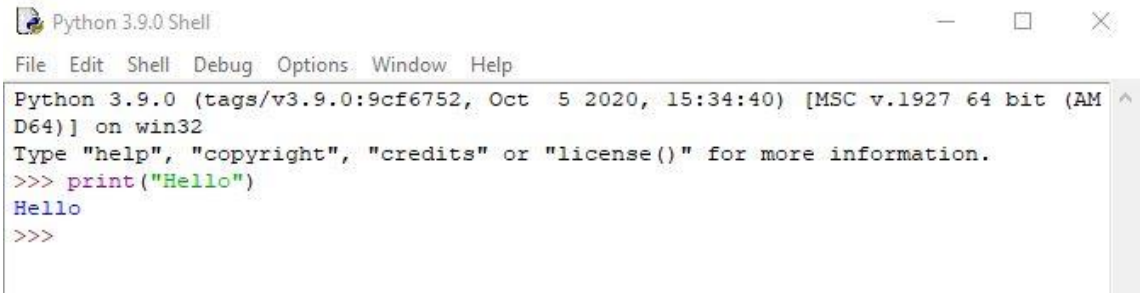


Рис. 3. Демонстрация работы IDEL

В оболочке есть подсказка из трех прямоугольных скобок:

```
>>>
```

Теперь напишите в подсказке следующий код рис. 3:

```
>>> print("Hello")
```

Нажмите Enter

```
>>> print("Hello")
Hello
```

Рис 4. Инициализация переменной

Также, если инициализировать переменную то можно вывести её прост напечатав название переменной и нажать на Enter как это показано на рис 4.

IDLE можно создавать файлы, где можно будет реализовать код. Затем их же можно и запустить. С верху надо нажать кнопку file -> new file затем нажимаем снова file -> save as…

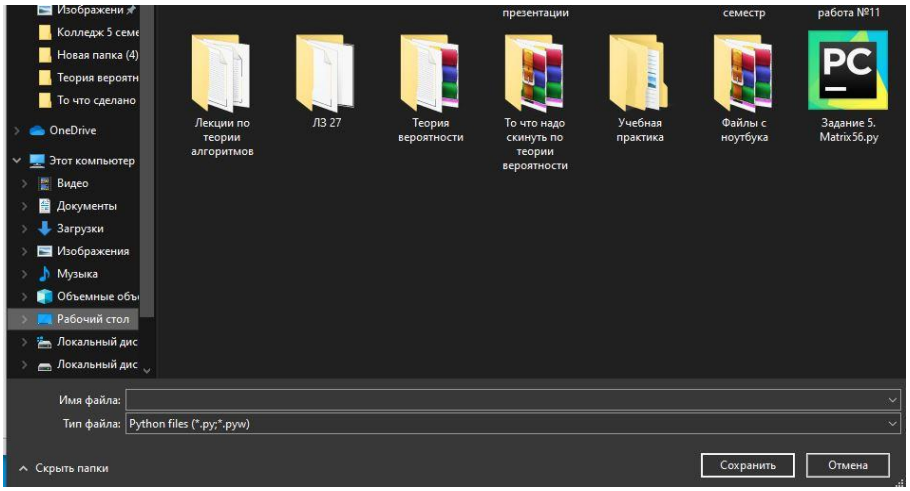


Рис. 5. Сохранение файла

Выбираем путь и в поле имя файла пишем названия файла нажимаем на кнопку сохранить и начинаем писать код пример на рис 5.

1.3 Техника работы с линейными и разветвляющимеся программами

Листинг 1. input.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание. Разработать программы по темам
- input
Функция input() в Python, ввод данных с клавиатуры.
'''
name = input('Enter your name: ')
print('Hello, ', name)
```

Листинг 2. print.py

```
'''
Выполнил: Митюшин Пётр. П1-18
- print
Функция print() в Python, печатает объект.
'''
import sys

print('Hello')
print("Hello")
objects = [1, 2, 3, 4, 5, 6, 7]
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=True)
#если True поток будет сброшен в указанный файл file принудительно.
#Значение по умолчанию False
print(*objects, sep='; ')
print('Hello, {0}, {1}!'.format(input(), input()))
print('{:f}!\n'.format(3.14))
```

Листинг 3. std.py, file.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
- stdin, stdout, stderr модуля sys
Объекты stdin, stdout, stderr модуля sys в Python.
'''
import sys
import time

stdout = sys.stdout
stderr = sys.stderr

lst = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
try:
    sys.stdout = open('file.txt', 'w')
```

```
    for i in lst:
        sys.stdout.write(i + '\n')
finally:
    sys.stdout.close()
    sys.stdout = stdout

def teleprint(*args, delay=0.05, str_join=' '):
    text = str_join.join(str(x) for x in args)
    n = len(text)
    for i, char in enumerate(text, 1):
        if i == n:
            char = f'{char}\n'
        sys.stdout.write(char)
        sys.stdout.flush()
        time.sleep(delay)

teleprint('Печать с задержкой!', 10, 12.5, 'Super!!!', delay=0.07)
```

Листинг 4. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Печать даты прописью
'''
date = input().split('.')

day_of_the_month = {'01': "Первое", '02': "Второе", '03': "Третье",
'04': "Четвётое", '05': "Пятое", '06': "Шестое", '07': "Седьмое",
'08': "Восьмое", '09': "Девятое", '10': "Десятое",
'11':"Одиннадцатое", '12': "Двенадцатое", '13': "Тринадцатое",
'14':"Четырнадцатое", '15': "Пятнадцатое", '16': "Шестнадцатое",
'17':"Семнадцатое", '18': "Восемнадцатое", '19': "Девятнадцатое",
'20': "Двадцатое", '21': "Двадцать первое", '22': "Двадцать второе",
'23': "Двадцать третье", '24': "Двадцать четвёртое", '25': "Двадцать
пятое", '26': "Двадцать шестое", '27': "Двадцать седьмое", '28':
"Двадцать восьмое", '29': "Двадцать девятое", '30': "Тридцатое", '31':
"Тридцать первое"}

month = {'01': "января",'02': "февралья",'03': "марта",'04':
"апрелья",'05': "майя",'06': "июнья",'07': "июлья",'08':
"августа",'09': "сентября",'10': "октября",'11': "ноября",'12':
"декабря"}

day_in_years = {'1': "первого",'2': "второго",'3': "третьего",'4':
"четвётого",'5': "пятого",'6': "шестого",'7': "седьмого",'8':
"восьмого",'9': "девятого"}

year_list_dec = \
['','десять','двадцать','тридцать','сорок','пятьдесят','шестьдесят',
```

```
'семьдесят','восемьдесят','девяносто']
year_list_dec2 = ['','десятого', 'двадцатого', 'тридцатого',
'сорокового', 'пятидесятого', 'шестидесятого', 'семидесятого',
'восемидесятого', 'девяностого']
year_list_unit = ['', 'первого', 'второго', 'третьего', 'четвёртого',
'пятого', 'шестого', 'седьмого', 'восьмого', 'девятого',
'одиннадцатого', 'двенадцатого', 'тринадцатого', 'четырнадцатого',
'пятнадцатого','шестнадцатого', 'семнадцатого', 'восемнадцатого',
'девятнадцатого']


year_list_hun = ['','сто','двести','триста','четыреста',
'пятьсот','шестьсот','семьсот','восемьсот','девятьсот']

year_list_hun2 = ['',
'сотого','двухсотого','трёхсотого','четырёхсотого','пятьсотого',
'шестьсотого','семьсотого','восьмисотого','девятьсотого']

year_list_th = ['','тысяча','две тысячи','три тысячи']

year_list_th2 = {'1': 'тысячного','2': 'двух тысячного',
'3': 'трёх тысячного'}

def definition_years(string):
    if (len(string) == 1):
        print(day_in_years[string[0]] + " года")
    elif (len(string) == 2):
        if string[1] == "0":
            print(year_list_dec2[int(string[0])],  "года")
        elif string[0] == "1":
            print(year_list_unit2[int(string[1])], "года")
        else:
            print(year_list_dec[int(string[0])],
day_in_years[string[1]],  "года")
    elif(len(string) == 3):
        if (string[0] != "0" and string[1] == "0" and string[2] == \
"0"):
            print(year_list_hun2[int(string[0])],  "года")
        elif (string[0] != "0" and string[1] != "0" and string[2] == \
"0"):
            print(year_list_hun[int(string[0])],
year_list_dec2[int(string[1])],  "года")
        else:
            print(year_list_hun[int(string[0])],
year_list_dec[int(string[1])], day_in_years[string[2]], "года")
    else:
        if string[1] == "0" and string[2] == '0' and string[3] == '0':
            print(year_list_th2[string[0]], "года")
        elif string[1] != "0" and string[2] == '0' and string[3] == \
'0':
```

```
            print(year_list_th2[string[0]],
year_list_hun2[int(string[1])], "года")
        else:
            print(year_list_th[int(string[0])], end=" ")# первая цифра
#в году
            if (string[1] == string[2] == string[3]):
                print(year_list_hun[int(string[1])], end=" ")
            else:
                print(year_list_hun[int(string[1])], end=" ")# вторая
#цифра в году

            if string[1] != "0" and string[3] == '0':
                print(year_list_dec2[int(string[2])], "года",end=" ")
#третья цифра в году

            elif string[1] == "0" and string[3] == '0':
                print(year_list_dec2[int(string[2])], "года")# третья
#цифра в году

            elif string[1] != string[2] == string[3]:
                print(year_list_unit2[int(string[3])],"года", end="")

            elif string[2] == "1" and string[3] != "0":
                print(year_list_unit2[int(string[3])],"года", end="")
#четвёртая цифра в году

            else:
                print(year_list_dec[int(string[2])],
day_in_years[string[3]],"года", end=" ")

date = input().split('.')

day = day_of_the_month[date[0]]
mon = month[date[1]]
print(day, mon, end=" ")
definition_years(date[2])
```

Листинг 5. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. Разработать программу с меню для демонстрации работы с
типами данных:
список(list), словарь(dict), множество(set)
Меню -> выбор типа данных -> выбор метода -> краткая справка
'''
def work_with_list(): # Списки
    print("1. append")
    print("2. extend")
    print("3. insert")
```

```
print("4. remove")
print("5. pop")
print("6. index")
print("7. count")
print("8. sort")
print("9. reverse")
print("10. copy")
print("11. clear")
print("exit")
while (True):
    com = input()
    if com == "1":
        print("list.append(x)")
        print("Добавляет элемент в конец списка")
    elif com == "2":
        print("list.extend(L)")
        print("Расширяет список a, ", end="")
        print("добавляя в конец все элементы списка b", end="")
    elif com == "3":
        print("list.insert(i, x)")
        print("Вставляет на i-ый элемент значение x")
    elif com == "4":
        print("list.remove(x)")
        print("Удаляет первый элемент в списке,имеющий значение")
        print("x. ValueError, ", end="")
        print("если такого элемента не существует", end="")
    elif com == "5":
        print("list.pop([i])")
        print("Удаляет i-ый элемент и возвращает его. ",end="")
        print("Если индекс не указан, ",end="")
        print("удаляется последний элемент", end="")
    elif com == "6":
        print("list.index(x, [start [, end]])")
        print("Возвращает положение первого элемента", end="")
        print("со значением x", end="")
        print("(при этом поиск ведется от start до end)", end="")
    elif com == "7":
        print("list.count(x)")
        print("Возвращает количество элементов со значением x")
    elif com == "8":
        print("list.sort([key=функция])")
        print("Сортирует список на основе функции")
    elif com == "9":
        print("list.reverse()")
        print("Разворачивает список")
    elif com == "10":
        print("list.copy()")
        print("Поверхностная копия списка")
    elif com == "11":
        print("list.clear()")
```

```
            print("Очищает список")
        elif com == "exit":
            main()
def work_with_dict(): # Словари
    print("1. clear")
    print("2. copy")
    print("3. fromkeys")
    print("4. get")
    print("5. items")
    print("6. keys")
    print("7. pop")
    print("8. popitem")
    print("9. setdefault")
    print("10. update")
    print("11. values")
    print("exit")
    while (True):
        com = input()
        if com == "1":
            print("dict.clear()")
            print("Очищает словарь.")
        elif com == "2":
            print("dict.copy()")
            print("Возвращает копию словаря.")
        elif com == "3":
            print("classmethod dict.fromkeys(seq[, value])")
            print("Создает словарь с ключами из seq ", end="")
            print("и значением value (по умолчанию None).", end="")
        elif com == "4":
            print("dict.get(key[, default])")
            print("Возвращает значение ключа, ", end="")
            print("но если его нет, не бросает исключение, ", end=""))
            print("а возвращает default (по умолчанию None).",end=""))
        elif com == "5":
            print("dict.items()")
            print("Возвращает пары (ключ, значение).")
        elif com == "6":
            print("dict.keys()")
            print("Возвращает ключи в словаре.")
        elif com == "7":
            print("dict.pop(key[, default])")
            print("Удаляет ключ и возвращает значение.", end="")
            print("Если ключа нет, возвращает default", end="")
            print("по умолчанию бросает исключение).", end="")
        elif com == "8":
            print("dict.popitem()")
            print("Удаляет и возвращает пару ")
            print("(ключ, значение). Если словарь пуст, ", end=""))
            print("бросает исключение KeyError. ", end=""))
```

```
            print("Помните, что словари неупорядочены.", end=""))
        elif com == "9":
            print("dict.setdefault(key[, default])")
            print("Возвращает значение ключа, ", end="")
            print("но если его нет, не бросает исключение,", end="")
            print(а создает ключ с значением default", end="")
            print("по умолчанию None).", end="")
        elif com == "10":
            print("dict.update([other])")
            print("Обновляет словарь, добавляя пары (ключ, значение)")
            print("из other", end="")
            print("Существующие ключи перезаписываются.", end="")
            print("Возвращает None (не новый словарь!).", end="")
        elif com == "11":
            print("dict.values()")
            print("Возвращает значения в словаре.")
        elif com == "exit":
            main()
def work_with_set(): #Множества
    #a = set()
    print("1. update")
    print("2. intersection_update")
    print("3. difference_update")
    print("4. symmetric_difference_update")
    print("5. add")
    print("6. remove")
    print("7. discard")
    print("8. pop")
    print("9. clear")
    print("10. copy")
    print("exit")
    while (True):
        com = input()
        if com == "1":
            print("set.update(other, ...); set |= other | ...")
            print("Объединение")
        elif com == "2":
            print("set.intersection_update(other, ...)
            print("set &= other & ...")
            print("Пересечение")
        elif com == "3":
            print("set.difference_update(other, ...)
            print("set -= other | ...")
            print("Вычитание")
        elif com == "4":
            print("set.symmetric_difference_update(other);")
            print("set ^= other", end="")
            print("Множество из элементов", end="")
            print("Встречающихся в одном множестве, ", end="")
```

```
            print("но не встречающиеся в обоих", end=""))
        elif com == "5":
            print("set.add(elem)")
            print("Добавляет элемент в множество.")
        elif com == "6":
            print("set.remove(elem)")
            print("Удаляет элемент из множества. ", end="")
            Print("KeyError, если такого элемента не существует")
       elif com == "7":
            print("set.discard(elem)")
            print("Удаляет элемент, если он находится в множестве")
        elif com == "8":
            print("set.pop()")
            print("Удаляет первый элемент из множества.", end=""))
            print("Так как множества не упорядочены", end="")
            print("Нельзя точно сказать, ", end="")
            print("какой элемент будет первым.", end=""))
        elif com == "9":
            print("set.clear()")
            print("Очистка множества")
        elif com == "10":
            print("set.copy() ")
            print("Копия множества")
        elif com == "exit":
            main()
def main():
    print("1. list")
    print("2. dict")
    print("3. set")
    com = input('Enter your choice: ')
    if com == "1":
        work_with_list()
    elif com == "2":
        work_with_dict()
    elif com == "3":
        work_with_set()
if __name__ == "__main__":
    main()
```

1.4. Техника работы с циклическими программами цикл while

Листинг 1. Задание 1.py

```
'''
Выполнил: Слепов Андрей. П1-18
К5_1 Техника работы с циклическими программами _ цикл while

Задание 1. На плоскости нарисован квадрат заданного размера с левой
нижней
вершиной в начале координат. В квадрат вписывается окружность.
Случайным образом в квадрате выбирается 1000 точек.
а) нужно определить, сколько точек попало внутрь круга
б) считая количество точек пропорциональным площади, найти отношение
площадей
круга и квадрата
в) по этому отношению определить приближённое значение числа пи
г) определить, насколько найденное значение отличается от
"библиотечного".
'''
import math
import random

NumberOfPoints = 1000

print("Введите сторону квадрата")
a = int(input())

Ssq = a
radius = a // 2
#Scr = math.pi() * radius1

points = []

for i in range(2):
    points.append([])
    for j in range(NumberOfPoints):
        elem = random.randint(-a/2, a/2)
        points[i].append(elem)

count = 0
for i in range(NumberOfPoints):
    if math.sqrt((points[0][i]**2 - 0) + (points[1][i]**2 - 0)) <= radius:
        count += 1

area_ratio = count / NumberOfPoints
pi = area_ratio * 4
dif_pi = math.pi - pi
print("Кол-во точек попавших внутрь круга: ", count)
print("Отношение площади круга и квадрата: ", area_ratio)
print("приближённое значение пи относительно наших вычислений", pi)
print("Разница между нашим числом пи и библиотечным", dif_pi)
```

Листинг 2. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К5_1 Техника работы с циклическими программами _ цикл while
Напишите программу, которая считывает целые числа с консоли по одному
числу в строке.

