



Государственное бюджетное образовательное учреждение высшего образования
Московской области

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

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)

Содержание отчёта

Раздел 1. Техника решения задач с использованием структурного и объектно-ориентированного программирования.	2
1.1 Установка интерпретатора Python 3 и настройка окружения	2
1.2 Техника работы в командной строке и среде IDLE.....	5
1.3 Техника работы с линейными и разветвляющимися программами	7
1.4 Техника работы с циклическими программами, цикл while	16
1.5 Техника работы с числами.....	24
1.6 Техника работы со строками	31
1.7 Техника работы со списками	36
1.8 Техника работы с циклом for и генераторами списков	41
1.9 Техника работы с функциями	48
1.10 Техника работы со словарями	54
1.11 Техника работы с множествами	64
1.12 Техника работы с кортежами.....	72
1.13 Техника работы с файлами	78
1.14 Техника работы с модулями	86
1.15 Техника работы с классами.....	102

Раздел 1. Техника решения задач с использованием структурного и объектно-ориентированного программирования.

1.1 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/>

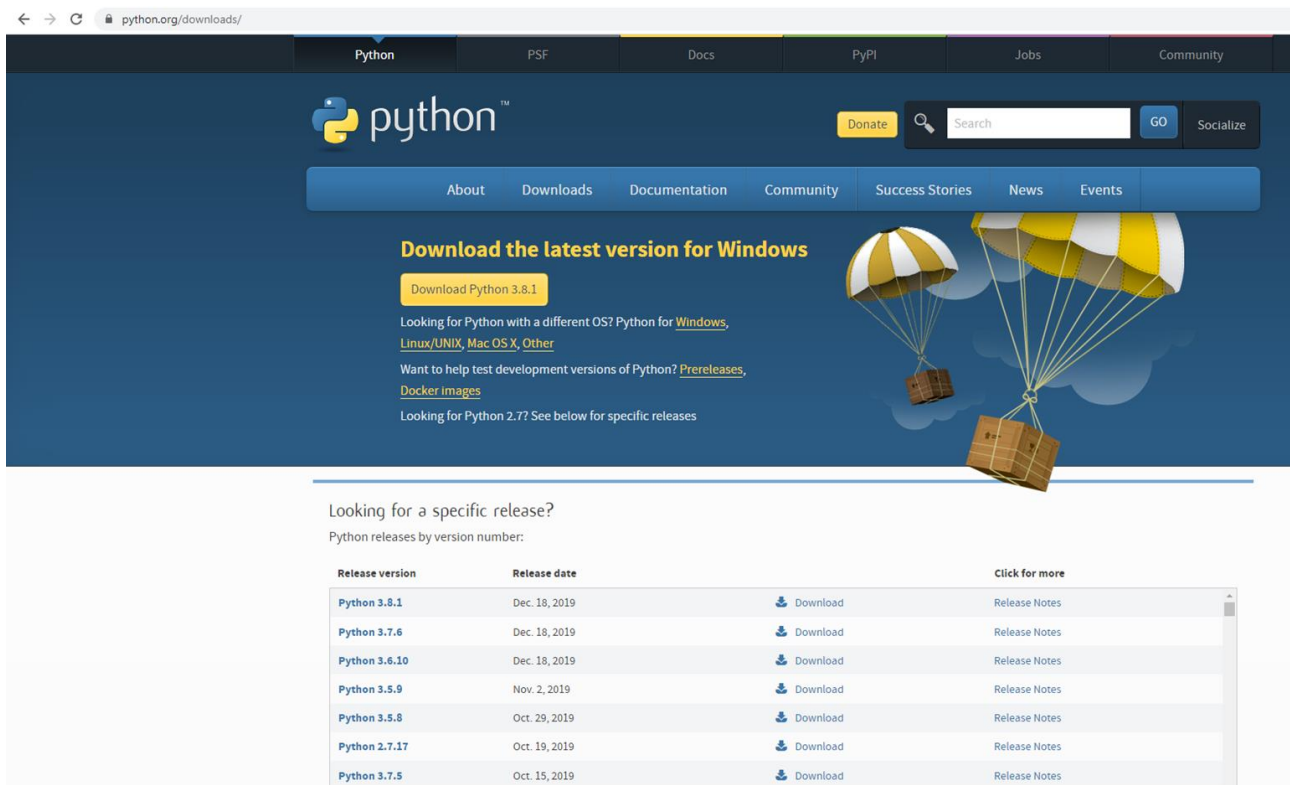


Рис. 1. Официальный сайт Python

Порядок установки на Windows:

1. Запустить скачанный установочный файл.
2. Выбрать способ установки.

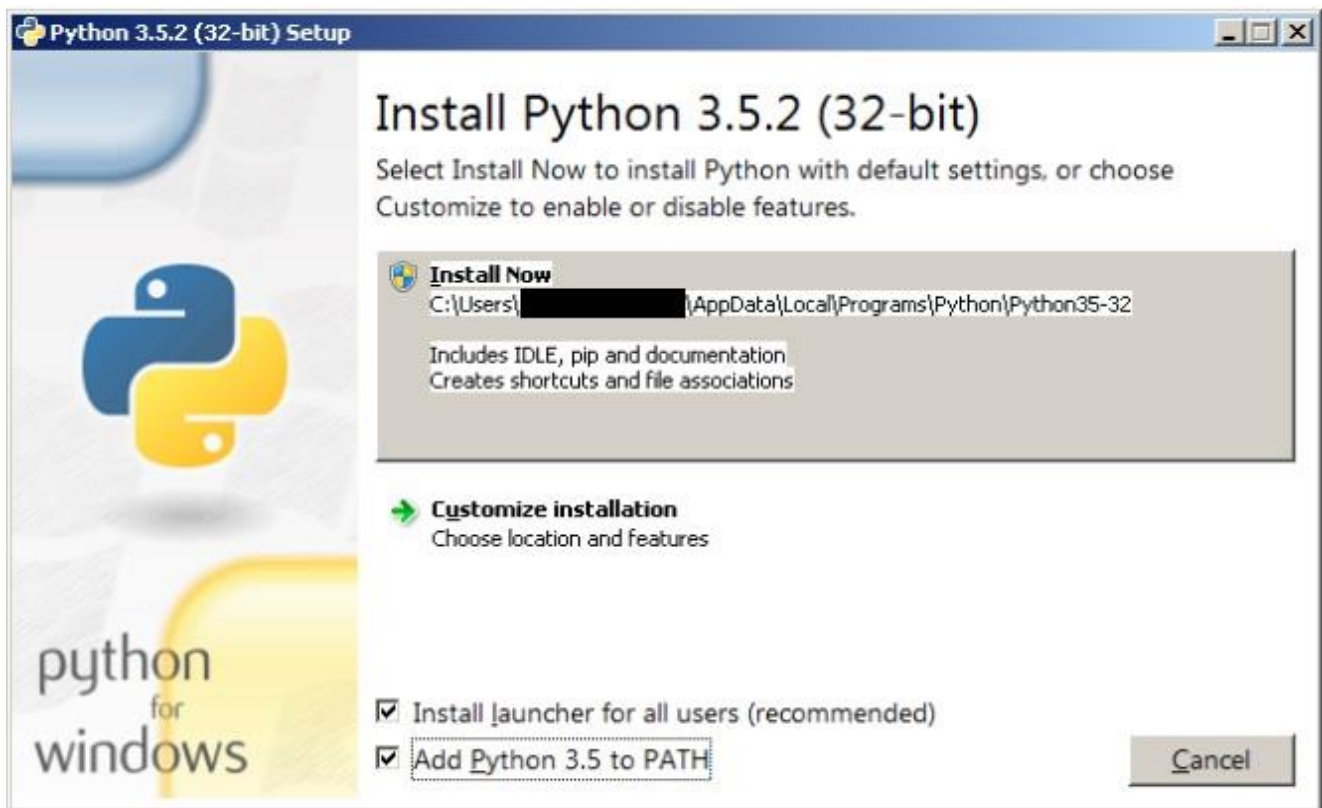


Рис. 2. Установщик Python

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)

