

Колледж космического машиностроения и технологий

ОТЧЕТ

По учебной практике УП.01.01 Разработка программных модулей

программного обеспечения для компьютерных систем

специальность 09.02.03 Программирование в компьютерных системах

Выполнили студенты:

Герасимов Д. А.

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

Груздев Р. И.

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

Гусятинер Л. Б.

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

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

Королев, 2020

Содержание отчёта

[**Раздел 1. Техника решения задач с использованием структурного и объектно-ориентированного программирования.** 2](#_Toc69212292)

[1.1 Установка интерпретатора Python 3 и настройка окружения 2](#_Toc69212293)

[1.2 Техника работы в командной строке и среде IDLE 5](#_Toc69212294)

[1.3 Техника работы с линейными и разветвляющимися программами 7](#_Toc69212295)

[1.4 Техника работы с циклическими программами, цикл while 16](#_Toc69212296)

[1.5 Техника работы с числами 24](#_Toc69212297)

[1.6 Техника работы со строками 31](#_Toc69212298)

[1.7 Техника работы со списками 36](#_Toc69212299)

[1.8 Техника работы с циклом for и генераторами списков 41](#_Toc69212300)

[1.9 Техника работы с функциями 48](#_Toc69212301)

[1.10 Техника работы со словарями 54](#_Toc69212302)

[1.11 Техника работы с множествами 64](#_Toc69212303)

[1.12 Техника работы с кортежами 72](#_Toc69212304)

[1.13 Техника работы с файлами 78](#_Toc69212305)

[1.14 Техника работы с модулями 86](#_Toc69212306)

[1.15 Техника работы с классами 102](#_Toc69212307)

[**Раздел 2. Техника решения задач с использованием библиотек** 113](#_Toc69212308)

[2.1 Установка и настройка среды JetBrains PyCharm 113](#_Toc69212309)

[2.2 Техника работы с базами данных 119](#_Toc69212310)

[2.3 Техника работы с библиотекой tkinter 123](#_Toc69212311)

[2.4 Техника работы с библиотекой NumPy 126](#_Toc69212312)

[2.5 Техника работы с библиотекой Matplotlib 128](#_Toc69212313)

[2.6 Элементы работы с библиотекой PyQt 128](#_Toc69212314)

[2.7 Элементы работы с библиотекой PyGame 128](#_Toc69212315)

[**Раздел 3. Разработка проекта с графическим интерфейсом** 140](#_Toc69212316)

[3.1 Изучение входной и выходной документации 150](#_Toc69212317)

[3.2 Разработка требований к проекту. Построение диаграммы использования. 152](#_Toc69212318)

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



Рис. 1. Официальный сайт Python

Порядок установки на Windows:

1. Запустить скачанный установочный файл.

2. Выбрать способ установки.



Рис. 2. Установщик Python

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



Рис. 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)



Рис. 4. «Продвинутые» опции установки

5. После успешной установки:



Рис. 5. Сообщение об успешной установке

## Техника работы в командной строке и среде IDLE

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



Рис. 6. 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



Рис. 7. Первая программа

## Техника работы с линейными и разветвляющимися программами

Приложения: K4\_1.py, K4\_2\_1.py, K4\_2\_2.py

Листинг 1. K4\_1.py makefile data.txt text.txt

'''

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К4\_1. Техника работы с линейными программами

Задание 1. Разработать программы по темам

- input

- print

- stdin, stdout, stderr

- форматная строка и метод формат

'''

import sys

stderr\_age = stderr\_value = sys.stderr

stdout\_f = stdout\_age = stdout\_value = sys.stdout

stdin\_f = sys.stdin

stdin\_f = open('text.txt',"r")

stdout\_f = open('data.txt', "w")

text\_list = []

i = 0 # Операнд строки(используется в виде индекса list'a), изначально равен 0

text\_list = stdin\_f.readlines() # Упаковываем каждую строчку из тиикстишника в элемент list'a

print("%s" % (text\_list[i]), end = "") # Узнаём имя товарища

name = str(input())

stdout\_f.write("Name: {} \n".format(name))

i += 1 # 1

print("%s" % (text\_list[i]), end = "")

i += 1 # 2

while True: # Узнаём сколько товарищу лет

print("%s" % (text\_list[i]), end="")

age = int(input())

if age < 1:

i += 1 # 3

stderr\_age.write(text\_list[i])

i -= 1 # 2

else:

stdout\_f.write("Age: {} \n".format(str(age)))

i += 2 # 4

stdout\_age.write(text\_list[i])

break

i += 1 # 5

while True: # Узнаём как товарищ оценивает данную программу

print("%s" % (text\_list[i]), end = "")

value = str(input())

if value == 'like':

stdout\_f.write("Review: {} \n".format(value))

i += 1 # 6

stdout\_value.write(text\_list[i])

stdin\_f.close() # Закрывает stdin

stdout\_f.close() # Закрывает stdout

exit(0)

elif value == 'dis':

i += 2 # 7

stderr\_value.write(text\_list[i]+'\n')

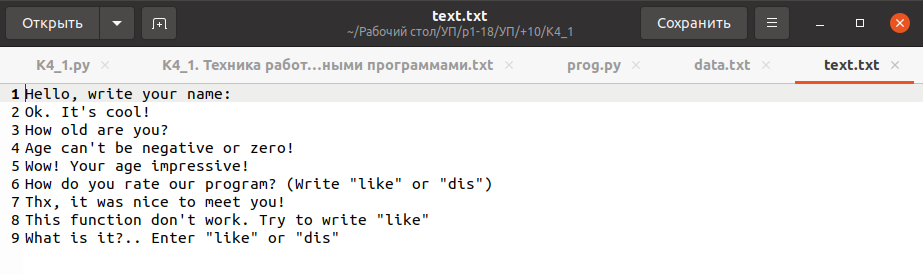
i -= 2

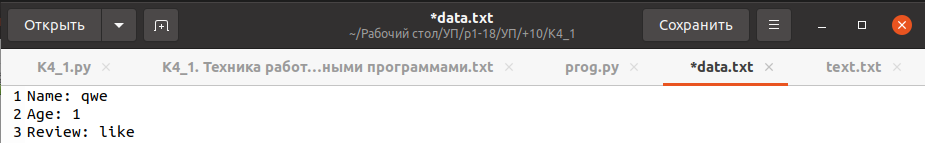
else:

i += 3

stderr\_value.write(text\_list[i]+'\n')

i -= 3

Рис. 8. text.txt

Рис. 9. data.txt

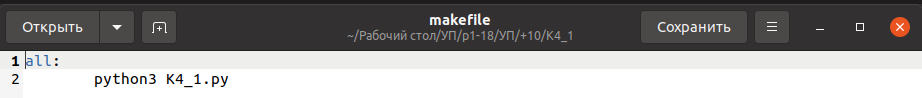


Рис. 10. makefile

Листинг 2. K4\_2\_1.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К4\_2. Техника работы с разветвляющимися программами

Задание 1. Разработать программу для печати даты прописью

Пример ввода: 15.12.1983

Пример вывода: Пятнадцатое декабря одна тысяча девятсот восемьдесят третьего года

'''

def get\_date(date):

day\_list = ['первое', 'второе', 'третье', 'четвёртое',

'пятое', 'шестое', 'седьмое', 'восьмое',

'девятое', 'десятое', 'одиннадцатое', 'двенадцатое',

'тринадцатое', 'четырнадцатое', 'пятнадцатое', 'шестнадцатое',

'семнадцатое', 'восемнадцатое', 'девятнадцатое', 'двадцатое',

'двадцать первое', 'двадцать второе', 'двадцать третье',

'двадацать четвёртое', 'двадцать пятое', 'двадцать шестое',

'двадцать седьмое', 'двадцать восьмое', 'двадцать девятое',

'тридцатое', 'тридцать первое']

month\_list = ['января', 'февраля', 'марта', 'апреля', 'мая', 'июня','июля', 'августа', 'сентября', 'октября', 'ноября', 'декабря']

# единицы

year\_list\_unit = ['','первого','второго','третьего', 'четвёртого', 'пятого', 'шестого', 'седьмого', 'восьмого', 'девятого']

year\_list\_unit2 = ['', 'одиннадцатого', 'двенадцатого', 'тринадцатого', 'четырнадцатого', 'пятнадцатого',

'шестнадцатого', 'семнадцатого', 'восемнадцатого', 'девятнадцатого']

# десятки

year\_list\_dec = ['','','двадцать','тридцать','сорок','пятьдесят','шестьдесят','семьдесят','восемьдесят','девяносто']

year\_list\_dec2 = ['', 'десятого', 'двадцатого', 'тридцатого', 'сорокового', 'пятидесятого', 'шестидесятого', 'семидесятого', 'восемидесятого',

'девяностого']

# сотни

year\_list\_hun = ['','сто','двести','триста','четыреста','пятьсот','шестьсот','семьсот','восемьсот','девятьсот']

year\_list\_hun2 = ['','сотого','двухсотого','трёхсотого','четырёхсотого','пятьсотого','шестьсотого','семьсотого','восьмисотого','девятьсотого']

# тысячи

year\_list\_th = ['','тысяча','две тысячи','три тысячи']

year\_list\_th2 = ['','тысячного','двух тысячного','трёх тысячного']

date\_list = date.split('.')

date\_y = [int(date\_list[2])//1000, int(date\_list[2])%1000//100, int(date\_list[2])%100//10,int(date\_list[2])%10]

if(date\_y[2] == 1):

return (day\_list[int(date\_list[0]) - 1] + ' ' +

month\_list[int(date\_list[1]) - 1] + ' ' +

year\_list\_th[date\_y[0]] + ' ' + year\_list\_hun[date\_y[1]-1] + ' ' + year\_list\_unit2[date\_y[3]] + ' ' + 'года')

elif (date\_y[0] == date\_y[1] == date\_y[2] == date\_y[3] == 0):

return (day\_list[int(date\_list[0]) - 1] + ' ' +

month\_list[int(date\_list[1]) - 1] + ' ' + 'нулевого года')

elif (date\_y[1] == date\_y[2] == date\_y[3] == 0):

return (day\_list[int(date\_list[0]) - 1] + ' ' +

month\_list[int(date\_list[1]) - 1] + ' ' +

year\_list\_th2[date\_y[0]] + ' ' + 'года')

elif(date\_y[2] == date\_y[3] == 0):

return (day\_list[int(date\_list[0]) - 1] + ' ' +

month\_list[int(date\_list[1]) - 1] + ' ' +

year\_list\_th[date\_y[0]] + ' ' + year\_list\_hun2[date\_y[1]] + ' ' + 'года')

elif(date\_y[3] == 0):

return (day\_list[int(date\_list[0]) - 1] + ' ' +

month\_list[int(date\_list[1]) - 1] + ' ' +

year\_list\_th[date\_y[0]] + ' ' + year\_list\_hun[date\_y[1]] + ' ' + year\_list\_dec2[

date\_y[2]] + ' ' + 'года')

return (day\_list[int(date\_list[0]) - 1] + ' ' +

month\_list[int(date\_list[1]) - 1] + ' ' +

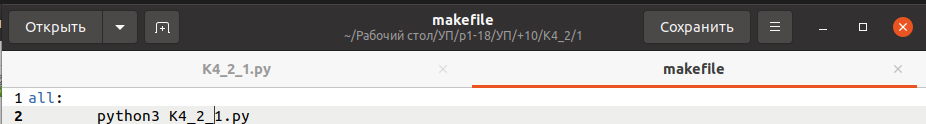
year\_list\_th[date\_y[0]] + ' ' + year\_list\_hun[date\_y[1]] + ' ' + year\_list\_dec[date\_y[2]] + ' ' + year\_list\_unit[date\_y[3]] + ' ' + ' года')

date = input()

while(date != 'stop'):

print(get\_date(date))

date = input()

Рис. 11. makefile

Листинг 3. K4\_2\_2.py dict.txt dict\_m.txt list.txt list\_m.txt set.txt set\_m makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К4\_2. Техника работы с разветвляющимися программами

Задание 2. Разработать программу с меню для демонстрации работы с типами данных:

список(list), словарь(dict), множество(set)

Меню -> выбор типа данных -> выбор метода -> краткая справка

'''

import sys

stdin\_f = sys.stdin

print('Добро пожаловать в меню:')

print('{| 1. Узнать про сипсок |}')

print('{| 2. Узнать про множество |}')

print('{| 3. Узнать про словарь |}')

print('{| 0. Выйти из программы |}')

print('Выберите пункт меню: ')

a = int(input())

if(a == 1):

print('1. Пояснение (что такое список)')

print('2. Некоторые методы списка')

print('Выберите пункт:')

b = int(input())

if(b == 1):

stdin\_f = open('list.txt', "r")

print(\*stdin\_f)

stdin\_f.close()

elif(b == 2):

stdin\_f = open('list\_m.txt', "r")

print(\*stdin\_f)

stdin\_f.close()

elif(a == 2):

print('1. Пояснение (что такое множество)')

print('2. Некоторые методы множества')

print('Выберите пункт:')

c = int(input())

if(c == 1):

stdin\_f = open('set.txt', "r")

print(\*stdin\_f)

stdin\_f.close()

elif(c == 2):

stdin\_f = open('set\_m.txt', "r")

print(\*stdin\_f)

stdin\_f.close()

elif(a == 3):

print('1. Пояснение (что такое словарь)')

print('2. Некоторые методы словаря')

print('Выберите пункт:')

d = int(input())

if(d == 1):

stdin\_f = open('dict.txt', "r")

print(\*stdin\_f)

stdin\_f.close()

elif(d == 2):

stdin\_f = open('dict\_m.txt', "r")

print(\*stdin\_f)

stdin\_f.close()

elif(a == 0):

print('Goodbye!')

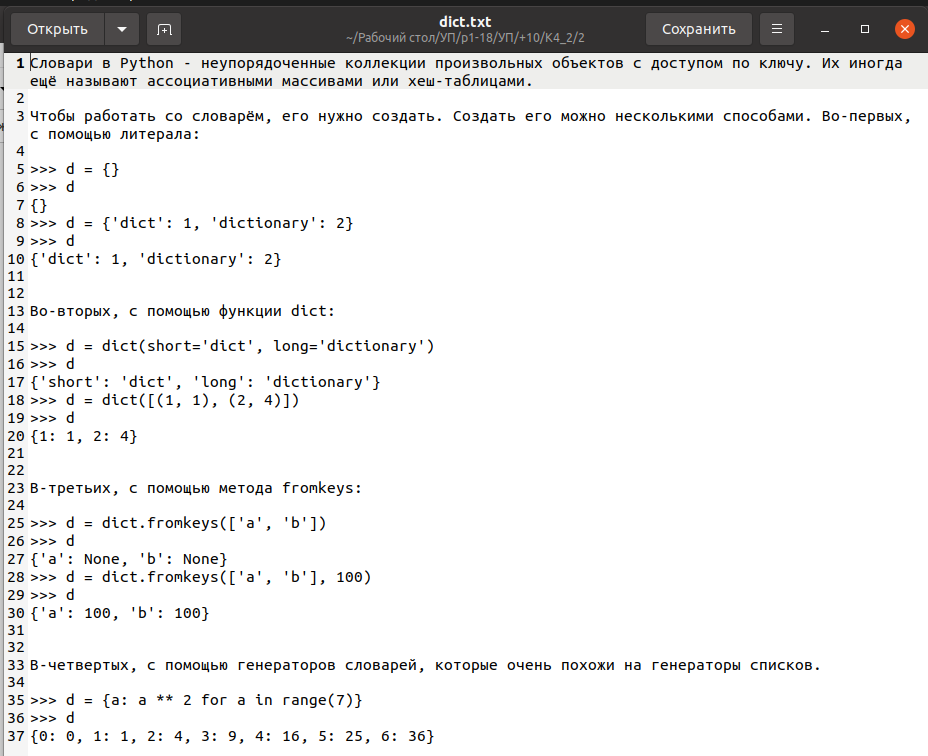
exit(0)

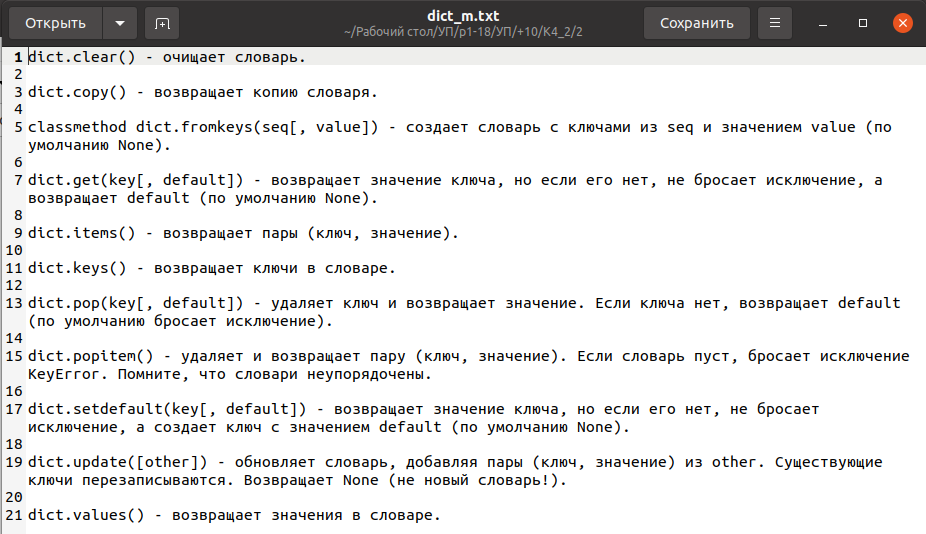
else:

while(a < 0 or a > 3):

print("Попробуйте ввести корректный пункт меню:")

a = int(input())

Рис. 12. dict.txt

Рис. 13. dict\_m.txt

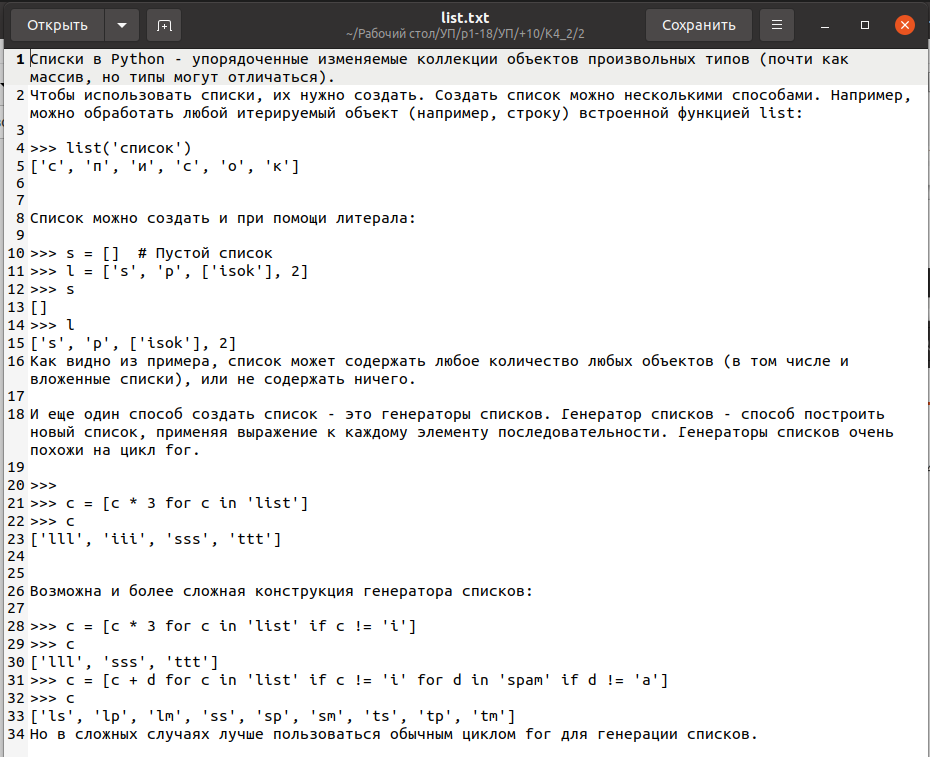
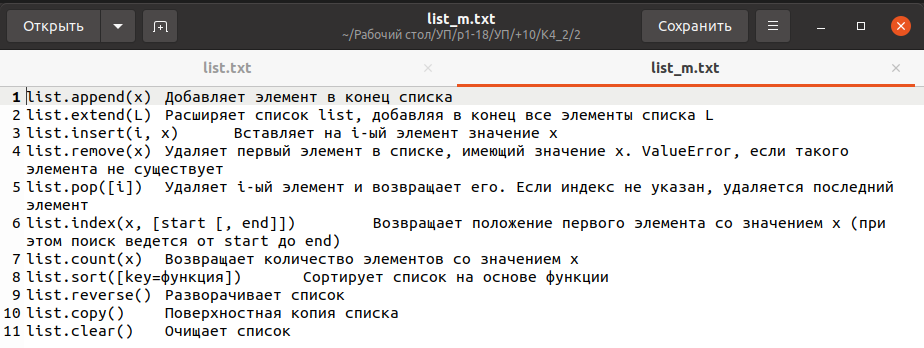
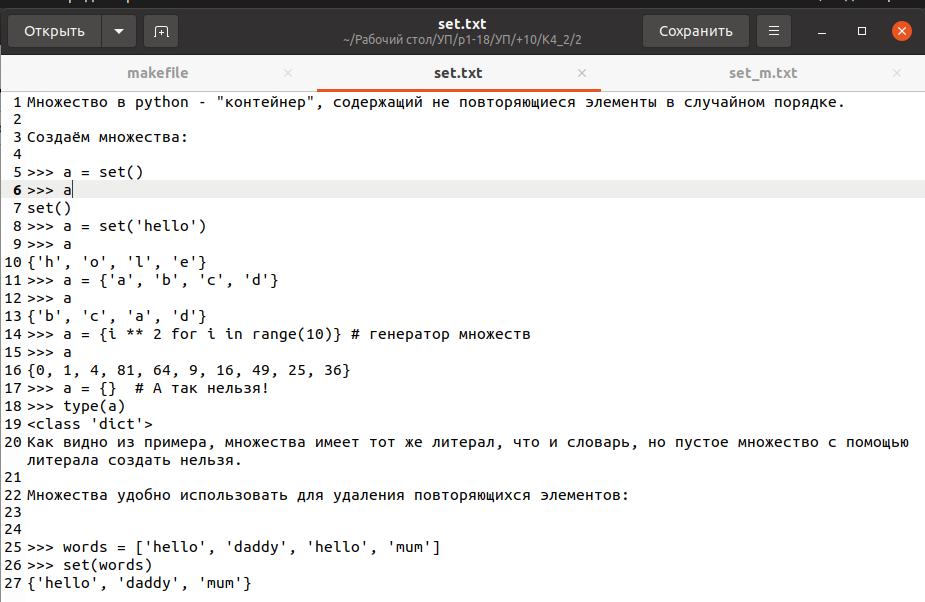
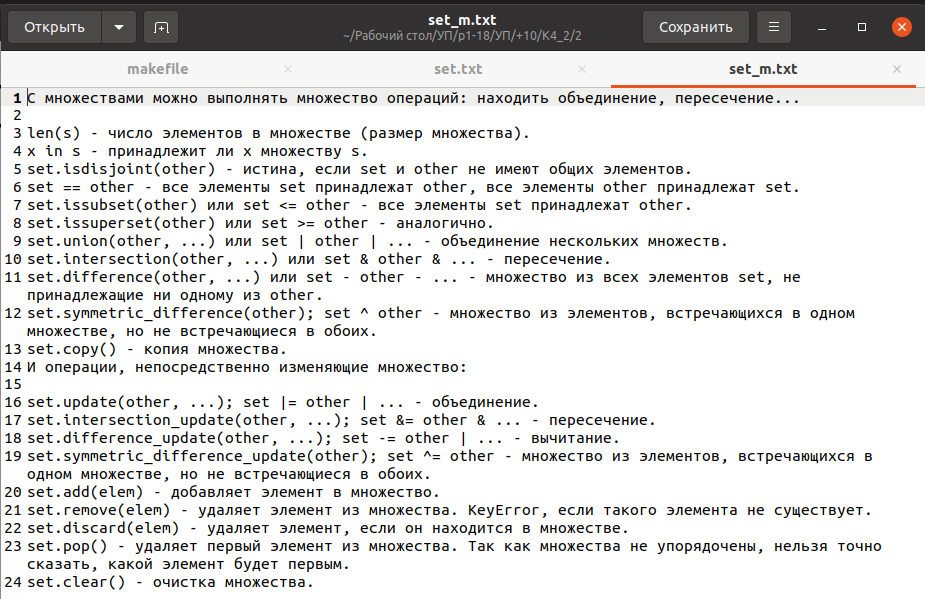
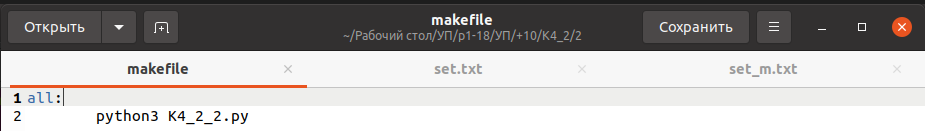


Рис. 14. list.txt

Рис. 15. list\_m.txt

Рис. 16. set.txt

Рис. 17. set\_m.txt

Рис. 18. makefile

## Техника работы с циклическими программами, цикл while

Приложения: K5\_1.py K5\_1\_2.py K5\_2\_1.py K5\_2\_2.py K5\_2\_3.py K5\_2\_4.py K5\_2\_5.py

Листинг 4. K5\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К5\_1. Техника работы с циклическими программами \_ цикл while

Задание 1. На плоскости нарисован квадрат заданного размера с левой нижней

вершиной в начале координат. В квадрат вписывается окружность.