Для каждого введённого числа проверить:
если число меньше 10, то пропускаем это число;
если число больше 100, то прекращаем считывать числа;
в остальных случаях вывести это число обратно на консоль в отдельной
строке.
'''
while True:
    num = int(input())
    if num < 10:
        continue
    elif num > 100:
        break
    else:
        print(num)

'''
Найти букву и вывести её номер
'''
string = input()
find_letter = input()
for i in range(string):
    if (string[i] == find_letter):
        print(i)
        break
else:
    print("Буква в строке не найдена")
```

Листинг 3. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К5_2 Техника работы с циклическими программами _ цикл while

Задание 1. Вычислить значение sin(x) с точностью до epsilon при помощи
разложения в ряд
Построить блок-схему
'''
import math

x = sin(float(input("x: ")))
epsilon = float(input("epsilon: "))
sign = 1
summ = 0
k = 3
res = (x**k)/math.factorial(k)
summ = x - res
while(res > epsilon):
    k += 2
    res = (x**k)/math.factorial(k)
    summ += sign * res
    sign *= -1
print(summ)
```

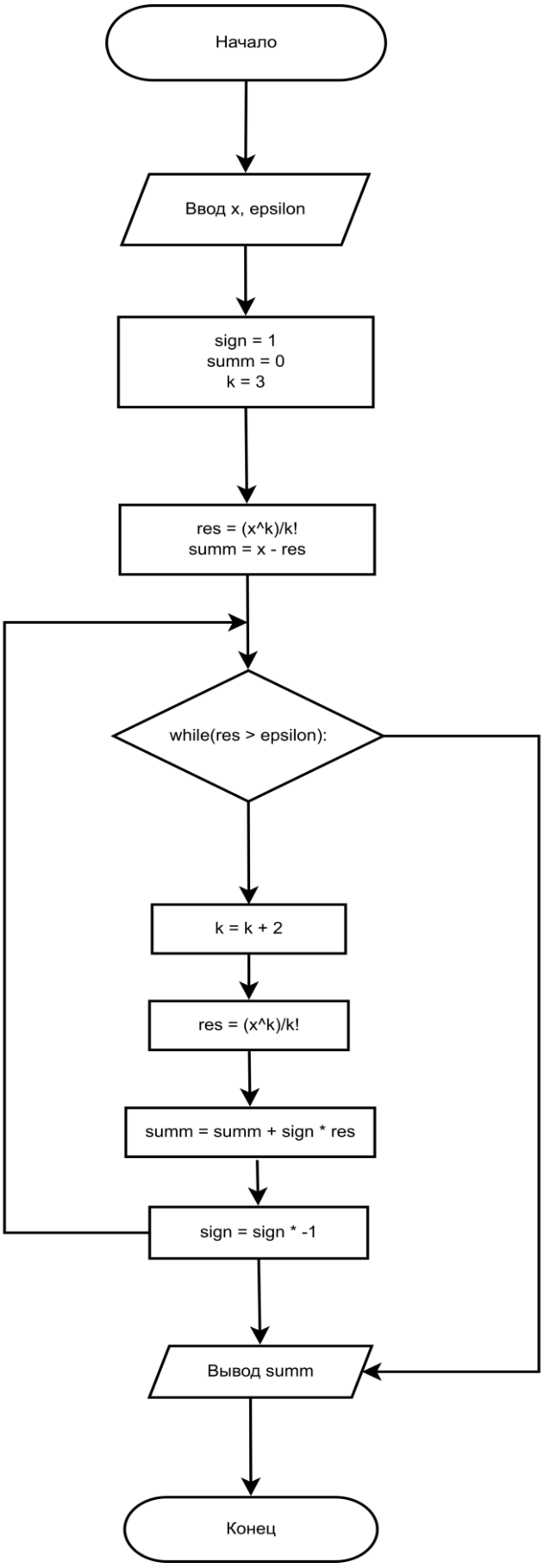


Рис 1. Блок-схема Задания 1

Листинг 4. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К5_2 Техника работы с циклическими программами _ цикл while

Задание 2. Напишите программу, которая считывает со стандартного ввода целые числа, по одному числу
в строке, и после первого введенного нуля выводит сумму полученных на вход чисел.
'''
def main():
    a = int(input())
    summ = 0
    while a != 0:
        summ = summ + a
        a = int(input())
    print(summ)

if __name__ == "__main__":
    main()
```

Листинг 5. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К5_2 Техника работы с циклическими программами _ цикл while

Задание 3. Разработать программу для нахождения наибольшего общего делителя
'''
def gcd(n, m):
    while (n != 0 and m != 0):
        if (n >= m):
            n = n % m
        else:
            m = m % n

    return n + m

def main():
    a, b = map(int, input().split())
    print(gcd(a, b))

if __name__ == "__main__":
    main()
```

Листинг 6. Задание4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К5_2 Техника работы с циклическими программами _ цикл while
```

```
Задание 4. С использованием результата задания 2 разработать программу
для нахождения наименьшего общего кратного
'''
def main():
    a, b = map(int, input().split())
    m = a * b
    while (n != 0 and m != 0):
        if (a > b):
            a = a % b
        else:
            b = b % a
    print(m // (a + b))

if __name__ == "__main__":
    main()
```

Листинг 7. Задание5.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К5_2 Техника работы с циклическими программами _ цикл while
Задание 5. Напишите программу, которая выводит часть
последовательности 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 ...
(число повторяется столько раз, чему равно).
На вход программе передаётся неотрицательное целое число n — столько
элементов
последовательности должна отобразить программа.
На выходе ожидается последовательность чисел, записанных через пробел
в одну строку.
Например, если n = 7, то программа должна вывести 1 2 2 3 3 3 4.
'''
n = int(input())
x = 0
i = 0
j = 0
if n == 1:
    print(1)
else:
    while (i < n):
        i += 1
        j = 0
        while (j < i):
            if x == n:
                break
            else:
                print(i, end=' ')
                x += 1
            j += 1
```

1.5. Техника работы с числами.

Листинг 1. Задание1.py

```
'''
Задание 1
Выполнил: Митюшин Пётр. П1-18
К6_1. Техника работы с числами
Составить и выполнить по 3 примера использования модулей для работы
с дробными числами (fractions), для точных вычислений (decimal).
'''
from fractions import *
from decimal import Decimal, ROUND_HALF_EVEN

number = Decimal("0.1")
number = number + number + number
print(number)

number = Decimal("0.69")
number = number * number
print(number)

a = Decimal("0.85843")
a = a.quantize(Decimal("1.00")) #quantize позволяет округлять числа
print(a)
b = Decimal("0.85843") + Decimal("0.69")
print(b.quantize(Decimal("1.00"), ROUND_HALF_EVEN))
print("-----------")

a = Fraction(1, 7)
b = Fraction(1, 3)
print(a + b)
print(a - b)
print(a / b)
print(a % b)
print(a ** b)
a.limit_denominator()
print(Fraction('3.14159265359').limit_denominator(1000))
print(Fraction('3.14159265359').limit_denominator(100))
print(Fraction('3.14159265359').limit_denominator(10))
print(Fraction('3.14159265359').limit_denominator(1))
print(Fraction(1, 4))
```

Инструкцию по использованию модулей fractions, decimal.

```
'''
Задание 2
Выполнил: Митюшин Пётр. П1-18
К6_1. Техника работы с числами
Подготовить инструкцию по использованию модулей fractions, decimal.
'''
```

Модуль fractions

Для начала в заголовке файла from fractions import,from decimal import *. Модуль fractions предоставляет поддержку рациональных чисел.

class fractions.Fraction(numerator=0, denominator=1)

class fractions.Fraction(other_fraction)

class fractions.Fraction(float)

class fractions.Fraction(decimal)

class fractions.Fraction(string)

Класс, представляющий собой рациональные числа. Экземпляр класса можно создать из пары чисел (числитель, знаменатель), из другого рационального числа, числа с плавающей точкой, числа типа decimal. Decimal, и из строки, представляющей собой число.

Fraction.limit_denominator(max_denominator=1000000) - ближайшее рациональное число со знаменателем не больше данного.

Также, помимо класса рациональных чисел, модуль fractions предоставляет функцию для нахождения наибольшего общего делителя.

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

Модуль decimal

Ключевым компонентом для работы с числами в этом модуле является класс Decimal. Для его применения нам надо создать его объект с помощью конструктора. В конструктор передается строковое значение, которое представляет число:

Листинг 2. Задание1.py

```
from decimal import Decimal
number = Decimal("0.1")
```

После этого объект Decimal можно использовать в арифметических операциях:

Листинг 3. Задание1.py

```
from decimal import Decimal
number = Decimal("0.1")
number = number + number + number
print(number)
```

В операциях с Decimal можно использовать целые числа:

Листинг 4. Задание1.py

```
number = Decimal("0.1")
number = number + 2
```

Однако нельзя смешивать в операциях дробные числа float и Decimal:

Листинг 5. Задание1.py

```
number = Decimal("0.1")
number = number + 0.1   # здесь возникнет ошибка
```

С помощью дополнительных знаков мы можем определить, сколько будет символов в дробной части числа:

Листинг 6. Задание1.py

```
number = Decimal("0.10")
number = 3 * number
print(number)       # 0.30
```

Строка "0.10" определяет два знака в дробной части, даже если последние символы будут представлять ноль. Соответственно "0.100" представляет три знака в дробной части.

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:

Листинг 7.

```
from decimal import Decimal
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
```

Используемая строка "1.00" указывает, что округление будет идти до двух знаков в дробной части.

По умолчанию округление описывается константой ROUND_HALF_EVEN, при котором число округляется в большую сторону, если оно нечетное, а предыдущее перед ним больше 4.

Листинг 8. Задание1.py

```
from decimal import Decimal, ROUND_HALF_EVEN
number = Decimal("10.025")
print(number.quantize(Decimal("1.00"), ROUND_HALF_EVEN))# 10.02
number = Decimal("10.035")
print(number.quantize(Decimal("1.00"), ROUND_HALF_EVEN))# 10.04
```

Можно переопределить, использовав одну из следующих констант:

ROUND_HALF_UP: округляет число в сторону повышения, если после него идет число 5 или выше

ROUND_HALF_DOWN: округляет число в сторону повышения, если после него идет число больше 5

Листинг 9. Задание1.py

```
number = Decimal("10.026")
print(number.quantize(Decimal("1.00"), ROUND_HALF_DOWN))# 10.03
number = Decimal("10.025")
print(number.quantize(Decimal("1.00"), ROUND_HALF_DOWN))# 10.02
```

ROUND_05UP - округляет только 0 до единицы, если после него идет 5

Листинг 10. Задание1.py

```
number = Decimal("10.005")
print(number.quantize(Decimal("1.00"), ROUND_05UP)) # 10.01
number = Decimal("10.025")
print(number.quantize(Decimal("1.00"), ROUND_05UP))# 10.02
```

ROUND_CEILING - округляет число в большую сторону вне зависимости от того, какое число идет после него

Листинг 11. Задание1.py

```
number = Decimal("10.021")
print(number.quantize(Decimal("1.00"), ROUND_CEILING))# 10.03
number = Decimal("10.025")
print(number.quantize(Decimal("1.00"), ROUND_CEILING))# 10.03
```

ROUND_FLOOR - не округляет число вне зависимости от того, какое число идет после него.

Листинг 12. Задание1.py

```
number = Decimal("10.021")
print(number.quantize(Decimal("1.00"), ROUND_FLOOR))# 10.02
number = Decimal("10.025")
print(number.quantize(Decimal("1.00"), ROUND_FLOOR))# 10.02
```

Листинг 13. Задание 1 cmath.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К6_2. Техника работы с числами
'''
import cmath

x = int(input())
print(cmath.polar(x))

r = int(input())
phi = int(input())
print(cmath.rect(r, phi))
print(cmath.exp(x))

y = int(input())
print(cmath.log(x, y))
print(cmath.log10(x))
print(cmath.sqrt(x))
print(cmath.acos(x))
print(cmath.asin(x))
print(cmath.atan(x))
print(cmath.cos(x))
print(cmath.sin(x))
print(cmath.tan(x))
print(cmath.acosh(x))
print(cmath.asinh(x))
print(cmath.atanh(x))
print(cmath.cosh(x))
print(cmath.sinh(x))
print(cmath.tanh(x))
print(cmath.isfinite(x))
print(cmath.isinf(x))
print(cmath.isnan(x))
print(cmath.pi)
print(cmath.e)
```

Листинг 14. Задание 2 math.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К6_2. Техника работы с числами
'''
import math

x = int(input())
print(math.ceil(x))

y = int(input())
print(math.copysign(x, y))
print(math.fabs(x))

print(math.factorial(x))
print(math.floor(x))
print(math.fmod(x, y))
print(math.frexp(x))

i = int(input())
print(math.ldexp(x, i))
print(math.fsum(1, 2, 3, 4, 5, 6))
print(math.isfinite(x))
print(math.isinf(x))
print(math.isnan(x))
print(math.modf(x))
print(math.trunc(x))
print(math.exp(x))
print(math.expm1(x))
print(math.log(x, y))
print(math.log1p(x))
print(math.log10(x))
print(math.log2(x))
print(math.pow(x, y))
print(math.sqrt(x))
print(math.acos(x))
print(math.asin(x))
print(math.atan(x))
print(math.atan2(y, x))
print(math.cos(x))
print(math.sin(x))
print(math.tan(x))
print(math.hypot(x, y))
print(math.degrees(x))
print(math.radians(x))
print(math.cosh(x))
print(math.sinh(x))
print(math.tanh(x))
print(math.acosh(x))
print(math.asinh(x))
print(math.atanh(x))
```

```
print(math.erf(x))
print(math.erfc(x))
print(math.gamma(x))
print(math.lgamma(x))
print(math.pi)
print(math.e)
```

1.6. Техника работы со строками

Листинг 1. Задание 1.py

```
"""
Выполнил: Митюшин Пётр. П1-18
К7_1. Техника работы со строками
С клавиатуры вводятся строки, последовательность заканчивается точкой.
Выведите буквы введенных слов в верхнем регистре, разделяя их
пробелами.
"""

print_string = ""
elem = input()
while (elem != "."):
    for i in range(0, len(elem)):
        if (i < len(elem) - 1):
            print_string += elem[i].upper() + ' '
        else:
            print_string += elem[i].upper()
    print(print_string)
    print_string = ""
    elem = input()
```

Листинг 2. Задание 2.py

```
"""
Выполнил: Митюшин Пётр. П1-18
К7_1. Техника работы со строками
Известно, что для логина часто не разрешается использовать строки
содержащие пробелы.
Но пользователю нашего сервиса особенно понравилась какая-то строка.
Замените пробелы в строке на символы нижнего подчеркивания, чтобы
строка
могла сгодиться для логина. Если строка состоит из одного слова,
менять ничего не нужно.
"""

string = str(input())
string = string.replace(' ', '_')
print(string)
```

Листинг 3. Задание 3.py

```
"""
Выполнил: Митюшин Пётр. П1-18
К7_1. Техника работы со строками
Уберите точки из введенного IP-адреса. Выведите сначала четыре числа
через пробел,
а затем сумму получившихся чисел.
"""

n = input().split('.')
per = int(n[0]) + int(n[1]) + int(n[2]) + int(n[3])
c = int(n[0]), int(n[1]), int(n[2]), int(n[3])
print(n[0], n[1], n[2], n[3])
print(per)
```

Листинг 4. Задание 4.py

```
"""
Выполнил: Митюшин Пётр. П1-18
К7_1. Техника работы со строками
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.
"""

string = input()
while (string != "."):
    if (string[0] == "!" and string[1] == "!"):
        print("предупреждение")
    elif (string[0] == "@" and string[1] == "@"):
        print("ошибка")
    elif (string[0] == "/" and string[1] == "/"):
        print("информация")
    elif (string[0] == "*" and string[1] == "*"):
        print("подробное сообщение")
    string = input()
```

Инструкцию по использованию. Форматирования строк “По старому”.

Листинг 5. File1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К7_2. Техника работы со строками
'''
name = "Peter"
print('Hello, %s' % name) # Вывод: "Hello, Peter"
a = 16
print('a: %d' %a) # Вывод: 16
```

%d или %s – Означает, где именно заменить значение имени, представленного в виде строки. %d – для вывода чисел, %s – для вывода строи.

Листинг 6. File2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
'''
errno = 50159747054
print('%x' % errno) # Вывод: 'badc0ffee'
```

%x для конвертации значения int в строку и представить его в качестве шестнадцатеричного числа.

Листинг 7. File3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К7_2. Техника работы со строками
'''
er = 5
name = 'Andrey'
print('Hey %s, there is a 0d% error!' % (name, er)) # 'Hey Andrey, there
#is a 5 error!'
```

Вывод нескольких переменных.

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

Форматирование строк обрабатывается вызовом .format() в строке.

Листинг 8. File4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К7_2. Техника работы со строками
'''
print('Hello, {}'.format(name)) # Вывод: 'Hello, Andrey'
print('Hey {name}, there is a 0x{errno:x} error!'.format(name=name,
errno=errno)
# Вывод: 'Hey Andrey, there is a 0xbadc0ffee error!'
```

Интерполяция строк. f-Строки.

Python 3.6 Добавил новый подход форматирования строк под названием форматированные строчные литералы, или “f-строки”. Этот новый способ форматирования строк позволяет вам использовать встроенные выражения Python внутрь строковых констант.

Листинг 9. File5.py

```
'''
Выполнил: Митюшин Пётр. П1-18
'''
name = 'Peter'
print(f'Hello, {name}!')
# Вывод: 'Hello, Peter!'
a = 5
b = 10
print(f'Five plus ten is {a + b} and not {2 * (a + b)}.')
def greet(name):
    return f"Hello, {name}!
print(printHi ('Bob'))
```

Передавать значения можно так же и в функцию. После передачи значений, переданные строки соединяются и составляют итоговую строку.

