

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

Выполнил студент:

Карандашев Н.К.

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

Попов В. Н.

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

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

Королев, 2019

Содержание отчёта

[**Раздел 1. Техника решения задач с использованием структурного программирования.** 3](#_Toc30197640)

[1.1. Установка интерпретатора Python 3 и настройка окружения 3](#_Toc30197641)

[1.2. Техника работы в командной строке и среде IDLE 8](#_Toc30197642)

[1.3. Техника работы с линейными и разветвляющимися программами 10](#_Toc30197643)

[1.4. Техника работы с циклическими программами, цикл while 13](#_Toc30197644)

[1.5. Техника работы с числами 16](#_Toc30197645)

[1.6. Техника работы со строками 19](#_Toc30197646)

[1.7. Техника работы со списками 22](#_Toc30197647)

[1.8. Техника работы с циклом for и генераторами списков 24](#_Toc30197648)

[1.9. Техника работы с функциями 27](#_Toc30197649)

[1.10. Техника работы со словарями 31](#_Toc30197650)

[1.11. Техника работы с множествами 35](#_Toc30197651)

[1.12. Техника работы с кортежами 37](#_Toc30197652)

[1.13. Техника работы с файлами 38](#_Toc30197653)

[**Раздел 2. Техника решения задач с использованием модульного и объектно-ориентированного программирования.** 45](#_Toc30197654)

[2.1. Техника работы с модулями 45](#_Toc30197655)

[2.2. Разработка проекта «Журнал посещения занятий» 47](#_Toc30197656)

[2.3. Установка и настройка среды JetBrains PyCharm 50](#_Toc30197657)

[2.4. Техника работы с библиотекой tkinter 57](#_Toc30197658)

[2.5 Доработка проекта «Журнал посещения занятий» с использованием графического интерфейса 61](#_Toc30197659)

[2.6. Техника работы с классами 67](#_Toc30197660)

[2.7. Доработка проекта «Журнал посещения занятий» с использованием классов 70](#_Toc30197661)

[2.8. Разработка документации по проекту 75](#_Toc30197662)

[**Раздел 3. Техника решения задач с использованием библиотек.** 82](#_Toc30197663)

[3.1. Техника работы с библиотекой NumPy 82](#_Toc30197664)

[3.2. Техника работы с библиотекой Matplotlib 86](#_Toc30197665)

[3.3 Техника работы с библиотекой PyQT 92](#_Toc30197666)

[3.4. Техника работы с библиотекой Pygame 98](#_Toc30197667)

[Список использованных источников 106](#_Toc30197668)

# **Раздел 1. Техника решения задач с использованием структурного программирования.**

## Установка интерпретатора Python 3 и настройка окружения

Для установки интерпретатора Python на компьютер, первое, что нужно сделать – это скачать дистрибутив. Загрузить его  можно с официального сайта, перейдя по ссылке <https://www.python.org/downloads/>



Рисунок 1. Официальный сайт Python

Порядок установки на Windows:

1. Запустить скачанный установочный файл.

2. Выбрать способ установки.



Рисунок 2. Установщик Python

3. Отметить необходимые опции установки (доступно при выборе Customize installation)



Рисунок 3. Опции установки

На этом шаге нам предлагается отметить дополнения, устанавливаемые вместе с интерпретатором Python. Выбираю:

* Documentation – установка документаций.
* pip – установка пакетного менеджера pip.
* tcl/tk and IDLE – установка интегрированной среды разработки (IDLE) и библиотеки для построения графического интерфейса (tkinter).

4. Выбираем место установки (доступно при выборе Customize installation)



Рисунок 4. Продвинутые опции установки

5. После успешной установки:



Рисунок 5. Сообщение об установке

Окружение Python представляет собой контекст, в котором выполняется код Python. Различают глобальные, виртуальные окружения и окружения Conda. Окружение состоит из интерпретатора, библиотеки и нескольких установленных пакетов. Вместе они определяют, какие языковые конструкции и синтаксис допустимы, какие возможности операционной системы доступны и какие пакеты можно использовать.

В Visual Studio для Windows есть окно **Окружения Python**, которое позволяет управлять окружениями и выбрать одно из них в качестве окружения по умолчанию для новых проектов.

Окружения, обнаруженные Visual Studio, отображаются в окне **Окружения Python**. Для открытия окна выберите команду меню **Просмотр** > **Другие окна** > **Окружения Python**.



Рисунок 6. Показать Python инструменты



Рисунок 7. Выбор версии Python

При выборе окружения в списке на вкладке **Обзор** Visual Studio отображаются различные свойства и команды для этого окружения. Например, как видно на рисунке выше, интерпретатор находится в папке C:\Python36-32. Четыре команды в нижней части вкладки **Обзор** открывают командную строку с выполняющимся интерпретатором.

Справа от каждого окружения в списке есть элемент управления, который позволяет открыть **интерактивное** окно для этого окружения.

## Техника работы в командной строке и среде IDLE

Выполняя (запуская) команду “python” в вашем терминале, вы получаете интерактивную оболочку Python.



Рисунок 8. Интерактивная оболочка Python

Существует несколько способов закрыть оболочку Python:

>>> exit()

или же

>>> quit()

Кроме того, CTRL + D закроет оболочку и вернет вас в командную строку терминала.

[IDLE](https://docs.python.org/2/library/idle.html) - простой редактор для Python, который поставляется вместе с Python.

Откройте IDLE в вашей системе выбора.

В оболочке есть подсказка из трех прямоугольных скобок:

>>>

Теперь напишите в подсказке следующий код:

>>> print("Hello, World")

Нажмите Enter .

>>> print("Hello, World")

Hello, World



Рисунок 9. Первая программа

## Техника работы с линейными и разветвляющимися программами

Приложение: Stepik\_1.12.6.py, Stepik\_1.12.2.py, Stepik\_1.10.5.py

Листинг 1. Stepik\_1.10.5.py

'''

Из передачи “Здоровье” Аня узнала, что рекомендуется спать хотя бы A часов в сутки, но пересыпать тоже вредно и не стоит спать более B часов.

Сейчас Аня спит H часов в сутки. Если режим сна Ани удовлетворяетрекомендациям передачи “Здоровье”, выведите “Это нормально”.

Если Аня спит менее A часов, выведите “Недосып”, если же более B часов, то выведите “Пересып”.

Входные данные:

6

10

8

Выходные данные:

Это нормально

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

05.09.2019

'''

a = int(input())

b = int(input())

h = int(input())

if h < a:

print('Недосып')

elif h > b:

print('Пересып')

else:

print('Это нормально')

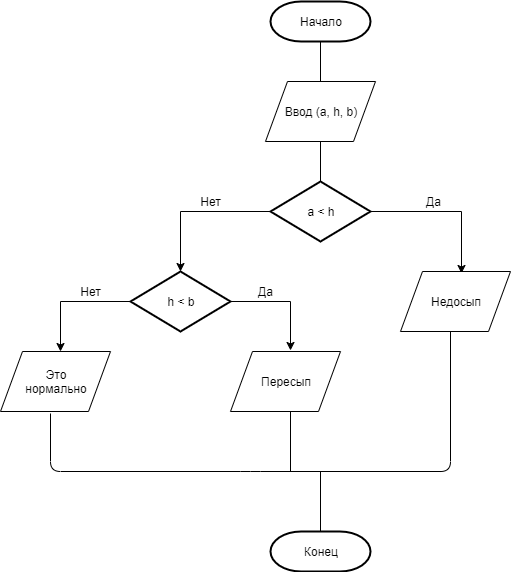


Рисунок 10. Блок-схема к листингу 1

Листинг 2. Stepik\_1.12.2.py

'''

Напишите программу, принимающую на вход целое число, которая выводит True, если переданное значение попадает

в интервал (−15,12]∪(14,17)∪[19,+∞) и False в противном случае (регистр символов имеет значение).

Входные данные:

20

Выходные данные:

True

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

05.09.2019

'''

figure = int(input())

if (12 >= figure > -15) or (17 > figure > 14) or ( figure >= 19):

print('True')

else: print('False')

Листинг 3. Stepik\_1.12.6.py

'''

В институте биоинформатики по офису передвигается робот. Недавно студенты из группы программистов написали для него программу,

по которой робот, когда заходит в комнату, считает количество программистов в ней и произносит его вслух: "n программистов".

Для того, чтобы это звучало правильно, для каждого n нужно использовать верное окончание слова.

Напишите программу, считывающую с пользовательского ввода целое число n (неотрицательное), выводящее это число в консоль

вместе с правильным образом изменённым словом "программист", для того, чтобы робот мог нормально общаться с людьми,

например: 1 программист, 2 программиста, 5 программистов.

Входные данные:

1

Выходные данные:

1 программист

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

05.09.2019

'''

num = int(input())

if num % 10 == 1 and num % 100 != 11:

print(num, 'программист')

elif num % 10 >= 2 and num % 10 <= 4 and (num % 100 < 10 or num % 100 > 20):

print(num, 'программиста')

else:

print(num, 'программистов')

## Техника работы с циклическими программами, цикл while

Приложения: while1.py, while2.py, while3.py, while4.py, while5.py, while6.py

Листинг 4. while1.py

"""

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.

Входные данные:

5

-3

8

4

0

Выходные данные:

14

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/67

"""

n = 1

sum = 0 #обнуляем сумму

while (n != 0): #условие выполняется, пока верно неравенство

n = int(input()) #ввод целых чисел

sum += n

print(sum)

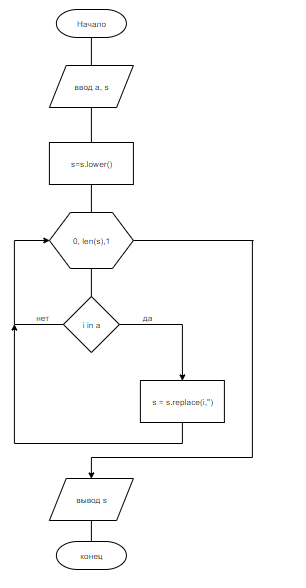


Рисунок 11. Блок-схема к листингу 4

Листинг 5. while2.py

"""

Крысолову нужно истребить всех крыс в городе, и ему нужна ваша помощь, чтобы

найти их. На вход программе подается натуральное число n и следом n строк.

Если в строке встречается подстрока "rat", выведите номер этой строки, начиная

с единицы. Если ни в одной строке на нашлось подстроки "rat", выведите -1.

Входные данные:

3

baby will cry without a rattle

rodents mutated because of radiation

rotten eggs smell bad

Выходные данные:

1

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

n = int(input())

k = 0

i = 1

while n > 0 :

s = str(input())#ввод строки

if 'rat' in s : #проверка на вхождение

k += 1 #счетчик

print(i)

i += 1

n -= 1

if k == 0 : #если ни в одной строке на нашлось подстроки "rat"

print(-1)

Листинг 6. while3.py

"""

Джон получил свою первую зарплату и хочет выгодно ей воспользоваться, то есть потратить как можно больше денег.

Он заходит в магазин и покупает все, что попадается ему под руку. Напишите программу, которая может сказать "Стоп, Джон!"

в тот момент, когда, с добавлением очередного товара в корзину, итоговая стоимость становится больше зарплаты Джона.

После этого нужно вывести стоимость всех добавленных ранее покупок.

Формат ввода: число n – сумма, которой располагает Джон. Далее идут произвольное количество строк – цен очередного товара.

Формат вывода: строка "Стоп, Джон!" и затем число – стоимость добавленных товаров.

Входные данные:

2500

500

280

120

1200

100

999

Выходные данные:

Стоп, Джон!

2200

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

n = int(input())

su = 0

while su <= n:

a = int(input()) #ввод чисел

su += a

print('Стоп, Джон!')

if su > n:

su -= a

print(su)

## Техника работы с числами

Приложения: Stepik\_1.8.8.py, Stepik\_1.8.6.py, Stepik\_1.8.7.py

Листинг 7. Stepik\_1.8.8.py

'''

Катя узнала, что ей для сна надо X минут. В отличие от Коли,

Катя ложится спать после полуночи в H часов и M минут.

Помогите Кате определить, на какое время ей поставить будильник,

чтобы он прозвенел ровно через X минут после того, как она ляжет спать.

На стандартный ввод, каждое в своей строке, подаются значения X, H и M.

Гарантируется, что Катя должна проснуться в тот же день, что и заснуть.

Программа должна выводить время, на которое нужно поставить будильник: в первой строке часы, во второй — минуты.

Входные данные:

475

1

55

Выходные данные:

9 50

Выполнил: Карандашев Н.К П1-17

Источник: https://stepik.org/courses/67

04.10.2019

'''

x = int(input())

h = int(input())

m = int(input())

a = x//60

b = x - a \* 60

m += b

z = m//60

h = h + z + a

m = m - z\*60

print(h, m)

Листинг 8. Stepik\_1.8.6.py

'''

Тимофей обычно спит ночью X часов и устраивает себе днем тихий час на Y минут.

Определите, сколько всего минут Тимофей спит в сутки.

Входные данные:

7

30

Выходные данные:

450

Выполнил: Карандашев Н.К П1-17

Источник: https://stepik.org/courses/67

4.10.2019

'''

X = int(input())

Y = int(input())

print(X\*60 + Y)

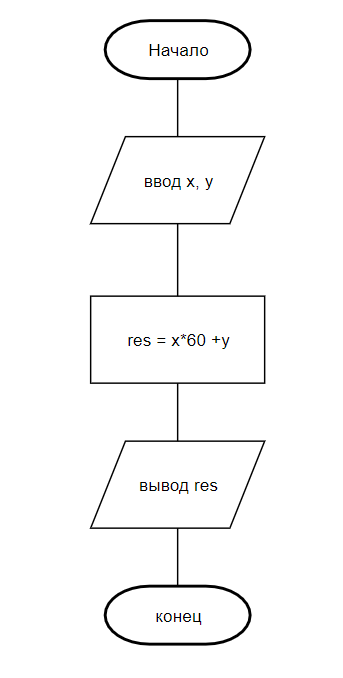


Рисунок 12. Блок-схема к листингу 8

Листинг 9. Stepik\_1.8.7.py

'''

Коля каждый день ложится спать ровно в полночь и недавно узнал,

что оптимальное время для его сна составляет X минут.

Коля хочет поставить себе будильник так, чтобы он прозвенел ровно через X минут

после полуночи, однако для этого необходимо указать время сигнала в формате часы, минуты.

