

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 студент:

Герасимов Д. А., Груздев Р. И.

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

Гусятинер Л. Б.

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

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

Королев, 2019

Содержание отчёта

[**Раздел 1. Техника решения задач с использованием структурного программирования.** 3](#_Toc44425580)

[1.1. Установка интерпретатора Python 3 и настройка окружения 3](#_Toc44425581)

[1.2. Техника работы в командной строке и среде IDLE 8](#_Toc44425582)

[1.3. Техника работы с линейными и разветвляющимися программами 10](#_Toc44425583)

[1.4. Техника работы с циклическими программами, цикл while 13](#_Toc44425584)

[1.5. Техника работы с числами 16](#_Toc44425585)

[1.6. Техника работы со строками 19](#_Toc44425586)

[1.7. Техника работы со списками 22](#_Toc44425587)

[1.8. Техника работы с циклом for и генераторами списков 24](#_Toc44425588)

[1.9. Техника работы с функциями 27](#_Toc44425589)

[1.10. Техника работы со словарями 31](#_Toc44425590)

[1.11. Техника работы с множествами 35](#_Toc44425591)

[1.12. Техника работы с кортежами 37](#_Toc44425592)

[1.13. Техника работы с файлами 38](#_Toc44425593)

[**Раздел 2. Техника решения задач с использованием модульного и объектно-ориентированного программирования.** 45](#_Toc44425594)

[2.1. Техника работы с модулями 45](#_Toc44425595)

[2.2. Разработка проекта «Журнал посещения занятий» 47](#_Toc44425596)

[2.3. Установка и настройка среды JetBrains PyCharm 50](#_Toc44425597)

[2.4. Техника работы с библиотекой tkinter 57](#_Toc44425598)

[2.5 Доработка проекта «Журнал посещения занятий» с использованием графического интерфейса 61](#_Toc44425599)

[2.6. Техника работы с классами 67](#_Toc44425600)

[2.7. Доработка проекта «Журнал посещения занятий» с использованием классов 70](#_Toc44425601)

[2.8. Разработка документации по проекту 75](#_Toc44425602)

[**Раздел 3. Техника решения задач с использованием библиотек.** 82](#_Toc44425603)

[3.1. Техника работы с библиотекой NumPy 82](#_Toc44425604)

[3.2. Техника работы с библиотекой Matplotlib 86](#_Toc44425605)

[3.3 Техника работы с библиотекой PyQT 92](#_Toc44425606)

[3.4. Техника работы с библиотекой Pygame 98](#_Toc44425607)

[Список использованных источников 106](#_Toc44425608)

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

## Установка интерпретатора Python 3 и настройка окружения

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



Рисунок 1. Официальный сайт Python

Порядок установки на Windows:

1. Запустить скачанный установочный файл.

2. Выбрать способ установки.



Рисунок 2. Установщик Python

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



Рисунок 3. Опции установки

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

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

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



Рисунок 4. Продвинутые опции установки

5. После успешной установки:



Рисунок 5. Сообщение об установке

## Техника работы в командной строке и среде IDLE

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



Рисунок 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



Рисунок 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))

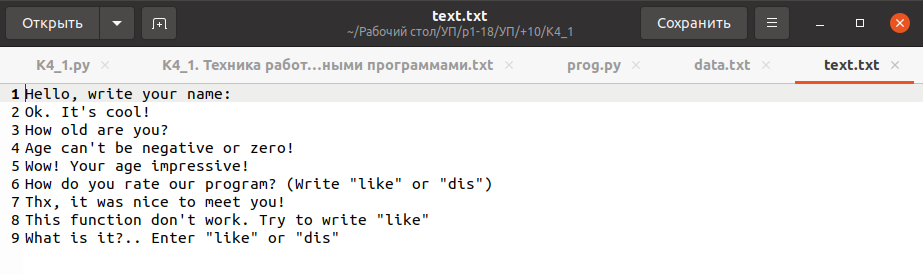
i += 1 # 6

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

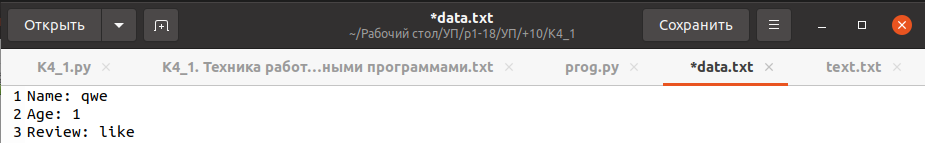
else:

i += 3

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

i -= 3

Рисунок 8. Входные данные text.txt

Рисунок 9. Исходные данные data.txt

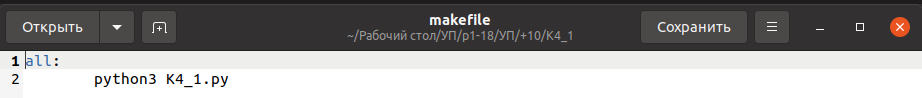


Рисунок 10. Мейкфайл makefile

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] + ' ' +

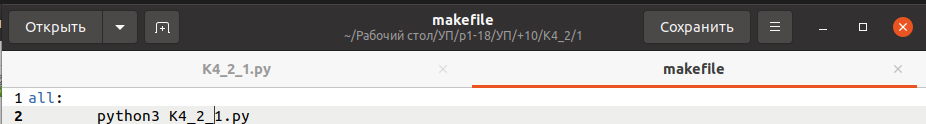
year\_list\_th[date\_y[0]] + ' ' + year\_list\_hun[date\_y[1]] + ' ' + year\_list\_dec[date\_y[2]] + ' ' + year\_list\_unit[date\_y[3]] + ' ' + ' года')

date = input()

while(date != 'stop'):

print(get\_date(date))

date = input()

Рисунок 11. Мейкфайл 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!')

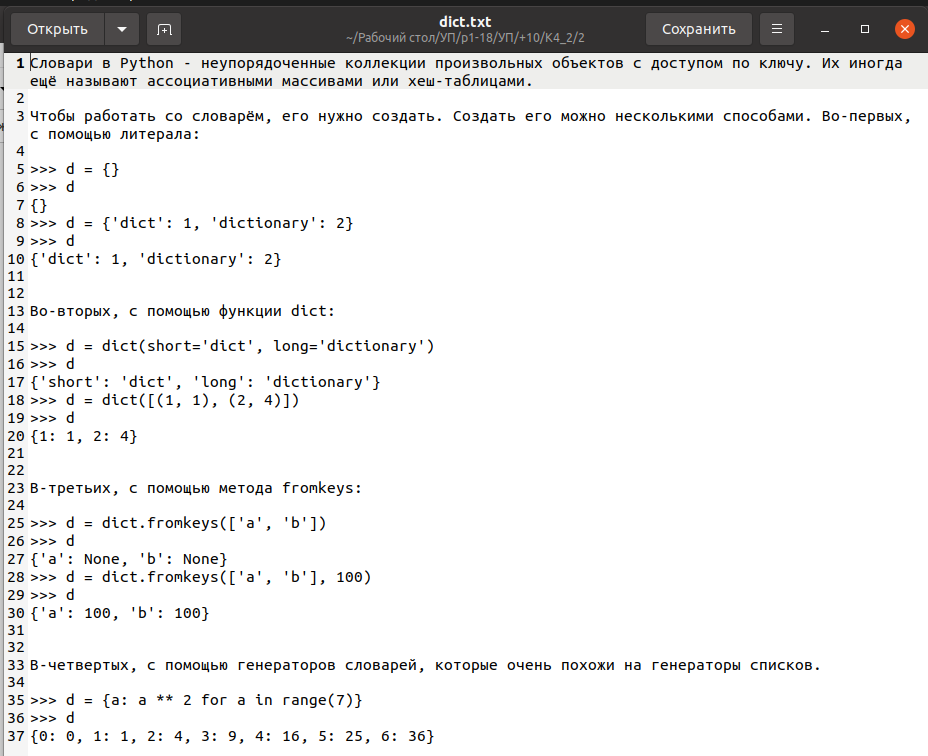
exit(0)