Листинг 10. File6.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К7_2. Техника работы со строками
'''
Name = 'Peter'
print(f"Hey {name}, there's a {er:#x} error!")
# Вывод: "Hey Peter, there's a 0xbadc0ffee error!"
```

Шаблонные строки (Стандартная библиотека Template Strings)

Template strings – это более простой и менее мощный механизм, но в ряде случаев он может быть именно тем, что вам нужно.

Листинг 11. File7.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К7_2. Техника работы со строками
'''
from string import Template
t = Template('Hey, $name!')
print(t.substitute(name=name)) # Вывод: 'Hey, Andrey!'

templ_string = 'Hey $name, there is a $error error!'
print(Template(templ_string).substitute(name=name, error=hex(errno)))
# Вывод: 'Hey Andrey, there is a 0xbadc0ffee error!'
templ_string = 'Hey $name, there is a $error error!'
```

1.7. Техника работы со списками

Листинг 1. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К8_1. Техника работы со списками
'''
lst = input().split()
count = 0
for i in range(2, len(lst)):
    if (int(lst[i - 2]) < int(lst[i - 1]) > int(lst[i])):
        count += 1
print(count)
```

Листинг 2. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К8_1. Техника работы со списками
'''
count = 0
lst = list(map(int, input().split()))
for i in range(len(lst)):
    for j in range(i+1, len(lst)):
        if lst[i] == lst[j]:
            count += 1
print(count)
```

Листинг 3. Задание 3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К8_1. Техника работы со списками
'''
import random

def search_equal_items(lst):
    print(list(set(lst)))

lst = []
n = int(input())
for i in range(n):
    lst.clear()
    m = random.randint(5, 10)
    for j in range(m):
        elem = random.randint(1, 50)
        lst.append(elem)
    search_equal_items(lst)
```

Листинг 4. Задание1.py

```
'''
Выполнил: Слепов Андрей
К8_2. Техника работы со списками
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.
'''

def bubble_sort(arr):
    for i in range(len(a) - 1):
        for j in range(len(a) - i - 1):
            if a[j] > a[j + 1]:
                a[j], a[j + 1] = a[j + 1], a[j]
                print(*a)
    return a


a = [int(i) for i in input().split()]
print(*bubble_sort(a))
```

Листинг 5. Задание2.py

```
'''
Выполнил: Слепов Андрей
К8_2. Техника работы со списками
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.
'''

def sort_sel(arr):
    i = 0
    while i < len(arr) - 1:
        m = i
        j = i + 1
        while j < len(arr):
```

```
            if arr[j] < arr[m]:
                m = j
            j += 1
        arr[i], arr[m] = arr[m], arr[i]
        i += 1
    return arr
a = [int(i) for i in input().split()]
print(*sort_sel(a))
```

Листинг 6. Задание3.py

```
'''
Выполнил: Слепов Андрей
К8_2. Техника работы со списками
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).
'''

def insertion(arr):
    for i in range(len(arr)):
        j = i - 1
        key = arr[i]
        while arr[j] > key and j >= 0:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr

a = [int(i) for i in input().split()]
print(*insertion(a))
```

1.8. Техника работы с циклом for и генераторами списков

Листинг 1. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К9_1. Техника работы с циклом for и генераторами списков
'''
import codecs

def find_name_command(find_command, group_command, lst):
    nc = 0
    index = 0
    find_command += '\r\n'
    for i in lst:
        #print(i)
        if (nc == 1 and enter_in_string(i)):
            break
        elif (nc == 1 and not enter_in_string(i)):
            print(i)
            group_command.append(i)
        if (find_command == i and nc == 0):
            group_command.append(i)
            nc += 1

def enter_in_string(string1):
    nc = 0
    for i in range(len(string1)):
        if (string1[i] == ":"):
            return False
    return True

with codecs.open("Конкурс проектов.txt", "r", "utf_8_sig") as file:
    lst = [i for i in file.readlines()]

students_and_position = [i for i in lst if not enter_in_string(i)]
possible_comm = [i for i in lst if enter_in_string(i)]

find_command = input()
group_command = []
find_name_command(find_command, group_command, lst)

print(students_and_position)
print(possible_comm)
print(group_command)
```

Листинг 2. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К9_2. Техника работы с циклом for и генераторами списков
Задание 1. Array55. 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.
'''
def main():
    a = list(map(int, input().split()))
    b = []
    for i in range(1, len(a), 2):
            b.append(int(a[i]))
    print("Размер массива: ", len(b))
    print(*b)

if __name__ == "__main__":
    main()
```

Листинг 3. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К9_2. Техника работы с циклом for и генераторами списков
Задание 2. Array57. Дан целочисленный массив A размера N. Переписать в
новый целочисленный массив B
того же размера вначале все элементы исходного массива с четными
номерами,
а затем — с нечетными:
A[0], A[2], A[4], A[6], ..., A[1], A[3], A[5], ... .
Условный оператор не использовать.
'''
a = []
    n = int(input())
    for i in range(n):
        elem = int(input())
        a.append(elem)
    b = []
    for i in range(1, len(a), 2):
        b.append(a[i])

    for i in range(0, len(a), 2):
        b.append(a[i])
    print(*b)

if __name__ == "__main__":
    main()
```

Листинг 4. Задание 3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К9_2. Техника работы с циклом for и генераторами списков
Задание 3. Array58. 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.
'''

def fillingZero(lst, b, n):
    b.append(lst[0])
    for i in range(1, n):
        b.append(0)

def main():
    b = []
    lst = []
    summ = 0
    n = int(input())
    for i in range(n):
        elem = float(input())
        lst.append(elem)
    fillingZero(lst, b, n)
    #b[0] = lst[0]
    for k in range(1, n):
        for j in range(k+1):
            b[k] += lst[j]
    print(b)
if __name__ == "__main__":
    main()
```

Листинг 5. Задание 4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К9_2. Техника работы с циклом for и генераторами списков
Задание 4. Matrix3. 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).
'''



n = int(input())
m = int(input())
mas = []
arr = []
```

```
for i in range(m):
    elem = int(input())
    arr.append(elem)

for i in range(m):
    mas.append([])
    for j in range(n):
        mas[i].append(arr[i])

print(mas)
```

Листинг 6. Задание 5.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К9_2. Техника работы с циклом for и генераторами списков
Задание 5. Matrix56. Дана матрица размера M x N (N — четное число).
Поменять местами
левую и правую половины матрицы.
'''

def swapMatrix(mas, a, b):
    for i in range(m):
        temp = mas[i][a]
        mas[i][a] = mas[i][b]
        mas[i][b] = temp

n = int(input())
m = int(input())
mas = []
for i in range(m):
    mas.append([])
    for j in range(n):
        elem = float(input())
        mas[i].append(elem)

for j in range(n//2):
    swapMatrix(mas, j, (n // 2) + j)
print(mas)
```

Листинг 7. Задание 6.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К9_2. Техника работы с циклом for и генераторами списков
Задание 6. Matrix88. Дана квадратная матрица порядка M. Обнулить
элементы матрицы,
лежащие ниже главной диагонали. Условный оператор не использовать.
'''
```

```
m = int(input())
mas = []
for i in range(m):
    mas.append([])
    for j in range(m):
        el = int(input())
        mas[i].append(el)

index_app = 0
for i in range(m):
    for j in range(m):
        if (j < index_app):
            mas[i][j] = 0
    index_app += 1

for i in range(m):
    for j in range(m):
        print(mas[i][j], end=" ")
    print()
```

1.9. Техника работы с функциями

Листинг 1. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К10_1. Техника работы с функциями
Задание 2. Func6. Описать функцию SumRange(A, B) целого типа,
находящую сумму всех целых
чисел от A до B включительно (A и B — целые). Если A > B, то функция
возвращает 0.
С помощью этой функции найти суммы чисел от A до B и от B до C, если
даны числа A, B, C.
'''
def SumRange(a, b, c):
    if a > b:
        return 0
    elif (a < b):
        summ = 0
        for i in range(a, b):
            summ += i
        for i in range(b, c+1):
            summ += i
        return summ
a = int(input())
b = int(input())
c = int(input())
print(SumRange(a, b, c))
```

Листинг 2. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К10_1. Техника работы с функциями
Задание 3. Func10. Описать функцию IsSquare(K) логического типа,
возвращающую True,
если целый параметр K (> 0) является квадратом некоторого целого
числа, и False
в противном случае. С ее помощью найти количество квадратов в наборе
из 10 целых
положительных чисел.
'''
import math
def IsSquare(K):
    result = math.sqrt(K)
    return (result % 1 == 0)

array = []
for i in range(11):
    elem = int(input())
    array.append(elem)
```

```
for i in range(11):
    print(IsSquare(array[i]))
```

Листинг 3. Задание 3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К10_1. Техника работы с функциями
Задание 4. Func33. Описать функцию SortInc3(X), меняющую содержимое
списка X из трех
вещественных элементов таким образом, чтобы их значения оказались
упорядоченными по
возрастанию (функция возвращает None). С помощью этой функции
упорядочить по
возрастанию два данных списка X и Y.
'''

def filling(X):
    for i in range(3):
        elem = int(input())
        X.append(elem)

def SortInc3(X):
    for i in range(3):
        for j in range(3):
            if (X[i] < X[j]):
                temp = X[i]
                X[i] = X[j]
                X[j] = temp
    return X
X = []
Y = []
filling(X)
filling(Y)
print(SortInc3(X))
print(SortInc3(Y))
```

Листинг 4. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К10_2. Техника работы с функциями
Задание 2. 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:
'''
lst = [i for i in map(int, iter(input, "."))]
p = list(map(lambda i: print(i ** 2, end=" "), lst[::-1]))
```

Листинг 5. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К10_2. Техника работы с функциями
Задание 3. Использовать lambda, filter.
Array55. Дан целочисленный массив A размера N (<= 15). Переписать в
новый целочисленный
массив B все элементы с нечетными порядковыми номерами (1, 3, ...) и
вывести размер
полученного массива B и его содержимое. Условный оператор не
использовать.
'''
n = int(input())
lst = [int(input()) for i in range(n)]
b = list(filter(lambda x: x % 2 != 0, lst))
print(b)
```

Листинг 6. Задание 3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К10_2. Техника работы с функциями
Задание 4. Использовать lambda, map.
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.
'''
lst = list(map(int, input().split()))
p = list(map(lambda i: print(i ** lst[2], end=" "), range(lst[0],
lst[1]+1)))
print(*p)
```

1.10. Техника работы со словарями

Листинг 1. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К11_1. Техника работы со словарями
Задание 1. 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.
'''
counter = {}
lst = input().split()
for i in lst:
    if i not in counter:
        counter[i] = 0
    else:
        counter[i] += 1
    print(counter[i], end=" ")
```

Листинг 2. Задание 2.py

```
'''
Выполнил: Слепов Андрей. П1-18
К11_1. Техника работы со словарями
Задание 2.Задача «Права доступа»
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 число 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.
'''
permission = {}
n = int(input())
for i in range(n):
    s = input().split()
    permission[s[0]] = s[1:]
for i in range(int(input())):
    perm, file = input().split()
    if perm == 'read':
        perm = 'R'
    if perm == 'write':
        perm = 'W'
    if perm == 'execute':
        perm = 'X'
    if perm in permission[file]:
        print('OK')
    else:
        print('Access denied')
```

Листинг 3. Задание 3.py

```
'''
Выполнил: Слепов Андрей. П1-18
К11_1. Техника работы со словарями
Задание 3. 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.
'''
dictt = {}
max = 0

for i in range(int(input())):
    for word in input().split():
        if word in dictt:
            dictt[word] += 1
        else:
            dictt[word] = 1
        if  dictt[word] > max:
            max = dictt[word]

for key, value in sorted(dictt.items()):
    if value == max:
        print(key)
        break
```

Листинг 4. Задание1.py

```
'''
Выполнил: Слепов Андрей. П1-18
К11_2. Техника работы со словарями
Задание 1. Телефонная книга. Этап 1. 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.
'''
phone_num = {}
for string in i(input, "."):
    for i in range(len(string)):
        if string[i] == " ":
            name = string[0:i]
            number = string[i+1:len(string)]
            phone_num[name] = number
    if (string.isalpha()):
        print(phone_num[string])
```

Листинг 5. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
К11_2. Техника работы со словарями
Задание 2. Телефонная книга. Этап 2.     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".
'''
data = input()
phone_book = {}
while data != '.':
    data = data.replace(',','').split()
    if len(data) == 1:
        name = ''.join(data)
```

```
        if name in phone_book:
            print(', '.join(phone_book[name]))
        else:
            print("Не найдено" )
    else:
        name, number = data[0],data[1:]
        phone_book[name] = phone_book.get(name,[]) + number
    data = input()
```

Листинг 6. Задание 3.ру

```
'''
Выполнил: Митюшин Пётр. П1-18
К11_2. Техника работы со словарями
Задание 3. Телефонная книга. Этап 3. Коле очень понравилась Ваша
программа, однако он стал
замечать, что иногда в его телефонную книгу попадают номера в
некорректном формате.
Чтобы не сохранять недействительные номера, он попросил Вас
обрабатывать только номера,
соответствующие критериям:
- номер должен начинаться либо с +7, либо с 8 и состоять из 11 цифр.
- блоки цифр могут разделяться пробелами или дефисами.
- вторая, третья и четвертая цифры могут помещаться в скобки.
Если программа встречает некорректный номер, она должна его
проигнорировать. В обратном
случае она должна привести номер к виду +7 (900) 800-70-60 и
запомнить. Остальной
функционал программы остается без изменений.
'''
def check_number(n):
    symbols = '+- ()1234567890' # То что должно содержаться в номере телефона
    count_sumb = 0
    if n.startswith('+7') or n.startswith('8'): # Начинается ли строка с
указанного префикса нужно для того что бы
        for counter in n:
            if counter not in symbols: # Проверка на то что не содержит ли
#строка символы которые не должны присутсвовать в номерах телефона
                return False
            if counter in '1234567890':
                count_sumb += 1 # Кол-во цифр в номере
        if count_sumb != 11: # Если длина номера телефона больше или меньше
#11
            return False
        else:
            return True
    else:
        return False
def mod_number(n):
    n = n.replace('+', '').replace('(', '').replace(')', '').replace(' ',
'').replace('-', '') # Все символя +, (, ), ' ', -, заменяются на на пустую
#строку
```

```
    number = '+7 (' + n[1:4] + ') ' + n[4:7] + '-' + n[7:9] + '-' + n[9:]
#Формируем правильный номер телефона
    return number
phonebook = {}
for counter in iter(input, '.'): # Считываем до поступления на вход точки
    if ' ' in counter: # Нужно для того, чтобы, если введут только имя то его
#номер надо вывести
        if counter.split()[0] not in phonebook: # Имя которое надо #записать
#в телефонную книгу
            phonebook[counter.split()[0]] = [] # Добавляем к ключу #которое
#считается за имя и по этому ключу добавляем словарь
        phones = counter.split(maxsplit = 1)[1]
        for phone in phones.split(', '): # На случай, если несколько #номеров
#телефонов и номера телефонов введены через запятую
            if check_number(phone): # Проверка на корректный номер #телефона
                phonebook[counter.split()[0]].append(mod_number(phone))
#Добавляем #телефонный номер в добавленный ранее список
    else:
        if counter.split()[0] not in phonebook or
phonebook[counter.split()[0]] == []: # Проверка на то есть ли в строке #номер
телефона
            print('Не найдено')
        else:
            print(*phonebook[counter.split()[0]], sep = ', ') # Вывод #номера
телефона
```

1.11. Техника работы с множествами

Листинг 1. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Задача «Количество различных чисел»
Условие. Дан список чисел. Определите, сколько в нем встречается
различных чисел.
'''
s = set(input().split())
print(len(s))
```

Листинг 2. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. Задача «Количество совпадающих чисел»
Условие. Даны два списка чисел. Посчитайте, сколько чисел содержится
одновременно как
в первом списке, так и во втором.
'''
n = set(input().split())
m = set(input().split())
p = 0
for elem in n:
    for j in m:
        if elem is j:
            p += 1
print(p)
```

Листинг 3. Задание 3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 3. Задача «Пересечение множеств»
Условие. Даны два списка чисел. Найдите все числа, которые входят как
в первый,
так и во второй список и выведите их в порядке возрастания.
'''
n = set(map(int, input().split()))
m = set(map(int, input().split()))
lst = []
fin = set()
for i in n:
    for j in m:
        if (i == j):
            lst.append(i)
print(*sorted(lst))
```

Листинг 4. Задание 4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 4.
Задача «Количество слов в тексте»
Условие. 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.
'''
def fillSetWords(string, set_words):
    start_index = 0
    for i in range(len(string)):
        if (string[i] == ' '):
            end_index = i
            set_words.add(string[start_index:end_index])
            start_index = i + 1
set_words = set()
n = int(input())
for i in range(n):
    string = input().split()
    for i in range(len(string)):
        set_words.add(string[i])
print((len(set_words)))
```

Листинг 5. Задание 5.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 5. https://pythontutor.ru/lessons/sets/problems/polyglotes/
Задача «Полиглоты»
Условие. 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.
```

```
'''
languages = set()
n = int(input())
nc = 0
knows_all_langth = True
for i in range(n):
    m = int(input())
    if (m == 1 and nc < 1):
        string = input()
        print(1)
        knows_all_langth = False
        nc += 1
        print(string)
    else:
        for c_langth in range(m):
            string = input()
            languages.add(string)

if knows_all_langth:
    print(len(languages))
    languages = sorted(languages)
    for i in languages:
        print(i)
print(len(languages))
languages = sorted(languages)
for i in languages:
    print(i)
```

Листинг 6. Задание 1.ру,

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1.
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.
'''
lst_words = set()
lst_string = set()
m = int(input())
for i in range(m):
```

```
    lst_words.update(input().lower().split())
n = int(input())
for i in range(n):
    lst_string.update(input().lower().split())

lst_string.difference_update(lst_words)
for i in lst_string:
    print(i)
```

Листинг 7. Задание 2.py, session.txt, disc.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. (Л.Б.) Сессия
В файле disc.txt хранится перечень дисциплин, выносимых на сессию,
например,
Теория алгоритмов
МДК.01.01
Основы экономики
...
В файле session.txt хранятся сведения о результатах сессии, например,
Грушников; П2-18; Теория алгоритмов; 5
Константинович; П2-18; Теория алгоритмов; 5
...
Студент считается сдавшим сессию, если у него сданы все предметы и нет
оценки "2".
Студент считается "отличником", если у него все пятерки
Требуется сформировать множества:
- студентов, сдавших сессию
- студентов-отличников
- дисциплин, по которым нет задолженностей
Результат вывести в файл output.txt
'''
import codecs

f_input1 = codecs.open("session.txt", "r", "utf_8_sig")
f_input2 = codecs.open("disc.txt", "r", "utf_8_sig")

lst = f_input1.readlines() #Из файла session.txt читаем и добавляем в
список lst
session = [i.strip().split('; ') for i in lst] #Удаляем '; ' из list
lst = f_input2.readlines()
disc = [i.strip() for i in lst] #Формирует список строк и благодаря
функции strip с обоих концов которой устранены указанные символы.
Формируется список из предметов
f_input1.close()
f_input2.close()
names_students = []

for data_students in session:
```

```
    if data_students[0] not in names_students:
        names_students.append(data_students[0]) #Добавляем имя с
список имён

excellent = set() #Отличники
sdali = set() #Множество сдавших ссесию
predmet = {*disc} #Название предмета

for i in range(len(names_students)):
    count_5 = 0 #Кол-во отличников. Так же каждая прокрутка цикла
обнуляет этот счётчик и для другого студента начинает считать заново
    count_2 = 0 #Кол-во двоек нужно для того чтобы узнать кто не
закрыл ссесию
    for j in range(len(session)):
        if(session[j][0] == names_students[i]): # Поиск имени студента
            if(session[j][3] == '5'):
                count_5 += 1
            elif(session[j][3] == '2'):
                count_2 += 1
                predmet.discard(session[j][2]) # Удаляет указанный
элемент из множества
    if(count_5 == len(disc)):
        excellent.add(names_students[i]) #Добавляет фамилии отличников
    if(count_2 == 0):
        sdali.add(names_students[i]) # Усли кол-во двоек равно нулю то
студент сдал ссесию (мне бы так)
output_file = open("out.txt", "w")
output_file.write('Сдали:\n'+'\n'.join(names_students)) #Имена тех кто
сдали
output_file.write('\n================================================
============================\n')
output_file.write('Отличники:\n'+'\n'.join(excellent)) #Имена
отличников
output_file.write('\n================================================
============================\n')
output_file.write('Предметы по которым нет
задолжности:\n'+'\n'.join(predmet)) #Предметы по которым нет
задолжностей
output_file.close()
```

1.12. Техника работы с кортежами

Листинг 1. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1.
Вывести чётные
Необходимо вывести все четные числа на отрезке [a; a * 10].
'''
n = int(input())
k = (n + 1) // 2 * 2
print(tuple(range(k, n * 10 + 1, 2)))
```