Помогите Коле определить, на какое время завести будильник.

Входные данные:

480

Выходные данные:

8

0

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

4.10.2019

'''

x = int(input())

a = x//60

b = x - a\*60

print(a)

print(b)

## Техника работы со строками

Приложения: str1.py, str2.py, sr3.py, str4.py, str5.py

Листинг 10. str1.py

"""

Уберите точки из введенного IP-адреса. Выведите сначала четыре числа через пробел, а затем сумму получившихся чисел.

Входные данные:

192.168.0.1

Выходные данные:

192 168 0 1

361

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

s = [int (x) for x in input().split('.')] #ввод строки, заполнение списка, разбиение строки, где разделитель - "."

print(\*s) #распаковка списка

print(sum(s))

Листинг 11. str2.py

"""

Уберите из введенного текста все знаки препинания и переведите все буквы из верхнего регистра в нижний.

﻿Гарантируется, что в введенных текстах будут встречаться знаки препинания из набора: {".", ",", "!", "?", ":", ";", "–"}

Входные данные:

Кажется, можно, наверное, предположить, что, конечно, запятых, бесспорно, слишком много, но, разумеется,

по всем, полагаю, правилам. Верно?

Выходные данные:

кажется можно наверное предположить что конечно запятых бесспорно слишком много но разумеется по всем

полагаю правилам верно

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

a = {".", ",", "!", "?", ":", ";", "–"}

s = input()

s = s.lower() #перевод в нижний регистр

for i in s:

if i in a:

s = s.replace(i,'')#убираем знаки ипрепинания

print(s)

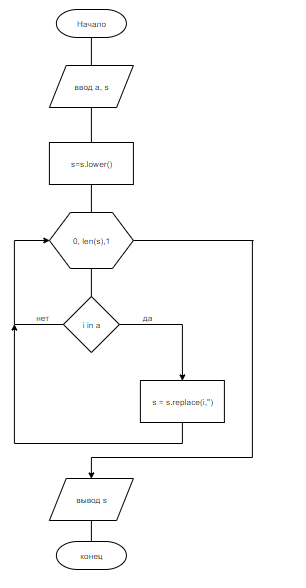


Рисунок 13. Блок-схема к листингу 11

Листинг 12. str3.py

"""

Вася составляет довольно примитивный шифр, планируя взлом Роскомнадзора. Одно из таких сообщений

попало в руки сотруднику организации. Помогите ему предотвратить диверсию и расшифровать сообщение.

Выведите каждое n-ое слово в тексте, начиная с k; n и k вводятся с клавиатуры (Вася случайно указал ключ в письме). ﻿

Входные данные:

Наш хлебушек это парад еще вздумай не черепаха конец черепица роскомнадзор

2

3

Выходные данные:

это еще не конец роскомнадзор

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

s = input().split()

n = int(input())

k = int(input())

for i in range (k-1, len(s), n): #начало, конец, шаг

print(s[i], end = ' ')

## Техника работы со списками

Приложение: Stepik(7-8)\_1.10.8.py,Stepik(7-8)\_1.10.9.py,Stepik(7-8)\_1.10.4.py, Stepik\_2.5.9.py, Stepik\_2.5.10.py, Stepik\_2.5.11.py

Листинг 13. Stepik(7-8)\_1.10.8.py

'''

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.

Входные данные:

python sila

Выходные данные:

python\_sila

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

15.10.2019

'''

a = input().split()

a = "\_".join(a)

print(a)

Листинг 14. Stepik(7-8)\_1.10.9.py

'''

Уберите точки из введенного IP-адреса. Выведите сначала четыре числа через пробел, а затем сумму получившихся чисел.

Входные данные:

192.168.0.1

Выходные данные:

192 168 0 1

361

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

15.10.2019

'''

s = input().split('.')

print(\*s)

print(sum(map(int, s)))

Листинг 15. Stepik(7-8)\_1.10.4.py

'''

В первой строке вводится натуральное число n, затем n чисел по одному в строке, числа не превышают 10^4.

Для переданного набора чисел вычислите контрольное значение M – сумму минимального нечетного и минимального четного элементов.

Затем увеличьте все элементы набора, меньшие M, на M. Если отсутствуют четные или нечетные числа, считайте соответствующий минимум равным нулю.

Выведите элементы обработанного набора чисел через пробел.

Входные данные:

5

30

99

27

90

42

Выходные данные:

87 99 84 90 99

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

09.10.2019

'''

n = int(input())

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

MinEven = MinOdd = 9999999

for x in Lst:

if MinEven > x and x % 2 == 0:

MinEven = x

elif MinOdd > x and x % 2 != 0:

MinOdd = x

for i in range(len(Lst)):

if Lst[i] < MinEven + MinOdd:

Lst[i] += MinEven + MinOdd

print(\*Lst)

## Техника работы с циклом for и генераторами списков

Приложения: for1.py, for2.py, for3.py, for4.py, for5.py, for6.py

Листинг 16. for1.py

"""

Напишите программу, которая считывает с клавиатуры два числа a и b, считает и выводит на

консоль среднее арифметическое всех чисел из отрезка [a; b][a;b], которые делятся на 3.

В приведенном ниже примере среднее арифметическое считается для чисел на отрезке [-5; 12].

Всего чисел, делящихся на 3, на этом отрезке 6: -3, 0, 3, 6, 9, 12. Их среднее арифметическое

равно 4.5

На вход программе подаются интервалы, внутри которых всегда есть хотя бы одно число, которое делится на 3.﻿

Входные данные:

-5

12

Выходные данные:

4.5

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

a = int(input())

b = int(input())

sum = 0

k = 0

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

if i % 3 == 0 :

sum += i

k += 1 #счетчик

i += 1

print(sum / k)

Листинг 17. for2.py

"""

С клавиатуры вводятся целые числа a > b. Выведите убывающую последовательность чисел по одному числу в строке.

Входные данные:

-2

-8

Выходные данные:

-2

-3

-4

-5

-6

-7

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

a = int(input())

b = int(input())

for i in range (a,b,-1) : #цикл с конца

print(i)

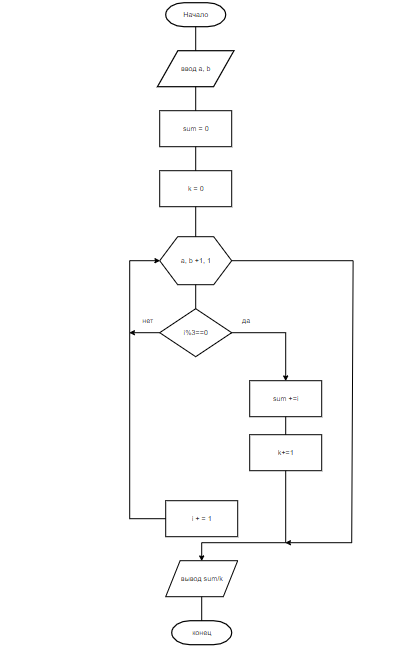


Рисунок 14. Блок-схема к листингу 17

Листинг 18. for3.py

"""

В первой строке вводится натуральное число n, затем n чисел по одному в строке, числа не

превышают 10^4. Для переданного набора чисел вычислите контрольное значение M – сумму минимального

нечетного и минимального четного элементов. Затем увеличьте все элементы набора, меньшие M, на M.

Если отсутствуют четные или нечетные числа, считайте соответствующий минимум равным нулю. Выведите

элементы обработанного набора чисел через пробел.

Входные данные:

5

30

99

27

90

42

Выходные данные:

87 99 84 90 99

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

a=[int(input()) for i in range(int(input()))] #генерация списка

x=min([i for i in a if i%2])#минимальный четный

y=min([i for i in a if not i%2])#минимальный нечетный

print(\*[i+x+y if i<x+y else i for i in a])#вывод через пробел обработанного списка

## Техника работы с функциями

Приложения: Stepik(7-8)\_2.4.1.py, Stepik(7-8)\_2.4.2.py, Stepik\_3.1.8.py, Stepik\_3.1.9.py

Листинг 19. Stepik(7-8)\_2.4.1.py

'''

С клавиатуры вводятся числа a и b, b > a. Найдите сумму факториалов чисел из диапазона [a; b]. Для вычисления факториала реализуйте функцию factorial(n).

Входные данные:

3

7

Выходные данные:

5910

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

09.10.2019

'''

def factorial(n):

res = 1

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

res \*= i

return res

n = int(input())

m = int(input())

sum = 0

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

sum += factorial(i)

print(sum)

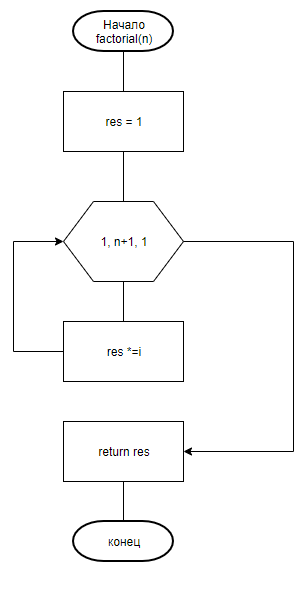


Рисунок 15. Блок-схема к листингу 19

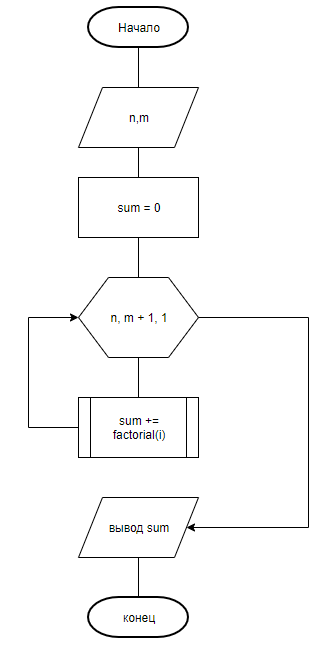


Рисунок 16. Блок-схема к листингу 19

Листинг 20. Stepik(7-8)\_2.4.2.py

'''

Вводится натуральное число n. Для этого числа вычислите n-ое число Фибоначчи и выведите сумму этого числа и его факториала.

Вычисление числа Фибоначчи и факториала вынесите в отдельные функции factorial(n) и fib(n). Считайте первое число Фибоначчи равным единице.

Входные данные:

10

Выходные данные:

3628855

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

12.10.2019

'''

def factorial(n):

res = 1

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

res \*= i

return res

def fib(n):

res\_fib = [1,1]

while len(res\_fib)!= n-2:

res\_fib.append(res\_fib[-1] + res\_fib[-2])

return sum(res\_fib)

n = int(input())

print(factorial(n) + fib(n)+1)

Листинг 21. Stepik\_3.1.8.py

'''

Напишите функцию f(x), которая возвращает значение следующей функции, определённой на всей числовой прямой:

Требуется реализовать только функцию, решение не должно осуществлять операций ввода-вывода.

Входные данные:

1

Выходные данные:

-0.5

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

09.10.2019

'''

def f(x):

if x <= -2:

return 1-(x+2)\*\*2

elif -2 < x <= 2:

return -(x/2)

elif 2 < x:

return (x-2)\*\*2 + 1

## Техника работы со словарями

Приложения: slv1.py, slv2.py, slv3.py

Листинг 22. slv1.py

"""

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.

Входные данные:

Ben 89001234050

Alice 210-220

Alice

Alice 404-502

Nick +16507811251

Ben

Alex +4(908)273-22-42

Alice

Nick

Robert 51234047129

Alex

.

Выходные данные:

210-220

89001234050

404-502

+16507811251

+4(908)273-22-42

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

d = {} #словарь

s = input()

while s != '.':

s = s.split()

if len(s) == 2:

d[s[0]] = s[1]

else:

print(d[s[0]])

s = input()

Листинг 23. slv2.py

"""

Программа должна считывать одну строку со стандартного ввода и выводить для каждого уникального слова

в этой строке число его повторений (без учёта регистра) в формате "слово количество" (см. пример вывода).

Порядок вывода слов может быть произвольным, каждое уникальное слово﻿ должно выводиться только один раз.

Входные данные:

a aa abC aa ac abc bcd a

Выходные данные:

ac 1

a 2

abc 2

bcd 1

aa 2

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

s = input()

s = s.lower()#приводим к нижнему регистру

s = s.split()

d = {}#словарь

for i in range(len(s)):

d[s[i]]= s.count(s[i])

for key, val in d.items():#перебор ключей и значений

print(key, val)

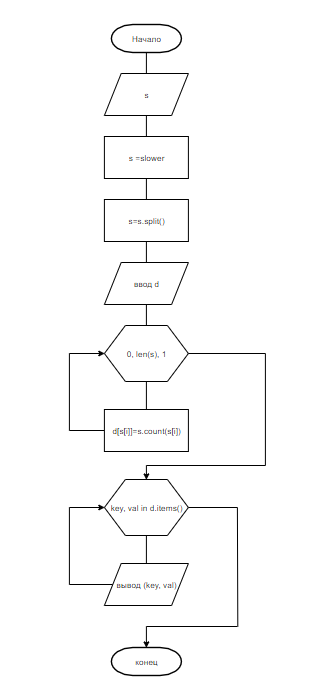


Рисунок 17. Блок-схема к листингу 23

Листинг 24. slv3 .py

"""

Напишите программу, которая умеет шифровать и расшифровывать шифр подстановки. Программа принимает

на вход две строки одинаковой длины, на первой строке записаны символы исходного алфавита, на второй

строке — символы конечного алфавита, после чего идёт строка, которую нужно зашифровать переданным ключом,

и ещё одна строка, которую нужно расшифровать.

Пусть, например, на вход программе передано:

abcd

\*d%#

abacabadaba

#\*%\*d\*%

Это значит, что символ a исходного сообщения заменяется на символ \* в шифре, b заменяется на d, c — на

% и d — на #.

Нужно зашифровать строку abacabadaba и расшифровать строку #\*%\*d\*% с помощью этого шифра.