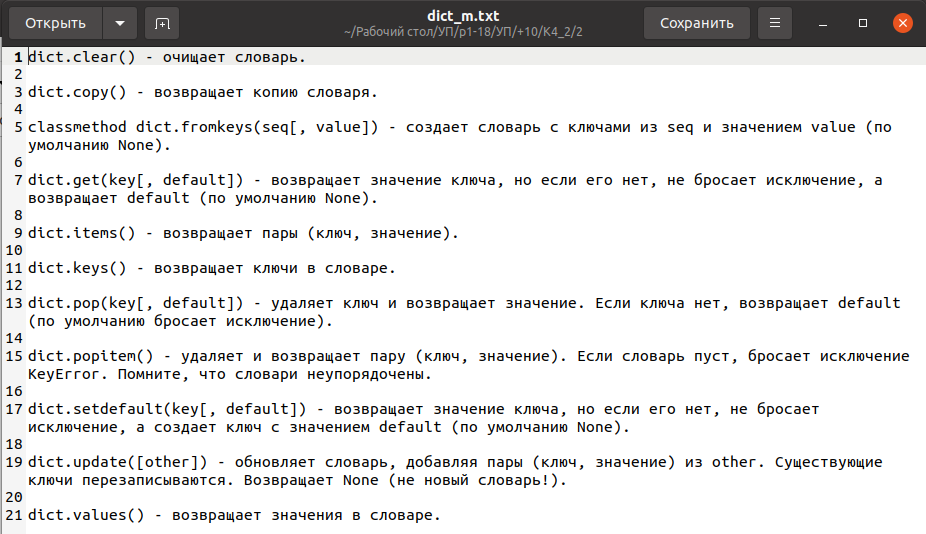
else:

while(a < 0 or a > 3):

print("Попробуйте ввести корректный пункт меню:")

a = int(input())