Листинг 2. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2.
Убывающий ряд.
С клавиатуры вводятся целые числа a > b. Выведите убывающую
последовательность чисел
по одному числу в строке.
'''
n = int(input())
m = int(input())
for i in tuple(range(n, m, -1)):
    print(i)
```

Листинг 3. Задание 3.py, Zadamie3.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 3. (Л.Б.) В каждой строке файла хранится информация о пунктах
и их координатах
относительно некоторого центра.
Требуется
1. Прочесть файл в список кортежей
2. Найти диаметр множества точек, то есть расстояние между наиболее
удалёнными точками.
Указать наиболее удалённые пары
3. Сформировать список пар городов, имеющих одинаковое расстояние до
центра
4. Отсортировать список одним из методов, реализованных в предыдущих
работах
Результаты вывести на экран
'''
```

```
import codecs
def BubbleSort(vals):
    for i in range(len(vals)):
        for j in range(i+1, len(vals)):
            if (vals[i] < vals[j]):
                temp = vals[i]
                vals[i] = vals[j]
                vals[j] = temp

fin = codecs.open("Zadamie 3.txt", 'r', "utf_8_sig")
d_coord = dict()
d_hyp = dict()
print()
for string in fin:
    l = string.split()
    d_coord[l[0]] = tuple(l[1:])
    d_hyp[l[0]] = (int(d_coord[l[0]][0])**2 + int(d_coord[l[0]][1])**2 )**0.5
fin.close()

vals_sort = BubbleSort(list(d_hyp.values()))

numb_city = len(d_coord)
for i in range(numb_city):
    for city in d_hyp:
        if d_hyp[city] == vals_sort[i]:
            print(f"{i+1}|", end = ' ')
            print(city + ":", *d_coord[city], end = '\n   ')
            print(f"До центра: {int(vals_sort[i])} км\n")
            break

count = 0
for city_i in d_hyp:
    for city_j in d_hyp:
        if d_hyp[city_i] == d_hyp[city_j] and city_i != city_j:
            print(f"{d_hyp[city_i]} == {d_hyp[city_j]}")
            count += 1
if count == 0:
    print("Пар городов, имеющих одинаковое расстояние до центра нет")
```

Листинг 4. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Класс namedtuple() модуля collections в Python.
'''
from collections import namedtuple

print("Primer 1\n")
Point = namedtuple('Point', ['x', 'y'])

x, y = int(input()), int(input())
p = Point(x, y)
print(p[0] - p[1])
print(p[0] * p[1])
print(p[0] / p[1])
print(p)
```

```
print("Primer 2\n")
lst = [1, 2]
p = Point._make(lst)
print(p)

print("Primer 3\n")
Home = namedtuple('home', ['color', 'width', 'height'])
p = Home("#ffff0000", "50", "150")
print(p.color)

print("Primer 4\n")
Persona = namedtuple('Person', 'name age group')
Peter = Persona(name='Peter', age='18', group='P1-18')
Andrey = Persona(name='Andrey', age='18', group='P1-18')
for i in [Peter, Andrey]:
    print(i)
print(Peter._replace(name="icefantik"))
print("Primer 5\n")
Point3D = namedtuple('Point3D', Point._fields + ('z',))
print(Point3D(10, 20, 30))
Point2D = namedtuple('Point2D', Point._fields)
print(Point2D(10, 20))

print("Primer 6\n")
Group = namedtuple("Group", ['name', 'age', 'group'])
for i in range(1):
    s_name = input()
    s_age = input()
    s_group = input()
    Group.__doc__ += str(i)
    Group.name.__doc__ = s_name
    Group.age.__doc__ = s_age
    Group.group.__doc__ = s_group
print("Primer 7")
lst = []
Group2 = namedtuple('Group2', ['name', 'age', 'group'])
for i in range(1, 2):
    s_name = input()
    s_age = input()
    s_group = input()
    lst.append(Group2(name=s_name, age=s_age, group=s_group))
for i in lst:
    print("Name: {0}, age: {1}, group: {2}".format(i.name, i.age, i.group))
```

1.13. Техника работы с файлами

Листинг 1. Задание 1.py, file.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1.
Text5. Дана строка S и текстовый файл. Добавить строку S в конец
файла.
'''
string = input()
with open("file.txt", "a") as file:
    file.write(string)
```

Листинг 2. Задание 2.py, Task2.txt, Task2_1.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2.
Text12. Дана строка S и текстовый файл. Заменить в файле все пустые
строки на строку S.
'''
string = input()
with open("Task2.txt", "r") as read_file:
    with open("Task2_1.txt", "w") as write_file:
        for i in read_file.readlines():
            print(i)
            if i != "\n":
                write_file.write(i)
            else:
                write_file.write(string+"\r")
```

Листинг 3. Задание 3.py, Text20.txt, Text20_1.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 3.
Text20. Дан текстовый файл. Заменить в нем все подряд идущие пробелы
на один пробел.
'''
with open("Text20.txt", "r") as f_read:
    with open("Text20_1.txt", "w") as f_write:
        nc = 0
        final_string = ""
        for string in f_read.readlines():
            nc = 0
            for j in range(len(string)):
                if (string[j] == " " and nc == 0):
                    nc += 1
                    final_string += string[j]
                elif (string[j] != " "):
```

```
                    nc = 0
                    final_string += string[j]
        f_write.write(final_string)
        print(final_string)
```

Листинг 4. Задание 4.py, Text44.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 4.
Text44. Дан текстовый файл, каждая строка которого изображает целое
число, дополненное слева и справа
несколькими пробелами. Вывести количество этих чисел и их сумму.
'''
lst = []
summ = 0
count = 0
formation = ""
amount_of_num = 0
with open("Text 44.txt", "r") as f_read:
    for i in f_read.read():
        if (i == ' ' or i == "\n"):
            if formation != '':
                lst.append(int(formation))
            formation = ""
            continue
        elif ('0' <= i and i <= '9' or i == '-'):
            formation += i
            count += 1
print(len(lst))
print(sum(lst))
```

Листинг 5. Задание 5.py, Text 53.txt, TextPunct.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 5.
Text53. Дан текстовый файл. Создать символьный файл, содержащий все
знаки препинания,
встретившиеся в текстовом файле (в том же порядке).
'''
punct_marks = []
with open("Text 53.txt", "r") as file:
    for i in file.read():
        if i == '.' or i == ',' or i == ';' or i == '!' or i == '?' or i == '-':
                punct_marks.append(i)
        elif i == '(' or i == ')' or i == '[' or i == ']' or i == '{' or i == '}':
                punct_marks.append(i)
        elif i == '"' or i == '<<' or i == '>>' or i == ':' or i == '/' or i == "'":
                punct_marks.append(i)
with open("TextPunct.txt", "w") as Punctfile:
```

```
    for i in punct_marks:
       Punctfile.write(i)
```

Листинг 6. Задание 1.ру, Информация о курсовых проектах.txt

```
import codecs

def direct_occurs_most_often(direction_students):  # Какое направление
встречается чаще всего

    max_nc = 1

    direction = direction_students[0]

    for i in range(len(direction_students)):

        nc = 1

        for k in range(i + 1, len(direction_students)):

            if direction_students[i] == direction_students[k]:

                nc += 1

        if nc > max_nc:

            max_nc = nc

            direction = direction_students[i]

    print(direction)

def appeared_in_diplomas(languages_student, years_students):   # Какие
#языки и среды появились в дипломах в 2017 г.

    for i in range(len(years_students)):

        if years_students[i] == '2017':

            print(languages_student[i], end="")

name_students = []           # имена студентов

group_students = []          # название группы студентов

years_students = []          # год студента

theme_students = []          # тема студентов

direction_students = []      # направление студента

languages_student = []       # языки программирования студентов

with codecs.open("Информация о курсовых проектах.txt", "r",
"utf_8_sig") as file:

    for line in file:

        lst = line.split(';')

        name_students.append(lst[0])

        group_students.append(lst[1])

        years_students.append(lst[2])
```

```
            theme_students.append(lst[3])
            direction_students.append(lst[4])
            languages_student.append(lst[5])
    direct_occurs_most_often(direction_students)
    appeared_in_diplomas(languages_student, years_students)
```

1.14. Техника работы с модулями

Листинг 1. Задание1.py, text.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. 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/
'''
import collections

lst = []
length = 0
with open("text.txt", "r") as file:
    for i in file.readline():
        lst.append(i)
        length += 1

dq = collections.deque(lst)

dq.reverse()
print(dq)

with open("text.txt", "w") as file:
    for i in range(length):
        file.write(dq[i])

n = 3
array = [40, 35, 30, 25, 20]
d = collections.deque(array)
summ_d = sum(d)
for i in array:
    summ_d += i - d.popleft()
    d.append(i)
print(summ_d / n)
to_check = input()
item_found = False
iterables = ['ABC', 'D', 'EF']
iterators = collections.deque(iterables)
for item in iterators:
    if to_check == item:
        print("Item found")
        item_found = True
if not item_found:
    print("Item not found")
```

Листинг 2. Задание2.py, text1.txt

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. Контейнерные типы данных модуля collections.
Класс Counter() модуля collections в Python.
https://docs-python.ru/standart-library/modul-collections-
python/klass-counter-modulja-collections/
'''
import collections
import re

cnt = collections.Counter(a=4, b=2, c=0, d=-2)
print(cnt)
print(*cnt.elements())

ct = collections.Counter("abbbaaaccacccascd")
s = set(ct)
print(ct.most_common(len(s)))

cnt1 = collections.Counter(a=3, b=6, c=6, d=5)
cnt1.subtract(cnt) # вычитает элементы текущего счетчика
print(cnt1)

cnt1.update(cnt) # складывает элементы текущего счетчика
print(cnt1)

print(cnt + cnt1) #Сложить два счетчика
print(cnt - cnt1) #Вычитание счетчиков
print(cnt & cnt1) #Пересечение счетчиков
print(cnt | cnt1) #Объединение счётчиков

print(cnt.items())
print(cnt.values())
cnt.clear()

string = ""
c = collections.Counter()
with open("text1.txt", "r") as file:
    for i in file:
        string += i

c = collections.Counter(string).most_common(len(string))
print(c)

cn = collections.Counter()
with open("text1.txt", "r") as file:
    for i in file:
        words = re.findall(r'\w+', file.read())   #findall
#используется для поиска всех непересекающихся совпадений в шаблоне
```

```
cn = collections.Counter(words).most_common(len(words))
print(cn)
```

Листинг 3. Задание 3.py

```
import collections
string = input()
string2 = "12345678910"
dq = collections.deque(string)
dq.append(string2)
print(dq)
dq.extend('ehwr')
print(dq)
dq.extendleft('ab')
print(dq)
print(dq.index('a', 1))
print(dq.pop())
print(dq.popleft())
dq.reverse()
print(dq)
dq.rotate(2)
print(dq)
dq.rotate(-4)
print(dq)
```

Листинг 4. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Контейнерные типы данных модуля collections.
https://docs-python.ru/standart-library/modul-collections-python/
Класс defaultdict() модуля collections в Python.
https://docs-python.ru/standart-library/modul-collections-
python/klass-defaultdict-modulja-collections/
'''
from collections import defaultdict
import random

lst = [("Ben", 89001234050), ("Alice", 210-220), ("Ben", 70504321009),
("Alice", 404-502), ("Nick", 16507811251),
       ("Robert", 51234047129), ("Alice", 894-455), ("Alice", 439-
495)]
d = defaultdict(list)
```

```
for i, elem in lst:
    d[i].append(elem)

print(d.items())

lst_number = [('a', 1), ('b', 2), ('c', 3), ('d', 4) , ('e', 5) ,
('f', 6), ('a', 2), ('a', 3)]
d_slov = {}
for i, elem in lst_number:
    d_slov.setdefault(i, []).append(elem**2)

print(sorted(d_slov.items()))

l = {}
for i, elem in lst:
    n = random.randint(1, 100)
    l.setdefault(i, []).append(n)
print(l)

string = "Hello world"
d = defaultdict(int)
for k in string:
    d[k] -= 1
print(d)

lst = ["qwertyui", "asdfghjkl", "zxcvbnm", "q"]
d = defaultdict(int)
for i in lst:
    d[i] += len(i)
print(d.items())

lst = [("Ben", 89001234050), ("Alice", 210-220), ("Ben", 70504321009),
("Alice", 404-502), ("Nick", 16507811251),("Robert", 51234047129),
("Alice", 894-455), ("Alice", 439-495)]
d = defaultdict(set)
index = 0
for i, elem in lst:
    if (index % 2 == 0):
        d[i].add(elem)
    index += 1
print(d.items())

for elem in lst:
    print(elem)

for i, elem in lst:
    print(i, elem)
```

Листинг 5. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. Контейнерные типы данных модуля collections.
Класс OrderedDict() модуля collections в Python.
https://docs-python.ru/standart-library/modul-collections-
python/klass-ordereddict-modulja-collections/
'''
from collections import *

c = Counter()
items = []
n = int(input())
for i in range(n):
    string = input()
    items.append(string)
for i in items:
    c[i] += 1
print(c)

defdict = defaultdict(list)
for i in range(n):
    defdict[i].append(i)
print(defdict)

d = OrderedDict.fromkeys('abcd')
d.move_to_end('b') # добавляет элемент из строки в конец
print(''.join(d.keys())) #p2rint(d.keys())
d.move_to_end('a')
print(''.join(d.keys()))

d.popitem('a') # удаляем элемент
d.popitem('b')
print(''.join(d.keys()))

d.move_to_end('d', last=True) #переносим d в перёд
print(''.join(d.keys()))
d.move_to_end('d', last=False) #переносим d в перёд
print(''.join(d.keys()))
```

Листинг 6. Задание1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Функция argv модуля sys в Python.
https://docs-python.ru/standart-library/modul-sys-python/funktsija-
argv-modulja-sys/
'''
```

```
import sys

print(len(sys.argv))
print(sys.argv[1])

for i in range(len(sys.argv)):
    print(sys.argv[i], end=" ")

w = None
print(len(sys.argv))
if len(sys.argv) > 1:
    if sys.argv[1] in ("-h", "-help"):
        print("Manual")
        word = 0
else:
    word = sys.argv[0]
```

Листинг 7. Задание2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. Имя используемой OS.
'''
import sys
import os

if sys.platform.startswith('linux'):
    print("This is linux {0}".format(os.name))
else:
    print("This is not linux. This is {0}".format(os.name))
    print(sys.getwindowsversion())
```

Листинг 8. Задание3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 3. Различные сведения о версии Python.
'''
import sys

string = sys.version_info
print(string)
print(sys.copyright)
print("API C languages: ", sys.api_version)
print(sys.version)
print("Hex version: ", sys.hexversion)
```

Листинг 9. Задание4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 4. Каталоги и пути интерпретатора Python.
'''
import sys
import os

print(sys.prefix)
print(sys.base_prefix)
print(sys.exec_prefix)
print(sys.base_exec_prefix)
print(sys.executable)
```

Листинг 10. Задание5.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 5. Объекты stdin, stdout, stderr модуля sys в Python.
'''
import sys
import time

stdin = sys.stdin
try:
    sys.stdin = open("text_zadania5.txt", "r")
    s = input()
    print("odna stroka: ", s)
finally:
    sys.stdin.close()
    sys.stdin = stdin

try:
    sys.stdin = open("text_zadania5.txt", "r")
    for i in sys.stdin:#s = input()
        print(i, end="")
finally:
    sys.stdin.close()
    sys.stdin = stdin
```

Листинг 11. Задание6.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 6. Функция exit() модуля sys в Python.
'''
import sys
if len(sys.argv) > 1:
    if sys.argv[1] in ("-exit", "-e"):
        sys.exit(0)
```

Листинг 12. Задание1.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 1. Вывод текущей директории
'''
import os
print("Текущая деректория:", os.getcwd())
```

Листинг 13. Задание2.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 2. Создание папки
'''
if not os.path.isdir("Митюшин и Слепов"):  # Проверка на отсутсвие
#папки с таким названием
    os.mkdir("Митюшин и Слепов")
```

Листинг 14. Задание3.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 3. Изменение директории
'''
import os

os.chdir("../")
if not os.path.isdir("Новая папка"):  # Проверка на отсутсвие папки с
#таким названием
    os.mkdir("Новая папка")

os.chdir("Новая папка")

print("Текущая директория изменилась на :", os.getcwd())
```

Листинг 15. Задание4.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 4. Создание вложенных папок
'''
import os

os.chdir("..")
path_dir = os.getcwd()
n = int(input("Введите количество вложенных папок, которые вы хотите \
создать: "))
for i in range(n):
    path = path_dir + "/Папка " + str(i)
```

```
    os.mkdir(path)
    path_dir += "/Папка " + str(i)
```

Листинг 16. Задание5.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 5. Создание файлов
'''
file = open("MyFirstFile.txt", "w")
file.write("Файл — именованная область данных на носителе \
информации.")
file.close()
```

Листинг 17. Задание6.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 6. Переименование файлов
'''
import os
path = input("Ввведите путь, где будет создан этот файл: ")
First_name = input("Введите имя файла: ")

os.chdir(path)
file = open(First_name + "txt", "w")
file.close()

Second_name = input("Как Вы хотите переименовать файл: ")
os.rename(First_name + "txt", Second_name)
```

Листинг 18. Задание7.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 7. Перемещение файлов
'''
import os
f = open("First_name.txt", "w")
f.close()
os.replace("First_name.txt", "../First_name.txt")
```

Листинг 19. Задание8.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 8. Список файлов и директорий
'''
import os
# распечатать все файлы и папки
```

```
for dirpath, dirnames, filenames in os.walk("."):
    # перебрать каталоги
    for dirname in dirnames:
        print("Каталог:", os.path.join(dirpath, dirname))
    # перебрать файлы
    for filename in filenames:
        print("Файл:", os.path.join(dirpath, filename))
# os.walk() — это генератор дерева каталогов.
# Он будет перебирать все переданные составляющие.
# Здесь в качестве аргумента передано значение «.», которое обозначает
#верхушку дерева.
# Метод os.path.join() был использован для объединения текущего пути с
#именем файла/папки.
```

Листинг 20. Задание9.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 9. Удаление файлов
'''
import os
File_name = input("Введите имя файла: ")
f = open(File_name + ".txt", "w")
f.close()
os.remove(File_name + ".txt")
```

Листинг 21. Задание10.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 10. Удаление директорий
'''
import os
print("Все папки и файлы: ", os.listdir())
del_name = input("Введите имя папки которую Вы хотите удалить: ")
os.rmdir(del_name)
```

Листинг 22. Задание11.py

```
'''
Выполнил: Слепов Андрей. П1-18
Задание 11. Получение информации о файлах
'''
import os
f = open("text.txt", "w")
f.write('Hello, World!')
f.close()
print(os.stat("text.txt"))
os.remove("text.txt")
```

1.15. Техника работы с классами

Листинг 1. Задание 1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 1. Создание класса
'''
class ColorSet:
    black = "#000000"
```

Листинг 2. Задание 2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 2. Создание объекта
'''
class ColorSet:
    black = "#000000"
    white = "#ffffff"

cl = ColorSet()
print(ColorSet.black)
```