Получаем следующие строки, которые и передаём на вывод программы:

\*d\*%\*d\*#\*d\*

dacabac

Входные данные:

abcd

\*d%#

abacabadaba

#\*%\*d\*%

Выходные данные:

\*d\*%\*d\*#\*d\*

dacabac

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

d = {}

s1 = list(input())

s2 = list(input())

s3 = list(input())

s4 = list(input())

s = ''

d = dict(zip(s1,s2))#объединение двух списков в один список пар

for i in range(len(s3)):

s += d[s3[i]]

print(s)

for i in range(len(s4)):

for k, v in d.items():#перебор ключей-значений

if v == s4[i]:

print(k,end = '')

## Техника работы с множествами

Приложения: mn1.py, mn2.py

Листинг 25. mn1.py

"""

В ходе исследований ученые делают некие замеры, результаты которых заносят в базу данных. Однако для анализа результатов нет необходимости держать в базе "лишние",повторяющиеся данные. Напишите программу, которая выводит максимальное количество записей, после удаления которых анализ результатов будет произведен верно. Список элементов вводится через пробел.

Входные данные:

6311 9423 142 142 8654 909 Error 6311 142 909 404 502 828 404 9423

Выходные данные:

6

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

s = input().split()

a = set(s)

print(len(s)-len(a))

Листинг 26. mn2.py

"""

Ученые вновь проводят какие-то таинственные измерения, и на этот раз их интересует корректность проделанных расчетов. Были произведены две серии опытов. Замеры считаются корректными, если доля совпадающих показаний превышает 70% от количества опытов.

Формат ввода: две строки одинаковой длины с числами, введенными через пробел (каждое число — отдельное измерение в серии опытов). Все числа в строке различны.

Формат вывода: строка "Корректно", если выполняется условие, и "Некорректно" в обратном случае.

Входные данные:

141 854 32 0.9 43 85.6 90.2 42.5 180 97.9

141 32 0.9 42.5 88 43 37 97.9 180 854

Выходные данные:

Корректно

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

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

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

d = set.symmetric\_difference(a,b) #создаем множество с отсутствующими в множестве b элементами

if (len(d)/(len(a)+len(b))\*100)<30:#процент несовпадающих показаний

print('Корректно')

else:

print('Некорректно')

Листинг 27. Stepik(7-8)3.2.5

'''

Учитель проверяет домашнее задание в классе и получил следующие ответы: из N школьников у M учеников домашнее задание съела собака, у K человек отключили свет, а P учеников постигли оба нечастья. Сколько человек выполнило домашнее задание?

Формат ввода: числа N, M, K и P, каждое в отдельной строке.

Формат вывода: количество учеников, сделавших домашнее задание.

Входные данные:

81

25

16

9

Выходные данные:

49

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/courses/67

'''

n, m, k, p = [int(input()) for i in range(4)]

print(n-(m+k)+p)

## 1.12. Техника работы с кортежами

Приложения: cort1.py, cort2.py

Листинг 28. cort1.py

"""

Необходимо написать программу, выводящую кортеж, содержащий числа в полуинтервале [a; b),

если a < b, и [b; a), если b < a. Числа a и b вводятся с клавиатуры.

Входные данные:

63

70

Выходные данные:

(63, 64, 65, 66, 67, 68, 69)

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

a = int(input())

b = int(input())

if a < b:

print(tuple(range(a,b)))#создание кортежа

else :

print(tuple(range(b,a)))

Листинг 29. cort2.py

"""

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

Входные данные:

2

Выходные данные:

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

Выполнил: Карандашев Н.К. П1-17

Источник: https://stepik.org/course/31182

"""

a = int(input())

if a % 2 == 0 :

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

else :

print(tuple(range(a + 1, a \* 10 + 1, 2)))#сразу к нечетному а прибавляем 1

## 1.13. Техника работы с файлами

Приложение: text14.py, text14.txt,

Листинг 30. text14.py

"""

Text14

Дан непустой файл.

Удалить последнюю строку

Выполнил: Карандашев Н.К. П1-17

Источник: задачник Абрамяна

"""

fname = 'text14.txt'

with open(fname,'r') as f:

line = f.readlines()

with open(fname,'w') as f:

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

f.write(line[i])

print('done')

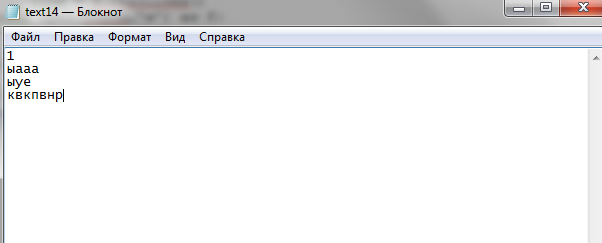


Рисунок 18. Исходные данные файла text14.txt

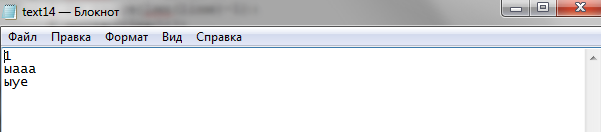


Рисунок 19. Результат выполнения программы

Листинг 31. State\_main.py, new.txt

import State\_funct

def main():

fname = 'new.txt'

states = []

st = State\_funct.State()

states = st.read(states)

print("""Выбор действия:

1-Добавление(Государство,Столица,Часть света,Численность,Площадь)

2-Вывод столицы по названию государства

3-Вывод названия гос-ва по городу-столице

4-Вывод всей информации о гос-ве

5-Вывод всех стран в данной части света

6-Расчет и вывод плотности населения

7-Вывод кол-ва гос-в в части света

8-Расчет общей площади гос-в в части света

0-Выход""")

while True: #меню

print('Введите номер действия')

n = int(input())

if n == 0:

break

elif n == 1:

st.add(fname,states)

elif n == 2:

print(st.state(states))

elif n == 3:

print(st.city(states))

elif n == 4:

st.information(states)#распаковка

elif n == 5:

st.states5(states)

elif n == 6:

st.population\_density(states)

elif n == 7:

print(st.states\_in\_partOfTheWorld(states))

elif n == 8:

print(st.common\_square(states))

print('Exit')

main()

class State:

def \_\_init\_\_(self, name=' ', capital=' ', part\_of\_the\_world=' ', population=3, area=4 ):

self.name = name

self.capital = capital

self.part\_of\_the\_world = part\_of\_the\_world

self.population = population

self.area = area

def add(self,fname,states):

"""

Функция добавления/дозаписи в файл информации

"""

print("Введите информацию о государстве: ")

f = open(fname, 'a+') #открываем для дозаписи

s = input()

f.write(s)

f.write('\n') #новая введенная информация в файле будет с новой строки

f.close()

f = open(fname, 'r')

for line in f: #считываем построчно,разбиваем на слова и добавляем в словарь

s = line

L = s.split()

st = State()

st.name = L[0]

st.capital = L[1]

st.part\_of\_the\_world = L[2]

st.population = L[3]

st.area = L[4]

states.append(st)

f.close()

return states

def state(self,states):

"""

Функция вывода столицы по государству

"""

print('Введите название гос-ва : ')

st = State()

s = input()

for st in states: #перебор ключей словаря

if st.name == s:

return st.capital #вывод значения ключа

return 'Не найдено'

def city(self,states):

"""

Функция вывода государства по столице

"""

st = State()

print('Введите название столицы : ')

s = input()

for st in states:

if st.capital == s:

return st.name #вывод ключа

return 'Не найдено'

def information(self,states):

"""

Функция вывода информации о государстве

"""

print('Введите название гос-ва : ')

st = State()

s = input()

for st in states:

if st.name == s:

print(st.capital,st.part\_of\_the\_world,st.population,st.area)

def states5(self,states):

"""

Функция вывода государств части света

"""

print('Введите часть света :')

st = State()

s = input()

for st in states:

if st.part\_of\_the\_world == s:

print(st.name)

def population\_density(self,states):

"""

Функция вывода плотности населения

"""

st = State()

print('Вывод плотности населения :')

for st in states:

people = float(st.population)

square = float(st.area)

print(st.name +' '+'%.2f' % (people/square\*1000)) #умножаем для соответствия единиц условия

def states\_in\_partOfTheWorld(self,states):

"""

Функция вывода кол-ва государств части света

"""

print('Введите часть света :')

st = State()

s = input()

k = 0

for st in states:

if st.part\_of\_the\_world == s:

k += 1 #счетчик

print('Кол-во государств ')

return k

def common\_square(self,states):

"""

Функция вывода общей площади в части света

"""

k = 0

print('Введите часть света :')

s = input()

st = State()

for st in states:

if st.part\_of\_the\_world == s:

k += int(st.area)

print('Общая площадь ')

return k

def read(self,states):

"""

Функция чтения файла и добавления информации в словарь

"""

states = []

fname = 'new.txt'

f = open(fname, 'r')

for line in f: #считываем построчно,разбиваем на слова и добавляем в словарь

s = line

L = s.split()

st = State()

st.name = L[0]

st.capital = L[1]

st.part\_of\_the\_world = L[2]

st.population = L[3]

st.area = L[4]

states.append(st)

f.close()

return states

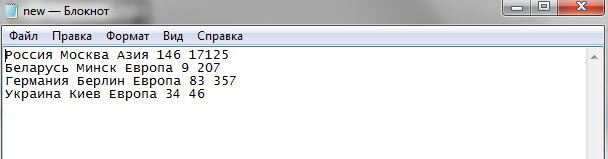


Рисунок 20. Содержание файла new.txt

Листинг 32. File13

'''

Дан файл целых чисел. Создать два новых файла, первый из которых содержит

положительные числа из исходного файла (в обратном порядке), а второй —

отрицательные (также в обратном порядке). Если положительные или отрицательные

числа в исходном файле отсутствуют, то соответствующий результирующий файл

оставить пустым.

Входные данные: number.txt

в файле number.txt: 11 -22 6 13 8 -7 3 -15

Выходные данные:

в файле positive.txt: 3 8 13 6 11

в файле negative.txt: -15 -7 -22

Выполнил: Карандашев Н.К. П1-17

Источник: Задачник Абрамян

'''

file\_name = input('Введите имя файла: ')

arr = []

f = open(file\_name, "r")

for line in f:

arr.extend(line.split(" "))

for item in arr:

arr[arr.index(item)] = int(item)

print(arr)

arr.reverse()

f.close()

f = open("positive.txt", "w")

f1 = open("negative.txt", "w")

for i in arr:

if (i >= 0):

f.write(str(i) + " ")

else:

f1.write(str(i) + " ")

f.close()

f1.close()

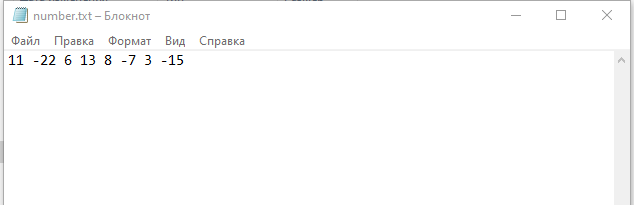


Рисунок 21. Содержание файла number.txt

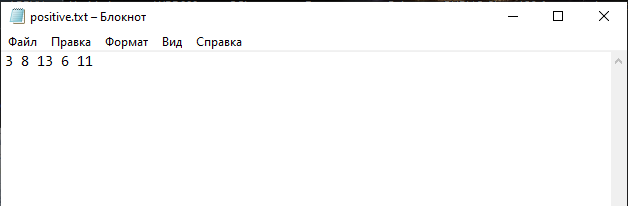


Рисунок 22. Содержание файла positive.txt

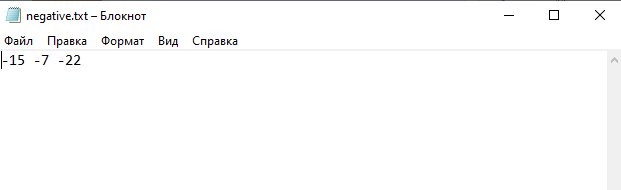


Рисунок 23. Содержание файла negative.txt

# **Раздел 2. Техника решения задач с использованием модульного и объектно-ориентированного программирования.**

## 2.1. Техника работы с модулями

Модулем в Python называется любой файл с программой. Каждая программа может импортировать модуль и получить доступ к его классам, функциям и объектам.

Подключить модуль можно с помощью инструкции import. К примеру, подключим [модуль](https://pythonworld.ru/moduli/modul-os.html) OS для получения текущей директории:

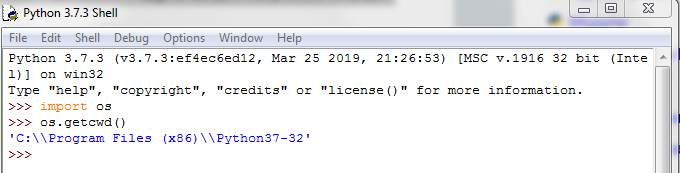


Рисунок 24. Подключение [модуль](https://pythonworld.ru/moduli/modul-os.html) OS

После ключевого слова import указывается название модуля.

После импортирования модуля его название становится переменной, через которую можно получить доступ к атрибутам модуля. Например, можно обратиться к константе e, расположенной в модуле [math](https://pythonworld.ru/moduli/modul-math.html" \t "_blank):

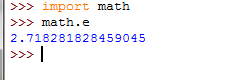


Рисунок 25. Подключение [модуль](https://pythonworld.ru/moduli/modul-os.html) [math](https://pythonworld.ru/moduli/modul-math.html) и вывод значения e

Если название модуля слишком длинное, или оно вам не нравится по каким-то другим причинам, то для него можно создать псевдоним, с помощью ключевого слова as.

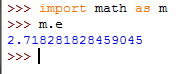


Рисунок 26. Создание псевдонима для модуля math и вывод е

Подключить определенные атрибуты модуля можно с помощью инструкции from

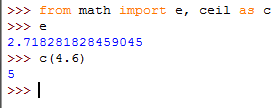


Рисунок 27. Подключение определенных атрибутов модуля math

## 2.2. Разработка проекта «Журнал посещения занятий»

Приложения: Journal\_v1.py, myfile.txt, DataStudent.txt.

Задание:

1. Создать проект «Журнал посещения занятий», который будет показывать список всех опоздавших на занятия учеников.
2. Добавить возможность считать число опозданий ученика.
3. Добавить возможность просмотра опозданий по заданному промежутку дат.
4. Решить без использования словарей.