Случайным образом в квадрате выбирается 1000 точек.

а) нужно определить, сколько точек попало внутрь круга

б) считая количество точек пропорциональным площади, найти отношение площадей

круга и квадрата

в) по этому отношению определить приближённое значение числа пи

г) определить, насколько найденное значение отличается от "библиотечного".

'''

import math

import random

n = int(input("Введите размер стороны квадрата: "))

S\_sq = n\*n #Площадь квадрата

r = n/2 #Радиус круга

S\_crc = (math.pi \* S\_sq)/4 #Площадь круга

NPOINTS = 1000

points = [[random.randint(0, n) for x in range(2)] for i in range(NPOINTS)]

#print(\*points)

num\_in\_points = 0 #Кол-во точек в круге

for i in range(NPOINTS):

#гипотенуза

if math.sqrt(points[i][0]\*\*2 + points[i][1]\*\*2) <= r:

num\_in\_points += 1

print("a) Количество точек попавших внутрь круга: ", num\_in\_points)

area\_S = S\_sq/S\_crc #Отношение площадей круга и квадрата

print("б) Отношение площадей круга и квадрата: ", area\_S)

pi = 4/area\_S #Наше число пи

print("в) Приближённое значение числа пи относительно наших вычислений: ", pi)

diff\_\_pi = math.pi - pi #Разница библиотечного числа пи от нашего

print("г) Разница \"библиотечного\" числа пи от пи который мы вычислили: ", pi)

print(" \"Библиотечный\" пи: ", math.pi)

print(" Пи который мы вычислили: ", pi)

print(" Разница: ", diff\_\_pi)

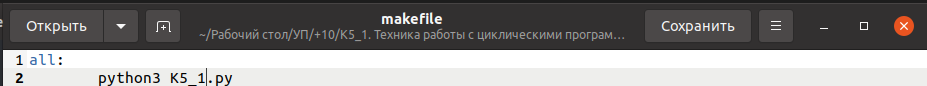


Рис. 19. Makefile

Листинг 5. K5\_1\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К5\_1. Техника работы с циклическими программами \_ цикл while

Задание 2.Придумать пример(ы) на использование break / continue /else.

'''

summ = 0

count = 0

while True:

count += 1

num = int(input("Введите число: "))

if num < 10:

continue

if num > 100:

summ += num

break

else:

summ += num

print(f"{count}| num: {num}, summ: {summ}")

print(f"{count}| num: {num}, summ: {summ}")

print()

print("Вводится строка и искомая буква. Надо найти номер буквы в строке и вывести её номер")

string = input()

find\_letter = input()

n = 0

for i in range(len(string)):

if (string[i] == find\_letter):

n = i+1

break

print(n) if n else print("Буква в строке не найдена")



Рис. 20. makefile

Листинг 6. K5\_2\_1.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К5\_2. Техника работы с циклическими программами \_ цикл while

Задание 1. Вычислить значение sin(x) с точностью до epsilon при помощи разложения в ряд

Построить блок-схему

'''

import math

def mysin(x, eps):

sum, an = 0.0, x

n = 1

while(math.fabs(an)>eps):

sum += an

n += 1

an \*= -x\*x/(2.0\*n-1.0)/(2.0\*n-2.0)

return sum

x = float(input())

epsilon = float(input())

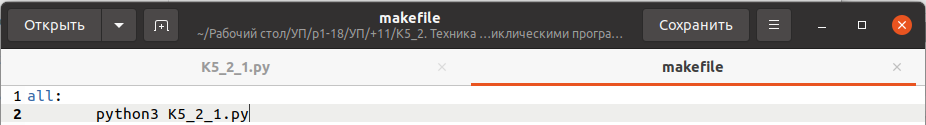
print(f"digit = {x}, mysin = {mysin(x, epsilon)}, math.sin = {math.sin(x)}")

Рис. 21. makefile

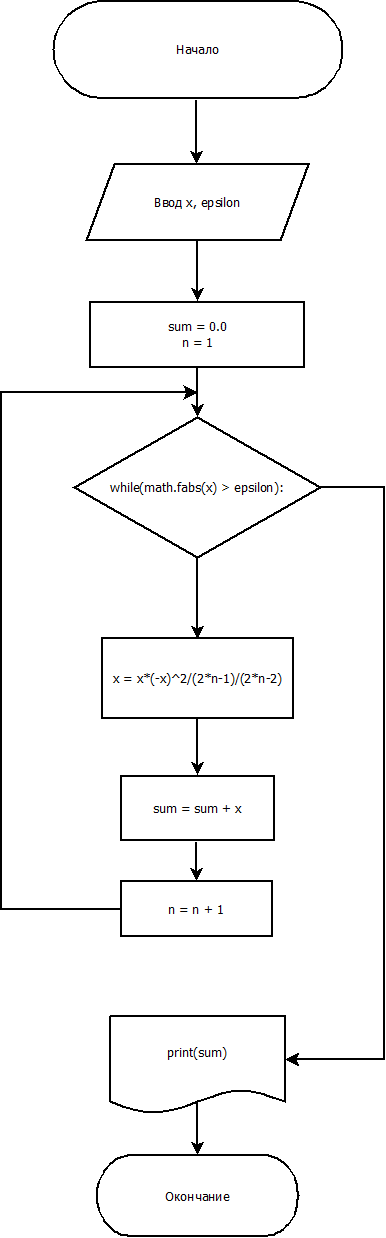


Рис. 22. makefile

Листинг 7. K5\_2\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К5\_2. Техника работы с циклическими программами \_ цикл while;

Задание 2.

https://stepik.org/lesson/3364/step/11?unit=947

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.

'''

n = -1

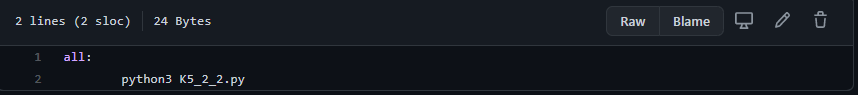
sum\_n = 0

while (n != 0):

n = int(input())

sum\_n += n

print(sum\_n)

Рис. 23. makefile

Листинг 8. K5\_2\_3.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К5\_2. Техника работы с циклическими программами \_ цикл while;

Задание 3.

Разработать программу для нахождения наибольшего общего делителя

'''

def nod(a, b):

assert a >= 0 and b >= 0

if a == 0 or b == 0:

return max(a, b)

return nod(b % a, a)

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

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

print( nod(a, b) )

Рис. 24. makefile

Листинг 9. K5\_2\_4.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К5\_2. Техника работы с циклическими программами \_ цикл while;

Задание 4.

С использованием результата задания 2 разработать программу для нахождения наименьшего

общего кратного

'''

def bruh():

n = -1

sum\_n = 0

while n != 0:

n = int(input())

sum\_n += n

return sum\_n

def gcd(a, b):

assert a >= 0 and b >= 0

if a == 0 or b == 0:

return max(a, b)

return gcd(b % a, a)

#least common multiple - lcm

def lcm(a, b):

assert a >= 0 and b >= 0

return a/gcd(a, b) \* b

# {--BASIC--}

def main():

#Ввод чисел, пока не ноль

print("Input numbers. To stop typint, enter 0")

a = bruh();

print("The sum of all entered numbers: ", a)

#Нахождение наимньшего общего кратного

print("\nInput number, to search \"least common multiple\": ")

b = int(input())

v\_lcm = lcm(a, b)

print("Least common multiple: ", v\_lcm)

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

main()

Рис. 25. makefile

Листинг 10. K5\_2\_5.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К5\_2. Техника работы с циклическими программами \_ цикл while;

Задание 5.

https://stepik.org/lesson/3369/step/8?unit=952

Напишите программу, которая выводит часть последовательности 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 ...

(число повторяется столько раз, чему равно).

На вход программе передаётся неотрицательное целое число n — столько элементов

последовательности должна отобразить программа.

На выходе ожидается последовательность чисел, записанных через пробел в одну строку.

Например, если n = 7, то программа должна вывести 1 2 2 3 3 3 4.

'''

def PartDigits(n):

count = 0

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

for j in range(i):

print(i, end = " ")

count += 1

if (count == n):

print()

return

# {--BASIC--}

def main():

n = int(input())

PartDigits(n)

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

main()

Рис. 26. makefile

## Техника работы с числами

Приложения: K6\_1.py K6\_2\_1.py K6\_2\_2.py

Листинг 11. K6\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К6\_1. Техника работы с числами

Задание 1.

Составить и выполнить по 3 примера использования модулей для работы

с дробными числами (fractions), для точных вычислений (decimal).

Задание 2.

Подготовить инструкцию по использованию модулей fractions, decimal.

'''

from decimal import Decimal, getcontext

from fractions import Fraction as frac

from fractions import Fraction

#{==== MENU ====}

def menu():

print("0. Выход")

print("1. Модуль \"decimal\"")

print("2. Модуль \"fractions\"")

#{==== MENU for Decimal ====}

def menuDecimal():

print("0. Назад")

print("1. Зачем нужен \"decimal\"")

print("2. Точность \"decimal\"")

print("3. Округление \"decimal\"")

def whatForDec():

#{ -- Example 1 -- }

print("1. Зачем нужен \"decimal\"")

print("Вобщем \"decimal\" нужен для того, чтобы вычислять более точные значения")

print("Если сложить числа 0.1 и 0.2, то получится:")

print(">>> 0.1 + 0.2 = ", 0.1 + 0.2)

print()

n1 = Decimal(0.1)

n2 = Decimal(0.2)

print("Но, воспользовавшись модулем (decimal), при сложении этих чисел, мы получим: ")

print(">>> Decimal(0.1) + Decimal(0.2) = ", n1 + n2)

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

print()

def accuracyDec():

#{ -- Example 2 -- }

print("2. Точность \"decimal\"")

print("Decimal(number) можно устанавливать точность. Т.е, кол-во знков после запятой.")

print("Для этого нужно прописать следующие:")

print()

print("Подключить модуль")

print(">>> from decimal import Decimal, getcontext")

print()

print("Установим точность 2")

print(">>> getcontext().prec = 2")

getcontext().prec = 2

print(">>> print(Decimal(\"4.341\")/1) = ", Decimal("4.341") / 1)

print(">>> print(Decimal(\"4.341\")/4) = ", Decimal("4.341") / 4)

print()

print("Установим точность 3")

print(">>> getcontext().prec = 3")

getcontext().prec = 3

print(">>> print(Decimal(\"4.341\")/1) = ", Decimal("4.341") / 1)

print(">>> print(Decimal(\"4.341\")/4) = ", Decimal("4.341") / 4)

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

print()

def roundingDec():

#{ -- Example 3 -- }

print("3. Округление \"decimal\"")

print("Decimal(number), можно округлять. Для этого нужно прописать следующее:")

print()

print("Установим точность округления")

print(">>> getcontext().prec = 4")

getcontext().prec = 4

print()

print("number = Decimal(\"2.12345678\")")

number = Decimal("2.123456789")

print()

print("Округляем число number")

print(">>> print(number.quantize(Decimal(\"1.000\")))", number.quantize(Decimal('1.000')))

print(">>> print(number.quantize(Decimal(\"1.00\")))", number.quantize(Decimal('1.00')))

print(">>> print(number.quantize(Decimal(\"1.0\")))", number.quantize(Decimal('1.0')))

print(">>> print(number.quantize(Decimal(\"1\")))", number.quantize(Decimal('1')))

print(">>> print(number.quantize(Decimal(\"10\")))", number.quantize(Decimal('10')))

print()

print("Но если мы введем:")

print(">>> print(number.quantize(Decimal(\"1.0000\")))")

print("То будет следующая ошибка: ", end="")

print("decimal.InvalidOperation: [<class 'decimal.InvalidOperation'>]")

print()

print("Чтобы избежать ее, необходимо поменять точность округления")

print(">>> getcontext().prec = 5")

getcontext().prec = 5

print()

print("Теперь ошибки не будет")

print(">>> print(number.quantize(Decimal(\"1.000\")))", number.quantize(Decimal('1.000')))

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

#{==== MENU for Fractions ====}

def whatForFrac():

#{ -- Example 1 -- }

print("1. Зачем нужен \"fractions\"")

print("Этот модуль пригодится в тех случаях, когда вам необходимо выполнить вычисления")

print("с дробями, или когда результат должен быть выражен в формате дроби.")

print(">>> from fractions import Fraction as frac")

print(">>> from fractions import Fraction")

print()

print(">>> print(Fraction(\'33.33\')")

print(Fraction('33.33'))

print()

print(">>> print(Fraction(33.33)")

print(Fraction(33.33))

print()

print("Модуль Fraction особенно полезен, потому что он автоматически уменьшает дробь.")

print("Выглядит это вот так:")

print(">>> Fraction(153, 272)")

print(Fraction(153, 272))

print()

print("Кроме того, вы можете выполнять бинарные (двоичные) операции над дробью также")

print("просто, как вы используете int или float . Просто добавьте две фракции:")

print(">>> Fraction(1, 2) + Fraction(3, 4) = Fraction(5, 4)")

print(Fraction(1, 2) + Fraction(3, 4))

print()

print("Теперь давайте попробуем возвести дробь в степень:")

print(">>> Fraction(1, 8) \*\* Fraction(1, 2)")

print(Fraction(1, 8) \*\* Fraction(1, 2))

print()

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

print()

#{==== MAIN ====}

def main():

n = -1

n\_loc = -1

while (n):

print()

menu()

n = int(input())

if (n == 1):

print()

while (n\_loc):

menuDecimal()

n\_loc = int(input())

print()

if (n\_loc == 1):

whatForDec()

elif (n\_loc == 2):

accuracyDec()

elif (n\_loc == 3):

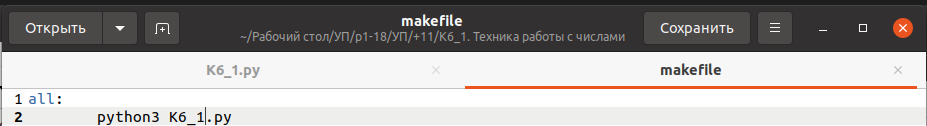
roundingDec()

n\_loc = -1

elif (n == 2):

whatForFrac()

n\_loc = -1

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

main()

Рис. 27. makefile

Листинг 12. K6\_2\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К6\_2. Техника работы с числами

cmath

'''

import cmath

def main():

print("Сложные функции")

print("cmath.polar(complex(1.0, 1.0)) =", cmath.polar(complex(1.0, 1.0)) ) # returns (1.4142135623730951, 0.7853981633974483)

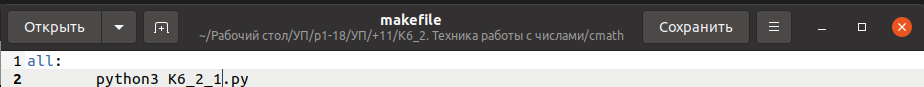
print("cmath.phase(complex(1.0, 1.0)) =", cmath.phase(complex(1.0, 1.0)) ) # returns 0.7853981633974483

print("abs(complex(1.0, 1.0)) =", abs(complex(1.0, 1.0)) ) # returns 1.4142135623730951

print("cmath.sqrt(complex(25.0, 25.0)) =", cmath.sqrt(complex(25.0, 25.0)) ) # returns (5.49342056733905+2.2754493028111367j)

print("cmath.cos(complex(25.0, 25.0)) =", cmath.cos(complex(25.0, 25.0)) ) # returns (35685729345.58163+4764987221.458499j)

print()



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

main()

Рис. 28. makefile

Листинг 13. K6\_2\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К6\_2. Техника работы с числами

math

'''

import math

def getsin(x):

multiplier = 1

result = 0

for i in range(1,20,2):

result += multiplier\*pow(x,i)/math.factorial(i)

multiplier \*= -1

return result

def main():

print("Арифметические функции")

print("sin(pi/2) =", getsin(math.pi/2)) # returns 1.0

print("math.pow(3, 2) =", math.pow(3,2))

print("math.pow(9, 0.5) =", math.pow(9,0.5))

print("math.sqrt(9) =", math.sqrt(9))

print("math.factorial(5) =", math.factorial(5))

print("Округление:")

print("math.ceil(1.001) =", math.ceil(1.001) ) # returns 2

print("math.floor(1.001) =", math.floor(1.001) ) # returns 1

print("math.factorial(10) =", math.factorial(10) ) # returns 3628800

print("math.gcd(10,125) =", math.gcd(10,125) ) # returns 5

print("math.trunc(1.001) =", math.trunc(1.001) ) # returns 1

print("math.trunc(1.999) =", math.trunc(1.999) ) # returns 1

print()

print("Тригонометрические функции")

print("math.sin(math.pi/4) =",math.sin(math.pi/4) ) # returns 0.7071067811865476

print("math.cos(math.pi) =",math.cos(math.pi) ) # returns -1.0

print("math.tan(math.pi/6) =",math.tan(math.pi/6) ) # returns 0.5773502691896257

print("math.hypot(12,5) =",math.hypot(12,5) ) # returns 13.0

print("math.atan(0.5773502691896257) =",math.atan(0.5773502691896257) ) # returns 0.5235987755982988

print("math.asin(0.7071067811865476) =",math.asin(0.7071067811865476) ) # returns 0.7853981633974484

print()

print("Гиперболические функции")

print("math.sinh(math.pi) =", math.sinh(math.pi) ) # returns 11.548739357257746

print("math.cosh(math.pi) =", math.cosh(math.pi) ) # returns 11.591953275521519

print("math.cosh(math.pi) =", math.cosh(math.pi) ) # returns 0.99627207622075

print("math.asinh(11.548739357257746) =", math.asinh(11.548739357257746) ) # returns 3.141592653589793

print("math.acosh(11.591953275521519) =", math.acosh(11.591953275521519) ) # returns 3.141592653589793

print("math.atanh(0.99627207622075) =", math.atanh(0.99627207622075) ) # returns 3.141592653589798

print()

print("Логарифмические функции")

print("math.exp(5) =", math.exp(5)) # returns 148.4131591025766

print("math.e\*\*5 =", math.e\*\*5 ) # returns 148.4131591025765

print("math.log(148.41315910257657) =", math.log(148.41315910257657)) # returns 5.0

print("math.log(148.41315910257657, 2) =", math.log(148.41315910257657, 2)) # returns 7.213475204444817

print("math.log(148.41315910257657, 10) =", math.log(148.41315910257657, 10)) # returns 2.171472409516258

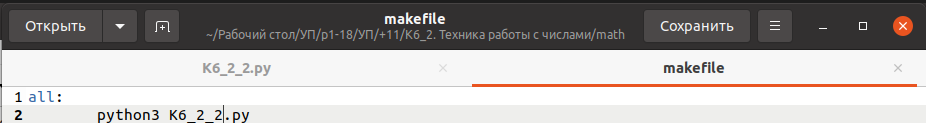
print("math.log(1.0000025) =", math.log(1.0000025)) # returns 2.4999968749105643e-06

print("math.log1p(0.0000025) =", math.log1p(0.0000025) ) # returns 2.4999968750052084e-06

print()

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

main()

Рис. 29. makefile

## Техника работы со строками

Приложение: K7\_1\_1.py K7\_1\_2.py K7\_1\_3.py K7\_1\_4.py

Листинг 14. K7\_1\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К7\_1. Техника работы со строками;

Задание 1.