Листинг 3. Задание 3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 3. Функция init
'''
class VisualElements:
    """color of object
    color_background of object
    """
    def __init__(self, color, color_background):
        self.color = color
        self.color_background = color_background
    black = "#000000"
    white = "#ffffff"

unit = VisualElements("#000000", color_background = "#ffffff")
print(unit.color)
print(unit.color_background)
```

Листинг 4. Задание 4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 4. Методы объектов
'''
class CreateCharacter:
    """Класс для передачи параметров персонже
    Например имени, раса и параметров здоровья
    Параметор здоровья по умолчанию 100
    """
    max_speed = 100
    def __init__(self, name,  race, health, damage = 10, armor = 5):
        """name object name
        race object race
        health object health
        damage object damage
        armor object armor
        """
        self.name = name
        self.race = race
        self.health = 100
        self.damage = damage
        self.armor = armor
    def print_haracteristic(self):
        """Вывод параметров персонажа"""
        print("Name:", self.name)
        print("Race:", self.race)
        print("Health:", self.health)
        print("Damage:", self.damage)
        print("Armor:", self.armor)

Player = CreateCharacter("Pers", 'Human', 100)
Player.print_haracteristic()
```

Листинг 5. Задание 5.py

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 5. Параметр self
'''
class Enemy:
    """Базовый класс врага с пораметрами жизни и пораметрами
    наносимого им урона
    """
    def __init__(self, name, enemy_health = 30, damage = 5):
        """name object name
        enemy_ health object enemy_ health
        damage object damage
        """
        self.name = "enemy"
```

```
        self.enemy_health = enemy_health
        self.damage = damage
    def hit(self, damage):
    """Передаем здоровье героя и вычитаем из
    здоровья урон наносимым врагом
    """
        self.enemy_health -= damage

class CreateCharacter:
    """Класс для передачи параметров персонже
    Например имени, раса и параметров здоровья
    Параметор здоровья по умолчанию 100
    """
    max_speed = 100
    def __init__(self, name,  race, health, damage = 10, armor = 5):
        """name object name
        race object race
        health object health
        damage object damage
        armor object armor
        """
        self.name = name
        self.race = race
        self.health = 100
        self.damage = damage
        self.armor = armor
    def hit(self, damage):
        """Передаем здоровье героя и вычитаем из
        здоровья урон наносимым врагом
        """
        self.health -= damage

Player = CreateCharacter("Pers", 'Human', 100)#Содаём экземпляр класса
enemy = Enemy("eewre")
Player.hit(enemy.damage)
enemy.hit(Player.damage)
print(Player.health)
print(enemy.enemy_health)
```

Листинг 6. Задание 6.ру

```
'''
Выполнил: Митюшин Пётр. П1-18
Задание 6. Изменение свойств объекта
'''
class CreateCharacter:
    max_speed = 100
    def __init__(self, name,  race, health, damage = 10, armor = 5):
        """name object name
        race object race
```

```
        health object health
        damage object damage
        armor object armor
        """
        self.name = name
        self.race = race
        self.health = 100
        self.damage = damage
        self.armor = armor
    def print_haracteristic(self):
        """Вывод параметров персонажа"""
        print("Name:", self.name)
        print("Race:", self.race)
        print("Health:", self.health)
        print("Damage:", self.damage)
        print("Armor:", self.armor)
Player = CreateCharacter("Pers", 'Human', 100)
Player.print_haracteristic()
Player.name = "Peter"
print("------------")
Player.print_haracteristic()
```

Листинг 7. Задание 7.py

```
'''
Задание 7. Удалить свойства объекта
'''
class CreateCharacter:
    max_speed = 100
    def __init__(self, name,  race, health, damage = 10, armor = 5):
        self.name = name
        self.race = race
        self.health = 100
        self.damage = damage
        self.armor = armor
    def print_haracteristic(self):
        """Вывод параметров персонажа"""
        print("Name:", self.name)
        print("Race:", self.race)
        print("Health:", self.health)
        print("Damage:", self.damage)
        print("Armor:", self.armor)

Player = CreateCharacter("Pers", 'Human', 100)
del Player.armor # Удаляет экземпляр класса
try:
    Player.print_haracteristic()
except AttributeError:
    print("Attribute was removed")
```

Листинг 8. Задание 8.py

```
'''
Задание 8. Удаление объектов
'''
class CreateCharacter:
    max_speed = 100
    def __init__(self, name,  race, health, damage = 10, armor = 5):
        """name object name
        race object race
        health object health
        damage object damage
        armor object armor
        """
        self.name = name
        self.race = race
        self.health = 100
        self.damage = damage
        self.armor = armor
    def print_haracteristic(self):
        """Вывод параметров персонажа"""
        print("Name:", self.name)
        print("Race:", self.race)
        print("Health:", self.health)
        print("Damage:", self.damage)
        print("Armor:", self.armor)
Player = CreateCharacter("Pers", 'Human', 100)
del Player
try:
    Player.print_haracteristic()
except NameError:
    print("Object player was removed")
```

Листинг 9. Задание1.py

```
'''
K16_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
'''
print("Задание 1")
class Class_Wolf:
    count_wolf = 0
    def __init__(self, age):
        """age object age"""
        self.name = "Wolf"
        self.color = "grey"
```

```
        self.age = age
        self.rem_atr = "Нужно удалить"
        Class_Wolf.count_wolf += 1
    def holw():
        """Вывод текста WOOO"""
        print("WOOOOOOOOOOOOOOOOO!!!")
    def how_much_wolf(self):
        """Выводит количество вызовов класса"""
        print("Воет {0} волк(ов)".format(Class_Wolf.count_wolf))

wolf = Class_Wolf(10)

print("\nЗадание 3")
Class_Wolf.holw()
wolf.how_much_wolf()
print("Вывод сколько воют волков не через функцию класса %d" %
Class_Wolf.count_wolf)

print(hastate(wolf, "rampart")) # Возвращает true усли атрибут сущ.
dilator(wolf, "rampart") # Удаляет атрибут rampart
print(hastate(wolf, "rampart"))# Проверить, есть ли в wolf атрибут
#rampart

atr = input("Введите имя атрибута: ")
print("Этот атрибут %s существует" %art)
print("Значение атрибута: %s", getter(wolf, art)) # возвращает значение
#атрибута 'age'
estate(wolf, art, 10) # Устанавливает атрибут art на 10
print(getter(wolf, art)) #Возвращает значение отребута

print("\nЗадание 4")

print("doc: ", Class_Wolf.__doc__) # строка документации класса
print("name: ", Class_Wolf.__name__) # имя класса
print("module: ", Class_Wolf.__module__) # Имя модуля в котором
#определён класс
print("bases: ", Class_Wolf.__bases__) # могут быть пустые tuple,
#содержащие базовые классы
print("dict: ", Class_Wolf.__dict__) #Словарь содер. Пространства имён

print("\nЗадание 5")

wolf1 = Class_Wolf(10)
wolf2 = wolf1
wolf3 = wolf1
print(id(wolf1), id(wolf2), id(wolf3)) # выведите id объектов
del wolf #Удаление объекта
del wolf1
del wolf2
del wolf3
```

```
try: #Успешно ли удалился объект
    print(id(wolf1), id(wolf2), id(wolf3))
except Name Error:
    print("Удаление экземпляров прошло успешно")
```

Листинг 10. Задание1.py

```
'''
К16_3. Техника работы с классами.
Задание 1. Наследование класса
Задание 2. Переопределение методов
Задание 3. Популярные базовые методы
Задание 4. Приватные методы и атрибуты класса
'''
print("Задание 1")
class Parant:
    """Класс родителья"""
    a = 10
    def set_attr(self, attr):
        """Возводим значение в квадрат"""
        Parant.a = attr ** 2
    def parent_method(self):
        """Функция выводит текст"""
        print("Вызов метода родителя")
    def get_attr(self):
        """Выводит переменную принадлежащую классу"""
        print("attr: ", Parant.a)
    def power(self, x, y):
        """Функция умножения x на y"""
        print(x * y)
    def method(self):
        """Функция выводит строку HELLO"""
        print("HELLO")
class Child(Parant):
    """Класс наследника"""
    def child_method(self): print("Вызов метода класса наследника")
    def power(self, x, y):
        """Выводим значение в x в степени y"""
        print(x ** y)
    def method(self):
        """Функция выводит строку HELLO"""
        print("HELLO")
c = Child()# экземпляр класса Child
c.child_method()# вызов метода child_method
c.parent_method()# вызов родительского метода parent_method
c.set_attr(4) # еще раз вызов родительского метода
c.get_attr()# снова вызов родительского метода

print("\nЗадание 2")
class Parant:
```

```
    """Класс родителья"""
    a = 10
    def power(self, x, y):
        """Выводим значение в x в степени y"""
        print(x * y) #Задание 2
    def method(self):
        """Функция выводит строку HELLO"""
        print("HELLO")#Задание 2

class Child(Parant):
    def power(self, x, y):
        """Выводим значение в x в степени y"""
        print(x ** y)#Задание 2
    def method(self):
        """Функция выводит строку HELLO"""
        print("HELLO")#Задание 2
c.power(10, 4)
c.method()

print("\nЗадание 3")
class ClassName:
    def __init__(self, x, y):
        """x object x"""
        """y object y"""
        self.x = x
        self.y = y

    def __str__(self):
        """Выводит строку со значением x и y"""
        return "Vector ({0}, {1})".format(self.x, self.y)

    def __mul__(self, other):
        """Возвращаем значения x нынешнее
        умножаем на x предыдушее и с y тоже самое
        """
        return ClassName(self.x * other.x, self.y * other.y)

v1 = ClassName(3, 10)
v2 = ClassName(5, 5)
print(v1 * v2)
print("\nЗадание 4")
class Counter:
    sec_count = 0
    x = 2
    def print_power(self):
        """Умножаем значение x на 2 и выводим"""
        self.x *= 2
        self.__y = 100
        print(self.x)
    def count(self):
```

```
        """Считаем количество вызова функции"""
        self.sec_count += 1
        print(self.sec_count)
coun = Counter()#Создание экземпляра класса
coun.count()#Вызываем функцию
coun.count()
print(coun.sec_count)Выводим кол-во вызова функции

coun.print_power()
coun.print_power()
print(coun.x)
```

Листинг 11. module.py

```
class ClassSize:
    def __init__(self, width, height):
        """width object width"""
        """height object height"""
        self.width = width
        self.height = height
```

Листинг 12. Задание1.py, module.py

```
import modul

rect = modul.ClassSize(17, 28)
print(rect.width)

class MyClass:
    def __init__(self, cont):
        """cont object cont"""
        self.cont = cont
    def __truediv__(self, other): # Деление с остатком
        """Строку умножаем на требуемую нам длину
        и возвращаем строку с передаваемым значением cont
        """
        line = "-" * len(other.cont)
        return "\n".join([self.cont, line, other.cont])

spam = MyClass("spam")
string = MyClass("hello!")
print(spam / string)

class ClassHi:
    countHello = 0
    def __init__(self, name):
        """name object name
        Считаем кол-во вызовов этого класса
        """
        self.name = name
```

```
        ClassHi.countHello += 1
    def printHi(self):
        """Выводим значение объекта который
        мы передали при вызове класса
        """
        print("Hello " + self.name)
    def printCount(self):
        """Вывод переменной класса которая
        считала значения вызовов этого класса
        """
        print(MultPoint.countHello)

sq = ClassHi ("Peter") #Создание экземпляра класса ClassHi
sq.printHi() #Выводит преветствие и передовемое имя
sq.printCount() #Выводит кол-во раз вызавов класса
```

Раздел 2. Техника решения задач с использованием библиотек

2.1. Установка и настройка среды JetBrains PyCharm

Открываем браузер и набираем в поисковике PyCharm и переходим на страницу официального сайта JetBrains: https://www.jetbrains.com/ru-ru/pycharm/. На главной странице нажимаем на кнопку download которая продемонстрирована на рис.1.

Рис. 1.Официальный сайт PyCharm

Нажмите «Download». Произойдет перенаправление на страницу загрузки PyCharm рис2.

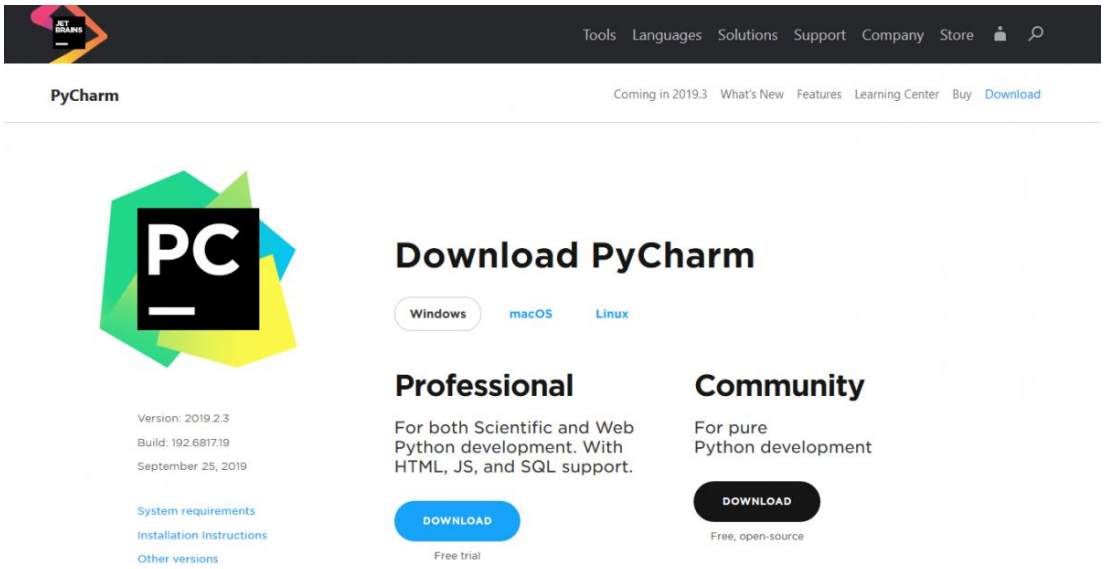


Рис. 2.

Дальше необходимо выбрать версию PyCharm: Community или Professional. Professional – является платной версией с расширенным функционалом для разработки. А Community – является бесплатной версией, но с базовым функционалом. После нажатия на кнопку «Download» на рис.2. скачивание начнется автоматически.

Рис. 3. Установщик PyCharm

Запускаем скаченный файл pycharm-community-2020.3.5.exe. Открывается диалоговое окно, которое представлено на рис.3

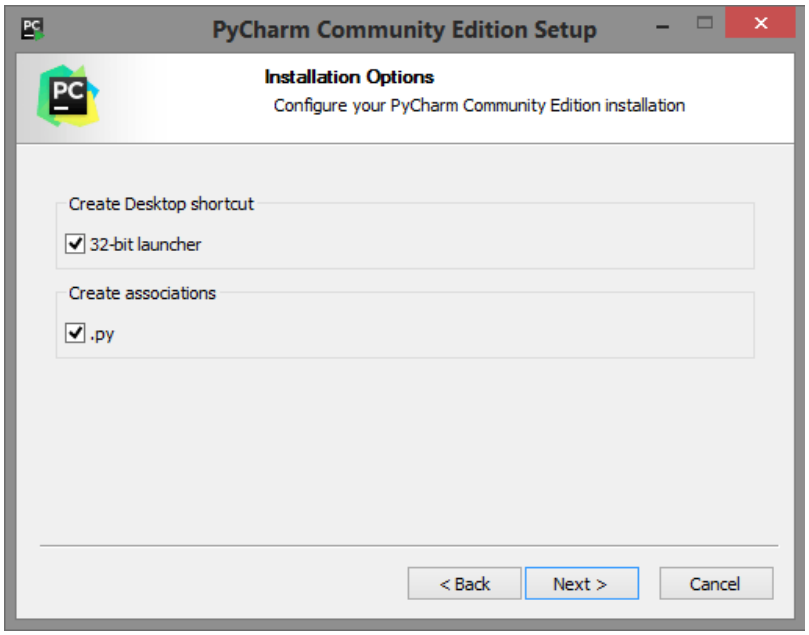


Рис. 4. Создание ярлыка PyCarm

Оставьте папку установки по умолчанию. Это папка C:\Program files(x86)\JetBrains\PyCharm Community Edition 2019.2. Если выбрана другая, лучше указать именно это направление. Нажмите «Next». Поставьте галочки напротив обоих пунктов как показано на рис.4, если нужен ярлык для рабочего стола. Нажмите «Next».

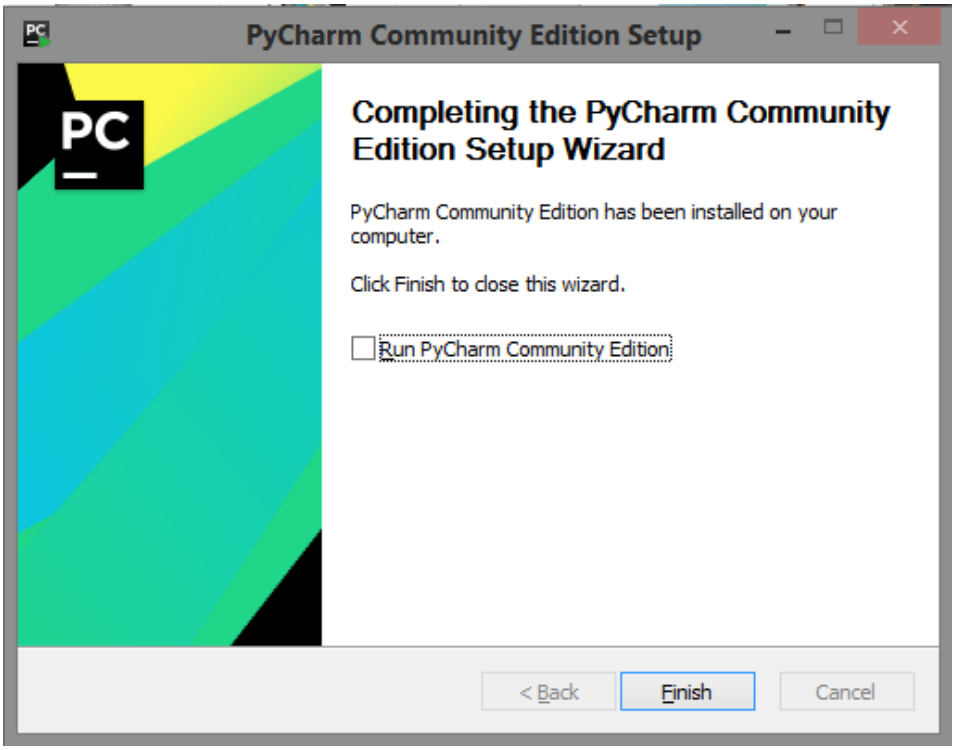


Рис. 5. Завершение установки

После нажатия Next установщик PyCharm завершает установку, как показано на рис. 5.

2.2. Техника работы с базами данных

Листинг 1. File.py