Рисунок 12. файл для вывода dict.txt

Рисунок 13. файл для вывода dict\_m.txt

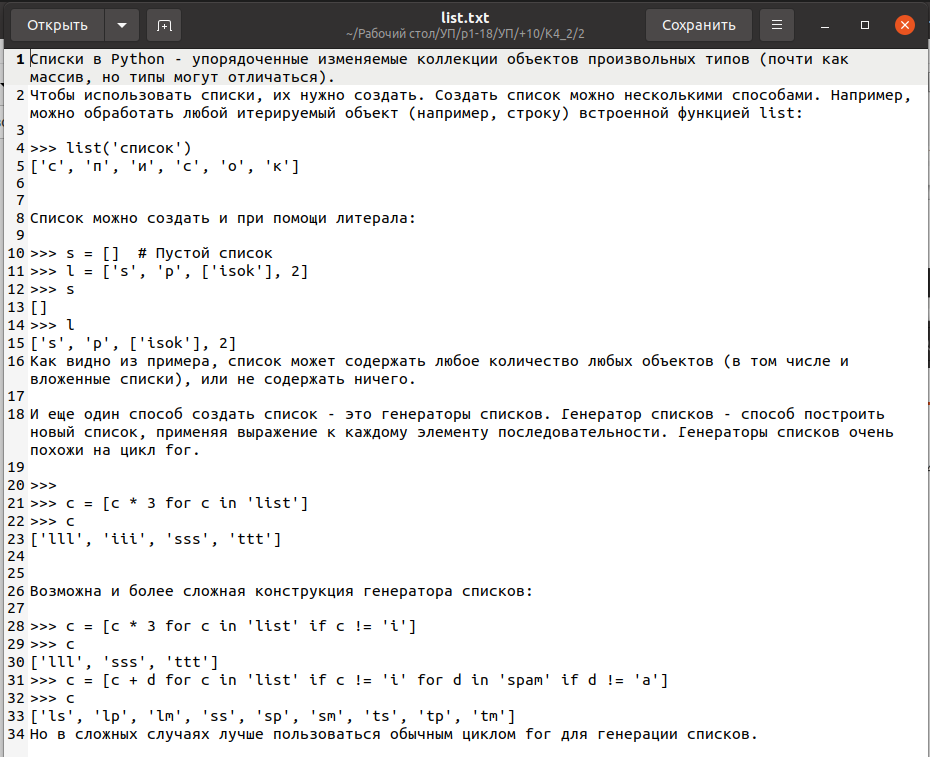
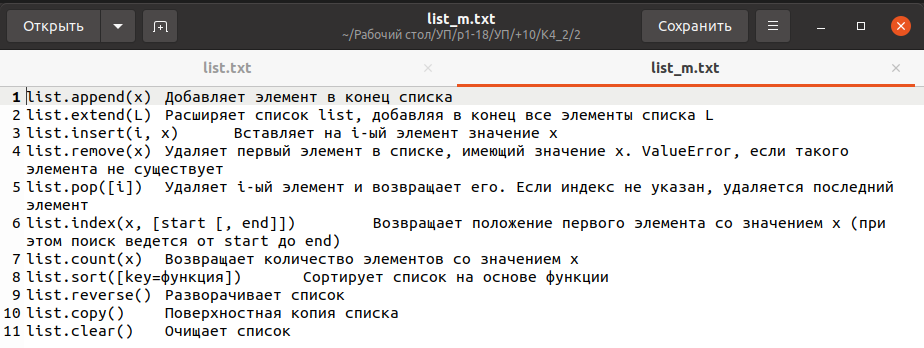
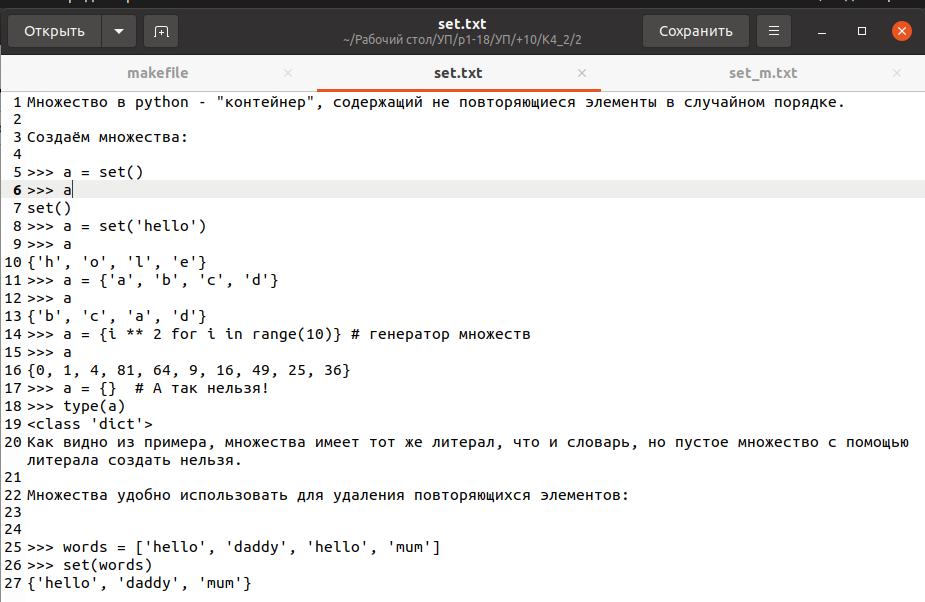
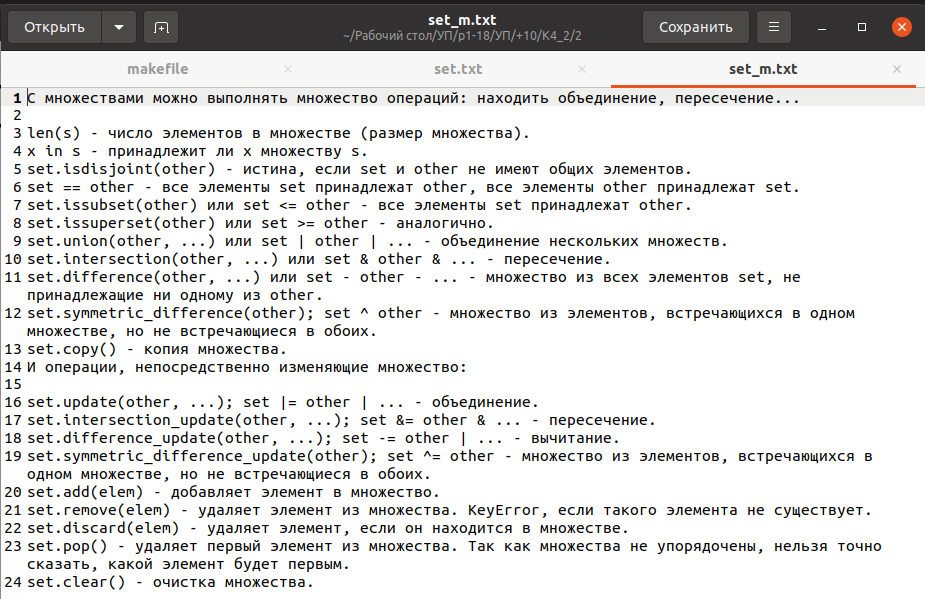
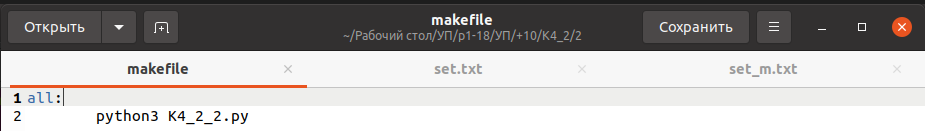