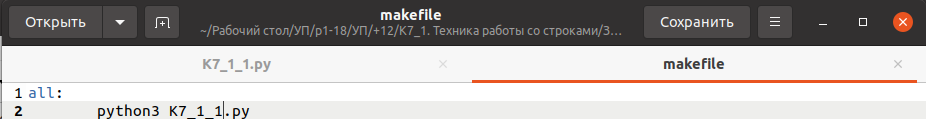
https://stepik.org/lesson/201702/step/5?unit=175778

С клавиатуры вводятся строки, последовательность заканчивается точкой.

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

'''

a = list()

[a.append(i.upper()) for i in iter(input, '.')]

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

Рис. 30. makefile

Листинг 15. K7\_1\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К7\_1. Техника работы со строками;

Задание 2.

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

?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

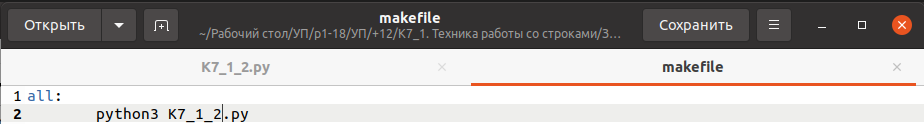
могла сгодиться для логина. Если строка состоит из одного слова, менять ничего не нужно.

Sample Input: python sila

Sample Output: python\_sila

'''

print(\*[\_ for \_ in input().split()], sep='\_')

Рис. 31. makefile

Листинг 16. K7\_1\_3.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К7\_1. Техника работы со строками;

Задание 3. https://stepik.org/lesson/201702/step/9?unit=175778

?Уберите точки из введенного IP-адреса. Выведите сначала четыре числа через пробел,

а затем сумму получившихся чисел.

Sample Input: 192.168.0.1

Sample Output:

192 168 0 1

361

'''

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

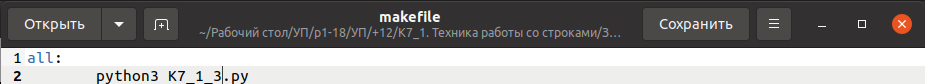
print(\*a, '\n'+str(sum(a)), sep=' ')

Рис. 32. makefile

Листинг 17. K7\_1\_4.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К7\_1. Техника работы со строками;

Задание 4. https://stepik.org/lesson/201702/step/14?unit=175778

Программист логирует программу, чтобы хорошо знать,

как она себя ведет (эта весьма распространенная и важная практика).

Он использует разные типы сообщений для вывода ошибок (error),

предупреждений (warning), информации (info) или подробного описания (verbose).

Сообщения отличаются по внешнему виду. Назовем модификаторами такие символы,

которые отличают сообщения друг от друга, позволяя программисту понять, к какому

из типов относится сообщения. Модификаторы состоят из двух одинаковых символов

и записываются по разу в начале и в конце строки.

@@ обозначает ошибку

!! обозначает предупреждение

// обозначает информационное сообщение

\*\* обозначает подробное сообщение

Напишите программу, которая принимает строки до точки и выводит,

какого типа это сообщение. Если сообщение не содержит модификаторов,

проигнорируйте его.

Sample Input:

!! cannot resolve this method !!

@@ invalid type @@

@@ StackOverFlowException @@

// here I change the variables name //

\*\* this class is used for operating with the database, including CRUD operations and registering new users \*\*

error on line 42

// TODO: optimize recursive calls //

.

Sample Output:

предупреждение

ошибка

ошибка

информация

подробное сообщение

информация

'''

m = list()

n = input()

while n != '.':

m.append(n)

n = input()

for i in m:

if i[:2] == '@@':

print('ошибка')

if i[:2] == '!!':

print('предупреждение')

if i[:2] == '//':

print('информация')

if i[:2] == '\*\*':

print('подробное сообщение')

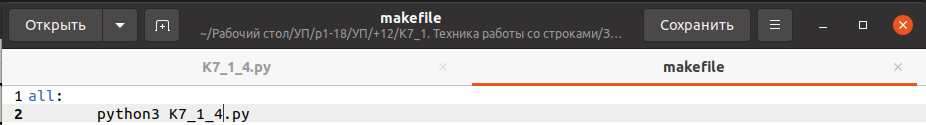


Рис. 33. makefile

Листинг 18. K7\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К7\_1. Техника работы со строками;

Задание 1.

Задание 1. Подготовить сравнительную инструкцию по использованию

форматирования строк

'''

print("1 Форматирование строк “По старинке” (оператор %)")

print("name = \"PYH\"")

name = "PYH"

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

print("Output: Hello, %s" %name)

print()

print("Вывод в шестнадцатиричного числа")

print("errno = 50159747054")

errno = 50159747054

print("print('%x' % errno)")

print("Output: %x" % errno)

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

print()

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

print("name = \"PYH\"")

name = "PYH"

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

print("Output:", 'Hello, {}'.format(name))

print()

print("Или")

print("errno = 50159747054")

errno = 50159747054

print("print(")

print("\t'Hey {name}, there is a 0x{errno:x} error!'.format("")")

print("\t\tname=name, errno=errno")

print("\t)")

print(")")

print(

'Output: Hey {name}, there is a 0x{errno:x} error!'.format(

name=name, errno=errno

)

)

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

print()

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

print("name = \"PYH\"")

name = "PYH"

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

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

print()

print("a = 5")

a = 5

print("b = 10")

b = 10

print("print(f'Five plus ten is {a + b} and not {2 \* (a + b)}.')")

print(f'Five plus ten is {a + b} and not {2 \* (a + b)}.')

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

print()

print("4 Шаблонные строки (Стандартная библиотека Template Strings)")

print("from string import Template")

from string import Template

print("name = \"PYH\"")

name = "PYH"

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

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

print("print(t.substitute(name=name))")

print(t.substitute(name=name))

print()

print("templ\_string = 'Hey $name, there is a $error error!'")

templ\_string = 'Hey $name, there is a $error error!'

print("print(")

print(" Template(templ\_string).substitute(")

print(" name=name, error=hex(errno)")

print(" )")

print(")")

print(

Template(templ\_string).substitute(

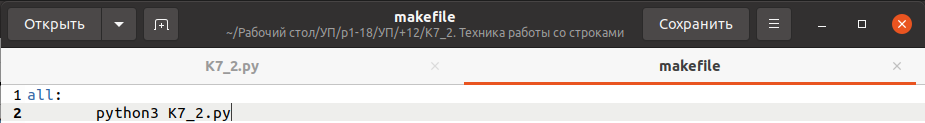
name=name, error=hex(errno)

)

)

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

print()

Рис. 34. makefile

## Техника работы со списками

Приложения: K8\_1\_1.py, K8\_1\_2.py, K8\_1\_3.py, К8\_2\_1, К8\_2\_2, К8\_2\_3

Листинг 19. K8\_1\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

Консультация 8\_1. Техника работы со списками

Задание 1. https://pythontutor.ru/lessons/lists/problems/more\_than\_neighbours/

Задача «Больше своих соседей»

Дан список чисел. Определите, сколько в этом списке элементов, которые больше двух

своих соседей, и выведите количество таких элементов. Крайние элементы списка никогда

не учитываются, поскольку у них недостаточно соседей.

'''

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

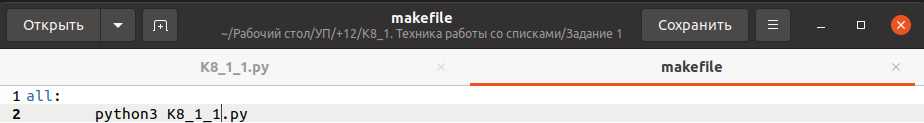
a = 0

for i in range(2,len(n)):

if n[i-2] < n[i-1] > n[i]:

a += 1

print(a)

Рис. 35. makefile

Листинг 20. K8\_1\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

Консультация 8\_1. Техника работы со списками;

Задание 2. https://pythontutor.ru/lessons/lists/problems/num\_equal\_pairs/

Задача «Количество совпадающих пар»

Дан список чисел. Посчитайте, сколько в нем пар элементов, равных друг другу.

Считается, что любые два элемента, равные друг другу образуют одну пару, которую

необходимо посчитать.

'''

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

a = 0

for i in range(len(n)):

for j in range(len(n)):

if n[i] == n[j]:

a += 1

a -=1

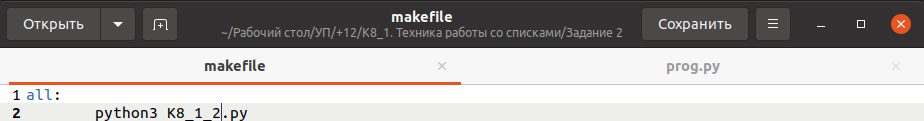
print(a/2)

Рис. 36. makefile

Листинг 21. K8\_1\_3.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

Консультация 8\_1. Техника работы со списками;

Задание 3. (Л.Б.)

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

случайным образом). Требуется сформировать

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

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

Решение без использования set - дополнительный бонус

'''

import random

n = int(input())

mtrx = [[random.randint(0, n) for i in range(n)] for j in range(n)] #Формируем матрицу

u\_every =[] #список значений, попадающих в каждый из N списков

u\_all =[] #список значений, попадающие хотя бы в один из N списков

#Формируем список значений, попадающие хотя бы в один из N списков

for i in range(n):

for j in range(n):

if mtrx[i][j] not in u\_all:

u\_all.append(mtrx[i][j])

#Формируем список значений, попадающих в каждый из N списков

for num in u\_all:

count = 0

for i in range(n):

if num in mtrx[i]:

count += 1

print(count)

if (count == n):

u\_every.append(num)

#Выводим данные

print("Матрица:", \*mtrx, sep='\n ')

print("Список значений, попадающих в каждый из N списков:\n", \*u\_every)

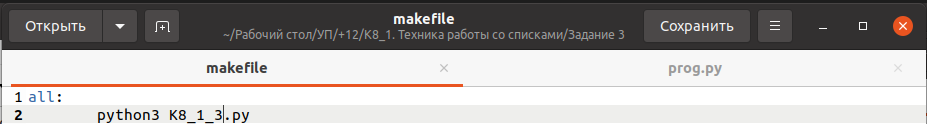
print("Список значений, попадающие хотя бы в один из N списков:\n", \*u\_all)

Рис. 37. makefile

Листинг 22. K8\_2\_1.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К8\_2. Техника работы со списками

Задание 1. Array112. Дан массив A размера N.

Упорядочить его по возрастанию методом сортировки

простым обменом («пузырьковой» сортировкой):

просматривать массив, сравнивая его соседние элементы

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

если левый элемент пары больше правого; повторить описанные

действия N 1 раз. Для контроля за выполняемыми действиями

выводить содержимое массива после каждого просмотра.

Учесть, что при каждом просмотре количество анализируемых

пар можно уменьшить на 1.

'''

from random import randint

n = int(input())

a = []

for i in range(n):

a.append(randint(1,100))

print(a)

for i in range(n-1):

for j in range(n-i-1):

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

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

print(a)

print(a)

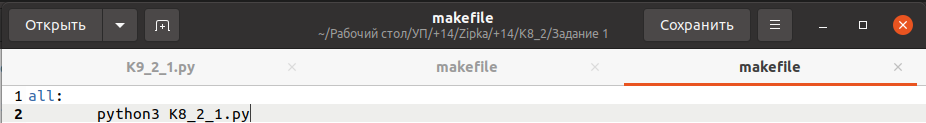


Рис. 38. makefile

Листинг 23. K8\_2\_2.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К8\_2. Техника работы со списками

Задание 2. Array113. Дан массив A размера N.

Упорядочить его по возрастанию методом сортировки простым

выбором: найти максимальный элемент массива и поменять его

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

действия N 1 раз, каждый раз уменьшая на 1 количество

анализируемых элементов и выводя содержимое массива.

'''

from random import randint

def sel\_sort(arr):

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

m = i

j = i + 1

while j < len(arr):

if arr[j] < arr[m]:

m = j

j = j + 1

arr[i], arr[m] = arr[m], arr[i]

print(arr)

n = int(input())

a = []

for i in range(n):

a.append(randint(1, 99))

print(a)

sel\_sort(a)

print(a)

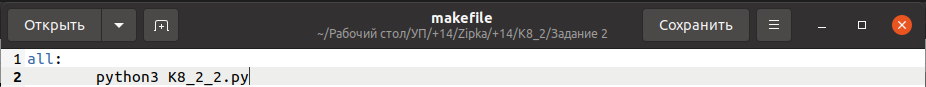


Рис. 39. makefile

Листинг 24. K8\_2\_3.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К8\_2. Техника работы со списками

Задание 3. Array114. Дан массив A размера N. Упорядочить

его по возрастанию методом сортировки простыми вставками:

сравнить элементы A0 и A1 и, при необходимости меняя их

местами, добиться того, чтобы они оказались упорядоченными

по возрастанию; затем обратиться к элементу A2 и

переместить его в левую (уже упорядоченную) часть массива,

сохранив ее упорядоченность; повторить этот процесс для

остальных элементов, выводя содержимое массива после

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

'''

from random import randint

def insertion(mas):

for i in range(len(mas)):

j = i - 1

key = mas[i]

while mas[j] > key and j >= 0:

mas[j + 1] = mas[j]

j -= 1

mas[j + 1] = key

print(mas)

return mas

n = int(input())

mas = []

for i in range(n):

mas.append(randint(1,100))

print(mas)

insertion(mas)

print(mas)

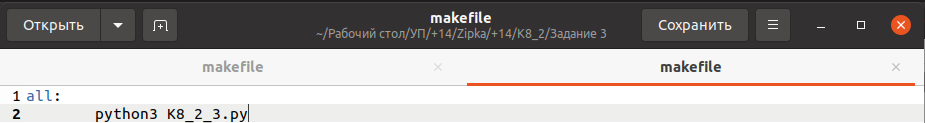


Рис. 40. makefile

## Техника работы с циклом for и генераторами списков

Приложения: K9\_1.py, K9\_1\_2.py, K9\_1\_3.py,

Листинг 25. K9\_1\_1.py file.txt makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К9\_1. Техника работы с циклом for и генераторами списков

Задание 1. (Л.Б.) Для проведения конкурса проектов в ККМТ формируются группы

из 4х участников: coder, writer, tester, designer, программирующих

на одном и том же языке.

Каждый студент может программировать только на одном языке

и занимать только одну позицию.

Дан текстовый файл, содержащий перечень студентов с указанием языка и позиции

(каждый студент с новой строки)

Требуется

1. Получить список студентов с указанием языка и позиции

2. Сформировать список всевозможных команд

3. Вывести список команд с указанием состава и названия команды:

Команда 1

coder: ...

designer: ...

tester: ...

writer: ...

Команда 2

...

Пункты 1 и 2 выполнить с использованием генераторов списка

'''

file = open("file.txt", "r")

def prof(mtrx,mas): # Проверка на "подходит ли чел в команду"

if (len(mtrx) == 0):

return True

count = 0

for i in range(len(mtrx)):

if(mtrx[i][1] != mas[1]):

count += 1

if(mtrx[i][2] == mas[2]):

count += 1

if(count == len(mtrx)\*2):

return True

return False

#---------1--------------------

list = file.readlines()

out = [i.strip().split() for i in list]

for i in range(len(out)):

print(f'{out[i][1]}: {out[i][0]} | {out[i][2]}')

print()

print(out)

print('Всего участников - ', len(out), '\n')

#---------1--------------------

#[

# [

# ['Dima', 'coder', 'C++'],

# ['Roma', 'designer', 'C++'],

# ['Ivan', 'tester', 'C++'],

# ['Stiv', 'writer', 'C++']

# ]

#]

#---------2--------------------

teams = [[]] #[[['Dima', 'coder', 'C++'], ['Roma', 'designer', 'C++'],['Ivan', 'tester', 'C++'],['Stiv', 'writer', 'C++']]]

k = 0

count = 0

i = 0

while(i != len(out)):

if(prof(teams[k],out[i])):

if(len(teams[k]) < 4):

teams[k].append(out[i])

out.pop(i)

i -= 1

if(len(teams[k]) == 4):

teams.append([])

k += 1

i = 0

continue

i += 1

if(i == len(out)):

teams.append([])

k += 1

i = 0

teams.pop(-1)

#---------2--------------------

'''3. Вывести список команд с указанием состава и названия команды:

Команда 1

coder: ...

designer: ...

tester: ...

writer: ...

Команда 2

...'''

#---------3--------------------

l = 0

for i in range(len(teams)):

if(len(teams[i]) < 4):

print("Неполная команда")

else:

l += 1

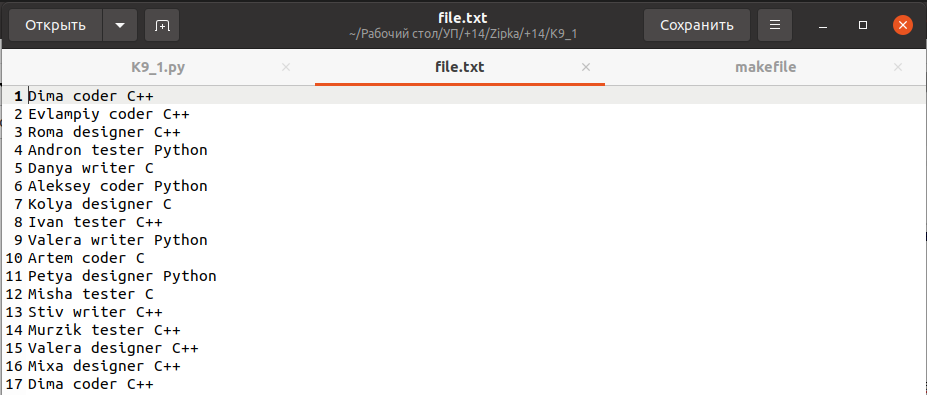
print("Команда", l)

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

print(f'{teams[i][j][1]}: {teams[i][j][0]} | {teams[i][j][2]}')

print()

#---------3--------------------



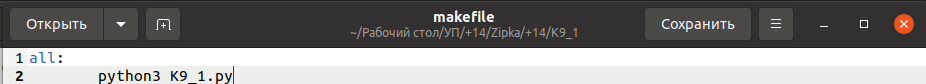
Рис. 41. file.txt

Рис. 42. makefile

Листинг 26. K9\_2\_1.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К9\_2. Техника работы с циклом for и генераторами списков

Задание 1. Array55. Дан целочисленный массив A размера N (<= 15). Переписать в новый целочисленный

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

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

'''

from random import randint

n = int(input())

a = []

b = []

for i in range(n):

a.append(randint(1, 100))

print(a)

print(len(a))

for i in range(1, len(a), 2):

b.append(a[i])

print(b)

print(len(b))

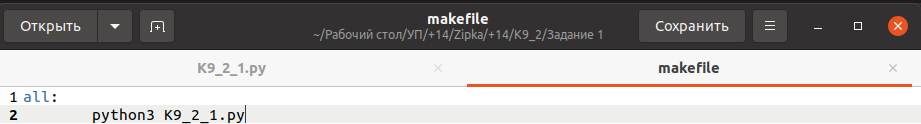


Рис. 43. makefile

Листинг 27. K9\_2\_2.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К9\_2. Техника работы с циклом for и генераторами списков

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

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

а затем — с нечетными:

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

Условный оператор не использовать.

'''

from random import randint

n = int(input())

a = []

b = []

for i in range(n):

a.append(randint(1, 100))

print(a)

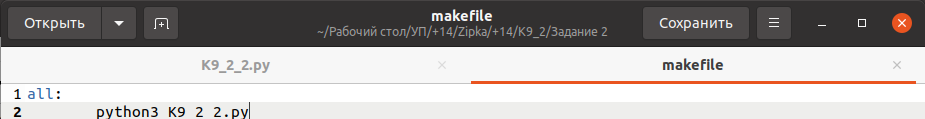
print(len(a))

for i in range(0, len(a), 2):

b.append(a[i])

for i in range(1, len(a), 2):

b.append(a[i])

print(b)

print(len(b))

Рис. 44. makefile

Листинг 28. K9\_2\_3.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К9\_2. Техника работы с циклом for и генераторами списков

Задание 3. Array58. Дан массив A размера N. Сформировать новый массив B того же размера по

следующему правилу: элемент B[K] равен сумме элементов массива A с номерами от 0 до K.

'''

from random import randint

n = int(input())

a = []

b = []

for i in range(n):

a.append(randint(1, 10))

print(a)

summ = 0

for i in range(len(a)):

summ += a[i]

b.append(summ)

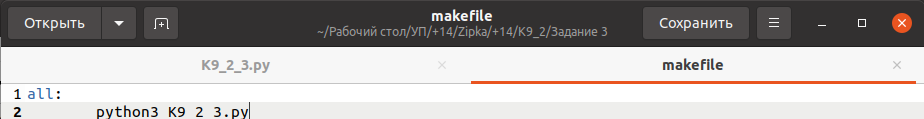
print(b)

Рис. 45. makefile

Листинг 29. K9\_2\_4.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К9\_2. Техника работы с циклом for и генераторами списков

Задание 4. Matrix3. Даны целые положительные числа M, N и набор из M чисел. Сформировать

матрицу размера M x N, у которой в каждом столбце содержатся все числа из исходного

набора (в том же порядке).

'''

from random import randint

m = int(input())

n = int(input())

a = [randint(1, 10) for i in range(m)]

print(a)

mtrx1 = []

for i in range(m):

list = []

for j in range(n):

list.append(a[i])

mtrx1.append(list)

[print(' '.join([str(j) for j in i])) for i in mtrx1]

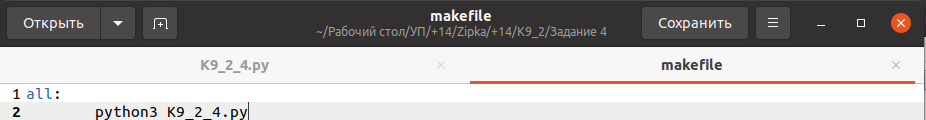


Рис. 46. makefile

Листинг 30. K9\_2\_5.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К9\_2. Техника работы с циклом for и генераторами списков

Задание 5. Matrix56. Дана матрица размера M x N (N — четное число). Поменять местами

левую и правую половины матрицы.