```
'''
Выполнил: Андрей Слепов. П1-18
'''
import sqlite3

class DataBase:
    def __init__(self, nameDB):
        self.db = sqlite3.connect(nameDB) #Подключение к БД
        self.cursor = self.db.cursor() #Позволяет взаимодействовать с БД

    def create_table(self, table): #Создаём таблицу
        self.cursor.execute(f"CREATE TABLE IF NOT EXISTS '{table}'(\
                            id INT PRIMARY KEY,\
                            'fio' TEXT, \
                            'group' TEXT,\
                            'direction' TEXT)")
        self.save()

    def add_stud(self, **kwargs): #Добавляем запись
        if self.cursor.fetchone() is None:
            self.cursor.execute("INSERT INTO 'students'\
            ('fio', 'group', 'direction') VALUES (?, ?, ?)",\ (kwargs['fio'],
kwargs['group'], kwargs['direction']))

    def print_table(self, table): #Печатаем в консоли таблицу
        for i in self.cursor.execute(f"SELECT * FROM {table}"):
            print(i)

    def save(self): #Сохранение изменений
        self.db.commit()

    def close(self): #Закрытие БД
        self.db.close()

db = DataBase("serv.db")
db.create_table('students')
db.add_stud(fio="Цыпков Илюффка Владимирович", group="П1-18",\
direction="Студент")
db.print_table('students')
db.close()
```

2.3. Техника работы с библиотекой tkinter

Листинг 1. Phonebook.py

```
'''
Выполнил: Митюшин Пётр. П1-18
В данной программе представлена работа с библиотекой tkinter. Данная
программа создаёт окно в котором находится два поля ввода имя и телефон.
'''
from tkinter import *

from tkinter import messagebox

def clickedAndEditInFile(): #При нажатии на кнопку добавляет в файл

    if (edit_telephon_file.get().isdigit() and len(edit_telephon_file.get()) == 11):

        resultString = ""

        with open("file.txt", "a", encoding='utf-8') as file:

            file.write(edit_name_file.get() + ' : ' + edit_telephon_file.get() + '\n')

        file.close()

    else:

        messagebox.showinfo("Ошибка", "Вы ввели не правильный телефонный номер")
#вывод сообщения если телефонный номер не правильный

window = Tk()

window.title("Телефонный справочник")

window.geometry("480x150")

edit_name_file = StringVar()

edit_telephon_file = StringVar()

label_name = Label(text="Имя: ")

label_name.pack(side="top")

edit_name = Entry(width = 20, textvariable = edit_name_file)

edit_name.pack(side="top")

label_telephon = Label(text="Телефонный номер: ")

label_telephon.pack(side="top")
edit_telephon = Entry(width = 20, textvariable = edit_telephon_file)

edit_telephon.pack(side="top")

bedit_in_file = Button(window, text="Добавить", width = 10,
command=clickedAndEditInFile)

bedit_in_file.pack(side="top")

window.mainloop()
```

Листинг 2. File2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
В данной программе из файла читается 5 строк первая строка является вопросом, вторая, третья, четвёртая является вариантами ответа к вопросу, пятая строка является номером ответа на вопрос.
'''
from tkinter import *

window = Tk()

window.title("Тестирование")

window.geometry('400x250')

def check_answer():

    if number_answer == var.get():

        print("Правильно")

    else:

        print("Ошибка")

var = IntVar()

with open('set_questions.txt', 'r') as file:

    question = file.readline()

    answer1 = file.readline()

    answer2 = file.readline()

    answer3 = file.readline()

    number_answer = int(file.readline())

    Label(text=question).place(x=0, y=0)

    ans1 = Radiobutton(window, text=answer1, value=1, variable=var).place(x=0, y=20)

    ans2 = Radiobutton(window, text=answer2, value=2, variable=var).place(x=0, y=50)

    ans3 = Radiobutton(window, text=answer3, value=3, variable=var).place(x=0, y=80)

    btn_answer = Button(window, text="Ответить", command=check_answer).place(x=0, y=110)

window.mainloop()
```

Листинг 3. Tk.py

```
'''
Выполнил: Андрей Слепов П1-18
'''
from tkinter import *

from tkinter import messagebox

class Window():

    def __init__(self):

        self.root = Tk()
```

```
        self.root.title("Заголовок окна")

        width = 400

        height = 500

        x = 500

        y = 250

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

        self.label = Label(self.root, text="Текст", bg="#a67fe4",\
relief=GROOVE, wraplength=170, font="TimeNewRoman 15",\
fg="#09f3d3").pack(anchor=N)

        self.menu_button()

        self.button = Button(self.root, width=2, height=1, bg="red",\
command=lambda: self.root.config(bg="red")).pack(anchor=W)

        self.button = Button(self.root, width=2, height=1, bg="orange",\
command=lambda: self.root.config(bg="orange")).pack(anchor=W)

        self.button = Button(self.root, width=2, height=1, bg="blue",\
command=lambda: self.root.config(bg="blue")).pack(anchor=W)

    def menu_button(self):

        self.button = Button(self.root, width=15, height=3, text="Press me",\
bg="#aaaaff", command=text_area).pack()

        self.button = Button(self.root, width=15, height=3, text="Press me",\
bg="#6baabb", command=text_field).pack()

        self.button = Button(self.root, width=15, height=3, text="Press me",\
bg="#5ff7cc", command=lambda:\
messagebox.showwarning("ВНИМАНИЕ!!!!!!!!!!!!!!", "ЫЪЫ")).pack()

def text_area():

    def smile():

        lable = Label(text, text=":)", bg="yellow")

        text.window_create(INSERT, window=lable)

    root = Tk()

    text = Text(root, width=50, height=10)

    text.pack()

    button = Button(root, text=":)", command=smile)

    button.pack()

def text_field():

    def insertText():

        s = "Илюффка Цыплаков"

        text.insert(1.0, s)

    def getText():
```

```
        s = text.get(1.0, END)
        label['text'] = s
    def deleteText():
        text.delete(1.0, END)
    root = Tk()
    text = Text(root, width=30, height=5)
    text.pack()
    frame = Frame(root)
    frame.pack()
    b_insert = Button(frame, text="Вставить", command=insertText)
    b_insert.pack(side=LEFT)
    b_get = Button(frame, text="Получить", command=getText)
    b_get.pack(side=LEFT)
    b_delete = Button(frame, text="Удалить", command=deleteText)
    b_delete.pack(side=LEFT)
    label = Label(root)
    label.pack()
    root.mainloop()
Window()
mainloop()
```

2.4. Техника работы с библиотекой NumPy

Листинг 1. File1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
'''
import numpy
lst = [i for i in range(1, 11)] #формируем список от 1 до 10
arr = numpy.array(lst) # трансформируем список в массив
print(arr)
print(arr.shape) #выводит размер массива
print(arr.dtype) #выводит тип элементов массива
print(arr.ndim) #число измерений массива
```

Листинг 2. File2.py

```
'''
Выполнил: Митюшин Пётр. П1-18
'''
import numpy
arr = (10--5) * numpy.random.random_sample((5,))-5 #5 случайных значений от
#-5 до 10
print(arr)
print(numpy.sin(arr)) #вычисляет тригонометрический синус элементов массива
print(numpy.cos(arr)) # вычисляет тригонометрический косинус элементов
#массива
print(numpy.tan(arr)) # вычисляет тригонометрический тангенс элементов
#массива
print(numpy.exp(arr)) #вычисляет число Эйлера для каждого элемента массива e^x,
#где x – элемента массива.
```

Листинг 3. File3.py

```
'''
Выполнил: Митюшин Пётр. П1-18
'''
import numpy
arr = (100 - 0) * numpy.random.random_sample((5,))-5 #5 случайных значений от
#0 до 100
```

```
arr = arr.astype(numpy.int64) # возвращает копию массива преобразованного к
#типу int64
print(arr)
print(arr.sum()) # Сумма всех элементов массива
print(arr.min()) # Минимальный элемент массива
print(arr.max()) # Максимальный элемент массива
print(arr.mean())# Среднее значение массива
print(arr.std()) # Стандартное отклонение
print(numpy.median(arr)) # Медиана
print(arr > 10)
tf = arr > 10
print(tf)
```

Листинг 4. File4.py

```
'''
Выполнил: Митюшин Пётр. П1-18
'''
import numpy
arr = (100 - 0) * numpy.random.random_sample((5,))-5 # 5 случайных значений
# от 0 до 100
arr = arr.astype(numpy.int64) # Превращение массива к типу int64
arr = numpy.insert(arr, 2, -20) # Вставка элемента в определённую позицию
print(arr)
arr = numpy.delete(arr, 2) # Удаление элемента
print(arr)
arr2 = numpy.sort(arr) # Сортировка массива
print(arr2)
print("arr bytes size: ", arr.itemsize) # Размер каждого элемента в байтах
print("arr2 bytes size: ", arr2.itemsize)
arr = numpy.concatenate((arr, arr2)) # Добавляет 2 массив в конец 1
print(arr)
print("arr bytes size: ", arr.itemsize)
```

Листинг 5. File6.py

```
'''
Выполнил: Митюшин Пётр. П1-18
'''
import numpy
lst_x = [i for i in range(1, 9)]
lst_y = [i for i in range(1, 9)]
x = numpy.array(lst_x, float)
y = numpy.array(lst_y, float)
print(x > y)
print(x == y)
print(x < y)
print(numpy.where(x != 0, 1 / x, x)) #если x != 0, 1/x иначе выводим просто x
a = numpy.array([2, 4, 5, 6, 8], float)
b = numpy.array([0, 0, 1, 3, 2, 1], int)
print(a.take(b))
```

2.5. Техника работы с библиотекой Matplotlib

Листинг 1. File1.py

```
'''
Выполнил: Митюшин Пётр. П1-18
'''
from matplotlib import pyplot as plt
x = [1, 5, 10, 15, 20]
y1 = [1, 7, 3, 5, 11]
y2 = [i**2 + 1 for i in y1]
y3 = [i**2 + 1 for i in y2]
plt.figure(figsize=(12, 7)) # создаёт числовые оси
plt.subplot(2, 2, 1) # получение фигуры и оси
plt.plot(x, y1, '-') # рисует график
plt.subplot(2, 2, 2)
plt.plot(x, y2, '--')
plt.subplot(2, 2, 3)
plt.plot(x, y3, '-.')
plt.savefig('grafig.png') # сохраняет в виде изображения
plt.show() # вывод на экран
```

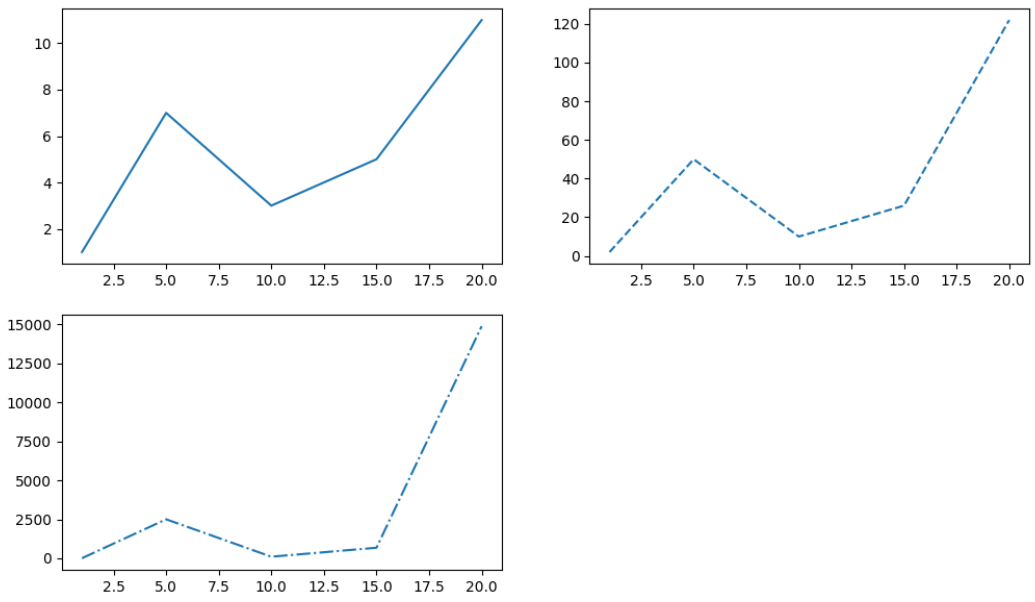


Рис.1. Демонстрация трёх графиков

Листинг 2. File2.py

```
'''

Выполнил: Митюшин Пётр. П1-18

'''

from matplotlib import pyplot as plt

x = [i for i in range(1, 20)]
y1 = [i ** 2 for i in x]
y2 = [i ** 3 for i in x]
y3 = [i ** 4 for i in x]

plt.plot(x, y1) # рисует график
plt.plot(x, y2)
plt.plot(x, y3)

plt.savefig('grafig.png') # сохраняет в виде изображения
plt.show()# вывод на экран
```

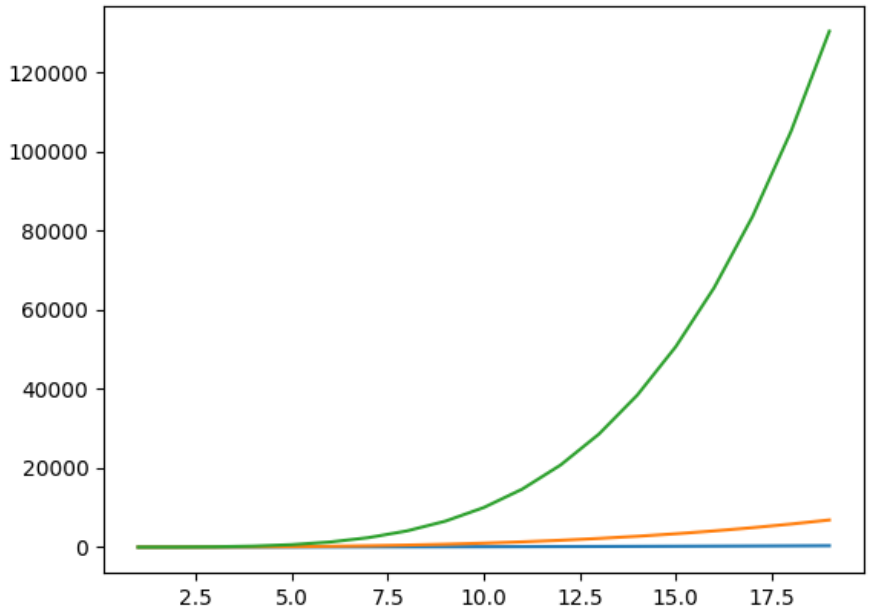


Рис.2. Демонстрация трёх разных функций

Листинг 3. File3.py

```
'''

Выполнил: Митюшин Пётр. П1-18

'''

from matplotlib import pyplot as plt
import numpy

fig, ax = plt.subplots()# получение фигуры и оси
```

```
x = numpy.array([-3, -2, -1, 0.1, 1, 2, 3])
y = numpy.array([9, 4, 1, 0.1, 1, 4, 9])
ax.plot(x, y) # рисует график
ax.plot(x, numpy.sin(x), color="blue", linestyle='-')
ax.plot(x, x + 4, color="yellow", linestyle='--')
ax.plot(x, x + 2, color="g", linestyle=':')
ax.plot(x, numpy.cos(x), color="#FF0000", linestyle='-.')
ax.plot(x, x, color="#FFDD44", linestyle='--')
ax.plot(x, numpy.tan(x), color="#FFFFFF", linestyle='-.')
ax.plot(x, x + 10, color="#000080", linestyle='--')
plt.savefig('grafig.png') # сохраняет в виде изображения
plt.show()# вывод на экран
```

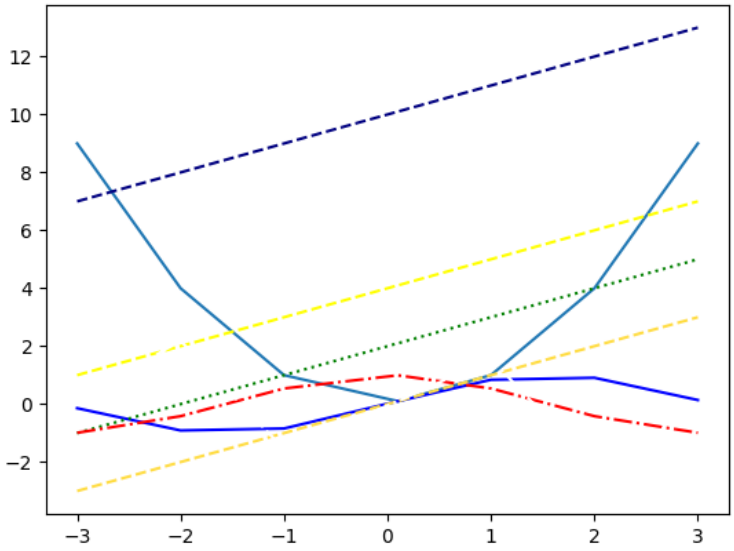


Рис.3. Работа с цветами графиков

Листинг 4. File4.py

```
'''

Выполнил: Митюшин Пётр. П1-18

'''

from matplotlib import pyplot as plt

fruits = ["apple", "peach", "orange", "bannana", "melon"]
number_sold_fruits = [35, 52, 43, 34, 17]
plt.bar(fruits, number_sold_fruits) # построение гистограммы
plt.title('Number of sold fruits') # заголовок
plt.xlabel('Fruit') # название оси x
plt.ylabel('Number of sold') # название оси y
plt.savefig('grafig.png') # сохраняет в виде изображения
plt.show()# вывод на экран
```

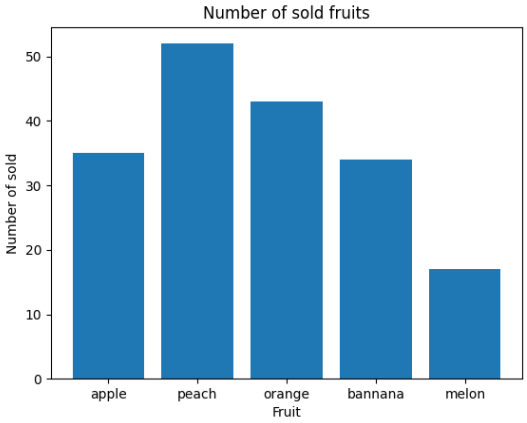


Рис.4. Гистограмма

2.6. Элементы работы с библиотекой PyQt

Листинг 1. lesson.py

```
'''
Выполнил: Митюшин Пётр. П1-18
'''
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QApplication, QMainWindow
import sys
def main():
    app = QApplication(sys.argv) #объект приложения (экземпляр QApplication)
    window = QMainWindow()
    window.setWindowTitle("Text")# устанавливает заголовок окна
    window.setGeometry(350, 250, 350, 250) #два парамитра для ширины обекта
    # следуюшие два параметра размер экрана
    main_text = QtWidgets.QLabel(window) #указываем к какому окну будет
#принадлежать объект
    main_text.setText("Just text") #задаем текст на экране
    main_text.move(100, 100) #Координаты текста на экране
    btn = QtWidgets.QPushButton(window)#создаёт кнопку на экране
    btn.move(70, 150)#Координаты книпки на экране
    btn.setText("Press me")#Текст книпки
    window.show() #отображает виджет на экране
    sys.exit(app.exec_())
if __name__ == "__main__":
    main()
```

Листинг 2. lesson2.py, lesson2.ui