Рисунок 14. файл для вывода list.txt

Рисунок 15. list\_m.txt

Рисунок 16. set.txt

Рисунок 17. set\_m.txt

Рисунок 18. makefile

## Техника работы с циклическими программами, цикл while

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

Листинг 4. K5\_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

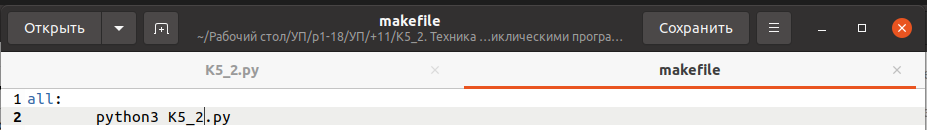
sum\_n = 0

while (n != 0):

n = int(input())

sum\_n += n

print(sum\_n)

Рисунок 19. makefile

Листинг 5. K5\_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:

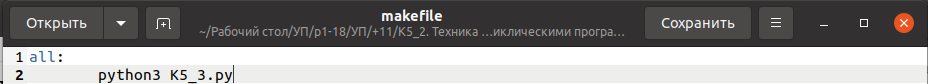
return max(a, b)

return nod(b % a, a)

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

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

print( nod(a, b) )

Рисунок 20. makefile

Листинг 6. K5\_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\": ")

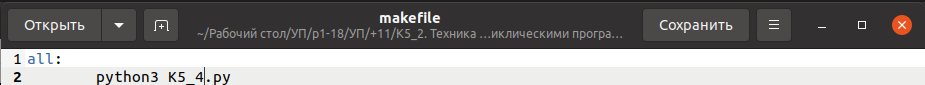
b = int(input())

v\_lcm = lcm(a, b)

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

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

main()

Рисунок 21. makefile

Листинг 7. K5\_4.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--}

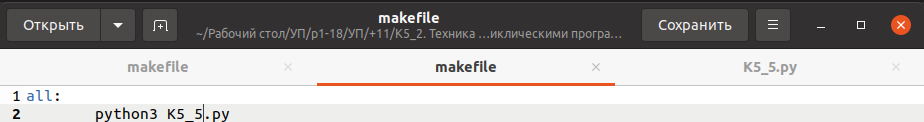
def main():

n = int(input())

PartDigits(n)

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

main()

Рисунок 22. makefile

## Техника работы с числами

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

Листинг 8. 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):

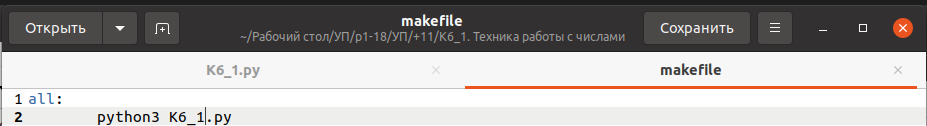
roundingDec()

n\_loc = -1

elif (n == 2):

whatForFrac()

n\_loc = -1

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

main()

Рисунок 23. makefile

## Техника работы со строками

Приложения: K6\_2\_1.py K6\_2\_2.py

Листинг 9. 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)

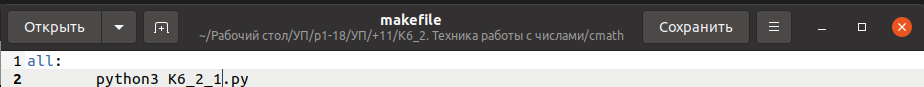
print("cmath.phase(complex(1.0, 1.0)) =", cmath.phase(complex(1.0, 1.0)) ) # returns 0.7853981633974483

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()

Рисунок 24. makefile

Листинг 10. 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

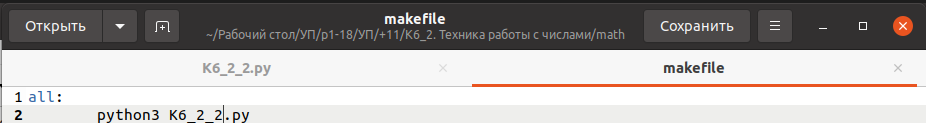
print("math.log(1.0000025) =", math.log(1.0000025)) # returns 2.4999968749105643e-06

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

print()

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

main()

Рисунок 25. makefile

## Техника работы со списками

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

Листинг 11. K7\_1\_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

К7\_1. Техника работы со строками;

Задание 1.