'''

from random import randint

def swap(a, b):

a, b = b, a

m = int(input()) # Количество строчек

n = int(input()) # Количество столбцов

mtrx =[[randint(1, 10) for j in range(n)] for i in range(m)] # Заполнение матрицы

print(\*['\n'.join((' '.join([str(j) for j in i])) for i in mtrx)])

n1 = n//2

[[swap(mtrx[j], mtrx[j - n1]) for j in range(n1)] for i in range(m)]

print()

print(\*['\n'.join((' '.join([str(j) for j in i])) for i in mtrx)])

'''

2 1 10 2 4 3 2 4 3 2 1 10

4 6 4 8 1 10 8 1 10 4 6 4

1 9 3 10 2 6 <=> 10 2 6 1 9 3

10 5 2 4 5 9 4 5 9 10 5 2

9 2 5 10 1 7 7 1 10 9 2 5

'''

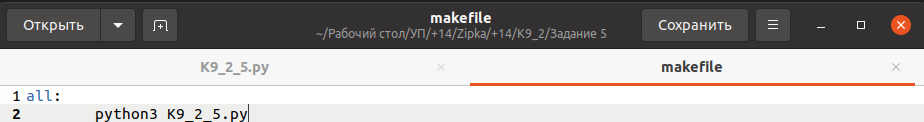


Рис. 47. makefile

Листинг 31. K9\_2\_6.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К9\_2. Техника работы с циклом for и генераторами списков

Задание 6. Matrix88. Дана квадратная матрица порядка M. Обнулить элементы матрицы,

лежащие ниже главной диагонали. Условный оператор не использовать.

'''

from random import randint

m = int(input()) # Количество строчек

n = int(input()) # Количество столбцов

mtrx =[[randint(1, 10) for j in range(n)] for i in range(m)] # Заполнение матрицы

print(\*['\n'.join((' '.join([str(j) for j in i])) for i in mtrx)])

for i in range(1,m):

for j in range(i):

mtrx[i][j] = 0

print()

print(\*['\n'.join((' '.join([str(j) for j in i])) for i in mtrx)])

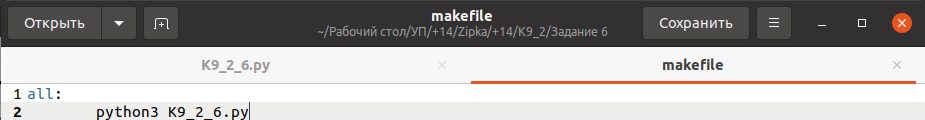


Рис. 48. makefile

## Техника работы с функциями

Приложения: K10\_1\_2.py, K10\_1\_3.py, K10\_1\_4.py, K10\_2\_2.py, K10\_2\_3.py, K10\_2\_4.py

Листинг 32. K10\_1\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

Консультация 8\_1. Техника работы со списками

Задание 1. https://pythontutor.ru/lessons/lists/problems/more\_than\_neighbours/

Задача «Больше своих соседей»

Дан список чисел. Определите, сколько в этом списке элементов, которые больше двух

своих соседей, и выведите количество таких элементов. Крайние элементы списка никогда

не учитываются, поскольку у них недостаточно соседей.

'''

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

a = 0

for i in range(2,len(n)):

if n[i-2] < n[i-1] > n[i]:

a += 1

print(a)

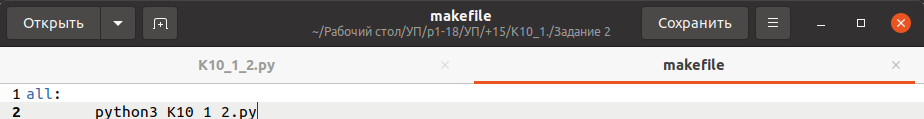


Рис. 49. makefile

Листинг 33. K10\_1\_3.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К10\_1. Техника работы с функциями

Задание 3. Func10. Описать функцию IsSquare(K) логического типа, возвращающую True,

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

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

положительных чисел.

'''

import math

def IsSquare(k):

if (k < 1):

return False

else:

sqrt\_k = math.sqrt(k)

if (sqrt\_k == int(sqrt\_k)):

return True

else:

return False

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

len\_a = len(a)

count = 0

for i in a:

count += IsSquare(i)

print(count)

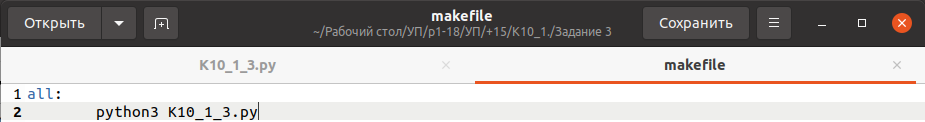


Рис. 50. makefile

Листинг 34. K10\_1\_4.py makefile

*#Выполнили: Герасимов Дмитрий, Груздев Роман  
#Группа: П1-18  
'''  
К10\_1. Техника работы с функциями  
  
Задание 4. Func33. Описать функцию SortInc3(X), меняющую содержимое списка X из трех   
вещественных элементов таким образом, чтобы их значения оказались упорядоченными по   
возрастанию (функция возвращает None). С помощью этой функции упорядочить по   
возрастанию два данных списка X и Y.  
'''*def SortInc3(a):  
 n = len(a)  
 for i in range(n):  
 for j in range(i+1,n):  
 if (a[i] > a[j]):  
 a[i], a[j] = a[j], a[i]  
   
print("Введите массив X:")  
x = [float(i) for i in input().split()]  
print("Введите массив Y:")  
y = [float(i) for i in input().split()]  
  
SortInc3(x)  
print(x)  
SortInc3(y)  
print(y)



Рис. 51. makefile

Листинг 35. K10\_2\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К10\_1. Техника работы с функциями

Задание 2. https://stepik.org/lesson/201702/step/13?unit=175778

Использовать map, lambda

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

обратном порядке, возводя их в квадрат.

Sample Input:

5

16

20

1

9

.

'''

a = [int(x)\*\*2 for x in iter(input, '.')]

#print(\*a[::-1], sep=' ')

# ИЛИ

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

print()

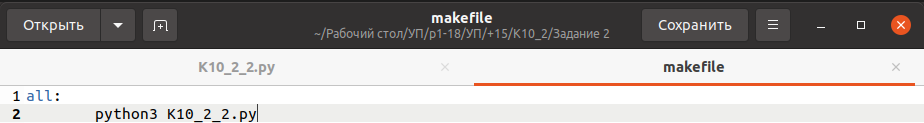


Рис. 52. makefile

Листинг 36. K10\_1\_4.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К10\_1. Техника работы с функциями

Задание 4. Func33. Описать функцию SortInc3(X), меняющую содержимое списка X из трех

вещественных элементов таким образом, чтобы их значения оказались упорядоченными по

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

возрастанию два данных списка X и Y.

'''

def SortInc3(a):

n = 3

for i in range(n):

for j in range(n-1):

if (a[i] < a[j]):

a[i], a[j] = a[j], a[i]

return None

print("Введите массив X:")

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

print("Введите массив Y:")

y = [float(i) for i in input().split()]

SortInc3(x)

print(x)

SortInc3(y)

print(y)

Рис. 53. makefile

Листинг 37. K10\_2\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К10\_2. Техника работы с функциями

Задание 2. https://stepik.org/lesson/201702/step/13?unit=175778

Использовать map, lambda

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

обратном порядке, возводя их в квадрат.

Sample Input:

5

16

20

1

9

.

'''

a = [int(x)\*\*2 for x in iter(input, '.')]

#print(\*a[::-1], sep=' ')

# ИЛИ

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

print()

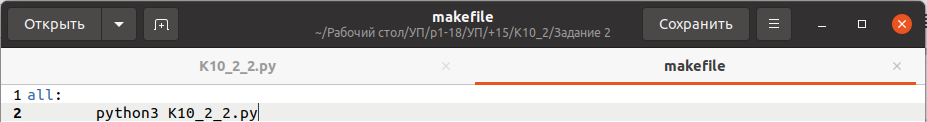


Рис. 54. makefile

Листинг 38. K10\_2\_3.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К10\_2. Техника работы с функциями

Задание 3. Использовать lambda, filter.

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

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

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

'''

n = int(input("Введите размер списка: "))

a = [int(input(f"{x}| ")) for x in range(n)]

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

print("\nСписок b:", \*b, sep=" ")

print("Размер списка b:", len(b))

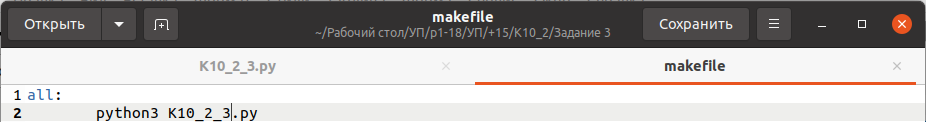


Рис. 55. makefile

Листинг 39. K10\_2\_4.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К10\_2. Техника работы с функциями

Задание 4. Использовать lambda, map.

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

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

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

числа получившегося списка через пробел.

Sample Input:

5 12 3

Sample Output:

125 216 343 512 729 1000 1331 1728

'''

a = int(input("Начало: "))

b = int(input("Конец: "))

power = int(input("Степень: "))

#arr = [x\*\*power for x in range(a, b+1)]

# ИЛИ

arr = list(map(lambda x: x\*\*power, range(a, b+1)))

print(\*arr)

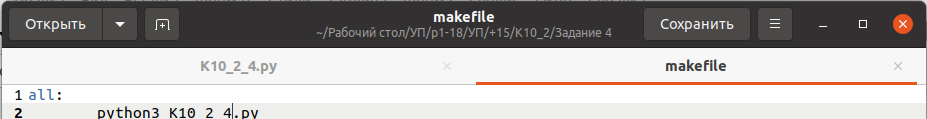


Рис. 56. makefile

## Техника работы со словарями

Приложения: K11\_1\_1.py, K11\_1\_2.py, K11\_1\_3.py, K11\_2\_1.py, K11\_2\_2.py, K11\_2\_3.py

Листинг 40. K11\_1\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К11\_1. Техника работы со словарями

Задание 1. https://pythontutor.ru/lessons/dicts/problems/occurency\_index/

Задача «Номер появления слова»

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

подсчитайте, сколько раз оно встречалось в этом тексте ранее.

Словом считается последовательность непробельных символов идущих подряд, слова разделены

одним или большим числом пробелов или символами конца строки.

'''

d = dict()

i = 0

for key in input().split():

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

print(d[key] - 1, end = ' ')

print()

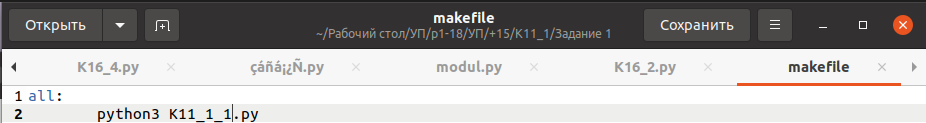


Рис. 57. makefile

Листинг 41. K11\_1\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К11\_1. Техника работы со словарями

Задание 2. https://pythontutor.ru/lessons/dicts/problems/permissions/

Задача «Права доступа»

Условие. В файловую систему одного суперкомпьютера проник вирус, который сломал контроль

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

нему обращаться:

запись W,

чтение R,

запуск X.

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

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

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

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

любое колличество запросов.

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

каждого запроса должна будет возвращать OK если над файлом выполняется допустимая

операция, или же Access denied, если операция недопустима.

'''

d = dict()

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

list = input().split()

d[list[0]] = list[1:]

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

command, name = input().split()

if command == 'read':

command = 'R'

elif command == 'write':

command = 'W'

elif command == 'execute':

command = 'X'

if command in d[name]:

print('OK')

else:

print('Access denied')

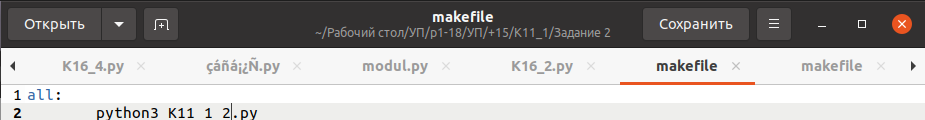


Рис. 58. makefile

Листинг 42. K11\_1\_3.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К11\_1. Техника работы со словарями

Задание 3. https://pythontutor.ru/lessons/dicts/problems/most\_frequent\_word/

Задача «Самое частое слово»

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

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

выведите то, которое меньше в лексикографическом порядке.

'''

text = list()

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

words = input().split()

for word in words:

text.append(word)

d = dict()

for key in text:

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

maxx = max(d.values())

for key, value in sorted(d.items()):

if (value == maxx):

print(key)

break

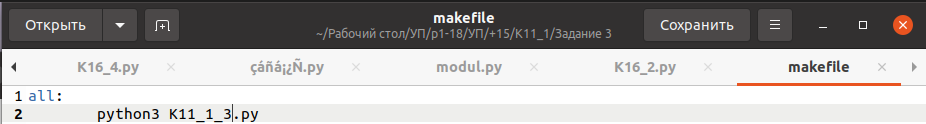


Рис. 59. makefile

Листинг 43. K11\_2\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К11\_2. Техника работы со словарями

Задание 1. https://stepik.org/lesson/243394/step/4?unit=215740

Телефонная книга. Этап 1. Коля устал запоминать телефонные номера и заказал у Вас

программу, которая заменила бы ему телефонную книгу. Коля может послать программе

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

или просто имя контакта. В первом случае программа должна добавить в книгу новый номер,

во втором – вывести номер контакта. Ввод происходит до символа точки. Если введенное

имя уже содержится в списке контактов, необходимо перезаписать номер.

Sample Input:

Ben 89001234050

Alice 210-220

Alice

Alice 404-502

Nick +16507811251

Ben

Alex +4(908)273-22-42

Alice

Nick

Robert 51234047129

Alex

.

Sample Output:

210-220

89001234050

404-502

+16507811251

+4(908)273-22-42

'''

d = dict()

for str in iter(input, '.'):

arr = str.split()

if len(arr) == 1:

print(d[arr[0]])

else:

d[arr[0]] = arr[1]

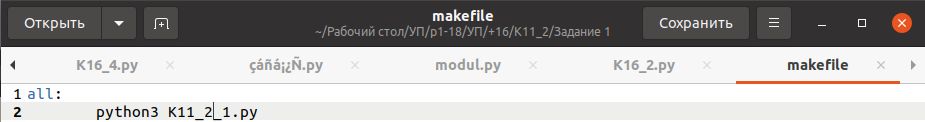


Рис. 60. makefile

Листинг 44. K11\_2\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К11\_2. Техника работы со словарями

Задание 2. https://stepik.org/lesson/243394/step/8?unit=215740

Телефонная книга. Этап 2. Коля понял, что у многих из его знакомых есть несколько

телефонных номеров и нельзя хранить только один из них. Он попросил доработать Вашу

программу так, чтобы можно было добавлять к существующему контакту новый номер или даже

несколько номеров, которые передаются через запятую. По запросу телефонного номера

должен выводиться весь список номеров в порядке добавления, номера должны разделяться

запятой. Если у контакта нет телефонных номеров, должна выводиться строка "Не найдено".

Sample Input:

Ben 89001234050, +70504321009

Alice 210-220

Alice

Alice 404-502, 894-005, 439-095

Nick +16507811251

Ben

Alex +4(908)273-22-42

Alice

Nick

Robert 51234047129, 92174043215

Alex

Robert

.

Sample Output:

210-220

89001234050, +70504321009

210-220, 404-502, 894-005, 439-095

+16507811251

+4(908)273-22-42

51234047129, 92174043215

'''

d = dict()

for str in iter(input, '.'):

arr = str.split()

if len(arr) == 1:

ret = d.get(arr[0], 0)

print(\*ret, sep=', ') if ret else print('Не найдено')

else:

key = arr[0]

d[key] = d.get(key, [])

len\_arr = len(arr)

for i in range(1, len\_arr):

if i == len\_arr-1:

d[key].append(arr[i])

else:

d[key].append(arr[i][:-1])

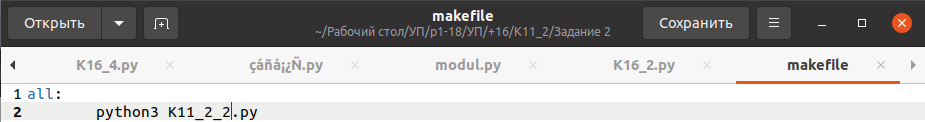
****

Рис. 61. makefile

Листинг 45. K11\_2\_3.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К11\_2. Техника работы со словарями

Задание 3. https://stepik.org/lesson/243394/step/13?unit=215740

Телефонная книга. Этап 3. Коле очень понравилась Ваша программа, однако он стал

замечать, что иногда в его телефонную книгу попадают номера в некорректном формате.

Чтобы не сохранять недействительные номера, он попросил Вас обрабатывать только номера,

соответствующие критериям:

- номер должен начинаться либо с +7, либо с 8 и состоять из 11 цифр.

- блоки цифр могут разделяться пробелами или дефисами.

- вторая, третья и четвертая цифры могут помещаться в скобки.

Если программа встречает некорректный номер, она должна его проигнорировать. В обратном

случае она должна привести номер к виду +7 (900) 800-70-60 и запомнить. Остальной

функционал программы остается без изменений.

Sample Input:

Ben 89001234050, +7 050 432 10-09

Alice 404-502, 894053212-65, 439-095

Nick +1(650)781 12-51

Ben

Alex +4(908)273-22-421, 8 (908) 273-22-42

Alice

Nick

Robert 51234047129, 89174043215

Alex

Robert

.

Sample Output:

+7 (900) 123-40-50, +7 (050) 432-10-09

+7 (940) 532-12-65

Не найдено

+7 (908) 273-22-42

+7 (917) 404-32-15

'''

#Фильтрует поступающие номера

def FilterNumbers(data, str):

index = 0 #наша позиция в строке

#data = [] #список данных

#Узнем имя

s = ''

while str[index] != ' ':

s += str[index]

index += 1

data.append(s) #кладем имя в список

s = ''

for i in range(index, len(str)):

#Filter

if str[i] == ',':

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

s2 = '+7' + s[1:]

data.append(s2)

elif (s[0] == '+' and s[1] == '7' and len(s) == 12):

data.append(s)

s = ''

#--Filter--

else:

if str[i].isdigit() or str[i] == '+':

s += str[i]

#Filter

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

s2 = '+7' + s[1:]

data.append(s2)

elif (s[0] == '+' and s[1] == '7' and len(s) == 12):

data.append(s)

s = ''

#--Filter--

#Выводим в правильной форме

def TrueForm(list):

#+7 (940) 532-12-65

len\_list = len(list)

for str in list:

if (list[len\_list-1] != str):

print(f"{str[:2]} ({str[2:5]}) {str[5:8]}-{str[8:10]}-{str[10:12]}", end = ', ')

else:

print(f"{str[:2]} ({str[2:5]}) {str[5:8]}-{str[8:10]}-{str[10:12]}")

d = dict()

for str in iter(input, '.'):

if ( len(str.split()) == 1):

ret = d.get(str, 0)

TrueForm(ret) if ret else print('Не найдено')

else:

data = []

FilterNumbers(data, str)

key = data[0]

d[key] = d.get(key, [])

len\_data = len(data)

for i in range(1, len\_data):

d[key].append(data[i])

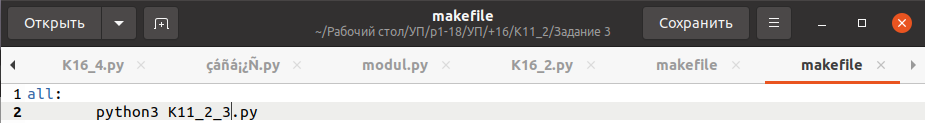


Рис. 62. makefile

## Техника работы с множествами

Приложения: K12\_1\_1.py, K12\_1\_2.py, K12\_1\_3.py, K12\_1\_4.py, K12\_1\_5.py, K12\_2\_1.py, K12\_2\_2.py

Листинг 46. K12\_1\_1.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К12\_1. Техника работы с множествами

Задание 1. https://pythontutor.ru/lessons/sets/problems/number\_of\_unique/

Задача «Количество различных чисел»

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

'''

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

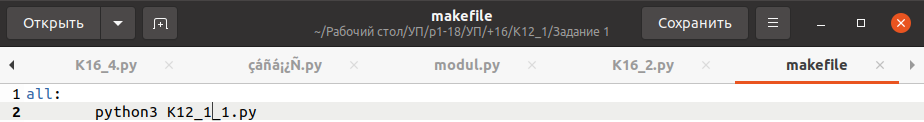


Рис. 63. makefile

Листинг 47. K12\_1\_2.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К12\_1. Техника работы с множествами

Задание 2. https://pythontutor.ru/lessons/sets/problems/number\_of\_coincidental/

Задача «Количество совпадающих чисел»

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

в первом списке, так и во втором.

'''

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

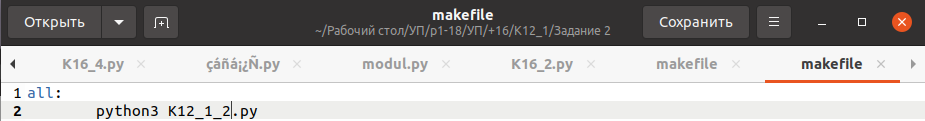


Рис. 64. makefile

Листинг 48. K12\_1\_3.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К12\_1. Техника работы с множествами

Задание 3. https://pythontutor.ru/lessons/sets/problems/sets\_intersection/

Задача «Пересечение множеств»

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

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

'''

print(\*sorted(set(input().split()) & set(input().split()), key=int))

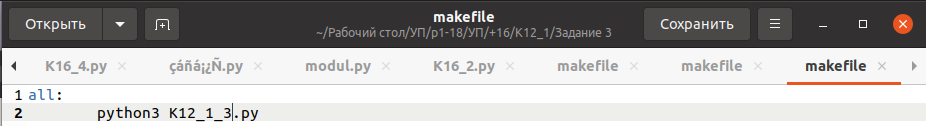


Рис. 65. makefile

Листинг 49. K12\_1\_4.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К12\_1. Техника работы с множествами

Задание 4. https://pythontutor.ru/lessons/sets/problems/number\_of\_words/

Задача «Количество слов в тексте»

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

Определите, сколько различных слов содержится в этом тексте.

Словом считается последовательность непробельных символов идущих подряд, слова разделены

одним или большим числом пробелов или символами конца строки.

'''

words = set()

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

words.update(input().split())

print(len(words))

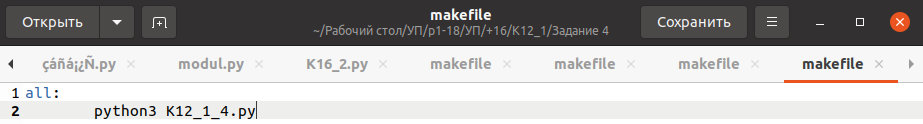


Рис. 66. makefile

Листинг 50. K12\_1\_5.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К12\_1. Техника работы с множествами

Задание 5. https://pythontutor.ru/lessons/sets/problems/polyglotes/

Задача «Полиглоты»

Условие. Каждый из некоторого множества школьников некоторой школы знает некоторое

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

знает хотя бы один из школьников.