Листинг 33. Journal\_v1.py

def CheckDate(s, dn, dk): #функция проверки попадания введенной даты в промежуток  
 date = [int(x) for x in s.split(".")]  
 b = 0  
 if dn[2] == dk[2]:  
 if dn[1] < date[1] < dk[1]:  
 b = 1  
 elif (dn[1] == date[1] == dk[1] and (dn[0] <= date[0] <= dk[0])):  
 b = 1  
 elif (dn[1] == date[1] and date[1] != dk[1] and dn[0] <= date[0]):  
 b = 1  
 elif (dk[1] == date[1] and date[1] != dn[1] and date[0] <= dk[0]):  
 b = 1  
 elif dn[2] != dk[2]:  
 if dn[2] < date[2] < dk[2]:  
 b = 1  
 elif (date[2] == dn[2] and date[1] > dn[1]):  
 b = 1  
 elif (date[2] == dn[2] and date[1] == dn[1] and date[0] >= dn[0]):  
 b = 1  
 elif (date[2] == dk[2] and date[1] < dk[1]):  
 b = 1  
 elif (date[2] == dk[2] and date[1] == dk[1] and date[0] <= dk[0]):  
 b = 1  
 return b  
  
late = {}  
print("Введите ограничения по дате (формат d.m.y):")  
datn = input() #ввод начальной даты  
datk = input() #ввод конечной даты  
datn = [int(x) for x in datn.split(".")]  
datk = [int(x) for x in datk.split(".")]  
print("Выберите формат ввода данных (0 - с консоли, 1 - из файла): ")  
choice = input()  
while (choice != '1' and choice != '0'):  
 choice = input()  
if choice == '0':  
 student = input() #входные данные о студентах  
 while student != '.':  
 student = student.split()  
 surn, time = student[1], student[2].split(":")  
 time[0], time[1] = int(time[0]), int(time[1])  
 if CheckDate(student[0], datn, datk) == 1:  
 if (time[0] == 9 and time[1] > 0 or time[0] > 9):  
 if surn in late:  
 late[surn] = [late[surn][0] + (time[0] - 9) \* 60 + time[1], late[surn][1] + 1]  
 else:  
 late[surn] = [(time[0] - 9) \* 60 + time[1], 1]  
 student = input()  
else:  
 file = open('DataStudent.txt','r')  
 students = file.read().split("\n")  
 for student in students:  
 student = student.split()  
 surn, time = student[1], student[2].split(":")  
 time[0], time[1] = int(time[0]), int(time[1])  
 if CheckDate(student[0], datn, datk) == 1:  
 if (time[0] == 9 and time[1] > 0 or time[0] > 9):  
 if surn in late:  
 late[surn] = [late[surn][0] + (time[0] - 9) \* 60 + time[1], late[surn][1] + 1]  
 else:  
 late[surn] = [(time[0] - 9) \* 60 + time[1], 1]  
f = open('myfile.txt', 'w')  
for x in late:  
 f.write(str(x) + ' ' + str(late[x][0]) + ' ' + str(late[x][1]) + ' \n')  
f.close()

Входные данные:

11.10.2019

13.10.2019

1



Рисунок 28. Файл с исходными данными

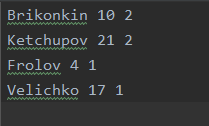


Рисунок 29. Результат работы программы

## 2.3. Установка и настройка среды JetBrains PyCharm

PyCharm - это кросс-платформенная среда разработки.

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

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



Рисунок 28. Установщик PyCharm



Рисунок 29. Место установки PyCharm



Рисунок 30. Процесс установки PyCharm



Рисунок 31. Стартовое окно PyCharm

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

Рисунок 32. Выбор места для нового проекта и выбор интерпретатора

Дальше откроется окно самого редактора.

Теперь вы готовы начать писать программы на Python!

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

Рисунок 33. Создание нового Phython файла

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

Справа откроется сам файл. Пока он пустой. Пишем:



Рисунок 34. Написания программы

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



Рисунок 35. Запуск программы

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



Рисунок 36. выполнения нашей программы

## 2.4. Техника работы с библиотекой tkinter

Tkinter – это графическая библиотека на основе Tk, которая входит в стандартную библиотеку Python.

Tkinter импортируется стандартно для модуля Python любым из способов: import tkinter, from tkinter import \*, import tkinter as tk

Чтобы написать GUI-программу, надо выполнить приблизительно следующее:

1. Создать главное окно.
2. Создать виджеты и выполнить конфигурацию их свойств (опций).
3. Определить события, то есть то, на что будет реагировать программа.
4. Определить обработчики событий, то есть то, как будет реагировать программа.
5. Расположить виджеты в главном окне.
6. Запустить цикл обработки событий.

Создадим кнопку, при нажатии на которую будет выводиться текст в консоль:

Листинг 34.

from tkinter import \*

root = Tk()

def Hello(event):

print ("Yet another hello world")

btn = Button(root, #родительское окно

text="Click me", #надпись на кнопке

width=30,height=5, #ширина и высота

bg="white",fg="black") #цвет фона и надписи

btn.bind("<Button-1>", Hello) #при нажатии ЛКМ на кнопку вызывается функция Hello

btn.pack() #расположить кнопку на главном окне

root.mainloop()

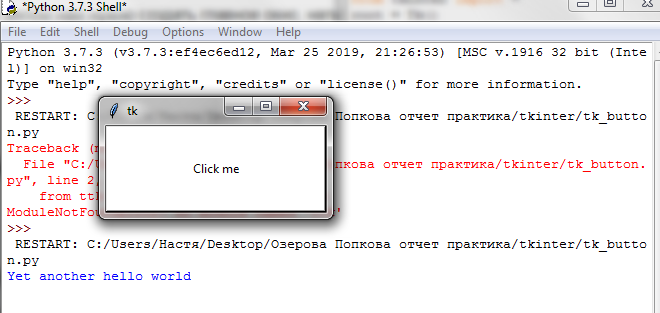


Рисунок 37. Кнопка

Функция pack() — это так называемый упаковщик, или менеджер расположения. Он отвечает за то, как виджеты будут располагаться на главном окне.

Всего упаковщиков три:  
  
pack(). Автоматически размещает виджеты в родительском окне:

Листинг 35.

from tkinter import \*

root = Tk()

Button(root, text = '1').pack(side = 'left')

Button(root, text = '2').pack(side = 'top')

Button(root, text = '3').pack(side = 'right')

Button(root, text = '4').pack(side = 'bottom')

Button(root, text = '5').pack(fill = 'both')

root.mainloop()

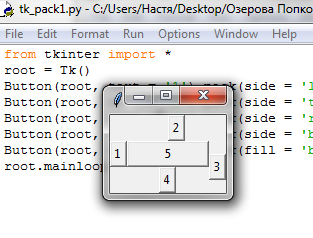


Рисунок 38. Кнопки

grid(). Размещает виджеты на сетке:

Листинг 36.

from tkinter import \*

root = Tk()

Button(root, text = '1').grid(row = 1, column = 1)

Button(root, text = '2').grid(row = 1, column = 2)

Button(root, text = '\_\_3\_\_').grid(row = 2, column = 1, columnspan =2)

root.mainloop()

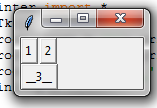


Рисунок 39. Кнопки

place(). Позволяет размещать виджеты в указанных координатах с указанными размерами:

Листинг 37.

from tkinter import \*

root = Tk()

Button(root, text = '1').place(x = 10, y = 10, width = 30)

Button(root, text = '2').place(x = 45, y = 20, height = 15)

Button(root, text = '\_\_3\_\_').place(x = 20, y = 40)

root.mainloop()

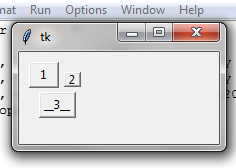


Рисунок 40. Кнопки

## 2.5 Доработка проекта «Журнал посещения занятий» с использованием графического интерфейса

Приложения: Journal\_v2.py, DataStudents.txt, myfile.txt.

Задание:

1. Добавить возможность ввода даты опоздания ученика.
2. Добавление опоздавших учеников в файл и чтение из него информацию о других опоздавших.
3. Использовать графический интерфейс

Листинг 38. Journal\_v2.py

from tkinter import \*  
  
def CheckDate(s, dn, dk): #функция проверки попадания введенной даты в промежуток  
 date = [int(x) for x in s.split(".")]  
 b = 0  
 if dn[2] == dk[2]:  
 if dn[1] < date[1] < dk[1]:  
 b = 1  
 elif (dn[1] == date[1] == dk[1] and (dn[0] <= date[0] <= dk[0])):  
 b = 1  
 elif (dn[1] == date[1] and date[1] != dk[1] and dn[0] <= date[0]):  
 b = 1  
 elif (dk[1] == date[1] and date[1] != dn[1] and date[0] <= dk[0]):  
 b = 1  
 elif dn[2] != dk[2]:  
 if dn[2] < date[2] < dk[2]:  
 b = 1  
 elif (date[2] == dn[2] and date[1] > dn[1]):  
 b = 1  
 elif (date[2] == dn[2] and date[1] == dn[1] and date[0] >= dn[0]):  
 b = 1  
 elif (date[2] == dk[2] and date[1] < dk[1]):  
 b = 1  
 elif (date[2] == dk[2] and date[1] == dk[1] and date[0] <= dk[0]):  
 b = 1  
 return b  
  
def click\_btn1(): #функция, которая выполняется при нажатии на первую кнопку  
 win1 = Tk()  
 win1.title("Выбор диапазона дат")  
 win1.geometry('400x175')  
 lbl1 = Label(win1, text="Введите ограничения по дате (формат dd.mm.yyyy), в рамках \nкоторой вы хотите увидеть информацию об опоздавших")  
 lbl1.pack()  
 lbl2 = Label(win1, text="Введите начальную дату:")  
 lbl2.pack()  
 txt1 = Entry(win1, width=70)  
 txt1.pack()  
 lbl3 = Label(win1, text="Введите конечную дату:")  
 lbl3.pack()  
 txt2 = Entry(win1,width=70)  
 txt2.pack()  
 def click\_Ok(): #функция, которая выполняется при нажатии на кнопку "OK"  
 late = {}  
 datn = txt1.get()  
 datk = txt2.get()  
 datn = [int(x) for x in datn.split(".")]  
 datk = [int(x) for x in datk.split(".")]  
 file = open('DataStudent.txt','r')  
 students = file.read().split("\n")  
 file.close()  
 for student in students:  
 student = student.split(" ")  
 surn = student[1]  
 time = student[2].split(":")  
 time[0], time[1] = int(time[0]), int(time[1])  
 if CheckDate(student[0], datn, datk) == 1:  
 if (time[0] == 9 and time[1] > 0 or time[0] > 9):  
 if surn in late:  
 late[surn] = [late[surn][0] + (time[0] - 9) \* 60 + time[1], late[surn][1] + 1]  
 else:  
 late[surn] = [(time[0] - 9) \* 60 + time[1], 1]  
 f = open('DataLateStudent.txt', 'w')  
 b = 0  
 for x in late:  
 b = b + 1  
 if b == 1:  
 f.write(str(x) + ' ' + str(late[x][0]) + ' ' + str(late[x][1]))  
 else:  
 f.write('\n' + str(x) + ' ' + str(late[x][0]) + ' ' + str(late[x][1]))  
 f.close()  
 win1.destroy()  
 win2 = Tk()  
 win2.title("Уведомление")  
 win2.geometry('300x80')  
 lbl = Label(win2, text = "Список опоздавших\nуспешно добавлен\nв текстовый файл DataLateStudent\nв текущей директории")  
 lbl.pack()  
 Ok = Button(win1, text="OK", command = click\_Ok)  
 Ok.pack()  
  
def click\_btn2(): #функция, которая выполняется при нажатии на вторую кнопку  
 win1 = Tk()  
 win1.title("Заполнение информации")  
 win1.geometry('400x200')  
 lbl1 = Label(win1, text="Введите дату появления студента:")  
 lbl1.pack()  
 txt1 = Entry(win1, width=70)  
 txt1.pack()  
 lbl2 = Label(win1, text="Введите фамилию студента:")  
 lbl2.pack()  
 txt2 = Entry(win1, width=70)  
 txt2.pack()  
 lbl3 = Label(win1, text="Введите время появления студента:")  
 lbl3.pack()  
 txt3 = Entry(win1, width=70)  
 txt3.pack()  
 def click\_Ok(): #функция, которая выполняется при нажатии на кнопку "OK"  
 f = open('DataStudent.txt', 'a')  
 f.write('\n' + str(txt1.get()) + ' ' + str(txt2.get()) + ' ' + str(txt3.get()))  
 f.close()  
 win1.destroy()  
 win2 = Tk()  
 win2.title("Уведомление")  
 win2.geometry('300x80')  
 lbl = Label(win2, text = "Информация о появлении студента\nуспешно добавлена\nв текстовый файл DataStudent\nв текущей директории")  
 lbl.pack()  
 Ok = Button(win1, text="OK", command = click\_Ok)  
 Ok.pack()  
  
def click\_btn3(): #функция, которая выполняется при нажатии на третью кнопку  
 win = Tk()  
 win.title("Список опоздавших")  
 win.geometry('400x200')  
 file = open('DataLateStudent.txt','r')  
 students = file.read()  
 file.close()  
 lbl1 = Label(win, text = "Фамилия студента / Время позданий (в мин) / Кол-во опозданий")  
 lbl1.pack()  
 lbl2 = Label(win, text=students)  
 lbl2.pack()  
   
def click\_btn4(): #функция, которая выполняется при нажатии на четвертую кнопку  
 win = Tk()  
 win.title("Наибольшее количество опозданий")  
 win.geometry('400x175')  
 lst = []  
 Text = ""  
 file = open('DataLateStudent.txt', 'r')  
 students = file.read().split("\n")  
 file.close()  
 max\_count = 0  
 for student in students:  
 student = student.split(" ")  
 if (int(student[2])) > (int(max\_count)):  
 max\_count = student[2]  
 for student in students:  
 student = student.split(" ")  
 if student[2] == max\_count:  
 lst.append(student)  
 for student in lst:  
 Text = Text + student[0] + " " + student[1] + " " + student[2] + "\n"  
 lbl1 = Label(win, text = "Фамилия студента / Время позданий (в мин) / Кол-во опозданий")  
 lbl1.pack()  
 lbl2 = Label(win, text = Text)  
 lbl2.pack()  
  
def click\_btn5(): #функция, которая выполняется при нажатии на пятую кнопку  
 win = Tk()  
 win.title("Наибольшее время опозданий")  
 win.geometry('400x175')  
 lst = []  
 Text = ""  
 file = open('DataLateStudent.txt', 'r')  
 students = file.read().split("\n")  
 file.close()  
 max\_time = 0  
 for student in students:  
 student = student.split(" ")  
 if (int(student[1])) > (int(max\_time)):  
 max\_time = student[1]  
 for student in students:  
 student = student.split(" ")  
 if student[1] == max\_time:  
 lst.append(student)  
 for student in lst:  
 Text = Text + student[0] + " " + student[1] + " " + student[2] + "\n"  
 lbl1 = Label(win, text = "Ф.И.О студента / Время позданий (в мин) / Кол-во опозданий")  
 lbl1.pack()  
 lbl2 = Label(win, text = Text)  
 lbl2.pack()  
   
window = Tk() #создание окна с кнопками  
window.title("Журнал посещений")  
window.geometry('290x219')   
btn1 = Button(window, text="Записать информацию о посещении", width = "40", pady = "10", background = "lime", command = click\_btn1)   
btn1.pack()  
btn2 = Button(window, text="Добавить информацию о появлении студента", width = "40", pady = "10", background = "yellow", command = click\_btn2)   
btn2.pack()  
btn3 = Button(window, text="Вывести список опоздавших", width = "40", pady = "10", background = "tan", command = click\_btn3)  
btn3.pack()  
btn4 = Button(window, text="Студент с наибольшим количеством опозданий", width = "40", pady = "10", background = "pink", command = click\_btn4)   
btn4.pack()  
btn5 = Button(window, text="Студент с наибольшим временем опозданий", width = "40", pady = "10", background = "orchid", command = click\_btn5)  
btn5.pack()  
window.mainloop()