```
'''
Выполнил: Митюшин Пётр. П1-18
'''
from PyQt5 import QtWidgets, uic
from PyQt5.QtWidgets import QApplication, QMessageBox, QMainWindow, QWidget
import sys
import sqlite3

nameDB = "database_grade.db"
```

```
class Database:
    def __init__(self, nameDB):
        self.db = sqlite3.connect(nameDB)
        self.cursor = self.db.cursor()

    def create_table(self):
        self.cursor.execute("CREATE TABLE IF NOT EXISTS 'students'(\
                             number_book INTEGER PRIMARY KEY,\
                              name_student TEXT,\
                              grade INTEGER)")
    def save(self):
        self.db.commit()

    def close(self):
        self.db.close()

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__() #вызываем конструктор QMainWindow
        uic.loadUi('lesson2.ui', self)
        self.show()
        self.pushButton.clicked.connect(self.editInDatabase) #потом допиши
#считывание из edli при нажатии на кнопку

    def editInDatabase(self):
        full_name = self.lineFull_name.text()
        book_number = self.lineNumber_book.text()
        grade = 0
        if (self.radioButton.isChecked()):
            grade = 2
        elif (self.radioButton_2.isChecked()):
            grade = 3
        elif (self.radioButton_3.isChecked()):
            grade = 4
        elif (self.radioButton_4.isChecked()):
            grade = 5
        db = Database(nameDB)
        try:
            db.cursor.execute("INSERT INTO 'students' VALUES (?, ?, ?)",
                              (book_number, full_name, grade))
            db.save()
        except sqlite3.Error:
            QMessageBox.about(self, "Message", "Номер зачётки не может быть\
             одинаковым")

app = QtWidgets.QApplication(sys.argv)
db = Database(nameDB)
db.create_table()
window = MainWindow()
db.close()
sys.exit(app.exec_())
```

Листинг 3. File2.py

```
'''

Выполнил: Митюшин Пётр. П1-18

'''

from PyQt5 import QtWidgets
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QLabel,
QLineEdit, QPushButton, QFileDialog
import sqlite3
import sys

nameDB = "database.db"
class Database:
    def __init__(self, nameDB):
        self.db = sqlite3.connect(nameDB) # Создаем соединение с нашей базой
#данных
        self.cursor = self.db.cursor() # Создаем объект который делает
#запросы и получает их результаты
        self.cursor.execute("CREATE TABLE IF NOT EXISTS 'users' (\
                            login TEXT,\
                            password TEXT)") # Делаем запрос к базе данных,
#используя обычный SQL-синтаксис
    def saveDatabase(self):
        self.db.commit() # Сохраняем транзакцию для вносения изменений в базу
#данных

class Window(QMainWindow):
    def __init__(self):
        QMainWindow.__init__(self)

        bar = self.menuBar() #строка меню
        operation_db = bar.addMenu("File")

        create = operation_db.addAction("Create") #Создаем кнопку
        create.triggered.connect(self.createDatabase) #вызываем функцию при
#нажатии

        open = operation_db.addAction("Open")
        open.triggered.connect(self.openDatabase)

        exit = bar.addAction("Exit")
        exit.triggered.connect(self.exit)

        self.setMinimumSize(QSize(320, 140)) # Установка минимального размера
        self.setWindowTitle("Title") # Текст загаловка окна

        self.loginLabel = QLabel(self) #Создаем QLabel
        self.loginLabel.setText('Name: ') # Текст лейбла
        self.loginLabel.move(20, 15)  # Координаты для label

        self.loginEdit = QLineEdit(self) # Создаем поля для ввода логина
        self.loginEdit.move(80, 20) # Выстовляем по координатам
#местонахождение на экране поля для ввода
        self.loginEdit.resize(200, 20) #Размер поля для ввода
```

```
        self.passwordLabel = QLabel(self)
        self.passwordLabel.setText('Password: ')
        self.passwordLabel.move(20, 55)  # Координаты для label

        self.passwordEdit = QLineEdit(self) # Создаем поля для ввода пароля
        self.passwordEdit.move(80, 60)
        self.passwordEdit.resize(200, 20)  # Размер поля для ввода

        reg_button = QPushButton('Register', self)
        reg_button.resize(100, 20) # Размер кнопки
        reg_button.move(180, 90) # Местоположение кнопки на окне

        db = Database(nameDB)
        reg_button.clicked.connect(self.editInDatatbase) # При нажатии на
#кнопку выполняется функция editInDatatbase

    def editInDatatbase(self):
        db = Database(nameDB)
        db.cursor.execute("INSERT INTO 'users' (login, password) VALUES (?,\
?)", (self.loginEdit.text(), self.passwordEdit.text())) # Заносим в БД логин
#и пароль
        db.saveDatabase()

    def createDatabase(self): #Вызывает диалоговое окно файлового менеджера
#для создание БД
        global nameDB
        nameDB = QFileDialog.getSaveFileName(self, "Выберите базу данных",
".db")[0] # Название файла
        if (nameDB[:4:-1] != 'bd.'): #если расширение файла не указано
            nameDB += '.db' #прибавляет расширения файла БД
        db = Database(nameDB)

    def openDatabase(self): #Вызывает диалоговое окно файлового менеджера для
#выбора какую БД открыть
        global nameDB
        nameDB = QFileDialog.getOpenFileName(self, "Выберите базу данных")[0]
# Название файла
        db = Database(nameDB)

    def exit(self):
        sys.exit(app.exec_())

def main():
    app = QApplication(sys.argv)
    window = Window()
    window.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```

2.7. Элементы работы с библиотекой PyGame

Листинг 1. MiniGameSquare.py

```
'''
Выполнил:Андрей Слепов. П1-18
'''
import pygame
from random import randint as rnd
pygame.init()
display_width = 900
display_height = 900
hero_x = 100
hero_y = 100
hero_width = 35
hero_height = 35
hero_color = (125, 125, 125)
speed = 2
display = pygame.display.set_mode((display_width, display_height))
pygame.display.set_caption('MyGame')
def DrawEvent(): # При попадании в определённые координаты меняется № # цвет
    global hero_color
    if hero_x >= 400 and hero_y >= 400:
        pygame.draw.rect(display, (rnd(1,255), rnd(1,255),\ rnd(1,255)),
(rnd(1,750), rnd(1,750), rnd(1, 100), rnd(1, 100)))
        pygame.draw.rect(display, (rnd(1, 255), rnd(1, 255), rnd(1,\ 255)),
(rnd(1, 750), rnd(1, 750), rnd(1, 100), rnd(1, 100)))
        hero_color = (rnd(1, 255), rnd(1, 255), rnd(1, 255))
def DrawLab(): # Отсрисовка прямоугольника
    pygame.draw.rect(display, (0, 0, 0), (250, 250, 20, 20))
def Dead():
    if hero_x in range(217, 270) and hero_y in range(216, 270):
        pygame.quit()
        quit()
def Move(): # Движение
    global hero_x, hero_y
```

```
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT] and hero_x > 10:
        hero_x -= speed
    if keys[pygame.K_RIGHT] and hero_x < 850:
        hero_x += speed
    if keys[pygame.K_UP] and hero_y > 10:
        hero_y -= speed
    if keys[pygame.K_DOWN] and hero_y < 850:
        hero_y += speed


def Update(): # Основная функция
    game = True
    while game:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
        display.fill((230, 230, 230))
        DrawEvent()
        Move()
        DrawLab()
        Dead()
        pygame.draw.rect(display, hero_color, (hero_x, hero_y,\ hero_width,
hero_height))
        pygame.display.update()
Update()
```

Раздел 3. Разработка проекта с графическим интерфейсом.

3.1. Изучение входной и выходной документации

На вход подается документ с порядковым номером книги, её название, автор и цена. Удаление данных из документа производится на основе порядкового номера.

Входные данные:

1. Документ на запись
    - 1.1 Данные книги
        - 1.1.1 Порядковый номер книги
        - 1.1.2 Название книги
        - 1.1.3 Автор
        - 1.1.4 Цена
    - 1.2 Данные для удаления по порядковому номеру
        - 1.1.1 Порядковый номер книги
    - 1.3 Добавление данных в документ
        - 1.1.1 Название книги
        - 1.1.2 Автор
        - 1.1.3 Цена

Выходные данные:

1.1 Демострация содержимого документа

- 1.1.1 Порядковый номер книги
- 1.1.2 Название книги
- 1.1.3 Автор
- 1.1.4 Цена

3.2 Разработка требований к проекту. Построение диаграммы использования.

Требования к проекту:

1) Разработать программу «Книжный магазин» с графическим интерфейсом.

2) Использовать базу данных (sqlite3).

3) Реализовать добавление / удаление / редактирование записей.

4) Реализовать вывод содержимого базы данных в табличном виде.

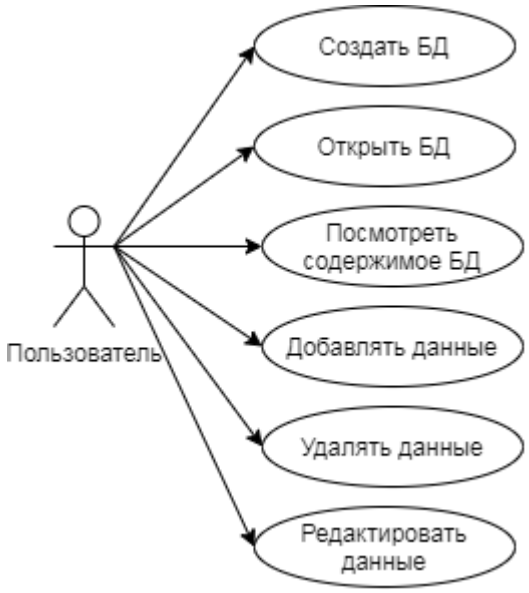


Рис.1 Диаграмма использования.

3.3 Разработка сценария проекта.

1) Запускаем приложение.

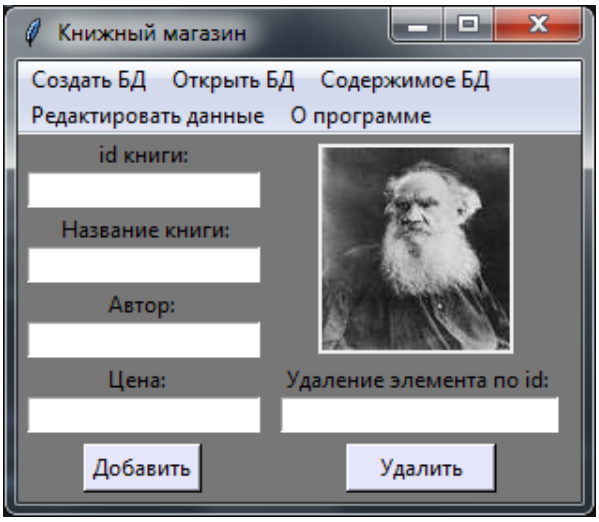


Рис.2 Главный экран.

2) Если нет БД, то нажимаем на Создать БД, иначе нажимаем на открыть БД и выбираем путь где лежит БД.

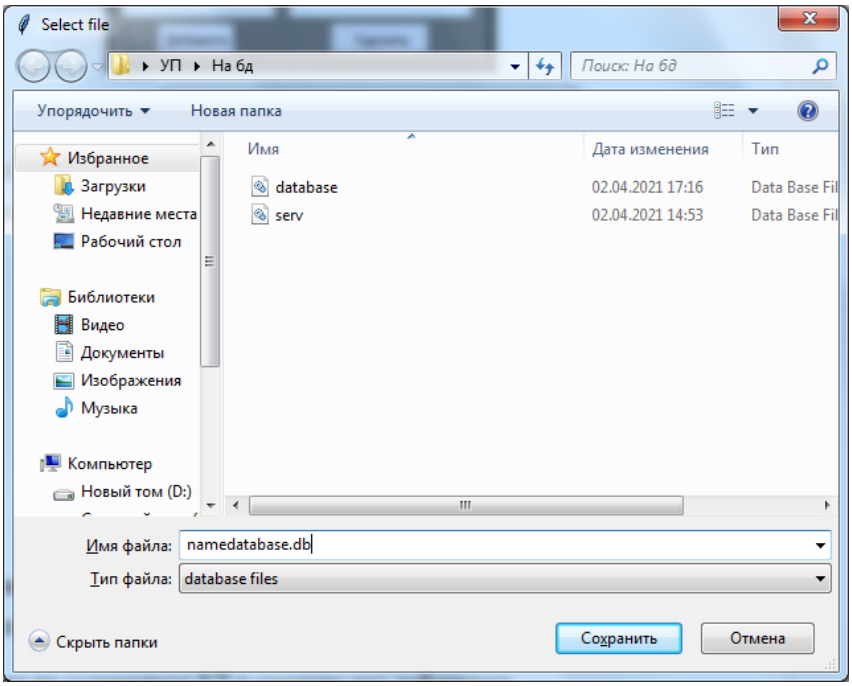


Рис.3 Создание БД.

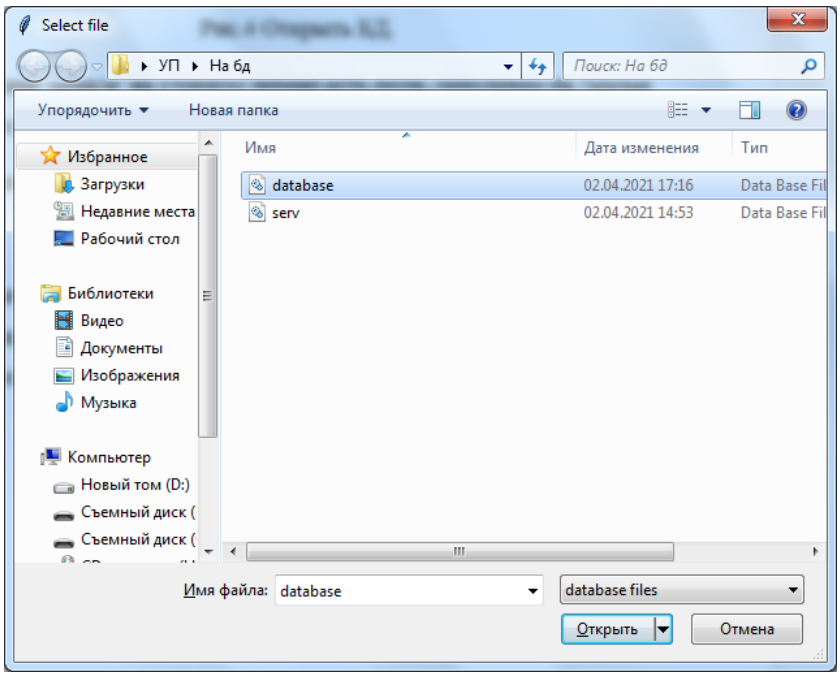


Рис.4 Открыть БД.

3) Добавление записи: на главном экране есть поля, заполняем их (кроме поля удалить по id) и нажимаем на кнопку добавить.

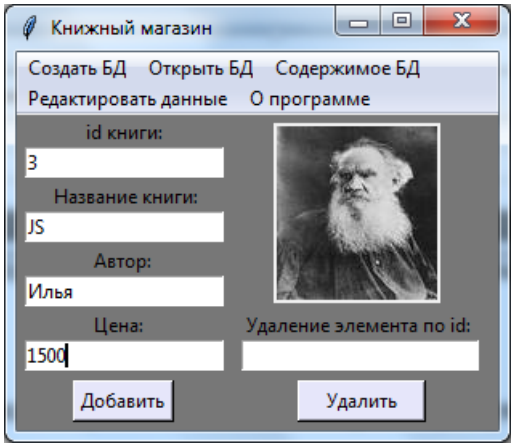
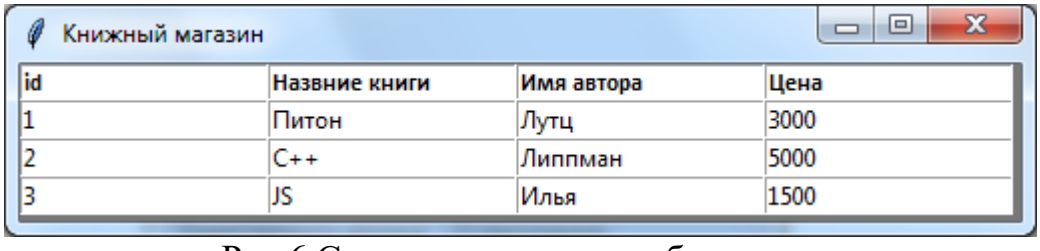


Рис.5 Добавление записи



id	Назвние книги	Имя автора	Цена
1	Питон	Лутц	3000
2	C++	Липпман	5000
3	JS	Илья	1500

Рис.6 Содержимое после добавления.

5) Нажимаем на редактировать данные, выбираем нужную книгу, нажимаем на кнопку изменить и вводим в полях что хотим изменить (для корректной работы программы надо заполнить все поля).

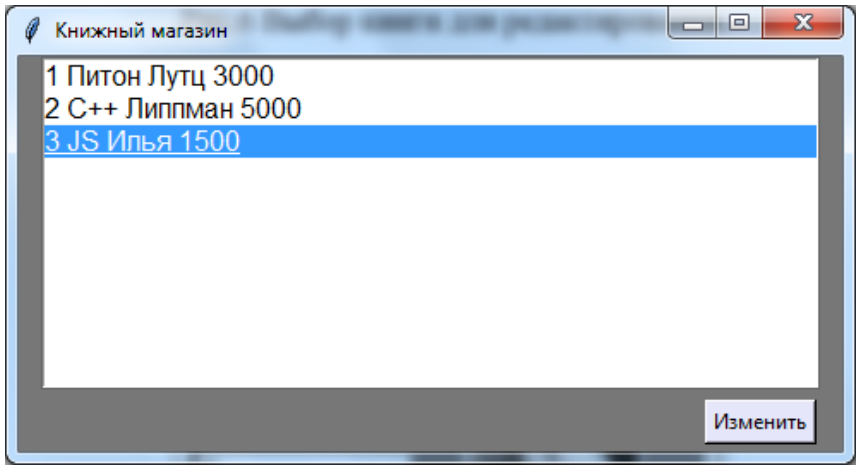


Рис.7 Выбор записи для редактирования.

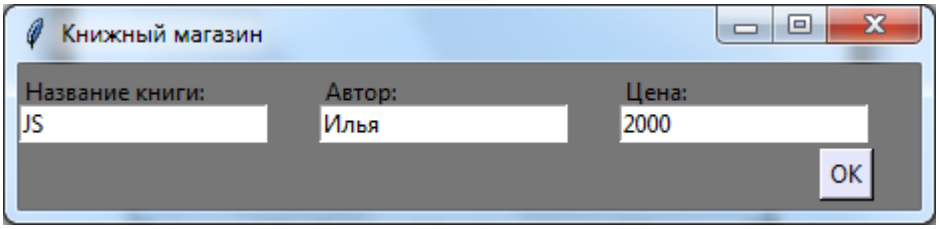


Рис.8 Редактирование записи.