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

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

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

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

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

лексикографическом порядке, по одному на строке.

'''

n = int(input())

language = []

for i in range(n):

k = int(input())

buff = set()

for j in range(k):

buff.add(input())

language.append(buff)

unic = set.union(\*language)

intersec = set.intersection(\*language)

print(len(intersec), '\n'.join(sorted(intersec)), len(unic), '\n'.join(sorted(unic)), sep='\n')

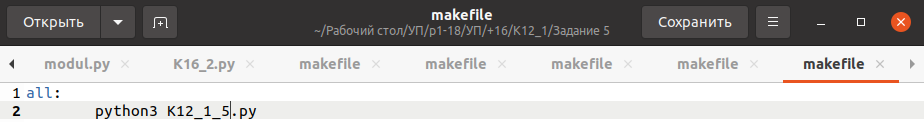


Рис. 67. makefile

Листинг 51. K12\_2\_1.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К12\_2. Техника работы с множествами

Задание 1. https://stepik.org/lesson/3380/step/3?unit=963

Простейшая система проверки орфографии может быть основана на использовании списка известных слов.

Если введённое слово не найдено в этом списке, оно помечается как "ошибка".

Попробуем написать подобную систему.

На вход программе первой строкой передаётся количество d известных нам слов, после чего

на d строках указываются эти слова.

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

Выведите уникальные "ошибки" в произвольном порядке. Работу производите без учёта регистра.

'''

count = int(input())

words = set()

for i in range(count):

s = input().lower()

words.add(s)

errors = set()

l = int(input())

for i in range(l):

s = input().lower().split()

for j in s:

if(j not in words and j not in errors):

errors.add(j)

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

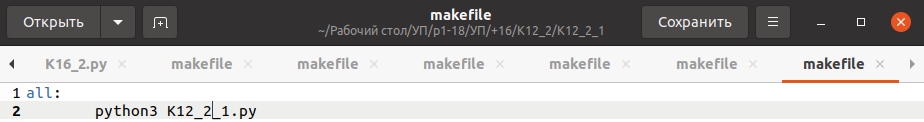


Рис. 68. makefile

Листинг 52. K12\_2\_2.py makefile disc.txt session.txt out.txt

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К12\_2. Техника работы с множествами

Задание 2. (Л.Б.) Сессия

В файле disc.txt хранится перечень дисциплин, выносимых на сессию, например,

Теория алгоритмов

МДК.01.01

Основы экономики

...

В файле session.txt хранятся сведения о результатах сессии, например,

Грушников; П2-18; Теория алгоритмов; 5

Константинович; П2-18; Теория алгоритмов; 5

...

Студент считается сдавшим сессию, если у него сданы все предметы и нет оценки "2".

Студент считается "отличником", если у него все пятерки

Требуется сформировать множества:

- студентов, сдавших сессию

- студентов-отличников

- дисциплин, по которым нет задолженностей

Результат вывести в файл output.txt

'''

file1 = open("session.txt", "r")

file2 = open("disc.txt", "r")

list = file1.readlines()

session = [i.strip().split('; ') for i in list]

list = file2.readlines()

disc = [i.strip() for i in list]

file1.close()

file2.close()

names = []

for str in session:

if str[0] not in names:

names.append(str[0])

otli4nik = set()

sdali = set()

predmet = {\*disc}

for i in range(len(names)):

count\_5 = 0

count\_2 = 0

for j in range(len(session)):

if(session[j][0] == names[i]):

if(session[j][3] == '5'):

count\_5 += 1

elif(session[j][3] == '2'):

count\_2 += 1

predmet.discard(session[j][2])

if(count\_5 == len(disc)):

otli4nik.add(names[i])

if(count\_2 == 0):

sdali.add(names[i])

file3 = open("out.txt","w")

file3.write('Сдали:\n'+'\n'.join(names))

file3.write('\n-----------------------------------------------------------------------------\n')

file3.write('Отличники:\n'+'\n'.join(otli4nik))

file3.write('\n-----------------------------------------------------------------------------\n')

file3.write('Предметы по которым нет задолжности:\n'+'\n'.join(predmet))

file3.close()

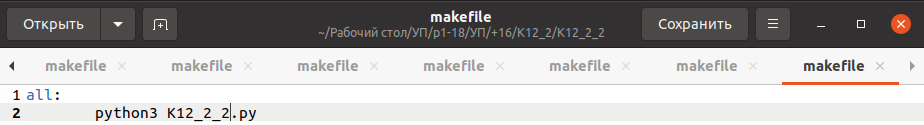


Рис. 69. makefile

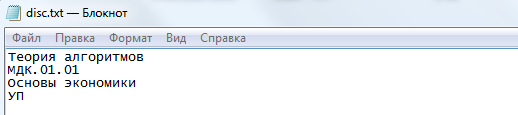


Рис. 70. disc.txt

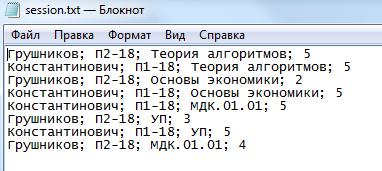


Рис. 71. session.txt

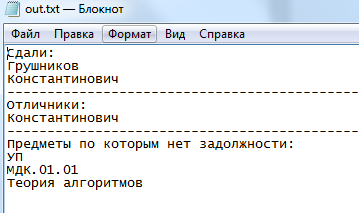


Рис. 72. out.txt

## Техника работы с кортежами

Приложения: K13\_1\_1.py, K13\_1\_2.py, K13\_1\_3.py, K13\_2

Листинг 53. K13\_1\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К13\_1. Техника работы с кортежами

Задание 1. https://stepik.org/lesson/193753/step/4?unit=168148

Вывести чётные

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

Sample Input:

2

Sample Output:

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

'''

a =int(input())

print(tuple([i for i in range(a+a%2,a\*10+1,2)]))

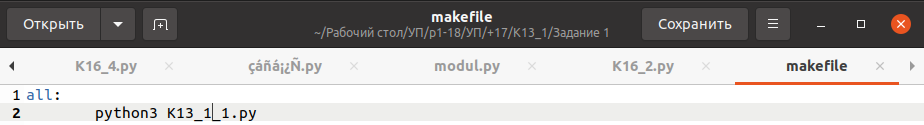


Рис. 73. makefile

Листинг 54. K13\_1\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К13\_1. Техника работы с кортежами

Задание 2. https://stepik.org/lesson/193753/step/5?unit=168148

Убывающий ряд.

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

по одному числу в строке.

Sample Input:

-2

-8

Sample Output:

-2

-3

-4

-5

-6

-7

'''

print(\*tuple([i for i in range(int(input()), int(input()), -1)]), sep='\n')

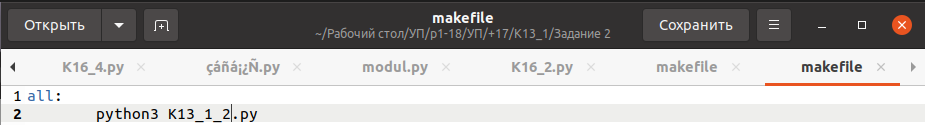


Рис. 74. makefile

Листинг 55. K13\_1\_3.py makefile input.txt

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К13\_1. Техника работы с кортежами

Задание 3. (Л.Б.) В каждой строке файла хранится информация о пунктах и их координатах

относительно некоторого центра.

Требуется

1. Прочесть файл в список кортежей

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

Указать наиболее удалённые пары

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

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

Результаты вывести на экран

Пример входного файла

Москва 0 0

Ивантеевка 20 15

Щёлково 10 30

Пушкино 15 5

'''

def BubbleSort(vals):

n = len(vals)

for i in range(n):

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

if (vals[i] < vals[j]):

vals[i], vals[j] = vals[j], vals[i]

return vals

fin = open("input.txt", 'r')

d\_coord = dict()

d\_hyp = dict()

print()

for str in fin:

l = str.split()

d\_coord[l[0]] = tuple(l[1:])

d\_hyp[l[0]] = ( int(d\_coord[l[0]][0])\*\*2 + int(d\_coord[l[0]][1])\*\*2 )\*\*0.5

fin.close()

vals\_sort = BubbleSort(list(d\_hyp.values()))

print("Сортированны по убыванию:")

numb\_city = len(d\_coord)

for i in range(numb\_city):

for city in d\_hyp:

if d\_hyp[city] == vals\_sort[i]:

print(f"{i+1}|", end = ' ')

print(city + ":", \*d\_coord[city], end = '\n ')

print(f"До центра: {int(vals\_sort[i])} км\n")

break

count = 0

for city\_i in d\_hyp:

for city\_j in d\_hyp:

if d\_hyp[city\_i] == d\_hyp[city\_j] and city\_i != city\_j:

print(f"{d\_hyp[city\_i]} == {d\_hyp[city\_j]}")

count += 1

if count == 0:

print("Пар городов, имеющих одинаковое расстояние до центра, не обнаруженно")

'''

for i in sorted(d\_hyp.values()):

print(i)

'''

'''

fout = open("output.txt", 'w')

fout.write('\n'.join(tup))

fout.close()

'''

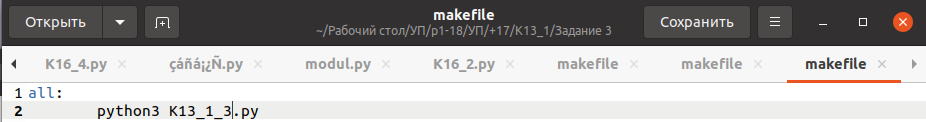


Рис. 75. makefile

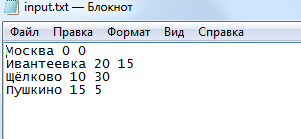


Рис. 76. input.txt

Листинг 56. K13\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К13\_2. Техника работы с кортежами

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

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

По приведённым примерам подготовить свои.

'''

print("Именнованные кортежи")

#Подключаем модуль namedtuple

print("#Подключаем модуль namedtuple")

print("from collections import namedtuple")

from collections import namedtuple

print()

#Создаем именнованый кортеж

print("#Создаем именнованый кортеж")

print("pos = namedtuple('pos', ['x', 'y'])")

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

print()

print("#Кортеж с позиционным параметром")

print("p1 = pos(20, 15)")

p1 = pos(20, 15)

print("print(p1)")

print(p1)

print('print(" Сумма: ", p1[0] + p1[1])')

print(" Сумма: ", p1[0] + p1[1])

print()

#Кортеж с именованным параметром

print("#Кортеж с именованным параметром")

print("p2 = pos(x = 30, y = 40)")

p2 = pos(x = 30, y = 40)

print("print(p2)")

print(p2)

print('print(" Сумма: ", p2[0] + p2[1])')

print(" Сумма: ", p2[0] + p2[1])

print()

#Можно распоковать, как обычный кортеж

print("#Можно распоковать, как обычный кортеж")

print("x1, y1 = p1")

x1, y1 = p1

print("print(x1, y1)")

print(x1, y1)

#Поля также доступны по названию

print("#Поля также доступны по названию")

print("#PS: которое мы присвоили в самом начале, т.е")

print("# pos = namedtuple('pos', ['x', 'y'])")

print("print(p1.x + p1.y)")

print(p1.x + p1.y)

print()

print("#Именованные кортежи поддерживают функцию getattr():")

print("print(getattr(p1, 'x'))")

print(getattr(p1, 'x'))

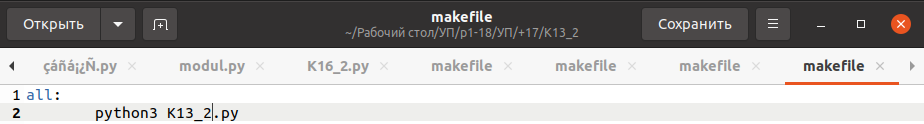


Рис. 77. makefile

## Техника работы с файлами

Приложения: K14\_1\_1.py, K14\_1\_2.py, K14\_1\_3.py, K14\_1\_4.py, K14\_1\_5.py, K14\_2.py

Листинг 57. K14\_1\_1.py makefile file.txt

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К14\_1. Техника работы с файлами

Задание 1. http://ptaskbook.com/ru/tasks/text.php

Text5. Дана строка S и текстовый файл. Добавить строку S в конец файла.

'''

fin = open("file.txt","a")

print("Enter your string:")

fin.write(input())

fin.close()

print("Pasted!")

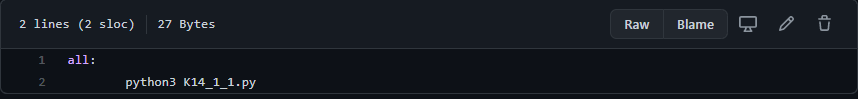


Рис. 78. makefile

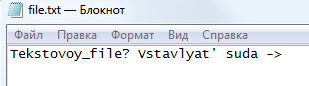


Рис. 79. file.txt

Листинг 58. K14\_1\_2.py makefile file.txt

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К14\_1. Техника работы с файлами

Задание 2. http://ptaskbook.com/ru/tasks/text.php

Text12. Дана строка S и текстовый файл. Заменить в файле все пустые строки на строку S.

'''

import os

fin = open("file.txt", "r")

list = fin.readlines()

fin.close()

os.remove("./file.txt")

s = input()

fout = open("file.txt","w")

for i in list:

for j in range(len(i)):

if(i[j-1] == '\n' and i[j] == '\n'):

fout.write(s+'\n')

else:

fout.write(i[j])

fout.close()



Рис. 80. makefile

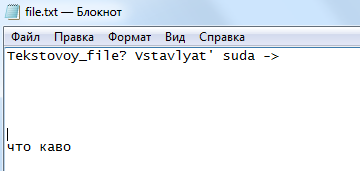


Рис. 81. file.txt

Листинг 59. K14\_1\_3.py makefile file.txt file2.txt

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К14\_1. Техника работы с файлами

Задание 3. http://ptaskbook.com/ru/tasks/text.php

Text20. Дан текстовый файл. Заменить в нем все подряд идущие пробелы на один пробел.

'''

fout = open("file2.txt","w")

with open("file.txt","r") as fin:

fout.write('\n'.join(' '.join(line.split()) for line in fin))

fout.close()

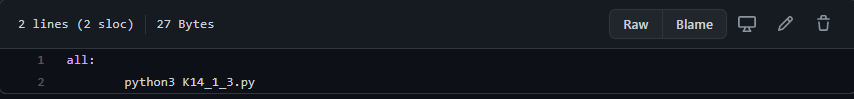


Рис. 82. makefile

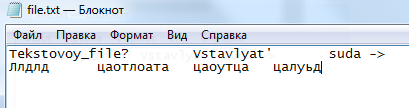


Рис. 83. file.txt

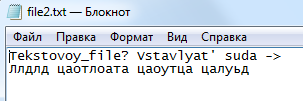


Рис. 84. File2.txt

Листинг 60. K14\_1\_4.py makefile file.txt

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К14\_1. Техника работы с файлами

Задание 4. http://ptaskbook.com/ru/tasks/text.php

Text44. Дан текстовый файл, каждая строка которого изображает целое число, дополненное слева и справа

несколькими пробелами. Вывести количество этих чисел и их сумму.

'''

with open("file.txt","r") as fin:

list = [line.split() for line in fin]

sum, count = 0, 0

for i in range(len(list)):

sum += (int(\*list[i]))

count += 1

print(\*list[i])

print("sum =",sum)

print("count =", count)



Рис. 85. makefile

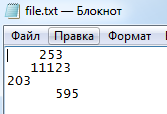


Рис. 86. file.txt

Листинг 61. K14\_1\_5.py makefile file.txt file2.txt

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К14\_1. Техника работы с файлами

Задание 5. http://ptaskbook.com/ru/tasks/text.php

Text53. Дан текстовый файл. Создать символьный файл, содержащий все знаки препинания,

встретившиеся в текстовом файле (в том же порядке).

'''

from string import punctuation

with open("file.txt", "r") as fin:

list = [line for line in fin]

punc = set()

for i in range(len(list)):

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

if list[i][j] in punctuation:

punc.add(list[i][j])

fout = open("file2.txt", "w")

fout.write("Знаки пунктуации:\n"+str(punc))

fout.close()



Рис. 87. makefile

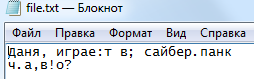


Рис. 88. file.txt

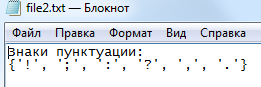


Рис. 89. file2.txt

Листинг 62. K14\_2.py makefile input.txt

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К14\_2. Техника работы с файлами

Задание 1. (Л.Б.)

При разработке курсовых проектов студентами 3 курса программистов ККМТ выбираются

различные направления, например, "графика", "базы данных"..

и предпочтения по языкам и средам "Си++", "Delphi"...

В каждой строке текстового файла хранятся следующие сведения о курсовых проектах:

Фамилия Имя Отчество; Группа; Год; Тема; Направления (список через запятую);

Языки и среды (список через запятую)

Например,

Иванов Иван Иванович;П1-21;2023;Картинки в базе;графика;Pascal,Lazarus

Программа должна читать входной файл и выдавать на экран ответы на вопросы

1. Какое направление встречается чаще всего

2. Какие языки и среды появились в дипломах в 2017 г.

'''

print()

fin = open("input.txt", 'r')

data\_user = list() #Список хранящий данные пользователей

num\_user = 0 #Кол-во пользователей

keys\_d = ['name', 'group', 'year', 'theme', 'direction', 'lang'] #Список названий (ключей) словаря

num\_keys = len(keys\_d) #кол-во ключей

#Чтение файла и запись в список (data\_user)

for str in fin:

l = str.strip().split(';') #Удаляем '\n' в конце и делим на части

len\_l = len(l) #кол-во частей

if (len\_l != num\_keys):

print('Не достаточно данных')

continue

data\_user.append(dict()) #Добавляем в список словарь, который будет хранить данные пользователя

#Запись обычных данных (не многомерных)

for i in range(len\_l-2):

data\_user[num\_user][keys\_d[i]] = l[i]

#Запись многомерных

for i in range(len\_l-2, len\_l):

data\_user[num\_user][keys\_d[i]] = l[i].split(',')

num\_user += 1 #Кол-во пользователей

fin.close()

#Вывод данных о пользователях

for i in range(len(data\_user)): #Номер каждого человека

print(f"---User #{i+1}---")

for key in data\_user[i]:

print(f"{key}: {data\_user[i][key]}") #

print()

print()

#--{1. Какое направление встречается чаще всего}--

d\_dir = dict() #Хранит имя\_направления: кол-во\_людей\_которые\_им\_занимаются

pos\_dir = -2 #Позиция "Направления" (direction) в списке ключей(keys\_d)

for i in range(num\_user): #номер человека

direct = data\_user[i][keys\_d[pos\_dir]] #Хранит список всех направлений человека

for word in direct:

d\_dir[word] = d\_dir.get(word, 0) + 1 #Подсчитываем кол-во каждого направления

num\_\_d\_dir = list(d\_dir.values()) #Список кол-ва направлений каждого человека

maxx = max(num\_\_d\_dir) #Наибольшее кол-во направлений

#Вывод наиболее часто встречающегося направления

print("Чаще всего встречается направление:", end=' ')

for i in d\_dir:

if (maxx == d\_dir[i]):

print(i, end=' ')

print('\n')

#--{2. Какие языки и среды появились в дипломах в 2017 г.}--

need\_year = '2017' #Год который мы ищем

lang = set() #Множество хранящее языки и среды нужного нам года

pos\_year = 2 #Позиция года в списке ключей(keys\_d)

pos\_lang = -1 #Позиция языков и сред в списке ключей(keys\_d)

for i in range(len(data\_user)):

year = data\_user[i][keys\_d[pos\_year]] #Год пользователя

if (year == need\_year):

set\_lang = set(data\_user[i][keys\_d[pos\_lang]]) #Хранит список всех языков и сред человека

lang.update(set\_lang) #Объединяем множества "языков текущего пользователя" со "всеми языками пользователя" за нужный нам год

#Вывод

print("Языки и среды появившиеся в дипломах 2017 года:", end=' ')

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

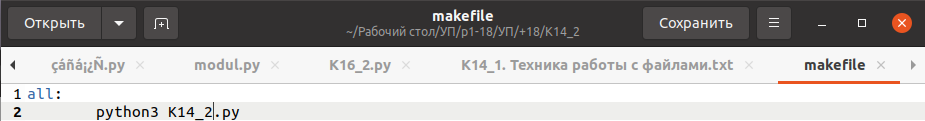


Рис. 90. makefile

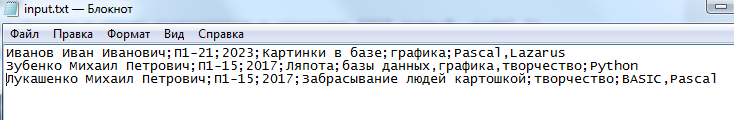


Рис. 91. input.txt

## Техника работы с модулями

Приложения: K15\_1\_1.py, K15\_1\_2.py, K15\_2\_1.py, K15\_2\_2.py, K15\_3\_1.py, K15\_3\_2.py, K15\_3\_3.py, K15\_3\_4.py, K15\_3\_5.py, K15\_3\_6.py, K15\_4.py, K15\_4\_1.py

Листинг 63. K15\_1\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К15\_1. Техника работы с модулями

Задание 1. Контейнерные типы данных модуля collections.

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

Класс deque() модуля collections в Python.

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

'''

import collections

string = input()

string2 = "234523"

dq = collections.deque(string)

dq.append(string2)

print(dq)

dq.extend('ehwr')

print(dq)

dq.extendleft('ab')

print(dq)

print()

print("dq.index('a', 1) =>", dq.index('a', 1))

print("dq.pop() =>", dq.pop())

print(dq)

print("dq.popleft() =>", dq.popleft())

print(dq)

print()

dq.reverse()

print(dq)

dq.rotate(1)

print(dq)

dq.rotate(2)

print(dq)

dq.rotate(-2)

print(dq)

dq.rotate(-1)

print(dq)

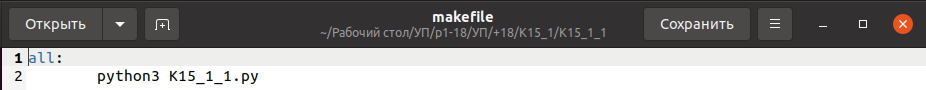


Рис. 92. makefile

Листинг 64. K15\_1\_2.py makefile text1.txt

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К15\_1. Техника работы с модулями

Задание 2. Контейнерные типы данных модуля collections.