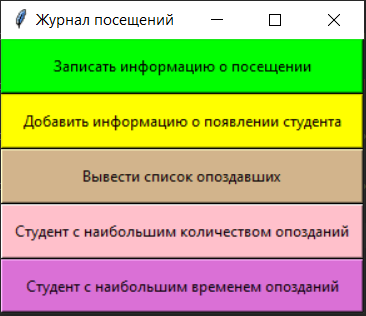


Рисунок 30. Графический интерфейс

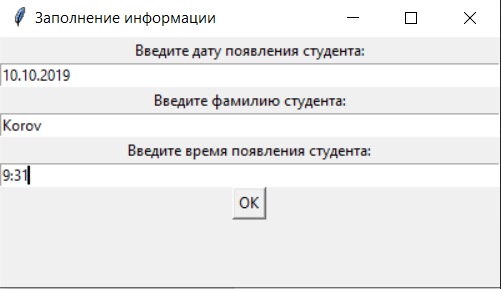


Рисунок 31. Добавление опоздавшего ученика

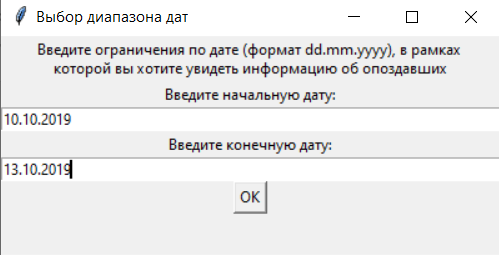


Рисунок 32. Входные данные



Рисунок 33. Файл с исходными данными



Рисунок 34. Результат работы программы

## 2.6. Техника работы с классами

Класс — образец, созданный программистом для объекта. Он определяет набор атрибутов, которые будут характеризовать любой объект, который создается из этого класса.

Объект — экземпляр класса. Это реализованная версия класса, где он проявляется в программе.

определим класс Shark, который имеет две связанные с ним функции:

Листинг 39.

class Shark:

def swim(self):

print("The shark is swimming.")

def be\_awesome(self):

print("The shark is being awesome.")

Поскольку эти функции объявлены под классом Shark, они называются методами. Методы — это особый вид функции, который определен внутри класса.

Аргументом этих функций является слово self, которое является ссылкой на объекты, созданные на основе этого класса. Чтобы ссылаться на экземпляры (или объекты) класса, self всегда будет первым параметром.

Объект является экземпляром класса. Создадим объект Shark с именем sammy. Мы инициализировали объект sammy как экземпляр класса, установив его равным Shark().

Листинг

sammy = Shark()

Теперь используем два метода с объектом Shark sammy:

Листинг

sammy = Shark()

sammy.swim()

sammy.be\_awesome()

Поскольку ключевое слово self было параметром методов класса Shark, объект sammy передается методам. Параметр self позволяет методам ссылаться на атрибуты объекта.

Но когда мы вызываем методы, в круглых скобках ничего не передается. Объект sammy автоматически передается с помощью оператора «.» (точки).

Добавим объект в программе:

Листинг 40.

class Shark:

def swim(self):

print("The shark is swimming.")

def be\_awesome(self):

print("The shark is being awesome.")

def main():

sammy = Shark()

sammy.swim()

sammy.be\_awesome()

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

main()

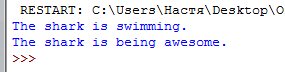


Рисунок 41. Результат выполнения программы

Объект sammy вызывает два метода в функции main(), указывая запустить эти методы.

Для инициализации данных используется метод конструктора - \_\_init\_\_

Листинг

class Person:

# конструктор

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

self.name = name # устанавливаем имя

def display\_info(self):

print("Привет, меня зовут", self.name)

person1 = Person("Tom")

person1.display\_info() # Привет, меня зовут Tom

person2 = Person("Sam")

person2.display\_info() # Привет, меня зовут Sam

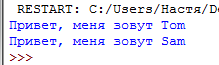


Рисунок 42. Результат выполнения программы

В качестве первого параметра конструктор также принимает ссылку на текущий объект - self. Нередко в конструкторах устанавливаются атрибуты класса. Так, в данном случае в качестве второго параметра в конструктор передается имя пользователя, которое устанавливается для атрибута self.name.

После окончания работы с объектом мы можем использовать оператор del для удаления его из памяти:

Листинг 41.

del person1 # удаление из памяти

#person1.display\_info() # Этот метод работать не будет, так как person1 уже удален из памяти

Мы можем определить определить в классе деструктор, реализовав встроенную функцию \_\_del\_\_, который будет вызываться либо в результате вызова оператора del, либо при автоматическом удалении объекта

Листинг 42.

class Person:

# конструктор

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

self.name = name # устанавливаем имя

def \_\_del\_\_(self):

print(self.name,"удален из памяти")

def display\_info(self):

print("Привет, меня зовут", self.name)

person1 = Person("Tom")

person1.display\_info() # Привет, меня зовут Tom

del person1 # удаление из памяти

person2 = Person("Sam")

person2.display\_info() # Привет, меня зовут Sam

del person2 # удаление из памяти

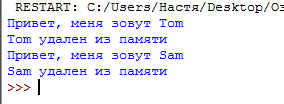


Рисунок 43. Результат выполнения программы

## 2.7. Доработка проекта «Журнал посещения занятий» с использованием классов

Приложения: Journal\_v3.py, DataStudent.txt, myfile.txt.

Задание:

1. Изменить проект используя классы

Листинг 43. Journal\_v3.py

from tkinter import \*  
  
class Person:  
 name = ""  
 data = ""  
 time = ""  
  
def CheckDate(s, dn, dk): #функция проверки попадания введенной даты в промежуток  
 date = [int(x) for x in s.split(".")]  
 b = 0  
 if dn[2] == dk[2]:  
 if dn[1] < date[1] < dk[1]:  
 b = 1  
 elif (dn[1] == date[1] == dk[1] and (dn[0] <= date[0] <= dk[0])):  
 b = 1  
 elif (dn[1] == date[1] and date[1] != dk[1] and dn[0] <= date[0]):  
 b = 1  
 elif (dk[1] == date[1] and date[1] != dn[1] and date[0] <= dk[0]):  
 b = 1  
 elif dn[2] != dk[2]:  
 if dn[2] < date[2] < dk[2]:  
 b = 1  
 elif (date[2] == dn[2] and date[1] > dn[1]):  
 b = 1  
 elif (date[2] == dn[2] and date[1] == dn[1] and date[0] >= dn[0]):  
 b = 1  
 elif (date[2] == dk[2] and date[1] < dk[1]):  
 b = 1  
 elif (date[2] == dk[2] and date[1] == dk[1] and date[0] <= dk[0]):  
 b = 1  
 return b  
  
def click\_btn1(): #функция, которая выполняется при нажатии на первую кнопку  
 win1 = Tk()  
 win1.title("Выбор диапазона дат")  
 win1.geometry('400x175')  
 lbl1 = Label(win1, text="Введите ограничения по дате (формат dd.mm.yyyy), в рамках \nкоторой вы хотите увидеть информацию об опоздавших")  
 lbl1.pack()  
 lbl2 = Label(win1, text="Введите начальную дату:")  
 lbl2.pack()  
 txt1 = Entry(win1, width=70)  
 txt1.pack()  
 lbl3 = Label(win1, text="Введите конечную дату:")  
 lbl3.pack()  
 txt2 = Entry(win1,width=70)  
 txt2.pack()  
 def click\_Ok(): #функция, которая выполняется при нажатии на кнопку "OK"  
 late = {}  
 datn = txt1.get()  
 datk = txt2.get()  
 datn = [int(x) for x in datn.split(".")]  
 datk = [int(x) for x in datk.split(".")]  
 file = open('DataStudent.txt','r')  
 students = file.read().split("\n")  
 file.close()  
 for x in students:  
 x = x.split(" ")  
 student = Person()  
 student.date = x[0]  
 student.name = x[1]  
 student.time = x[2].split(":")  
 student.time[0], student.time[1] = int(student.time[0]), int(student.time[1])  
 if CheckDate(student.date, datn, datk) == 1:  
 if (student.time[0] == 9 and student.time[1] > 0 or student.time[0] > 9):  
 if student.name in late:  
 late[student.name] = [late[student.name][0] + (student.time[0] - 9) \* 60 + student.time[1], late[student.name][1] + 1]  
 else:  
 late[student.name] = [(student.time[0] - 9) \* 60 + student.time[1], 1]  
 f = open('DataLateStudent.txt', 'w')  
 b = 0  
 for x in late:  
 b = b + 1  
 if b == 1:  
 f.write(str(x) + ' ' + str(late[x][0]) + ' ' + str(late[x][1]))  
 else:  
 f.write('\n' + str(x) + ' ' + str(late[x][0]) + ' ' + str(late[x][1]))  
 f.close()  
 win1.destroy()  
 win2 = Tk()  
 win2.title("Уведомление")  
 win2.geometry('300x80')  
 lbl = Label(win2, text = "Список опоздавших\nуспешно добавлен\nв текстовый файл DataLateStudent\nв текущей директории")  
 lbl.pack()  
 Ok = Button(win1, text="OK", command = click\_Ok)  
 Ok.pack()  
  
def click\_btn2(): #функция, которая выполняется при нажатии на вторую кнопку  
 win1 = Tk()  
 win1.title("Заполнение информации")  
 win1.geometry('400x200')  
 lbl1 = Label(win1, text="Введите дату появления студента:")  
 lbl1.pack()  
 txt1 = Entry(win1, width=70)  
 txt1.pack()  
 lbl2 = Label(win1, text="Введите фамилию студента:")  
 lbl2.pack()  
 txt2 = Entry(win1, width=70)  
 txt2.pack()  
 lbl3 = Label(win1, text="Введите время появления студента:")  
 lbl3.pack()  
 txt3 = Entry(win1, width=70)  
 txt3.pack()  
 def click\_Ok(): #функция, которая выполняется при нажатии на кнопку "OK"  
 f = open('DataStudent.txt', 'a')  
 f.write('\n' + str(txt1.get()) + ' ' + str(txt2.get()) + ' ' + str(txt3.get()))  
 f.close()  
 win1.destroy()  
 win2 = Tk()  
 win2.title("Уведомление")  
 win2.geometry('300x80')  
 lbl = Label(win2, text = "Информация о появлении студента\nуспешно добавлена\nв текстовый файл DataStudent\nв текущей директории")  
 lbl.pack()  
 Ok = Button(win1, text="OK", command = click\_Ok)  
 Ok.pack()  
  
def click\_btn3(): #функция, которая выполняется при нажатии на третью кнопку  
 win = Tk()  
 win.title("Список опоздавших")  
 win.geometry('400x200')  
 file = open('DataLateStudent.txt','r')  
 students = file.read()  
 file.close()  
 lbl1 = Label(win, text = "Фамилия студента / Время позданий (в мин) / Кол-во опозданий")  
 lbl1.pack()  
 lbl2 = Label(win, text=students)  
 lbl2.pack()  
   
def click\_btn4(): #функция, которая выполняется при нажатии на четвертую кнопку  
 win = Tk()  
 win.title("Наибольшее количество опозданий")  
 win.geometry('400x175')  
 lst = []  
 Text = ""  
 file = open('DataLateStudent.txt', 'r')  
 students = file.read().split("\n")  
 file.close()  
 max\_count = 0  
 for student in students:  
 student = student.split(" ")  
 if (int(student[2])) > (int(max\_count)):  
 max\_count = student[2]  
 for student in students:  
 student = student.split(" ")  
 if student[2] == max\_count:  
 lst.append(student)  
 for student in lst:  
 Text = Text + student[0] + " " + student[1] + " " + student[2] + "\n"  
 lbl1 = Label(win, text = "Фамилия студента / Время позданий (в мин) / Кол-во опозданий")  
 lbl1.pack()  
 lbl2 = Label(win, text = Text)  
 lbl2.pack()  
  
def click\_btn5(): #функция, которая выполняется при нажатии на пятую кнопку  
 win = Tk()  
 win.title("Наибольшее время опозданий")  
 win.geometry('400x175')  
 lst = []  
 Text = ""  
 file = open('DataLateStudent.txt', 'r')  
 students = file.read().split("\n")  
 file.close()  
 max\_time = 0  
 for student in students:  
 student = student.split(" ")  
 if (int(student[1])) > (int(max\_time)):  
 max\_time = student[1]  
 for student in students:  
 student = student.split(" ")  
 if student[1] == max\_time:  
 lst.append(student)  
 for student in lst:  
 Text = Text + student[0] + " " + student[1] + " " + student[2] + "\n"  
 lbl1 = Label(win, text = "Ф.И.О студента / Время позданий (в мин) / Кол-во опозданий")  
 lbl1.pack()  
 lbl2 = Label(win, text = Text)  
 lbl2.pack()  
   
window = Tk() #создание окна с кнопками  
window.title("Журнал посещений")  
window.geometry('290x219')   
btn1 = Button(window, text="Записать информацию о посещении", width = "40", pady = "10", background = "lime", command = click\_btn1)   
btn1.pack()  
btn2 = Button(window, text="Добавить информацию о появлении студента", width = "40", pady = "10", background = "yellow", command = click\_btn2)   
btn2.pack()  
btn3 = Button(window, text="Вывести список опоздавших", width = "40", pady = "10", background = "tan", command = click\_btn3)  
btn3.pack()  
btn4 = Button(window, text="Студент с наибольшим количеством опозданий", width = "40", pady = "10", background = "pink", command = click\_btn4)   
btn4.pack()  
btn5 = Button(window, text="Студент с наибольшим временем опозданий", width = "40", pady = "10", background = "orchid", command = click\_btn5)  
btn5.pack()  
window.mainloop()

## 2.8. Разработка документации по проекту

Назначение программы.