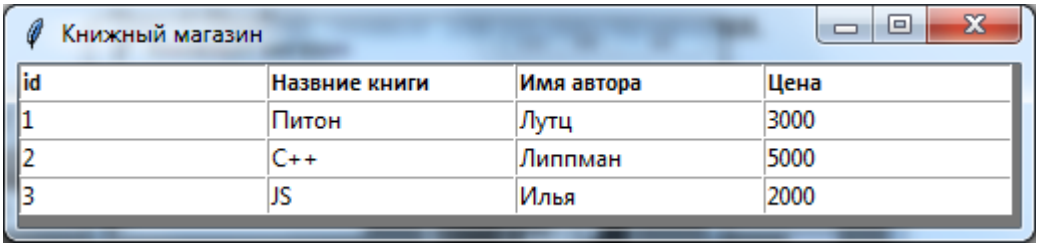


Рис.9 Просмотр содержимого после изменений.

3.4 Построение диаграммы классов.

В данном разделе предоставлена диаграмма классов нашего приложения (рисунок 10).

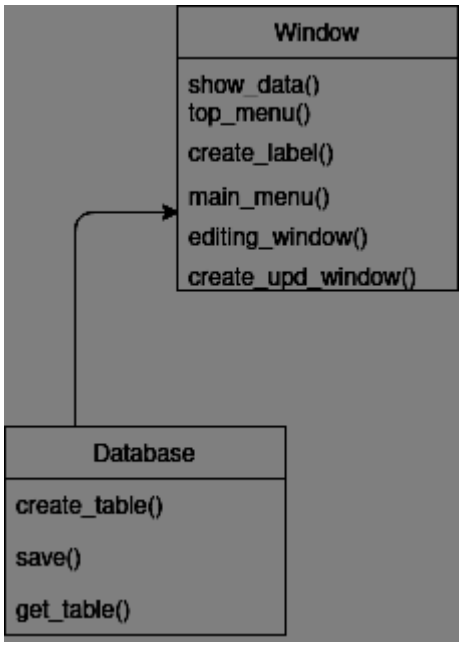


Рис.10 Диаграмма классов

3.5 Разработка базы данных.

В базу данных этого проекта входит одна таблица, в которую входят 4 столбца: book_id, book_name, author_name и price.

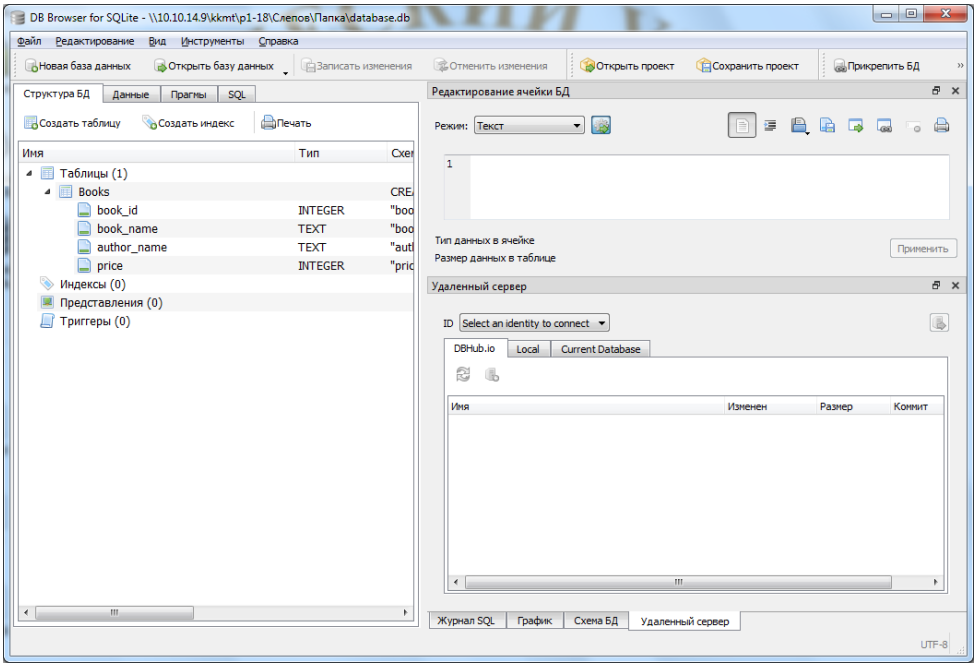


Рис.10 Структура базы данных.

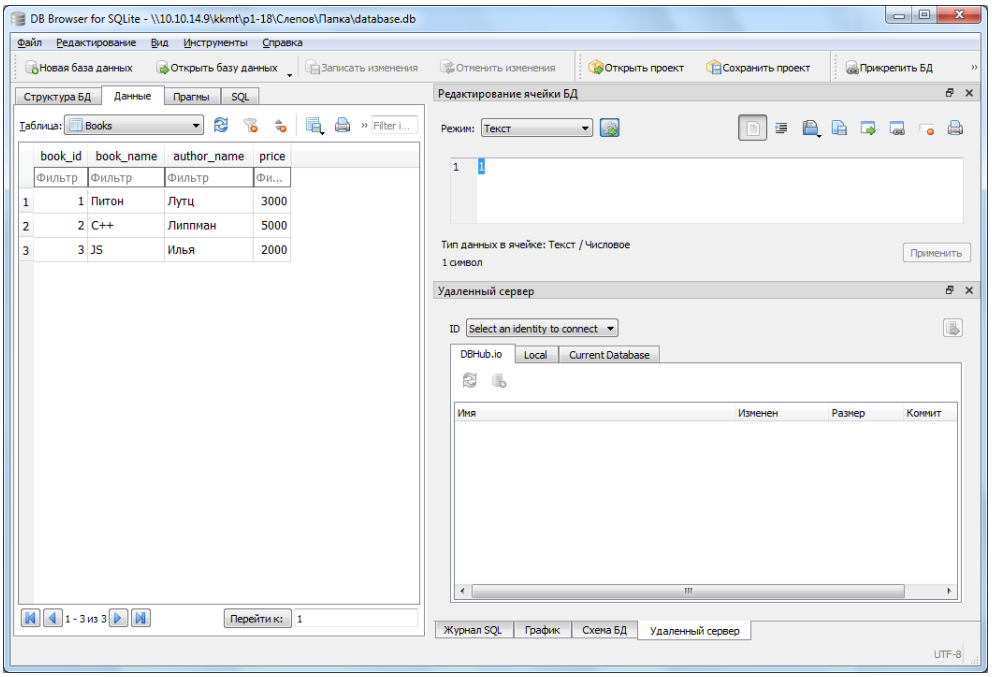


Рис.11 Заполнение базы данных.

3.6 Разработка главного модуля.

Приложения: main.py, database.db.

```
from tkinter import *
from tkinter import messagebox, PhotoImage
from tkinter import filedialog
import sqlite3
WINDOW_TITLE = "Книжный магазин"
WINDOW_GEOMERTY = "300x215"
LABEL_ID_BOOK = "id книги: "
LABEL_NAME_BOOK = "Название книги: "
LABEL_AUTHOR = "Автор: "
LABEL_PRICE = "Цена: "
LABEL_DEL_BY_ID = "Удаление элемента по id: "
TEXT_BUTTON_EDIT = "Добавить"
TEXT_BUTTON_DEL = "Удалить"
TEXT_CREATE_FILE = "Создать БД"
TEXT_OPEN_FILE = "Открыть БД"
SHOW_DATA = "Содержимое БД"
NAMETABLE = "Books"
nameDB = "database.db"
BUTTON_COLOR = "#E6E6FA"
WINDOW_COLOR = "#777777"
ACT_BUTTON_COLOR = "#B0C4DE"
LABEL_BG = "#777777"
UPDATE_DATA = "Редактировать данные"
class Database:
    def __init__(self, nameDB):
        self.db = sqlite3.connect(nameDB)  # Если файл не был создал или
открыт по дефолту создастся файл database.txt
        self.cursor = self.db.cursor()     # Курсор для обращения к БД
    def create_table(self, table):  # Создание таблицы
        self.cursor.execute(f"CREATE TABLE IF NOT EXISTS '{table}' (\
                            book_id INTEGER PRIMARY KEY, \
                            book_name TEXT, \
```

```
                                author_name TEXT, \
                                price INTEGER)")
    def save(self):     # Сохранение изменений
        self.db.commit()
    def get_table(self, table):     # Печать таблицы в консоль (для отладки)
        return [i for i in self.cursor.execute(f"SELECT * FROM {table}")]
class Window:
    def __init__(self):
        self.window = Tk()
        self.window.title(WINDOW_TITLE) # Заголовок программы
        self.window.geometry(WINDOW_GEOMERTY) # Размеры программы
        self.window["bg"] = WINDOW_COLOR # Цвет фона программы
        self.id_book = StringVar()  # Присваивает строку из виджета поля для
#ввода текста
        self.name_book = StringVar()
        self.author = StringVar()
        self.book_price = StringVar()
        self.id_book_del = StringVar()  # примечание разработчика переменная
#возможно не нужена
        self.background_image = PhotoImage(file="/home/andrey/Рабочий \
стол/УП/На бд/LNT.png") # Изображение на фон
        self.upd_book_name = StringVar()
        self.upd_author = StringVar()
        self.upd_price = StringVar()
        self.top_menu()
        self.main_menu()
    def show_data(self):      # Экран "Содержимое БД"
        db = Database(nameDB)
        win = Toplevel(self.window)
        win.geometry('500x250')
        win['bg'] = WINDOW_COLOR
        titles_columns = ['id', 'Назвние книги', 'Имя автора', 'Цена']
        for i in range(len(titles_columns)):
            titles = Entry(win, font='Calibri 9 bold', width=20)
            titles.insert(0, titles_columns[i])
```

```
            titles.grid(row=0, column=i)
        book_list = [i for i in db.cursor.execute(f"SELECT * FROM \
{NAMETABLE}")]
        col = len(book_list)
        row = 4
        for i in range(col):
            for j in range(row):
                b = Entry(win)
                b.insert(0, book_list[i][j])
                b.grid(row=i+1, column=j)
    def top_menu(self): # Меню
        menu = Menu(self.window)  # создаем объект меню для секций
        menu.add_cascade(label=TEXT_CREATE_FILE, command=creat_db)
        menu.add_cascade(label=TEXT_OPEN_FILE, command=open_db)
        menu.add_cascade(label=SHOW_DATA, command=self.show_data)
        menu.add_cascade(label=UPDATE_DATA, command=self.editing_window)
        menu.add_cascade(label="О программе", command=lambda: \
messagebox.showinfo("О программе", "Разработал Митюшин Пётр и Андрей \
Слепов."))  # Меню о программе добавляется на панель меню
        self.window.config(menu=menu)  # доступ к атрибуту объекта после его
#инициализации
    def create_label(self, text_label, x_label, y_label):   # Функция
#принимает строку и координаты и создаёт текст
      Label(text=text_label, bg=LABEL_BG).place(x=x_label, y=y_label)
#Отображает текст в окне по переданным координатам
    def main_menu(self):  # Функция отображения полей ввода в главном меню
        self.create_label(LABEL_ID_BOOK, 40, 0)  # Отображаем текст
#хранящейся в переменной label_id_book
        idb = Entry(width=20, textvariable=self.id_book).place(x=5, y=20)
#Определим элемент Entry который представляет собой поле для ввода текста
        self.create_label(LABEL_NAME_BOOK, 20, 45)
        book = Entry(width=20, textvariable=self.name_book).place(x=5, y=65)
        self.create_label(LABEL_AUTHOR, 45, 90)
        author = Entry(width=20, textvariable=self.author).place(x=5, y=110)
        self.create_label(LABEL_PRICE, 45, 140)
        price = Entry(width=20, textvariable=self.book_price).place(x=5, \
y=160)
```

```
        Button(self.window, text=TEXT_BUTTON_EDIT, command=insert_data,\
bg=BUTTON_COLOR, activebackground=ACT_BUTTON_COLOR).place(x=35, y=185)

        self.create_label(LABEL_DEL_BY_ID, 170, 130)

        delete = Entry(width=24, textvariable=self.id_book_del).place(x=175,\
y=160)

        Button(self.window, text=TEXT_BUTTON_DEL, width=10,\
command=delete_book_by_id, bg=BUTTON_COLOR,\
activebackground=ACT_BUTTON_COLOR).place(x=195, y=185)



        background = Label(self.window, image=self.background_image,\
width=100, height=108).place(x=180, y=5)

    def editing_window(self): # Экран выбора записи для редактирования

        global lb

        global win_update

        db = Database(nameDB)

        win_update = Toplevel(self.window)

        win_update.geometry('480x230')

        win_update['bg'] = WINDOW_COLOR

        book_list = [i for i in db.cursor.execute(f"SELECT * FROM \
{NAMETABLE}")]

        lb = Listbox(win_update, width=50, font=14)

        for book in book_list:

            book_str = " ".join([str(field) for field in book])

            lb.insert(END, book_str)

        lb.pack()

        change_button = Button(win_update, text="Изменить", bg=BUTTON_COLOR,
activebackground=ACT_BUTTON_COLOR,command=self.create_upd_window).place(x=40,
y=200)

    def create_upd_window(self):    # Экран редактирования записи

        global win_update

        upd_window = Toplevel(win_update) # Дочернее окно

        upd_window.geometry('450x100')

        upd_window['bg'] = WINDOW_COLOR

        Label(upd_window, text=LABEL_NAME_BOOK, bg=LABEL_BG).place(x=0, y=3)
# Текст

        Entry(upd_window, width=20,\
textvariable=self.upd_book_name).place(x=0, y=20) # Поле ввода

        Label(upd_window, text=LABEL_AUTHOR, bg=LABEL_BG).place(x=150, y=3)
```

```
        Entry(upd_window, width=20,\
textvariable=self.upd_author).place(x=150, y=20)

        Label(upd_window, text=LABEL_PRICE, bg=LABEL_BG).place(x=300, y=3)

        Entry(upd_window, width=20, \

        textvariable=self.upd_price).place(x=300, y=20)

        done_button = Button(upd_window, text="OK", bg=BUTTON_COLOR,\
activebackground=ACT_BUTTON_COLOR,command=update_data).place(x=400, y=42)

def update_data():  # Редактирование данных

    db = Database(nameDB)

    try:

        selection = lb.curselection()

        select = lb.get(selection[0])

        select_item = list(select.split())

    except IndexError:

        messagebox.showinfo("Error", "You are not select book")

    if(str(window.upd_price.get()).isdigit()):

        db.cursor.execute("""UPDATE `books`  # Запрос редактирования данных

                          SET `book_name` = (?),

                          `author_name` = (?),

                          `price` = (?)

                          WHERE `book_id` = (?)

                    """, (window.upd_book_name.get(),
window.upd_author.get(), window.upd_price.get(), select_item[0]))

    else:

        messagebox.showerror("Error", "Price can only contain numbers")

    db.save()

    win_update.destroy()

def creat_db():  # При вызове функции открывается диалоговое окно, где в поле
имя файлов надо ввести имя файла которое вы хотите создать

    global nameDB

    path_db = filedialog.asksaveasfilename(initialdir="/", title="Select \
file", filetypes=(("database files", "*.db"), ("all files", "*.*")))

    nameDB = path_db

    db = Database(nameDB)

    db.create_table(NAMETABLE)

def delete_book_by_id(): # Удаление по id
```

```
    db = Database(nameDB)

    cursor = db.cursor

    cursor.execute('DELETE FROM {0} WHERE book_id = {1}'.format(NAMETABLE,
window.id_book_del.get()))

    db.save()

def insert_data():

    db = Database(nameDB)

    if (str(window.id_book.get()).isdigit() and\
str(window.book_price.get()).isdigit()):

        db.cursor.execute("INSERT INTO books (book_id, book_name,\
author_name, price) VALUES (?,?,?,?)",

            (window.id_book.get(), window.name_book.get(),
window.author.get(), window.book_price.get()))

        db.save()

    else:

        messagebox.showerror("Error", "Id and price can only contain\
numbers")

def open_db():  # При вызове функции открывается диалоговое окно, где
#пользователь выбирает какой файл ему нужно открыть

    global nameDB

    path_db = filedialog.askopenfilename(initialdir="/", title="Select file",
filetypes=(("database files", "*.db"), ("all files", "*.*")))  # ор является
#путём и его надо передать в функции добавления и удаления из файла

    nameDB = path_db

window = Window()

mainloop()  # Отображает главное окно со всеми виджетами
```

3.7 Тестирование и отладка

В ходе написания проекта при запуске программы были получены ошибки:

```
C:\Python38\python.exe Z:/Слепов/Папка/File2-Копия.py
Traceback (most recent call last):
  File "Z:/Слепов/Папка/File2-Копия.py", line 207, in <module>
    window = Window()
  File "Z:/Слепов/Папка/File2-Копия.py", line 62, in __init__
    self.main_menu()
  File "Z:/Слепов/Папка/File2-Копия.py", line 106, in main_menu
    self.create_labe(LABEL_PRICE, 45, 120)
AttributeError: 'Window' object has no attribute 'create_labe'
```

Рис.12 Сообщение об ошибке

При поиске и исправления ошибок программа запустилась.

```
C:\Users\P1-18\PycharmProjects\untitled\venv\Scripts\python.exe D:/Папка/main.py
BUILD SUCCESSFUL!
```

Рис.13 Исправление ошибки

3.8 Разработка документации к проекту.

Назначение программы.

Функциональное назначение.

Данная программа предназначена для добавления, удаления, редактирования и просмотра записей. Главный плюс этой программы – удобный интерфейс для пользователя. При этом программа взаимодействует с базой данных.

Эксплуатационное назначение программы.

С помощью «Книжный магазин» пользователь может отслеживать количество книг, редактировать, например, цену книги или её название, добавлять новые книги или удалять их.

Состав функций:

1) Проверка попадания введенной даты в промежуток;
2) Запись информации о посещениях;
3) Добавление информации о появлении студента;
4) Вывод списка опоздавших;
5) Поиск студента с наибольшим количеством опозданий;
6) Поиск студента с наибольшим временем опозданий.

Условия выполнения программы

Минимальные требования для аппаратных средств.

Для работы программы «Книжный магазин» требуется:

- ОС: Windows XP, 7, Vista, 8, 8.1, 10
- Процессор: Intel Celeron 1800 MHz
- Оперативная память: 256 MB ОЗУ
- Видеокарта: Intel HD Graphics
- DirectX: Версии 9.0

- Место на диске: 15 MB

Минимальный состав программных средств

Системные программные средства, используемые специальным программным обеспечением «Муром», должны быть представлены локализованной версией операционной системы Windows XP, Windows Vista или Windows 7

Требования к персоналу (пользователю)

Пользователь программы (оператор) должен обладать начальными навыками работы с компьютером.