Класс Counter() модуля collections в Python.

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

'''

import collections

import re

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

print(cnt)

print(\*cnt.elements())

ct = collections.Counter("abbbaaaccacccascd")

s = set(ct)

print(ct.most\_common(len(s)))

cnt1 = collections.Counter(a=3, b=6, c=6, d=5)

cnt1.subtract(cnt) # вычитает элементы текущего счетчика

print(cnt1)

cnt1.update(cnt) # складывает элементы текущего счетчика

print(cnt1)

print()

print("#print(cnt + cnt1)")

print(cnt + cnt1) #Сложить два счетчика

print("#print(cnt - cnt1)")

print(cnt - cnt1) #Вычитание счетчиков

print("#print(cnt & cnt1)")

print(cnt & cnt1) #Пересечение счетчиков

print("#print(cnt | cnt1)")

print(cnt | cnt1) #Объединение счётчиков

print()

print(cnt.items())

print(cnt.values())

cnt.clear()

string = ""

c = collections.Counter()

with open("text1.txt", "r") as file:

for i in file:

string += i

c = collections.Counter(string).most\_common(len(string))

print(c)

cn = collections.Counter()

with open("text1.txt", "r") as file:

for i in file:

words = re.findall(r'\w+', file.read()) #findall используется для поиска всех непересекающихся совпадений в шаблоне

cn = collections.Counter(words).most\_common(len(words))

print(cn)

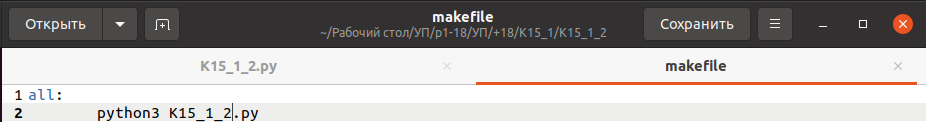


Рис. 92. makefile

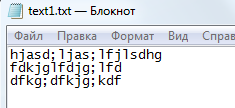


Рис. 93. text1.txt

Листинг 64. K15\_2\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К15\_2. Техника работы с модулями

Задание 1. Контейнерные типы данных модуля collections.

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

Класс defaultdict() модуля collections в Python.

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

'''

from collections import defaultdict

import random

print("1.--------------")

lst = [("Ben", 89001234050), ("Alice", 210-220), ("Ben", 70504321009), ("Alice", 404-502), ("Nick", 16507811251),

("Robert", 51234047129), ("Alice", 894-455), ("Alice", 439-495)]

d = defaultdict(list)

for i, elem in lst:

d[i].append(elem)

print(d.items())

print()

print("2.--------------")

lst\_number = [('a', 1), ('b', 2), ('c', 3), ('d', 4) , ('e', 5) , ('f', 6), ('a', 2), ('a', 3)]

d\_slov = {}

for i, elem in lst\_number:

d\_slov.setdefault(i, []).append(elem\*\*2)

print(sorted(d\_slov.items()))

print()

print("3.--------------")

l = {}

for i, elem in lst:

n = random.randint(1, 100)

l.setdefault(i, []).append(n)

print(l)

print()

print("4.--------------")

string = "Hello world"

d = defaultdict(int)

for k in string:

d[k] -= 1

print(d)

print()

print("5.--------------")

lst = ["qwertyui", "asdfghjkl", "zxcvbnm", "q"]

d = defaultdict(int)

for i in lst:

d[i] += len(i)

print(d.items())

print()

print("6.--------------")

lst = [("Ben", 89001234050), ("Alice", 210-220), ("Ben", 70504321009), ("Alice", 404-502), ("Nick", 16507811251),

("Robert", 51234047129), ("Alice", 894-455), ("Alice", 439-495)]

d = defaultdict(set)

index = 0

for i, elem in lst:

if (index % 2 == 0):

d[i].add(elem)

index += 1

print(d.items())

print()

print("7.--------------")

for elem in lst:

print(elem)

print()

print("8.--------------")

for i, elem in lst:

print(i, elem)

print()

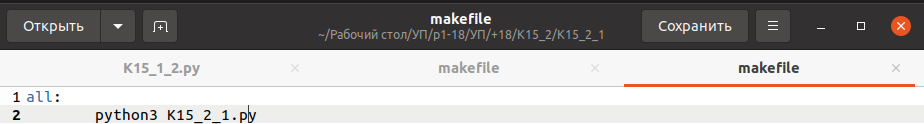


Рис. 94. makefile

Листинг 65. K15\_2\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К15\_2. Техника работы с модулями

Задание 2. Контейнерные типы данных модуля collections.

Класс OrderedDict() модуля collections в Python.

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

'''

from collections import \*

c = Counter()

items = []

n = int(input())

for i in range(n):

string = input()

items.append(string)

for i in items:

c[i] += 1

print(c)

print()

defdict = defaultdict(list)

for i in range(n+1):

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

defdict[i].append(j)

print(defdict)

print()

d = OrderedDict.fromkeys('abcd')

d.move\_to\_end('b') # добавляет элемент из строки в конец

print(''.join(d.keys())) #print(d.keys())

d.move\_to\_end('a')

print(''.join(d.keys()))

d.popitem('a') # удаляем элемент

d.popitem('b')

print(''.join(d.keys()))

d.move\_to\_end('d', last=True) #переносим d вперёд

print(''.join(d.keys()))

d.move\_to\_end('d', last=False) #переносим d вперёд

print(''.join(d.keys()))

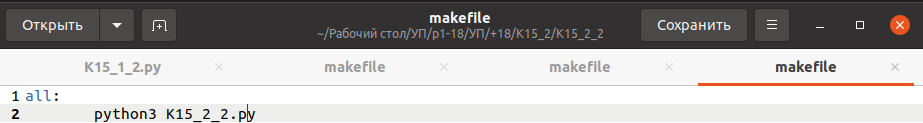


Рис. 95. makefile

Листинг 66. K15\_3\_1.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К15\_3. Техника работы с модулями

Модуль sys в Python.

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

Задание 1. Функция argv модуля sys в Python.

https://docs-python.ru/standart-library/modul-sys-python/funktsija-argv-modulja-sys/

'''

import sys

print("Количество аргументов:")

print(len(sys.argv))

print("Последний аргумент:")

print(sys.argv[-1])

print("Все аргументы:")

for i in range(len(sys.argv)):

print(sys.argv[i], end=" ")

print(len(sys.argv))

if len(sys.argv) > 1:

if ('-h' in sys.argv) or ('-help' in sys.argv):

print("Помощи нет")

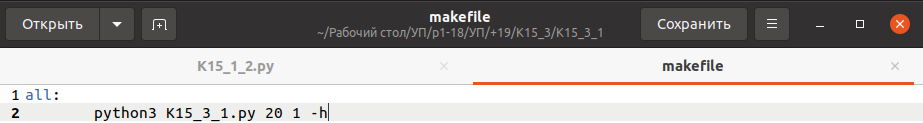


Рис. 96. makefile

Листинг 67. K15\_3\_2.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К15\_3. Техника работы с модулями

Модуль sys в Python.

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

Задание 2. Имя используемой OS.

https://docs-python.ru/standart-library/modul-sys-python/imja-ispolzuemoj-os/

'''

import sys

import os

if sys.platform.startswith('linux'):

print(f"This is linux {os.name}")

else:

print(f"This is not linux. This is {os.name}")

print(sys.getwindowsversion())

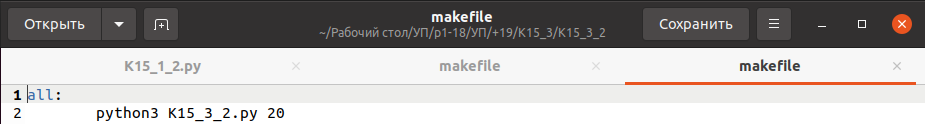


Рис. 97. makefile

Листинг 68. K15\_3\_3.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К15\_3. Техника работы с модулями

Модуль sys в Python.

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

Задание 3. Различные сведения о версии Python.

https://docs-python.ru/standart-library/modul-sys-python/razlichnye-svedenija-versii/

'''

import sys

string = sys.version\_info

print(string)

print(sys.copyright)

print("API C languages: ", sys.api\_version)

print(sys.version)

print("Hex version: ", sys.hexversion)

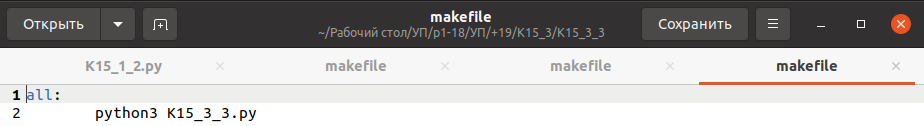


Рис. 98. makefile

Листинг 69. K15\_3\_4.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К15\_3. Техника работы с модулями

Модуль sys в Python.

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

Задание 4. Каталоги и пути интерпретатора Python.

https://docs-python.ru/standart-library/modul-sys-python/katalogi-puti-interpretatora/

'''

import sys

import os

print()

print("Совет! Не вводите большие числа")

print(sys.prefix)

path\_dir = "."

path\_const = path\_dir

n = int(input())

for i in range(0, n):

path = path\_dir + "/Dir " + str(i)

os.mkdir(path)

path\_const = "./Dir " + str(i+1)

os.mkdir(path\_const)

path\_dir += "/Dir " + str(i)

print(os.getcwd())

if n > 5:

print("Зря...")

print(sys.base\_prefix)

print(sys.exec\_prefix)

print(sys.base\_exec\_prefix)

print(sys.executable)

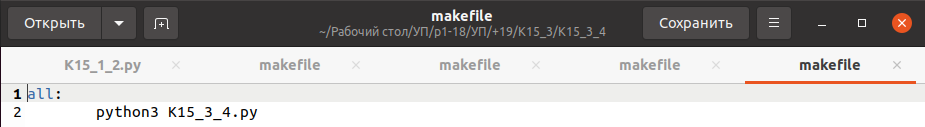


Рис. 99. makefile

Листинг 70. K15\_3\_5.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К15\_3. Техника работы с модулями

Модуль sys в Python.

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

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

https://docs-python.ru/standart-library/modul-sys-python/obekty-stdin-stdout-stderr-modulja-sys/

'''

import sys

stdin = sys.stdin

try:

sys.stdin = open("text.txt", "r")

s = input()

print("odna stroka: ", s)

finally:

sys.stdin.close()

sys.stdin = stdin

try:

sys.stdin = open("text.txt", "r")

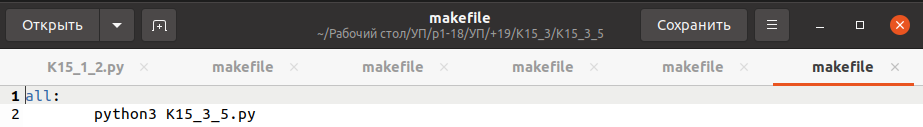
for i in sys.stdin:

print(i, end="")

finally:

sys.stdin.close()

sys.stdin = stdin



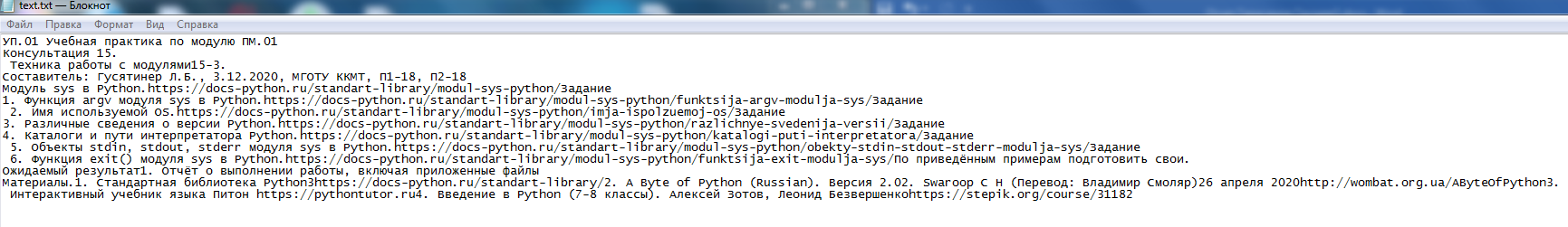
Рис. 100. makefile

Рис. 101. text.txt

Листинг 71. K15\_3\_6.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

К15\_3. Техника работы с модулями

Модуль sys в Python.

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

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

https://docs-python.ru/standart-library/modul-sys-python/funktsija-exit-modulja-sys/

'''

import sys

if len(sys.argv) > 1:

if ("-exit" in sys.argv) or ("-e" in sys.argv):

print("Ну вы вышли")

sys.exit(0)

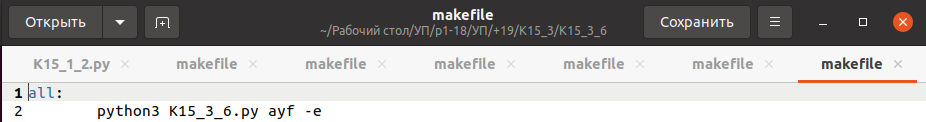


Рис. 102. makefile

Листинг 72. K15\_4.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К15\_1. Техника работы с модулями

Задание 1. Вывод текущей директории

Задание 2. Создание папки

Задание 3. Изменение директории

Задание 4. Создание вложенных папок

Задание 5. Создание файлов

Задание 6. Переименование файлов

Задание 7. Перемещение файлов

Задание 8. Список файлов и директорий

Задание 9. Удаление файлов

Задание 10. Удаление директорий

Задание 11. Получение информации о файлах

'''

import os

#Задание 1. Вывод текущей директории

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

print()

#Задание 2. Создание папки

if not os.path.isdir("Балдеж"): # Проверка на отсутсвие папки с таким названием

os.mkdir("Балдеж")

#Задание 3. Изменение директории

os.chdir("./Балдеж")

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

os.mkdir("Новая папка")

os.chdir("Новая папка")

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

print()

#Задание 4. Создание вложенных папок

os.chdir("..")

path\_dir = os.getcwd()

n = int(input("Введите количество вложенных папок, которые вы хотите создать: "))

for i in range(n):

path = path\_dir + "/Папка " + str(i)

os.mkdir(path)

path\_dir += "/Папка " + str(i)

print()

#Задание 5. Создание файлов

file = open("file.txt", "w")

file.write("Уж тварь ли я дрожащая или право имею?")

file.close()

#Задание 6. Переименование файлов

file = open("newfile.txt", "w")

file.write("Ну, да, я файл. И что")

file.close()

Second\_name = input("Как Вы хотите переименовать файл: ")

os.rename("newfile.txt", Second\_name)

print()

#Задание 7. Перемещение файлов

file = open("First\_name.txt", "w")

file.close()

os.replace("First\_name.txt", "Новая папка/First\_name.txt")

#Задание 8. Список файлов и директорий

# распечатать все файлы и папки

for dirpath, dirnames, filenames in os.walk("."): #Перебирает все переданные составляющие.

# перебрать каталоги

for dirname in dirnames:

print("Каталог:", os.path.join(dirpath, dirname))

# перебрать файлы

for filename in filenames:

print("Файл:", os.path.join(dirpath, filename))

print()

#Задание 9. Удаление файлов

File\_name = input("Введите имя файла: ")

f = open(File\_name + ".txt", "w")

f.close()

os.remove(File\_name + ".txt")

#Задание 10. Удаление директорий

os.mkdir('Bruh')

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

print("Все папки и файлы: ", os.listdir())

del\_name = input("Введите имя папки которую Вы хотите удалить: ")

os.rmdir(del\_name)

print()

#Задание 11. Получение информации о файлах

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

f.write('Hell, World')

f.close()

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

os.remove("text.txt")

print()

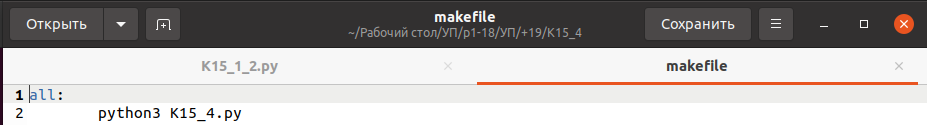


Рис. 103. makefile

## Техника работы с классами

Приложения: K16\_1.py, K16\_2.py, K16\_3.py, K16\_4.py

Листинг 73. K16\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К16\_1. Техника работы с классами.

Задание 1. Создание класса

Задание 2. Создание объекта

Задание 3. Функция init

Задание 4. Методы объектов

Задание 5. Параметр self

Задание 6. Изменение свойств объекта

Задание 7. Удалить свойства объекта

Задание 8. Удаление объектов

'''

#Создадим класс

class BruhClass:

x = 20

#Создадим объект

p1 = BruhClass()

print(p1.x)

#Функция init

class DataHero:

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

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

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

self.name = name

self.age = age

#Методы класса

def Welcome(self):

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

def DataHero(bruh):

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

print(f"Имя: {bruh.name}")

print(f"Возраст: {bruh.age}")

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

print()

hero1 = DataHero("Владимир", 20)

hero1.Welcome()

hero1.DataHero()

#Свойство объекта можно менять

hero1.age = 9999

hero1.DataHero()

#Свойства объектов можно удалять

del hero1.age

#Также можно удалять сам объект

del hero1

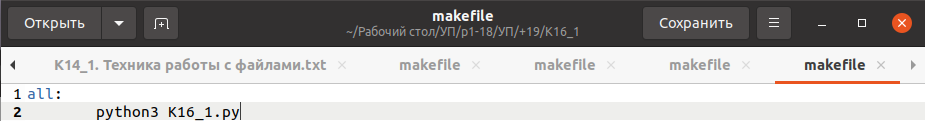


Рис. 104. makefile

Листинг 74. K16\_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К16\_2. Техника работы с классами.

Задание 1. Создание классов

Задание 2. Создание экземпляров класса

Задание 3. Доступ к атрибутам

Задание 4. Встроенные атрибуты класса

Задание 5. Уничтожение объектов (сбор мусора)

'''

#1. Создание классов

class DataHero:

"""Функции героя, статистика и другое о нем"""

obj\_count = 0

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

self.name = name

self.age = age

DataHero.obj\_count += 1

#Методы класса

#Количество объектов класса

def ActCount(self):

print(f"Количество объектов класса: {DataHero.obj\_count}")

#Приветствие

def Welcome(self):

print(f"Добро пожаловать, в долину силы, мой юный друг {self.name}")

#Данные героя

def DataHero(bruh):

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

print(f"Имя: {bruh.name}")

print(f"Возраст: {bruh.age}")

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

print()

#----------------------------------

#2. Создание экземпляров класса

hero1 = DataHero("Владимир", 20) #1ый объект класса DataHero

hero2 = DataHero("icefanik", 33) #2ой объект класса DataHero

#Документация класса

print(hero1.\_\_doc\_\_)

print(DataHero.\_\_doc\_\_)

#----------------------------------

#3. Доступ к атрибутам

hero1.Welcome()

hero1.DataHero()

print(f"Количество героев: {DataHero.obj\_count}")

#Свойство объекта можно менять

hero1.age = 9999

hero1.DataHero()

#Свойства объектов можно удалять

del hero2.age

#Также можно удалять сам объект

del hero2

print(hasattr(hero1, 'age')) #Возвращает True, если атрибут 'age' существует

print(getattr(hero1, 'age')) #Возвращает значение атрибута 'age'

delattr(hero1, 'age') #Удаляет атрибут 'age'

setattr(hero1, 'age', 8) #Устанавливает атрибут 'age' на 8

print()

#----------------------------------

#4. Встроенные атрибуты класса

#По объекту класса

print("#По объекту класса")

print(f"hero1.\_\_doc\_\_: {hero1.\_\_doc\_\_}") #Документация класса

print(f"hero1.\_\_dict\_\_: {hero1.\_\_dict\_\_}") #Словарь, содержащий пространство имен класса.

print()

#По классу

print("#По классу")

print(f"DataHero.\_\_doc\_\_: {DataHero.\_\_doc\_\_}") #Документация класса

print(f"DataHero.\_\_name\_\_: {DataHero.\_\_name\_\_}") #Наименование класса

print(f"DataHero.\_\_module\_\_: {DataHero.\_\_module\_\_}") #Имя модуля, в котором определяется класс. Этот атрибут \_\_main\_\_ в интерактивном режиме.

print(f"hero1.\_\_bases\_\_: {DataHero.\_\_bases\_\_}") #Могут быть пустые tuple, содержащие базовые классы, в порядке их появления в списке базового класса.

print(f"hero1.\_\_dict\_\_: {DataHero.\_\_dict\_\_}") #Словарь, содержащий пространство имен класса.

print()

#----------------------------------

#5. Уничтожение объектов (сбор мусора)

class Point:

def \_\_init\_\_(self, x=0, y=0):

self.x = x

self.y = y

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

class\_name = self.\_\_class\_\_.\_\_name\_\_

print(f"Класс {class\_name} уничтожен")

ptr1 = Point()

print(f"id\_ptr1 = {id(ptr1)}, ptr1 = {ptr1.x} {ptr1.y}")

ptr1 = Point(1, 2)

print(f"id\_ptr1 = {id(ptr1)}, ptr1 = {ptr1.x} {ptr1.y}")

ptr2 = ptr1

ptr3 = ptr1

print(id(ptr1), id(ptr2), id(ptr3))

del ptr1

del ptr2

del ptr3

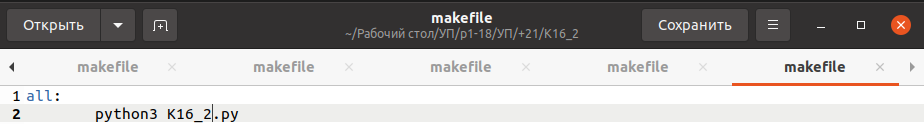


Рис. 105. makefile

Листинг 75. K16\_3.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К16\_3. Техника работы с классами.