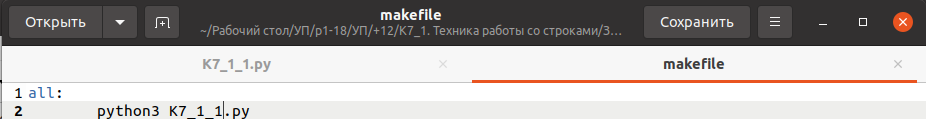
https://stepik.org/lesson/201702/step/5?unit=175778

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')

Рисунок 26. makefile

Листинг 12. 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

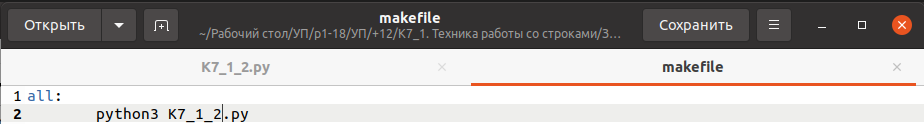
могла сгодиться для логина. Если строка состоит из одного слова, менять ничего не нужно.

Sample Input: python sila

Sample Output: python\_sila

'''

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

Рисунок 27. makefile

Листинг 13. 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('.')]

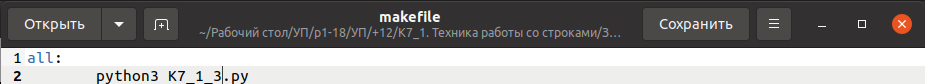
print(\*a, '\n'+str(sum(a)), sep=' ')

Рисунок 28. makefile

Листинг 14. 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')

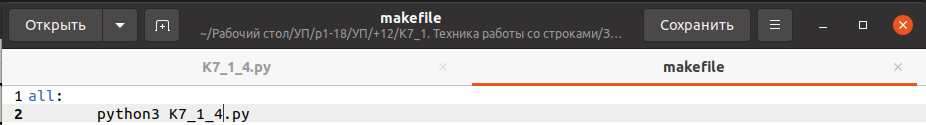


Рисунок 29. makefile

Листинг 15. 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(

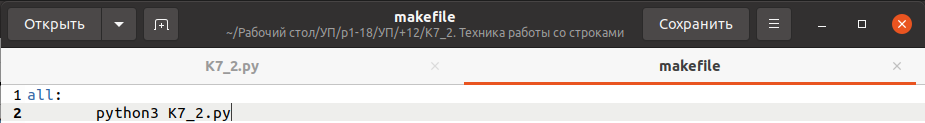
name=name, error=hex(errno)

)

)

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

print()

Рисунок 30. makefile

## Техника работы с циклом for и генераторами списков

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

Листинг 16. 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()]

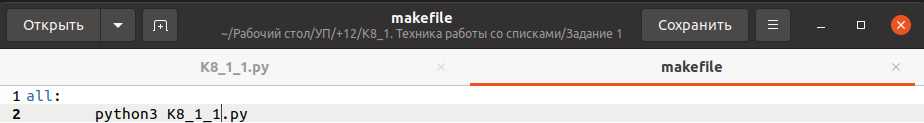
a = 0

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

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

a += 1

print(a)

Рисунок 31. makefile

Листинг 17. 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

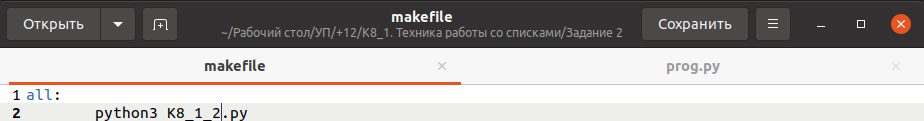
print(a/2)

Рисунок 32. makefile

Листинг 18. 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)

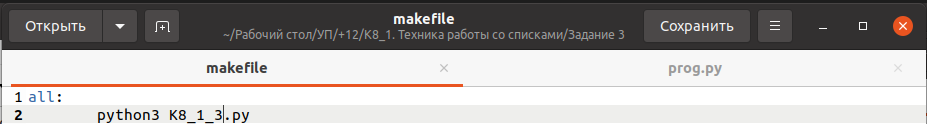
print("Список значений, попадающие хотя бы в один из N списков:\n", \*u\_all)

Рисунок 33. makefile