Функциональное назначение.

Данная программа предназначена для записи и добавления информации о посещении студентов, вывода списка опоздавших, данные студента с наибольшим кол-во опозданий и с наибольшим временем опозданий. Главный плюс этой программы – удобный интерфейс для пользователя. При этом программа взаимодействует с файлами.

Эксплуатационное назначение программы.

С помощью «Журнала посещений» пользователь может отслеживать опоздания всех студентов и узнавать данные студента с наибольшим количеством опозданий и с наибольшим временем опозданий.

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 «Журнал поспещений» тербуется:

* 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.

Выполнение программы

Запуск программы

При запуске программы открывается окно-меню

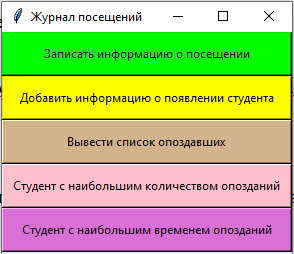


Рисунок 44. Окно-меню "Журнал посещений"

В открывшемся окне пользователь может выбрать необходимую для него функцию, которая должна выполнить программа.

Пользователь должен записать в текстовый файл информацию об опоздавших студентах (дата, фамилия, время прихода)

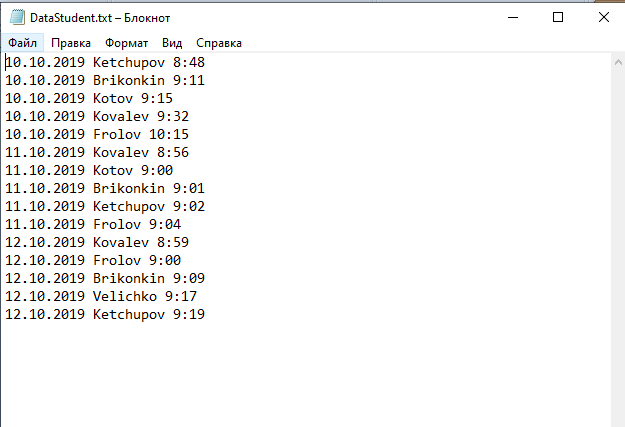


Рисунок 45. Исходный текстовый файл

Выполнение программы

При нажатии на кнопку «Записать информацию о посещении» открывается окно «Выбор диапазона дат»

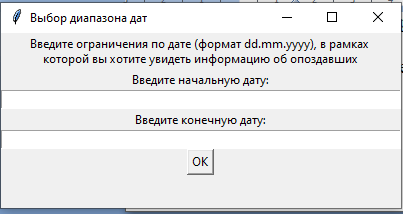


Рисунок 46. Окно "Выбор диапазона дат"

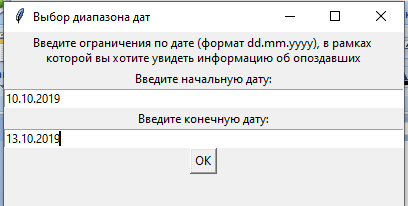


Рисунок 47. Ввод ограничения по дате

После введения ограничения по дате и нажатии «ОК» открывается окно «Уведомление», которое заверяет пользователя об успешном добавлении списка опоздавших в файл DataLateStudent.

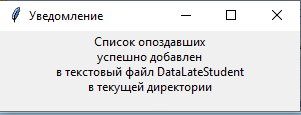


Рисунок 48. Уведомление после ввода диапазона дат

Результат этой кнопки – запись информации опоздавших в файл DateLateStudent в формате фамилия, минуты опозданий, количество опозданий.

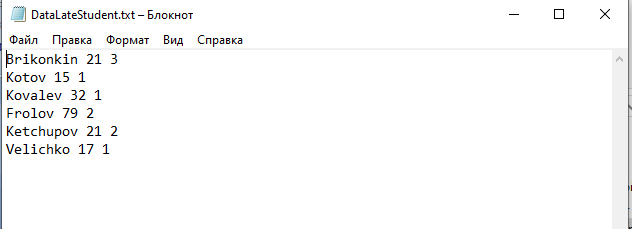


Рисунок 49. Результативный текстовый файл

При нажатии на кнопку «Добавить информацию о появлении студента» открывается окно «Заполнение информации», в котором пользователь должен ввести дату появления студента, фамилию студента, время появления студента.

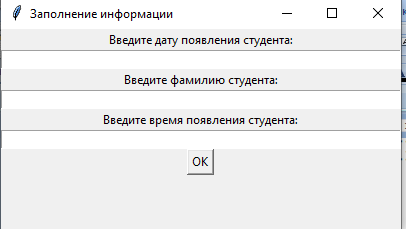


Рисунок 50. Заполнение информации

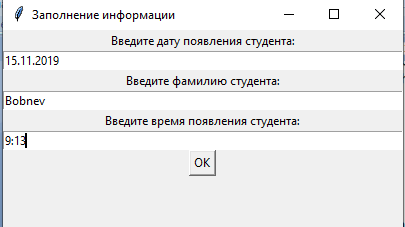


Рисунок 51. Ввод информации опоздавшего

После введения информации об опоздавшем и нажатии «ОК» открывается окно «Уведомление», которое заверяет пользователя об успешном добавлении введенной информации в файл DataStudent.

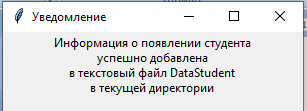


Рисунок 52. Уведомление об успешном добавлении записи в текстовый файл

Результат нажатия кнопки «Добавить информацию о появлении студента» – добавление информации об опоздавших в файл DateStudent.

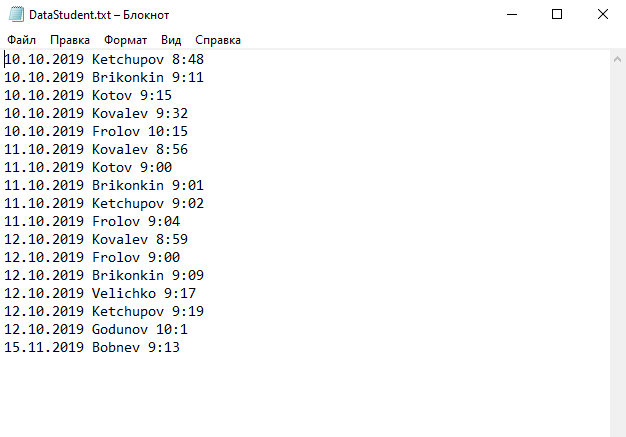


Рисунок 53. Результативный текстовый файл

При нажатии на кнопку «Вывести список опоздавших» открывается окно «Список опоздавших», содержащее информацию об опоздавших студентах в формате фамилия студента, время опозданий (мин), количество опозданий.

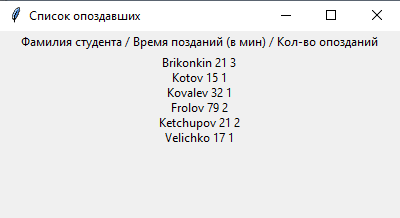


Рисунок 54. Окно "Список опоздавших"

При нажатии на кнопку «Студент с наибольшим количеством опозданий» открывается окно «Наибольшее кол-во опозданий», которое содержит информацию о студенте с наибольшим количеством опозданий.

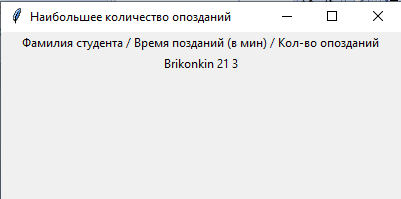


Рисунок 55. Окно "Наибольшее количество опозданий"

При нажатии на кнопку «Студент с наибольшим временем опозданий открывается окно «Наибольшее время опозданий», которое содержит информацию о студенте с наибольшим временем опозданий.

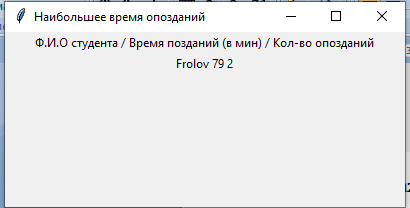


Рисунок 56. Окно "Наибольшее время опозданий"

# **Раздел 3. Техника решения задач с использованием библиотек.**

## 3.1. Техника работы с библиотекой NumPy

Приложения: np1.py, np2.py, np3.py, np4.py, np5.py, np6.py, np7.py, np8.py, np9.py, np10.py, np11.py, np12.py.

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

Установка:

• Через терминал командой: “pip install numpy”

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

Импорт модуля numpy:

import numpy

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

Листинг 44.

import numpy as np

a = np.array([1, 4, 5, 8], float)

print(a)

Вывод:

[1. 4. 5. 8.]

Пример двумерного массива (матрица):

Листинг 45.

import numpy as np

a = np.array([[1, 2, 3], [4, 5, 6]], float)

print(a)

Вывод:

[[1. 2. 3.]

[4. 5. 6.]]

Метод shape возвращает количество строк и столбцов в матрице:

Листинг 46. np3.py

import numpy as np

a = np.array([[1, 2, 3], [4, 5, 6]], float)

print(a.shape)

Вывод:

(2, 3)

Метод len возвращает длину первого измерения (оси):

Листинг 47.

import numpy as np

a = np.array([[1, 2, 3], [4, 5, 6]], float)

print(len(a))

Вывод:

2

Списки можно тоже создавать с массивов:

Листинг 48.

import numpy as np

a = np.array([1, 2, 3], float)

a.tolist()

print(list(a))

Вывод:

[1.0, 2.0, 3.0]

Заполнение массива одинаковым значением.

Листинг 49.

import numpy as np

a = np.array([1, 2, 3], float)

a.fill(0)

print(a)

Вывод:

[0. 0. 0.]

Транспонирование массивов:

Листинг 50.

import numpy as np

a = np.array(range(6), float).reshape((2, 3))

print(a.transpose())

Вывод:

[[0. 3.]

[1. 4.]

[2. 5.]]

Многомерный массив можно переконвертировать в одномерный при помощи метода flatten:

Листинг 51.

import numpy as np

a = np.array([[1, 2, 3], [4, 5, 6]], float)

print(a.flatten())

Вывод:

[1. 2. 3. 4. 5. 6.]

Два или больше массивов можно сконкатенировать при помощи метода concatenate:

Листинг 52.

import numpy as np

a = np.array([1,2], float)

b = np.array([3,4,5,6], float)

c = np.array([7,8,9], float)

print(np.concatenate((a, b, c)))

Вывод:

[1. 2. 3. 4. 5. 6. 7. 8. 9.]

NumPy предоставляет методы для работы с полиномами. Передавая список корней, можно получить коэффициенты уравнения:

Листинг 53.

import numpy as np

print(np.poly([-1, 1, 1, 10]))

Вывод:

[ 1. -11. 9. 11. -10.]

NumPy предоставляет еще некоторые методы для работы со статистическими данными в массивах.  
Медиана может быть найдена так:

Листинг 54.

import numpy as np

a = np.array([1, 4, 3, 8, 9, 2, 3], float)

print(np.median(a))

Вывод:

3.0

Важная часть каждой симуляции это способность генерировать случайные числа. Для этого мы используем встроенный в NumPy генератор псевдослучайных чисел в под-модуле random. Массив случайных чисел из полуинтервала [0.0, 1.0) может быть сгенерирован так:

Листинг 55.

import numpy as np

print(np.random.rand(5))

Вывод:

[0.24658796 0.22621755 0.59782307 0.26044404 0.60875373]

## 3.2. Техника работы с библиотекой Matplotlib

Приложения: mp1.py, mp2.py, mp3.py, mp4.py, mp5.py, mp6.py.

Для установки можно воспользоваться менеджером pip и установить Matplotlib самостоятельно, для этого введите в командной строке вашей операционной системы следующие команды:

>python -m pip install -U pip

>python -m pip install -U matplotlib

Первая из них обновит ваш pip, вторая установит matplotlib со всеми необходимыми зависимостями.

Листинг 56.

import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4, 5], [1, 2, 3, 4, 5])

plt.show()

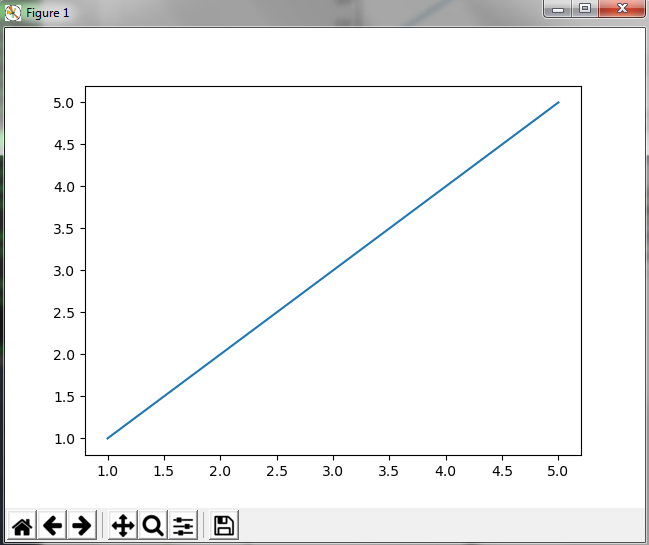


Рисунок 57. Результат выполнения программы

Для начала построим простую линейную зависимость, дадим нашему графику название, подпишем оси и отобразим сетку.