Задание 1. Наследование класса

Задание 2. Переопределение методов

Задание 3. Популярные базовые методы

Задание 4. Приватные методы и атрибуты класса

'''

#1. Наследование класса

print("#Задание 1")

class Parent:

"""Родительский класс"""

parent\_attr = 18

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

print("Вызов родительского класса")

def parent\_method(self):

print('Вызов родительского метода')

def set\_attr(self, attr):

Parent.parent\_attr = attr

def get\_attr(self):

print(f"Вызов родителя: {Parent.parent\_attr}")

#для задания 2

def my\_method(self):

print('Вызов родительского метода')

#Ссылается на класс Parent

class Child(Parent):

"""Класс наследник"""

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

print("Вызов класса наследника")

def child\_method(self):

print("Вызов метода класса наследника")

def set\_attr\_c(self, attr):

Parent.parent\_attr = attr

#для задания 2

def my\_method(self):

print('Вызов метода наследника')

#----------------------------------

c = Child() # экземпляр класса Child

c.child\_method() # вызов метода child\_method

c.parent\_method() # вызов родительского метода parent\_method

c.set\_attr(200) # еще раз вызов родительского метода

c.get\_attr() # снова вызов родительского метода

c.set\_attr\_c(300) # еще раз вызов родительского метода

c.get\_attr() # снова вызов родительского метода

#Возвращает True, если 1ый аргумент подкласс 2го

print(issubclass(Child, Parent))

print(issubclass(Parent, Child))

#Возвращает True, если 1ый аргумент является экземпляром подкласса класса

print(isinstance(c, Parent))

print(isinstance(c, Child))

del c

print()

#Задание 2. Переопределение методов

print("#Задание 2")

v1 = Child()

v1.my\_method()

print()

v2 = Parent()

v2.my\_method()

del v1, v2

print()

# Т.е если у классов метод совпадает, то вызывается идет обращение к классу,

#к которому принадлежит наш объект

#Задание 3. Популярные базовые методы

class Vector:

def \_\_init\_\_(self, a, b):

self.a = a

self.b = b

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

return f"Vector({self.a}, {self.b})"

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

return f"Vector('{self.a}', '{self.b}')"

def \_\_add\_\_(self, other):

#return f"Vector({self.a + other.a}, {self.b + other.b})"

return Vector(self.a + other.a, self.b + other.b)

v1 = Vector(2, 10)

v2 = Vector(10, -2)

print(str(v1)) #\_\_str\_\_

print(repr(v1)) #\_\_repr\_\_

print(v1 + v2) #\_\_add\_\_

del v1, v2

#Задание 4. Приватные методы и атрибуты класса

class Class:

\_\_secret\_count = 0

def count(self):

self.\_\_secret\_count += 1

print(self.\_\_secret\_count)

counter = Class()

counter.count()

counter.count()

print(counter.\_Class\_\_secret\_count) #Обращаемся к приватному атрибуту

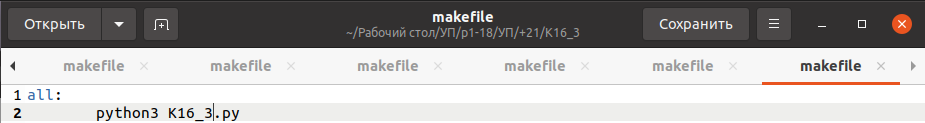


Рис. 106. makefile

Листинг 76. K16\_4.py modul.py makefile

---------------------K16\_4.py---------------------

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К16\_4. Техника работы с классами.

Задание 1. Придумать собственный класс

Задание 2. Неформально описать функционал класса

Задание 3. Реализовать класс в модуле

Задание 4. Разработать скрипт для демонстрации работы с классом (импортировать модуль,

создать экземпляры, вызвать методы)

'''

class Hero:

"""Данные героя"""

obj\_count = 0

damage = 5

helthy = 10

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

self.name = name

Hero.obj\_count += 1

def count(self):

return Hero.obj\_count

def InputHero(self, damage, helthy):

self.damage = damage

self.helthy = helthy

def printDataHero(self):

print("Name:", self.name)

print("Damage:", self.damage)

print("Helthy:", self.helthy)

def DataHero(self):

return [self.name, self.damage, self.helthy]

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

class\_name = self.\_\_class\_\_.\_\_name\_\_

print(f"Герой \"{self.name}\" удален. Класс \"{class\_name}\"")

Hero.obj\_count -= 1

qwe = Hero("Qwe")

print(qwe.DataHero())

print(f"Heroes: {qwe.count()}")

print()

fire = Hero("Fire")

fire.InputHero(15, 30)

print(fire.DataHero())

print(f"Heroes: {fire.count()}")

print()

import modul

predmets = modul.SubjectsHero()

predmets.sword()

predmets.shield()

predmets.printSubjectsHero()

print()

---------------------modul.py---------------------

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К16\_4. Техника работы с классами.

Задание 1. Придумать собственный класс

Задание 2. Неформально описать функционал класса

Задание 3. Реализовать класс в модуле

Задание 4. Разработать скрипт для демонстрации работы с классом (импортировать модуль,

создать экземпляры, вызвать методы)

'''

class SubjectsHero:

"""Предметы героя"""

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

self.num\_subj = 0

self.damage = 0

self.helthy = 0

self.subjects = list()

def numSubj(self):

return self.num\_subj

def sword(self):

self.damage += 5

self.num\_subj += 1

self.subjects.append('sword')

def shield(self):

self.helthy += 10

self.num\_subj += 1

self.subjects.append('shield')

def printSubjectsHero(self):

print("Damage:", self.damage)

print("Helthy:", self.helthy)

print("Subjects hero:", end=' ')

for i in range(self.num\_subj):

print(self.subjects[i], end = ' ')

print()

def SubjectsHero(self):

return [self.damage, self.helthy, self.subjects]

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

class\_name = self.\_\_class\_\_.\_\_name\_\_

print(f"Класс \"{class\_name}\" удален")

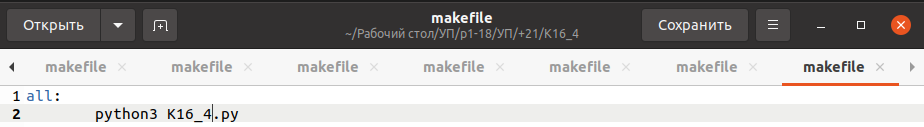


Рис. 107. makefile

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

## 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 108. Установщик PyCharm



Рисунок 109. Место установки PyCharm



Рисунок 110. Процесс установки PyCharm



Рисунок 111. 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 112. 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 113. 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 114. Написание программы

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



Рисунок 115. Запуск программы

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



Рисунок 116. Выполнение программы

# Техника работы с базами данных

Приложения: sql\_bd.py

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

Листинг 77. sql\_bd.py

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

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

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



Рис. 117. Установщик DB Browser

После установки, открываем DB Browser для работы с ним.

Рис. 118. Окно DB Browser

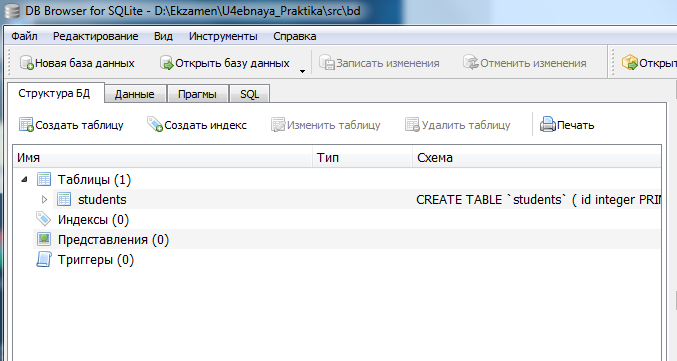
В открывшееся окно DB Browser’a перетаскиваем файл с БД.

Рис. 119. Таблица students в DB Browser

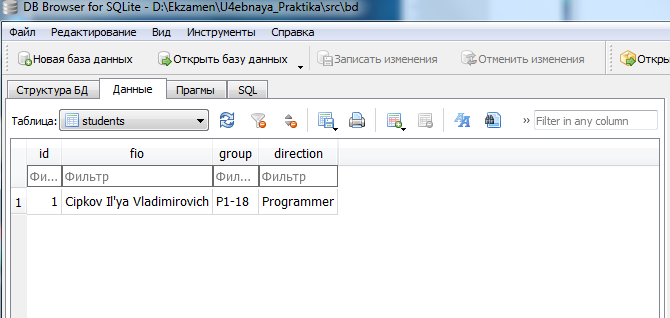
Чтобы просмотреть содержимое БД необходимо нажать на кнопку «Данные».

Рис. 120. Данные таблицы students

Видно, что в таблицу занеслись корректные данные.

# Техника работы с библиотекой tkinter

Приложения: MyTable.py

Библиотека Tkinter установлена в Python в качестве стандартного модуля, поэтому нам не нужно устанавливать что-либо для его использования. Чтобы импортировать его, пропишем import tkinter. В качестве примера для демонстрации tkinter будет приведена программа, которая принимает данные из формы и добавляет их в текстовый файл, также в программе есть возможность очищать весь файл полностью.

Листинг 78: MyTable.py

from tkinter import \*  
  
def main():  
 *#Функция save сохраняет введённые данные в строку и записывает строчку в файл* def save():  
 str = get\_name() + ' ' + group\_get() + ' ' + show\_parametrs() + m\_g()  
 with open("test.txt", "a") as f:  
 f.write(str + '\n')  
  
 *# Функция group\_get возвращает выбранную группу пользователем* def group\_get():  
 return my\_var.get()  
  
 *# Функция show\_parametrs возвращает все языки, которые выбрал пользователь* def show\_parametrs():  
 str = ""  
 for name, var in parametrs:  
 if var.get():  
 str += name + ' '  
 return str  
  
 *# Функция get\_name возвращает имя введённое пользователем* def get\_name():  
 return my\_var2.get()  
  
 *# Функция delete очищает текстовоый файл полностью* def delete():  
 f = open('test.txt', 'w+')  
 f.seek(0)  
 f.close()  
  
 *# Функция m\_g возвращает выбранный пол пользователем* def m\_g():  
 if v.get() == 1:  
 return "Мужской"  
 else:  
 return "Женский"  
  
 *# Создание объекта окна верхнего уровня (на нём будут распологаться все элементы* root = Tk()  
 *# Задание размера окна и его координат расположения* root.geometry("300x300+700+300")  
 *# StringVar используется для легкого отслеживания изменений в переменных tkinter* my\_var = StringVar('')  
 my\_var2 = StringVar('')  
 parametrs = (("Java", IntVar()), ("C#", IntVar()), ("C++", IntVar()),\  
 ("Python", IntVar()), ("Pascal", IntVar()))

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

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

Форма имеет вид:

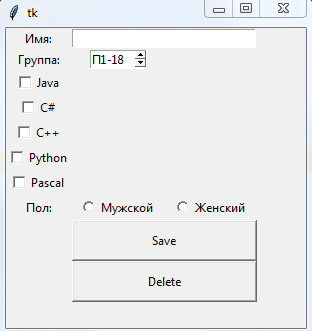


Рис. 121. Вид формы

Заполним форму данными:

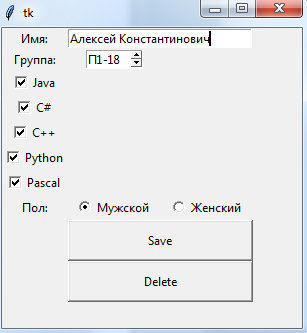


Рис. 122. Заполненная форма

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

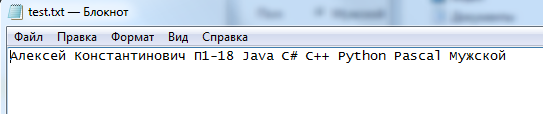


Рис. 123. Вывод в файл

# Техника работы с библиотекой NumPy

Приложения: primer.py

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

Установка:

• Через терминал командой: “pip install numpy”

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

Импорт модуля numpy:

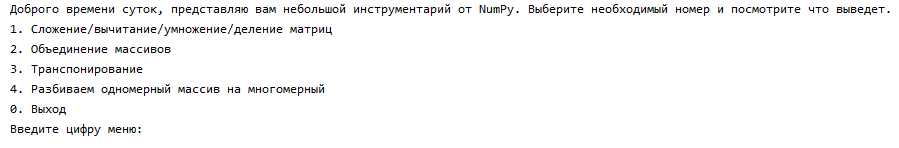
import numpy

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

Листинг 79: primer.py

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

main()

Рис. 124 Вывод меню в консоль

# Техника работы с библиотекой Matplotlib

# Элементы работы с библиотекой PyQt

# Элементы работы с библиотекой PyGame

Приложения: main.py

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

Листинг 82. main.py

import pygame

import random

class Hero:

    positions = list()

    colors = {

        'RED': (237, 77, 12),

        'PURPLE': (130, 12, 247),

        'TURQUOISE': (22, 224, 214),

        'YELLOW': (213, 247, 12),

        'ORANGE': (240, 121, 40)

    }

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

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

*self*.width = size[0]

*self*.height = size[0]

        else:

*self*.width = size[0]

*self*.height = size[1]

*self*.type\_hero = type\_hero

*self*.bgc = (125, 125, 125)

*self*.x = 100

*self*.y = 100

*self*.speed = 1

*self*.target = {'y': '', 'x': ''}

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

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

    def \_\_repr\_\_(self):

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

    def update\_positions(self):

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

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

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

        if pos == 'x':

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

        else:

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

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

        # Устанавливаем границы

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

            x\_start = x\_start + *self*.width

            x\_end = x\_end - *self*.width

            y\_start = y\_start + *self*.width

            y\_end = y\_end - *self*.width

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

            x\_end = x\_end - *self*.width

            y\_end = y\_end - *self*.height

*self*.mapp = {

            'x\_start':x\_start,

            'x\_end': x\_end,

            'y\_start': y\_start,

            'y\_end': y\_end

        }

    def set\_color(self, \*\*kwargs):

        if kwargs.get('hex') != None:

*self*.bgc = kwargs['hex']

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

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

        else:

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

    def set\_type(self, type\_hero):

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

            if *self*.width > *self*.height:

*self*.height = *self*.width

            else:

*self*.width = *self*.height

*self*.type\_hero = type\_hero

*self*.set\_map(\

*self*.mapp['x\_start'],

*self*.mapp['x\_end'],

*self*.mapp['y\_start'],

*self*.mapp['y\_end'],

        )

    def set\_speed(self, speed):

*self*.speed = speed

    def add\_speed(self, speed):

        if speed < 0:

*self*.sub\_speed(speed)

        else:

*self*.speed += speed

        print("speed: ", *self*.speed)

    def sub\_speed(self, speed):

        if speed < 0:

*self*.add\_speed(speed)

        else:

*self*.speed -= speed

            if *self*.speed < 0:

*self*.speed = 0

        print("speed: ", *self*.speed)

    def set\_pos(self, \*args):

        if args:

*self*.x = args[0]

*self*.y = args[1]

        else:

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

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

*self*.update\_positions()

    def draw(self, display):

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

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

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

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

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

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

    def set\_target(self, \*args):

        if not args:

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

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

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

*self*.target['x'] = args[0]

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

*self*.target['y'] = args[1]

        print(*self*.target)

    def navigation(self):

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

*self*.x -= *self*.speed

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

*self*.x += *self*.speed

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

*self*.y += *self*.speed

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

*self*.y -= *self*.speed

        # Уперся в другой объект

        is\_altered = False

        for pos in *self*.positions:

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

                is\_altered = True

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

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

*self*.target['x'] = 'right'

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

*self*.target['x'] = 'left'

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

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

*self*.target['y'] = 'bottom'

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

*self*.target['y'] = 'top'

        # Уперся в границу

        if not is\_altered:

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

*self*.target['x'] = 'right'

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

*self*.target['x'] = 'left'

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

*self*.target['y'] = 'bottom'

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

*self*.target['y'] = 'top'

*self*.update\_positions()

#

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

    global stop, mode

    operating\_speed = 0

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            pygame.quit()

            quit()

        elif mode == 'press':

            if event.type == pygame.KEYDOWN:

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

                    stop = not stop

                    print(stop)

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

                    mode = 'touch'

        elif mode == 'touch':

            if event.type == pygame.KEYDOWN:

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

                    stop = not stop

                    print(stop)

                if not stop:

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

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

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

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

                        hero.set\_speed(1)

                        hero.set\_pos()

                        hero.set\_target()

                        # hero.set\_pos(50, 100)

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

                        hero.set\_color()

                        heroes.append(hero)

                        del hero

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

                        operating\_speed = 1

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

                        operating\_speed = -1

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

                        mode = 'press'

    display.fill(bgc)

    if not stop:

        if mode == 'press':

            keys = pygame.key.get\_pressed()

            if keys[pygame.K\_SPACE]:

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

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

                hero.set\_speed(1)

                hero.set\_pos()

                hero.set\_target()

                hero.set\_color()

                heroes.append(hero)

                del hero

            elif keys[pygame.K\_UP]:

                operating\_speed = 1

            elif keys[pygame.K\_DOWN]:

                operating\_speed = -1

        for hero in heroes:

            if operating\_speed == 1:

                hero.add\_speed(1)

            elif operating\_speed == -1:

                hero.sub\_speed(1)

            hero.navigation()

            hero.draw(display)

    if operating\_speed != 0:

        operating\_speed = 0

    pygame.display.update()

if \_\_name\_\_ == '\_\_main\_\_':

    pygame.init()  # Инициализация окна

    # Работа с окном

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

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

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

    # bgc = (255, 255, 255)

    bgc = (40, 40, 40)

    # Размер карты

    map\_size = {

        "x\_start": 0,

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

        "y\_start": 0,

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

    }

    # Создаем героя в виде шарика

    heroes = list()  # Список, где будут все герои

    hero = Hero((25,), 'circle')  # Создаем героя

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

    hero.set\_speed(1)  # Скорость передвижения

    hero.set\_target()  # В какую сторону полетит

    hero.set\_pos()  # Позиция на карте

    hero.set\_color()  # Цвет героя

    heroes.append(hero)  # Добавляем в реест героев

    del hero

    game = True  # Статус игры

    stop = False  # Игра ли приостоновлена

    mode = "touch"  # Есть 2 мода "press" и "touch". Благодаря "press" функции будут выполняться при зажиме

    # А при "touch" функци будут срабатывать лишь от нажатия

    while game:

        Update(display, map\_size, heroes)

Начнем с конструкции main:

Сначала мы осуществляем соединение с библиотекой pygame:

pygame.init()

Далее мы проводим некую работу с окном

Размеры окна:

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

Создаем окно:

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

Устанавливаем имя окна:

    pygame.display.set\_caption("Bruh") # Задаем имя окну

Фон окна:

    bgc = (40, 40, 40) # Фон окна

Устанавливаем размер карты:

    map\_size = {

        "x\_start": 0,

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

        "y\_start": 0,

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

    }

Создаем героя

Список со всеми героями:

    heroes = list()  # Список, где будут все герои

Создаем героя (фигуру в виде шарика):

    hero = Hero((25,), 'circle')  # Создаем героя

Территориальные границы героя:

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

Скорость передвижения героя:

    hero.set\_speed(1)  # Скорость передвижения

В какую сторону направится герой:

    hero.set\_target()  # В какую сторону полетит

Позиция на карте:

    hero.set\_pos()  # Позиция на карте

Цвет героя:

    hero.set\_color()  # Цвет героя

Добавляем героя в реестр героев

    heroes.append(hero)  # Добавляем в реестр героев

Освобождаем переменную

    del hero

Системные переменные:

Статус игры. Запущена ли игра

game = True  # Статус игры

Пауза или Пользователь играет

    stop = False  # Игра ли приостановлена

Мод. Есть 2 мода "press" и "touch". При "press", каждое ваше нажатие воспринимается как зажатие, от этого вызванная вами функция будет выполняться многократно. А при "touch", вызов функции будет только один раз

    mode = "touch"

Далее мы запускаем нашу игру

while game:

        Update(display, map\_size, heroes)

Теперь идет функция Update где прописан наш основной функционал

Что делать со скоростью. Если 1, поднять скорость, если -1 опустить, если 0, то ничего не делать.

operating\_speed = 0

Запрашиваем данные о происходящих на данный момент событии, и кладем в переменную event.

for event in pygame.event.get()

Выход

if event.type == pygame.QUIT:

pygame.quit()

quit()

Дальше у нас идет разделение на 2 мода "press" и "touch". Если активный мод "touch", то для начала мы проверяем статус кнопки (опущена ли она):

if event.type == pygame.KEYDOWN

Далее, если пользователь нажал "escape", то игра приостановится

if event.key == pygame.K\_ESCAPE:

stop = not stop

print(stop)

Если игра не поставлена на паузу у нас доступен функционал, а именно:

Создание нового героя. При нажатии на пробел. Создаётся также, как и в конструкции main

if event.key == pygame.K\_SPACE:

hero = Hero((25,), 'circle')

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

map\_size["y\_start"], map\_size["y\_end"] )

hero.set\_speed(1)

hero.set\_pos()

hero.set\_target()

hero.set\_color()

heroes.append(hero)

del hero

Добавление или убавление скорости героя(-ев). При нажатии на стрелку вверх, скорость повышается, а при нажатии на стрелку вниз, скорость, соответственно, понижается.:

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

operating\_speed = 1

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

operating\_speed = -1

Смена мода. Происходит при нажатии на Tab. Меняется с "touch" на "press".

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

mode = 'press'

Следующий мод это "press"

При нажатии (не при зажиме) на Escape игра приостановится, а нажатии на Tab поменяется мод на "touch"

elif mode == 'press':

if event.type == pygame.KEYDOWN:

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

stop = not stop

print(stop)

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

mode = 'touch'

Идем дальше. Нужно вновь задать фон, ибо если мы не зададим фон, игра нам ничего не отобразит.

display.fill(bgc)

Наверняка вы заметили, что мы не полностью прописали функционал моду "press". Давайте сделаем это.

Мы должны быть убеждены что игра не приостановлена и что мод равен “press”

if not stop:

if mode == 'press':

Получаем все зажатые клавиши

keys = pygame.key.get\_pressed()

Ну а теперь по классике. Если зажат пробел, то новый герой (круг), а если зажата стрелка вверх или вниз, то добавление или убавление скорости.

if keys[pygame.K\_SPACE]:

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

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

hero.set\_speed(1)

hero.set\_pos()

hero.set\_target()

hero.set\_color()

heroes.append(hero)

del hero

elif keys[pygame.K\_UP]:

operating\_speed = 1

elif keys[pygame.K\_DOWN]:

operating\_speed = -1

Далее нам нужно обновить позиции каждого героя, и отрисовать их. А так же проверить было ли добавление или убавление скорости. При этом игра не приостановлена.

for hero in heroes:

if operating\_speed == 1:

hero.add\_speed(1)

elif operating\_speed == -1:

hero.sub\_speed(1)

hero.navigation() # Перемещение героя

hero.draw(display) # Отрисовка позиции героя

Теперь, нужно вернуть переменную являющаяся датчиком изменения скорости, вернуть в прежний вид.

if operating\_speed != 0:

operating\_speed = 0

И обновить экран

pygame.display.update()

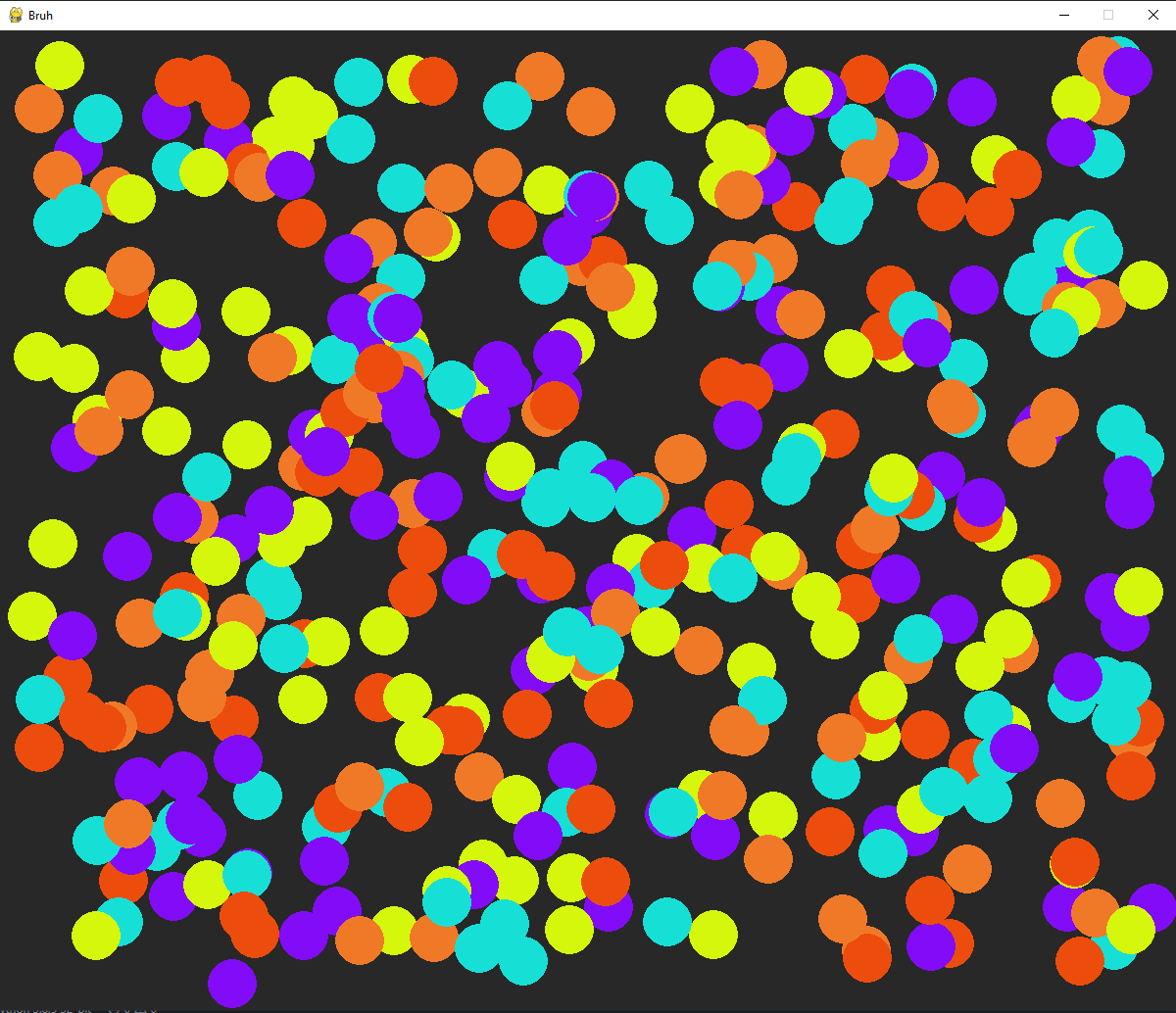


Рисунок 124. Скриншот игры

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

Приложения: visual.py, Cashier.py, sqlighter.py, bd

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

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

Чтобы импортировать его, пропишем

import tkinter

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

import sqlite3

Листинг 1. visual.py

from tkinter import \*

from tkinter import messagebox

import tkinter as tk

from datetime import datetime

from Cashier import Deal, Report

from sqlighter import SQLigter

class DocumentWorker(tk.Toplevel):

    def \_\_init\_\_(self):

        root = Toplevel()

*# self.title("Мини-ауф")*

        root.title("Работа с документом клиента")

*# self.geometry('450x500')*

*# Верхний отступ*

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

*# Текст окна*

        fio = deal.getFIO()

        text = fio['lastname'] + ' ' + fio['name'] + '\n' + fio['patronymic']

        tk.Label(root, text=text, font=(80), pady=20).pack(pady=7)

*# Кнопки в окне*

        self.btn\_print\_payment\_doc = tk.Button(root, text='Квитанция об оплате', command=self.print\_payment\_doc)

        self.btn\_file\_payment\_doc = tk.Button(root, text='Печатать квитанции об оплате', command=self.print\_file\_payment\_doc)

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

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

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

*# Позиционирование кнопок*

        self.btn\_print\_payment\_doc.pack(padx=30, ipadx=51)

        self.btn\_file\_payment\_doc.pack(ipadx=20)

        self.btn\_print\_report.pack(ipadx=72)

        self.btn\_file\_report.pack(ipadx=39)

        self.btn\_close.pack(ipadx=92)

*# Нижний отступ*

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

    def print\_payment\_doc(self):

        print("Квитанция об оплате:")

        deal.printPaymentDoc()

        messagebox.showinfo("Квитанция", deal.getInfoPaymentDoc())

        print()

    def print\_file\_payment\_doc(self):

        name\_file = deal.printFilePaymentDoc()

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

        print("Квитанция об оплате в файле: ", name\_file)

        print()

    def print\_report(self):

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

        cashier.printReport()

        print()

    def print\_file\_report(self):

        name\_file = cashier.createNameFile()

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

        print("Отчет в файле: ", name\_file)

        print()

class HelpCashier(tk.Toplevel):

    def \_\_init\_\_(self, parent):

        super().\_\_init\_\_(parent)

        self.label = tk.Label(self, text='New window')

        self.title("Окно ввода данных")

*# Имя документа*

        self.doc = StringVar()

        lbl\_doc = tk.Label(self, text="Имя документа:").grid(row=0, column=0, sticky=W, pady=10, padx=10)

        ent\_doc = tk.Entry(self, textvariable=self.doc).grid(row=0, column=1, columnspan=3, sticky=W+E, padx=10)

*# Оплата клиента*

        self.money = IntVar()

        lbl\_money = tk.Label(self, text="Оплата клиента:").grid(row=1, column=0, sticky=W, pady=10, padx=10)

        ent\_money = tk.Entry(self, textvariable=self.money).grid(row=1, column=1, columnspan=3, sticky=W+E, padx=10)

*# Нал/Безнал*

        self.payment = StringVar()

        lbl\_payment = tk.Label(self, text="Вид оплаты (нал/безнал):").grid(row=2, column=0, sticky=W, pady=5, padx=10)

        ent\_payment = tk.Entry(self, textvariable=self.payment, width=30).grid(row=2, column=1, columnspan=3, sticky=W+E, padx=10)

*# Кнопки*

        btn\_canel = tk.Button(self, text='Отменить', bg='red', command=self.destroy).grid(row=3, column=2, pady=10)

        btn\_submit = tk.Button(self, text='Принять', bg='lightgreen', command=self.accept\_data).grid(row=3, column=3, padx=10)

*# self.doc = 'doc.txt'*

*# self.money = 7000*

*# self.payment = 'нал'*

*#mainloop()*

    def accept\_data(self):

        data = {

            "doc": self.doc.get(),

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

            "payment": self.payment.get()

        }

*# data = {*

*#   "doc": 'doc.txt',*

*#   "money": 7000,*

*#   "payment": 'нал'*

*# }*

        try:

            global deal

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

*# Если не хватило денег*

            if deal.enoughMoney() == False:

                print()

                print("Недостаточно")

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

*# Денег хватает*

            else:

                print()

                print("Докумет принят")

                self.destroy()

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

                data['money'] = deal.money

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

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

*# Если его нет в бд с клиентами, добавляем*

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

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

                    s = ''

                    for key in d\_fio:

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

                    print("Добавлен новый клиент: ", s)

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

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

*# Добавляем в реестр заказов новый заказ*

                d = deal.getDataProduct()

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

                db.addOrderToRegister(\

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

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

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

                    payment\_method=deal.getPaymentMethod(), \

                    client\_id=client\_id, \

                    date=date, \

                    time=time

                )

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

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

                for x in data:

                    s += str(x) + ' | '

                print("Добавлен в реестр: ", s)

                print()

*# Работа с поступившем документом*

*#user.workWithDocuments(deal, cashier)*

                deal.setPaymentDoc()

                doc\_work = DocumentWorker()

*# doc\_work.grab\_set()*

        except:

            print("Error")

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

class App(tk.Tk):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title("Мини-кассир")

*# self.geometry('450x500')*

*# Верхний отступ*

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

*# Текст окна*

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

*# Кнопки в окне*

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

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

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

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

*# Позиционирование кнопок*

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

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

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

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

        self.is\_accept = False

*# Нижний отступ*

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

    def help\_cashier\_window(self):

        help\_cashier = HelpCashier(self)

        help\_cashier.grab\_set()

    def print\_report(self):

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

        cashier.printReport()

        print()

    def print\_file\_report(self):

        name\_file = cashier.createNameFile()

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

        print("Отчет в файле: ", name\_file)

        print()

cashier = Report(0, 0)

db = SQLigter("bd")

deal = None

if \_\_name\_\_ == '\_\_main\_\_':

    app = App()

    app.mainloop()

Листинг 2. Cashier.py

*# Разработчики:*

*#   Герасимов Дмитрий*

*#   Груздев Роман*

from datetime import datetime

from os import name

from pathlib import Path

class Deal:

    """Класс для работы с данными поступающими кассиру"""

*# Принять документ и деньги*

    def \_\_init\_\_(self, doc, money, payment\_method):

        self.doc = doc

        self.money = float(money)

        self.payment\_method = payment\_method

        self.data\_doc = self.getDataDoc()

    def \_\_str\_\_(self):

        return str(self.doc), str(self.money)

    def \_\_repr\_\_(self):

        return self.\_\_class\_\_.\_\_name\_\_ + '(' + str(self.doc) + ', ' + str(self.money) + ')'

*# Вернет данные документа в виде словаря*

    def getDataDoc(self):

*# self.payment\_doc = ""*

*#! Считываем данные из поданного документа*

        f = open(self.doc, 'r')

*# Сами данные*

        d\_data = dict()

        name = ''

        for s in f:

*# Удаление '\n' с конца*

            if s[-1] == '\n':

                s = s[:-1]

*# Если строка не пустая*

            if s != '':

*# Если нам поступило новое имя*

                if (s[0] != '\t') and (s[-1] == ':'):

                    name = s[:-1]

                else:

                    if name != '':

*# Если словаря с таким именем не существует, создаем*

                        if d\_data.get(name, None) == None:

                            d\_data[name] = {}

                        l = [x.strip() for x in s.split(':')]   *# Удаляем ['\t', '\n', ' '] и делим по ':'*

                        if len(l) > 1:

                            if s[0] == '\t':

                                d\_data[name][l[0]] = l[1]

                            else:

                                d\_data[l[0]] = l[1]

        return d\_data

    def getFIO(self):

        """Вернет ФИО пользователя"""

        return {

            'lastname': self.data\_doc['Контакты']['Фамилия'],

            'name': self.data\_doc['Контакты']['Имя'],

            'patronymic': self.data\_doc['Контакты']['Отчество']

        }

    def getPhoneNumber(self):

        """Вернет номер телефона"""

        return self.data\_doc['Контакты']['Номер телефона']

    def getPaymentMethod(self):

        """Вернет тип оплаты (нал/безнал)"""

        return self.payment\_method

    def getDataProduct(self, \*args):

        """Вернет данные о товаре"""

        if (len(args) == 0) or (args[0] == dict):

*#d = dict()*

            print(self.data\_doc['О товаре'])

            d = {

                'product': self.data\_doc['О товаре']['Товар'],

                'number': self.data\_doc['О товаре']['Количество'],

                'unit\_price': self.data\_doc['О товаре']['Цена за единицу'],

                'total\_price': self.data\_doc['О товаре']['Итоговая цена'],

            }

            return d

        elif args[0] == list:

            return list(self.data\_doc['О товаре'].values())

*# Хватает ли денег*

    def enoughMoney(self):

        if (self.money >= float(self.data\_doc['О товаре']['Итоговая цена'].split()[0])):

            return True

        return False

*# Обновляет сумму денег*

    def updateMoney(self):

        self.money = float(self.data\_doc['О товаре']['Итоговая цена'].split()[0])

*# Создать квитанцию*

    def setPaymentDoc(self):

*#! Устанавливаем дату оплаты*

        today = str(datetime.today()).split()[0]

        self.data\_doc.pop('Дата составления', None)

        self.data\_doc['Дата оплаты'] = today

        self.data\_doc['Статус'] = "ОПЛАЧЕНО"

    def setInfoPaymentDoc(self):

*#! Заполняем квитанцию*

        message = "Квитанция об оплате\n" +\

            '\n' +\

            "Статус: " + self.data\_doc["Статус"] + '\n' +\

            '\n' +\

            'Контакты:' + '\n'

*# Добавляем контактные данные*

        contacts = ['Фамилия', 'Имя', 'Отчество', 'Номер телефона']

        for name in contacts:

*#message += f"\t{name}: {self.data\_doc['Контакты'][name]}\n"*

            message += "\t" + name + ": " + str(self.data\_doc['Контакты'][name]) + "\n"

*# Добавляем данные о товаре*

        message += "\n" +\

            "О товаре:" + '\n'

        about\_product = ['Товар', 'Количество', 'Цена за единицу', 'Итоговая цена']

        for name in about\_product:

*#message += f"\t{name}: {self.data\_doc['О товаре'][name]}\n"*

            message += "\t" + name + ": " + str(self.data\_doc['О товаре'][name]) + "\n"

        message += '\n' +\

            "Дата оплаты: " + self.data\_doc['Дата оплаты']

        self.info = message

*# Печатает на экран данные квитанции*

    def printPaymentDoc(self, \*args):

        info = self.getInfoPaymentDoc()

*#! Вывод на экран*

        if len(args) == 0:

            print(info)

*#! Вывод в файл(-ы)*

        else:

            for name\_f in args:

                f = open(name\_f, 'w')

                f.write(info)

*# Вернет квитанцию об оплате. Печать документу поставлена*

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

        """Вернет квитанцию об оплате"""

        if kwargs.get(name) == None:

*#! Создаем имя документа*

            name\_payment\_doc = "Квитанция " + \

                self.data\_doc['Контакты']['Фамилия'] + '.' +\

                self.data\_doc['Контакты']['Имя'][0] + '.' +\

                self.data\_doc['Контакты']['Отчество'][0] + ' ' +\

                '[' + self.data\_doc['Дата оплаты'] + ']' +\

                ".txt"

*# Проверка на существование такого файла*

            count = 1

            while(Path(name\_payment\_doc).is\_file()):

                index\_end = name\_payment\_doc.index('.txt')

                if (name\_payment\_doc[-5] == ')'):

                    index\_end = name\_payment\_doc.index(']')+1

*#name\_payment\_doc = name\_payment\_doc[:index\_end] + f'({count}).txt'*

                name\_payment\_doc = name\_payment\_doc[:index\_end] + "(" + str(count) + ").txt"

                count += 1

        else:

            name\_payment\_doc = kwargs['name']

*# Печатаем в файл*

        self.printPaymentDoc(name\_payment\_doc)

        return name\_payment\_doc

    def getInfoPaymentDoc(self):

        if not hasattr(self, 'info'):

            self.setInfoPaymentDoc()

        return self.info

class Report:

    def \_\_init\_\_(self, money\_cash, money\_non\_cash):

        self.money\_cash = float(money\_cash)

        self.money\_non\_cash = float(money\_non\_cash)

    def add(self, money, payment):

        """Плюс к деньгам"""

        if payment == 'нал':

            self.money\_cash += money

        else:

            self.money\_non\_cash += money

    def difference(self, money, payment):

        """Минус от денег"""

        if payment == 'нал':

            self.money\_cash -= money

        else:

            self.money\_non\_cash -= money

    def getReport(self):

        """Возращает итоговый отчет"""

        return {

            'cash': self.money\_cash,

            'non-cash': self.money\_non\_cash,

            'state': self.money\_cash + self.money\_non\_cash

            }

    def createNameFile(self):

        """Создаем имя документа"""

        name\_payment\_doc = "Отчет кассира " + \

            '[' + str(datetime.today()).split()[0] + ']' +\

            ".txt"

*# Проверка на существование такого файла*

        count = 1

        while(Path(name\_payment\_doc).is\_file()):

            index\_end = name\_payment\_doc.index('.txt')

            if (name\_payment\_doc[-5] == ')'):

                index\_end = name\_payment\_doc.index(']')+1

*#name\_payment\_doc = name\_payment\_doc[:index\_end] + f'({count}).txt'*

            name\_payment\_doc = name\_payment\_doc[:index\_end] + "(" + str(count) + ").txt"

            count += 1

        self.printReport(name\_payment\_doc)

        return name\_payment\_doc

    def getFormReport(self):

        day = str(datetime.today()).split()

        message = "Отчет продавца:\n" +\

            "Дата: " + day[0] + '\n' +\

            "Время: " + day[1] + '\n' +\

            "Денег наличными: " + str(self.money\_cash) + '\n' +\

            "Денег безналичными: " +  str(self.money\_non\_cash) + '\n' +\

            "Общее количество: " + str(self.money\_cash + self.money\_non\_cash)

        return message

    def printReport(self, \*args):

        """Вывод отчета на экран или в файл"""

        info = self.getFormReport()

*#! Вывод на экран*

        if len(args) == 0:

            print(info)

*#! Вывод в файл(-ы)*

        else:

            for name\_f in args:

                f = open(name\_f, 'w')

                f.write(info)

Листинг 3. sqlighter.py

import sqlite3

from sqlite3.dbapi2 import connect

class SQLigter:

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

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

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

        self.cursor = self.connection.cursor()

*# ! Обобщенные функции (для всех таблиц)*

    def translateData(self, args):

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

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

            return args[0].values()

        elif type(args) == dict:

            return args.values()

        return args

    def getLastId(self, table):

        """Вернет последний индекс"""

        with self.connection:

            try:

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

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

            except Exception:

                return 0

*# ! Работа с таблицей данных клиента `clients`*

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

        """Добавляем нового пользователя в базу"""

        with self.connection:

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

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

            print(fio)

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

            self.connection.commit()

            return result

    def getIdClient(self, kwargs):

        """Вернет id клиента"""

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

        with self.connection:

            return self.cursor.execute(\

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

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

            ).fetchall()[0][0]

    def getClientMoney(self, id\_client):

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

        with self.connection:

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

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

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

        with self.connection:

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

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

            return amount\_summ

    def clientExists(self, fio):

        """Существует ли клиент"""

        with self.connection:

            try:

                result = self.cursor.execute(\

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

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

                ).fetchall()

            except:

                result = ''

            return bool(len(result))

*# ! Работа с таблицей реестра заказов `registry`*

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

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

        with self.connection:

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

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

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

            kwargs['id'] = need\_id

            result = self.cursor.execute(\

                "INSERT INTO `registry` (\

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

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

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

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

            ).fetchall()

            self.connection.commit()

            return result

# 3.1 Изучение входной и выходной документации

Входные данные:

1. Документ на оплату
   1. Контактные данные
      1. Фамилия
      2. Имя
      3. Отчество
      4. Номер телефона
   2. Данные о товаре
      1. Товар
      2. Количество
      3. Цена за единицу товара
      4. Итоговая цена
   3. Дата составления
2. Сумма денег клиента на оплату
3. Вид оплаты (нал/безнал)
4. База данных

Выходные данные

1. Квитанцию об оплате
   1. Статус оплаты
   2. Контактные данные
      1. Фамилия
      2. Имя
      3. Отчество
      4. Номер телефона
   3. Данные о товаре
      1. Товар
      2. Количество
      3. Цена за единицу товара
      4. Итоговая цена
   4. Дата оплаты
2. Отчёт кассира
   1. Дата
   2. Время
   3. Денег наличными
   4. Денег безналичными
   5. Общее количество
3. База данных

# 3.2 Разработка требований к проекту. Построение диаграммы использования.

Требования к проекту:

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

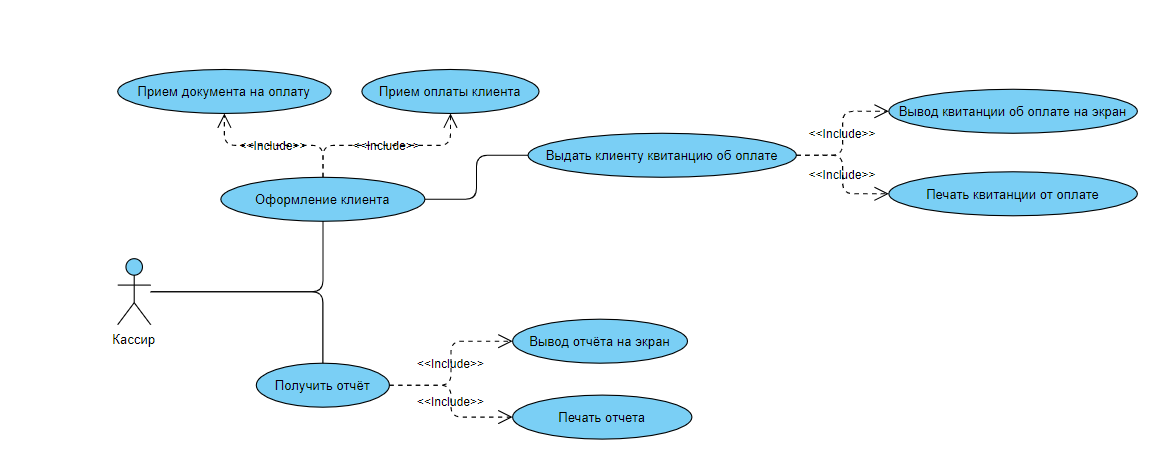


Рисунок 125. Диаграмма использования