Листинг 57.

import matplotlib.pyplot as plt

import numpy as np

# Независимая (x) и зависимая (y) переменные

x = np.linspace(0, 10, 50)

y = x

# Построение графика

plt.title("Линейная зависимость y = x") # заголовок

plt.xlabel("x") # ось абсцисс

plt.ylabel("y") # ось ординат

plt.grid() # включение отображение сетки

plt.plot(x, y) # построение графика

plt.show()

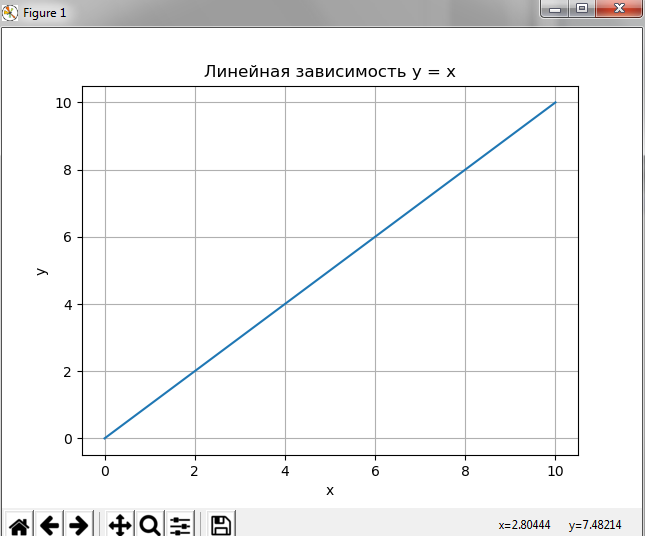


Рисунок 58. Результат выполнения программы

Изменим тип линии и ее цвет, для этого в функцию plot(), в качестве третьего параметра передадим строку, сформированную определенным образом, в нашем случае это “r–”, где “r” означает красный цвет, а “–” – тип линии – пунктирная линия.

Листинг 58.

import matplotlib.pyplot as plt

import numpy as np

# Независимая (x) и зависимая (y) переменные

x = np.linspace(0, 10, 50)

y = x

# Построение графика

plt.title("Линейная зависимость y = x") # заголовок

plt.xlabel("x") # ось абсцисс

plt.ylabel("y") # ось ординат

plt.grid() # включение отображение сетки

plt.plot(x, y, "r--") # построение графика

plt.show()

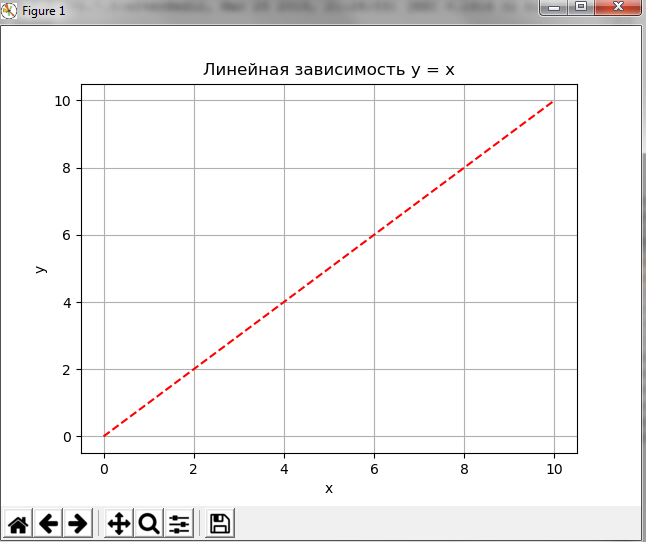


Рисунок 59. Результат выполнения программы

Построим несколько графиков на одном поле, для этого добавим квадратичную зависимость:

Листинг 59.

import matplotlib.pyplot as plt

import numpy as np

# Линейная зависимость

x = np.linspace(0, 10, 50)

y1 = x

# Квадратичная зависимость

y2 = [i\*\*2 for i in x]

# Построение графика

plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок

plt.xlabel("x") # ось абсцисс

plt.ylabel("y1, y2") # ось ординат

plt.grid() # включение отображение сетки

plt.plot(x, y1, x, y2) # построение графика

plt.show()

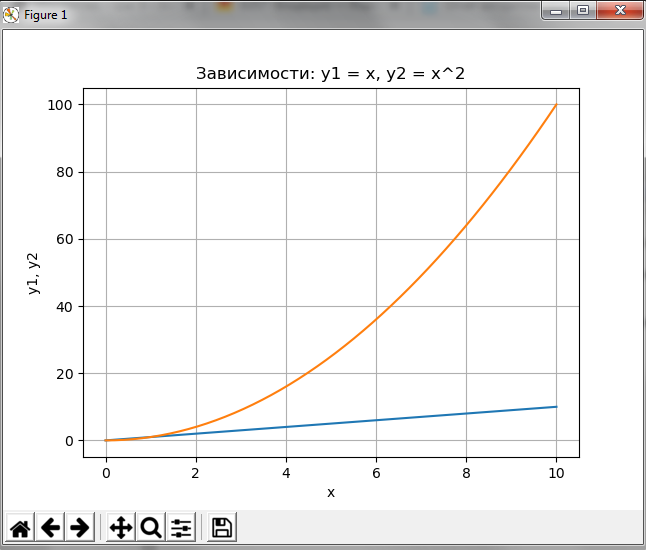


Рисунок 60. Результат выполнения программы

В приведенном примере в функцию plot() последовательно передаются два массива для построения первого графика и два массива для построения второго.

Отобразить два или более различных поля, на которых будет отображено по одному или более графику.

Листинг 60.

import matplotlib.pyplot as plt

import numpy as np

# Линейная зависимость

x = np.linspace(0, 10, 50)

y1 = x

# Квадратичная зависимость

y2 = [i\*\*2 for i in x]

# Построение графиков

plt.figure(figsize=(9, 9))

plt.subplot(2, 1, 1)

plt.plot(x, y1) # построение графика

plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок

plt.ylabel("y1", fontsize=14) # ось ординат

plt.grid(True) # включение отображение сетки

plt.subplot(2, 1, 2)

plt.plot(x, y2) # построение графика

plt.xlabel("x", fontsize=14) # ось абсцисс

plt.ylabel("y2", fontsize=14) # ось ординат

plt.grid(True) # включение отображение сетки

plt.show()

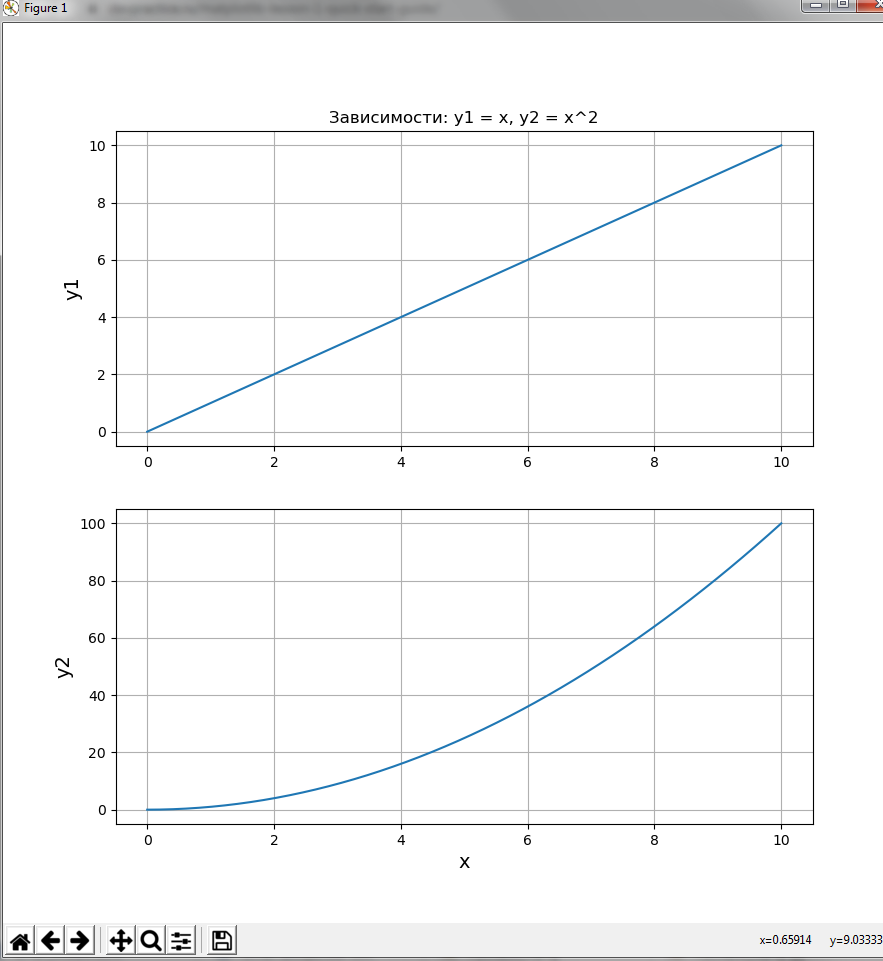


Рисунок 61. Результат выполнения программы

Листинг 61.

import matplotlib.pyplot as plt

import numpy as np

fruits = ["apple", "peach", "orange", "bannana", "melon"]

counts = [34, 25, 43, 31, 17]

plt.bar(fruits, counts) #Для вывода диаграммы

plt.title("Fruits!")

plt.xlabel("Fruit")

plt.ylabel("Count")

plt.show()

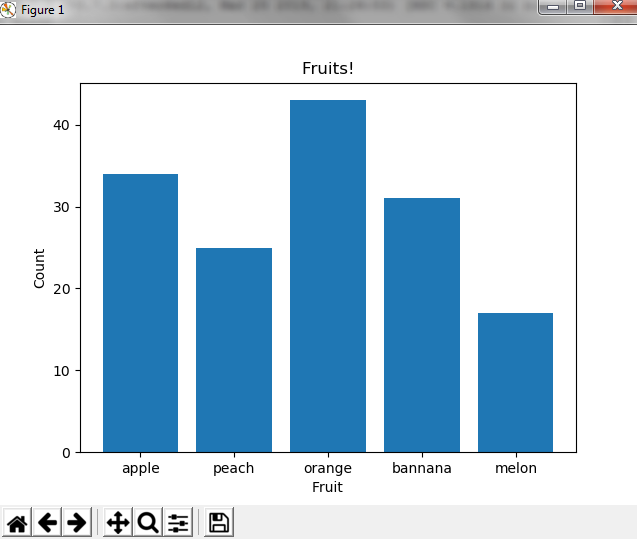


Рисунок 62. Результат выполнения программы

## 3.3 Техника работы с библиотекой PyQT

Для установки библиотеки используется команда:

pip install PyQt5

которую необходимо написать в командной строке.

PyQt5 является одним из наиболее часто используемых модулей для создания GUI приложений в Python. С помощью PyQt5 можно создавать графический интерфейс для программ.

Листинг 62. Создание Gui

import sys

from PyQt5.QtWidgets import (QApplication, QWidget)

app = QApplication(sys.argv)

w = QWidget()

w.resize(300,300)

w.setWindowTitle('Тест1')

w.show()

sys.exit(app.exec\_())

from PyQt5.QtWidgets import QApplication, QWidget

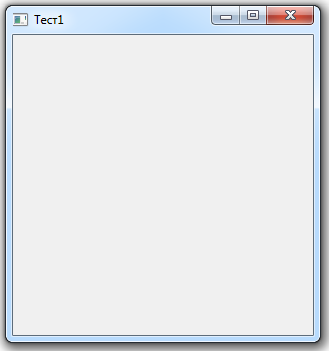


Рисунок 63. Результат выполнения программы

Этот оператор импортирует все модули, необходимые для создания GUI, в текущее пространство имен. Модуль QtWidgets содержит все основные виджеты, которые будут использованы далее.

app = QApplication(sys.argv)

Здесь создаётся объект класса QApplication. Этот шаг необходим для PyQt5; каждое приложение пользовательского интерфейса должно создавать экземпляр QApplication в качестве своего рода точки входа в приложение. sys.argv — это список параметров командной строки, которые можно передать приложению при запуске через оболочку или при автоматизации интерфейса.

w = QWidget()

Далее создаётся объект класса QWidget. QWidget является базовым классом всех объектов пользовательского интерфейса в Qt, и практически всё, что видит пользователь в приложении, является виджетом.

w.resize(300,300)

Метод resize класса QWidget позволяет устанавливать размеры любого виджета. В этом случае размер окна составляет 300 \* 300 пикселей.

w.setWindowTitle('Тест1')

С помощью метода setWindowTitle () можно дать имя окну.

w.show()

show() отображает виджет на экране монитора.

sys.exit(app.exec\_())

Метод app.exec\_() запускает цикл событий Qt / C ++.

Листинг 63. Заполнение окна

import sys

from PyQt5.QtWidgets import (QApplication, QWidget, QLabel, QPushButton, QMessageBox)

def dialog():

mbox = QMessageBox()

mbox.setWindowTitle('Новое окно')

mbox.setText("Вы совершили открытие")

mbox.setDetailedText("Немного текста для примера")

mbox.setStandardButtons(QMessageBox.Ok | QMessageBox.Cancel)

mbox.exec\_()

app = QApplication(sys.argv)

w = QWidget()

w.resize(300,300)

w.setWindowTitle('Тест2')

label = QLabel(w)

label.setText('Нажмите "Открыть" для открытия')

label.move(70,130)

label.show()

btn = QPushButton(w)

btn.setText('Открыть')

btn.move(110,150)

btn.show()

btn.clicked.connect(dialog)

w.show()

sys.exit(app.exec\_())

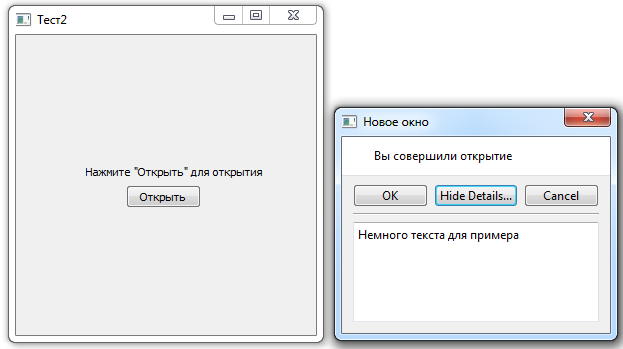


Рисунок 64. Результат выполнения второй программы

def dialog():

mbox = QMessageBox()

mbox.setWindowTitle('Новое окно')

mbox.setText("Вы совершили открытие")

mbox.setDetailedText("Немного текста для примера")

mbox.setStandardButtons(QMessageBox.Ok | QMessageBox.Cancel)

mbox.exec\_()

Здесь определяется метод с именем dialog, который создает виджет окна сообщения и устанавливает некоторый текст для кнопок и других полей. Метод dialog вызывается из основного блока программы, когда кнопка нажимается в определенном виджете (в данном случае, btn PushButton).

label = QLabel(w)

btn = QPushButton(w)

Два новых виджета: QLabel используется для вывода нередактируемого текста или заполнителей внутри виджета, а QPushButton используется для создания кликабельной кнопки.

label.move(70,130)

btn.move(110,150)

move() используется для установки позиции виджета относительно его родительского виджета. В первом случае метка будет перемещена на 100 пикселей с левой стороны и на 130 пикселей сверху окна.

btn.clicked.connect(dialog)

Это пример сигналов и слотов в PyQt. При поступлении сигнала (clicked – кнопка нажата) выполняется действие (dialog – создание, заполнение и открытие нового окна)

Другие виджеты PyQt5: QRadioButton – это поле ввода с выбираемой кнопкой, аналогичное переключателям в html.

rad = QRadioButton(w)

rad.setText('Имя кнопки')

rad.setChecked(False) #Так кнопка «нажата» по умолчанию

Безымянный

Рисунок 65 Пример RadioButton

QLineEdit – это поле ввода, которое позволяет пользователю вводить одну строку текста.

line = QLineEdit(w)

Безымянный

Рисунок 66 Пример QlineEdit

PyQt имеет мощный механизм управления компоновкой, который можно использовать для создания расширенных пользовательских интерфейсов приложений. Рассмотрим два класса, которые используются в Qt для создания и управления макетами: QBoxLayout и QGridLayout.

QBoxLayout используется для выравнивания дочерних виджетов макета в горизонтальной или вертикальной строке. Два основных класса, которые наследуются от QBoxLayout, это:

QHBoxLayout: используется для выравнивания дочерних виджетов по горизонтали.

QVBoxLayout: используется для выравнивания дочерних виджетов по вертикали.

Листинг 64. Вставка виджетов

import sys

from PyQt5.QtWidgets import \*

app = QApplication([])

w = QWidget()

w.setWindowTitle("Богатыри1")

z = QWidget()

z.setWindowTitle("Богатыри2")

z.resize(200, 100)

btn1 = QPushButton("Алёша Попович")

btn2 = QPushButton("Добыня Никитич")

btn3 = QPushButton("Илья Муромец")

btn11 = QPushButton("Алёша Попович")

btn22 = QPushButton("Добыня Никитич")

btn33 = QPushButton("Илья Муромец")

hbox = QHBoxLayout(w)

hbox.addWidget(btn1)

hbox.addWidget(btn2)

hbox.addWidget(btn3)

zbox = QVBoxLayout(z)

zbox.addWidget(btn11)

zbox.addWidget(btn22)

zbox.addWidget(btn33)

w.show()

z.show()

sys.exit(app.exec\_())

sys.exit(app.exec\_())

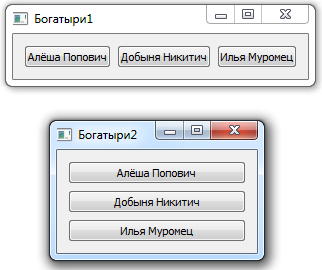


Рисунок 67 Результат выполнения третьей программы

Метод addWidget() используется для вставки виджетов в макет HBox или VBox. Он также используется в других макетах, где принимает другое количество параметров.

QGridLayout используется для создания интерфейсов, в которых виджеты расположены в виде сетки (например, матрицы или 2D-массива). Чтобы вставить элементы в сетку, можно использовать матричное представление, чтобы определить количество строк и столбцов в сетке, а также положение этих элементов.

Листинг 65. Создание кнопок

import sys

from PyQt5.QtWidgets import \*

app = QApplication([])

w = QWidget()

w.setWindowTitle("Тест4")

grid = QGridLayout(w)

grid.addWidget(QPushButton("Кнопка1"),0,0)

grid.addWidget(QPushButton("Кнопка2"),0,1)

grid.addWidget(QPushButton("Кнопка3"),1,0)

grid.addWidget(QPushButton("Кнопка4"),1,1)

w.show()

sys.exit(app.exec\_())

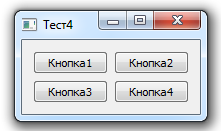


Рисунок 68 Результат выполнения четвертой программы

## 3.4. Техника работы с библиотекой Pygame

Pygame – это библиотека модулей для языка Python, созданная для разработки 2D игр. Также Pygame могут называть фреймворком. В программировании понятия "библиотека" и "фреймворк" несколько разные. Но когда дело касается классификации конкретного инструмента, не все так однозначно.

В любом случае, фреймворк является более мощным по-сравнению с библиотекой, он накладывает свою специфику на особенности программирования и сферу использования продукта. С точки зрения специфики Pygame – это фреймворк. Однако его сложно назвать "мощным инструментом". По своему объему и функционалу это скорее библиотека.

Также существует понятие "игрового движка" как программной среды для разработки игр. По своему назначению Pygame можно считать игровым движком. В то же время, с точки зрения классификации программного обеспечения, Pygame является API для Питона к API библиотеки SDL.

API – это интерфейс (в основном набор функций и классов) для прикладного (часто более высокоуровневого) программирования, который предоставляет, например, та или иная библиотека. SDL – это библиотека, которая работает с мультимедийными устройствами компьютера.

В этом смысле Pygame можно сравнить с Tkinter, который через свои функции и классы предоставляет Питону доступ к графической библиотеке Tk.

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

python3 -m pip install -U pygame --user

или

sudo apt-get install python3-pygame

Для Windows:

Открыв командную строку (cmd) прописываем:

pip install pygame

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

Python

Import pygame

Листнинг 66.

# здесь подключаются модули

import pygame

# здесь определяются константы, классы и функции

FPS = 60

# здесь происходит инициация, создание объектов и др.

pygame.init()

pygame.display.set\_mode((600, 400))

clock = pygame.time.Clock()

# если надо до цикла отобразить объекты на экране

pygame.display.update()

# главный цикл

while True:

# задержка

clock.tick(FPS)

# цикл обработки событий

for i in pygame.event.get():

if i.type == pygame.QUIT:

exit()

# --------

# изменение объектов и многое др.

# --------

# обновление экрана

pygame.display.update()

Начнем с функции rect() модуля draw:

pygame.draw.rect(sc, (255, 255, 255), (20, 20, 100, 75))

pygame.draw.rect(sc, (64, 128, 255), (150, 20, 100, 75), 8)

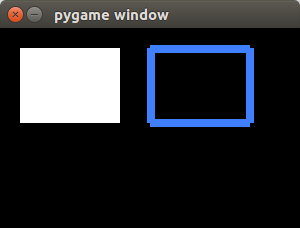


Рисунок 69. Результат выполнения программы

Если указывается толщина контура (последний аргумент во второй строке), то прямоугольник будет незаполненным, а цвет определит цвет рамки. Третьим аргументом является кортеж из четырех чисел. Первые два определяют координаты верхнего левого угла прямоугольника, вторые – его ширину и высоту.

Следует отметить, что в функцию draw.rect() и некоторые другие третьим аргументом можно передавать не кортеж, а заранее созданный экземпляр Rect.

pygame.draw.line(sc, WHITE, [10, 30], [290, 15], 3)

pygame.draw.line(sc, WHITE, [10, 50], [290, 35])

pygame.draw.aaline(sc, WHITE, [10, 70], [290, 55])

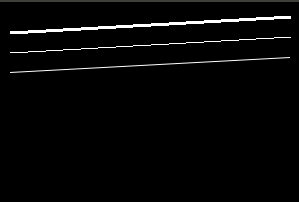


Рисунок 70. Результат выполнения программы

Координаты можно передавать как в виде списка, так и кортежа.

Функции lines() и aalines() рисуют ломанные линии.

Функция polygon() рисует произвольный многоугольник. Задаются координаты вершин.

pygame.draw.polygon(sc, WHITE, [[150, 10], [180, 50], [90, 90], [30, 30]])

pygame.draw.polygon(sc, WHITE, [[250, 110], [280, 150], [190, 190], [130, 130]])

pygame.draw.aalines(sc, WHITE, True, [[250, 110], [280, 150], [190, 190], [130, 130]])

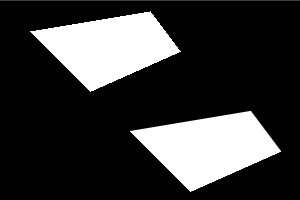


Рисунок 71. Результат выполнения программы

Сглаженная ломаная здесь повторяет контур многоугольника, чем сглаживает его ребра.

Так же как в случае rect() для polygon() можно указать толщину контура.

Функция circle() рисует круги. Указывается центр окружности и радиус:

pygame.draw.circle(sc, YELLOW, (100, 100), 50)

pygame.draw.circle(sc, PINK, (200, 100), 50, 10)

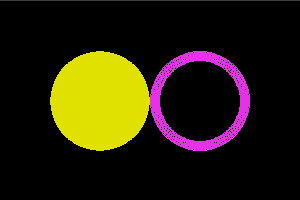


Рисунок 72. Результат выполнения программы

В случае эллипса передается описывающая его прямоугольная область:

pygame.draw.ellipse(sc, GREEN, (10, 50, 280, 100))

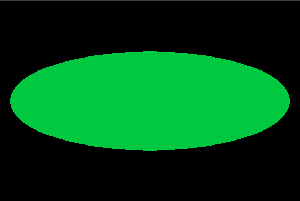


Рисунок 73. Результат выполнения программы

Наконец, дуга:

pi = 3.14

pygame.draw.arc(sc, WHITE, (10, 50, 280, 100), 0, pi)

pygame.draw.arc(sc, PINK, (50, 30, 200, 150), pi, 2\*pi, 3)

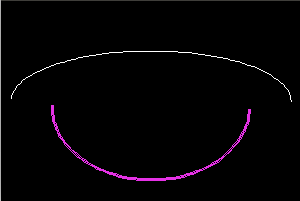


Рисунок 74. Результат выполнения программы

Указывается прямоугольник, описывающий эллипс, из которого вырезается дуга. Четвертый и пятый аргументы – начало и конец дуги, выраженные в радианах. Нулевая точка справа.

У обоих типов событий клавиатуры есть атрибуты key и mod. В key записывается конкретная клавиша, которая была нажата или отжата. В mod – клавиши-модификаторы (Shift, Ctrl и др.), которые были зажаты в момент нажатия или отжатия обычной клавиши. У событий KEYDOWN также есть поле unicode, куда записывается символ нажатой клавиши (тип данных str).

Рассмотрим, как это работает. Пусть в центре окна имеется круг, который можно двигать по горизонтали клавишами стрелок клавиатуры:

1. import pygame
3. FPS = 60
4. W = 700 # ширина экрана
5. H = 300 # высота экрана
6. WHITE = (255, 255, 255)
7. BLUE = (0, 70, 225)
9. pygame.init()
10. sc = pygame.display.set\_mode((W, H))
11. clock = pygame.time.Clock()
13. # координаты и радиус круга
14. x = W // 2
15. y = H // 2
16. r = 50
18. while 1:
19. sc.fill(WHITE)
21. pygame.draw.circle(sc, BLUE, (x, y), r)
23. pygame.display.update()
25. for i in pygame.event.get():
26. if i.type == pygame.QUIT:
27. exit()
28. elif i.type == pygame.KEYDOWN:
29. if i.key == pygame.K\_LEFT:
30. x -= 3
31. elif i.key == pygame.K\_RIGHT:
32. x += 3
34. clock.tick(FPS)

От классов pygame.font.Font и pygame.font.SysFont создаются объекты-шрифты. Второй класс берет системные шрифты, поэтому конструктору достаточно передать имя шрифта. Конструктору Font надо передавать имя файла шрифта. Например:

pygame.font.SysFont('arial', 36)

pygame.font.Font('/usr/share/fonts/truetype/msttcorefonts/Arial.ttf', 36)

Второй аргумент – это размер шрифта в пикселях.

Узнать, какие шрифты есть в системе, можно с помощью функции get\_fonts():

>>> pygame.font.get\_fonts()

['cmmi10', 'umeminchos3', 'kacstbook', 'freesans', 'lohitpunjabi', ...

Узнать адрес конкретного шрифта:

>>> pygame.font.match\_font('verdana')

'/usr/share/fonts/truetype/msttcorefonts/Verdana.ttf'

Вы можете скопировать шрифт в каталог программы и обращаться к нему без адреса:

pygame.font.Font('Verdana.ttf', 24)

В pygame есть шрифт по-умолчанию. Чтобы использовать его, вместо имени файла в конструктор надо передать объект None:

pygame.font.Font(None, 24)

От обоих классов (Font и SysFont) создаются объекты типа Font.

Метод render() экземпляра Font создает поверхность (экземпляр Surface), на которой "написан" переданный в качестве аргумента текст, шрифтом, к которому применяется метод. Вторым аргументом указывается сглаживание (0 – нет, 1 – есть), третьим – цвет текста. При необходимости четвертым аргументом можно указать цвет фона.

import pygame

pygame.init()

sc = pygame.display.set\_mode((300,200))

sc.fill((255, 255, 255))

f1 = pygame.font.Font(None, 36)

text1 = f1.render('Hello Привет', 1, (180, 0, 0))

f2 = pygame.font.SysFont('serif', 48)

text2 = f2.render("World Мир", 0, (0, 180, 0))

sc.blit(text1, (10, 50))

sc.blit(text2, (10, 100))

pygame.display.update()

while 1:

for i in pygame.event.get():

if i.type == pygame.QUIT:

exit()

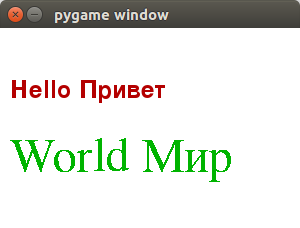


Рисунок 75. Результат выполнения программы

## Список использованных источников

1. <https://stepik.org/course/67/syllabus>
2. <https://www.python.org/>
3. Марк Лутц - "Изучаем Python"
4. <https://younglinux.info/pygame>
5. <https://python-scripts.com/pyqt5>
6. <https://www.jetbrains.com/ru-ru/pycharm/>
7. https://ru.wikiversity.org/wiki/%D0%9A%D1%83%D1%80%D1%81\_%D0%BF%D0%BE\_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B5\_Tkinter\_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0\_Python