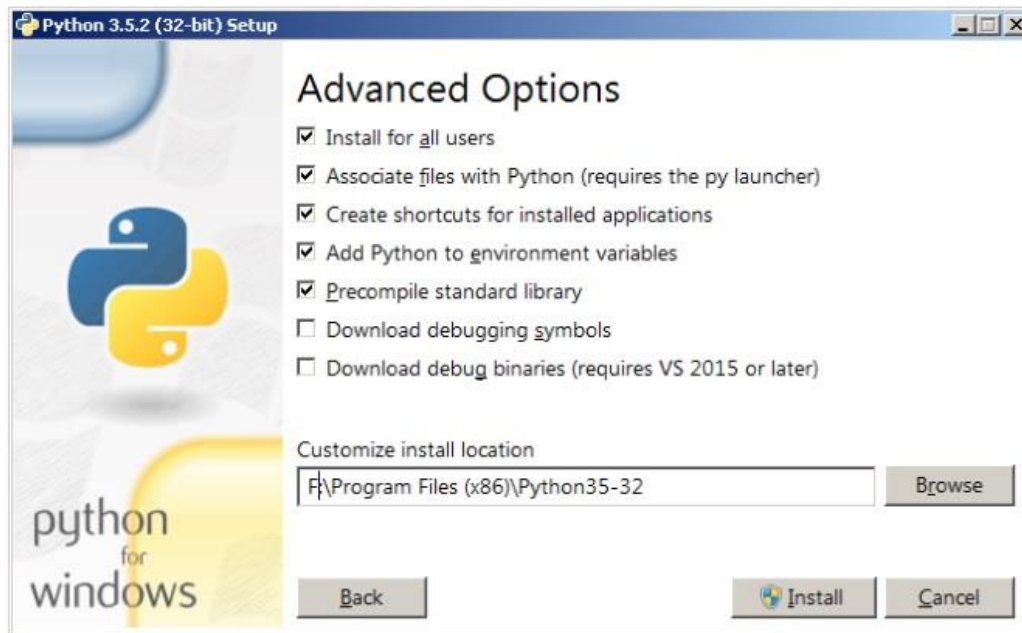


Рис. 4. «Продвинутые» опции установки

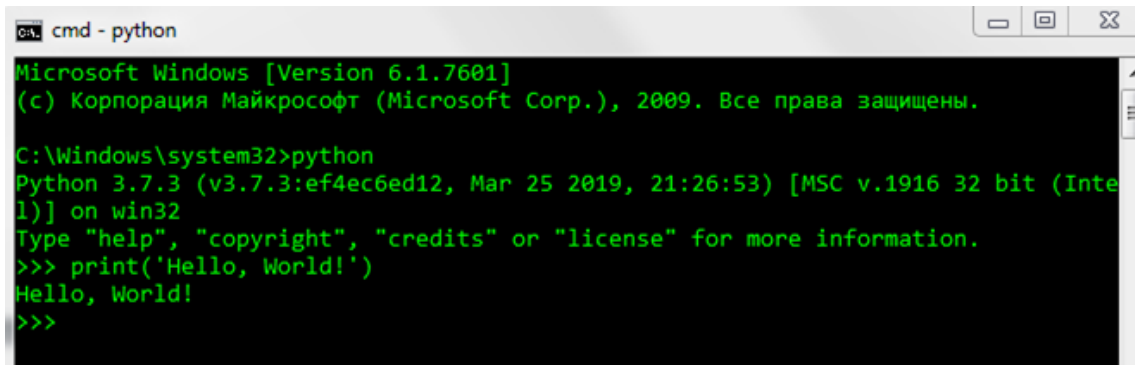
5. После успешной установки:



Рис. 5. Сообщение об успешной установке

1.2 Техника работы в командной строке и среде IDLE

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



```
cmd - python
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Windows\system32>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello, World!')
Hello, World!
>>>
```

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 - 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
```

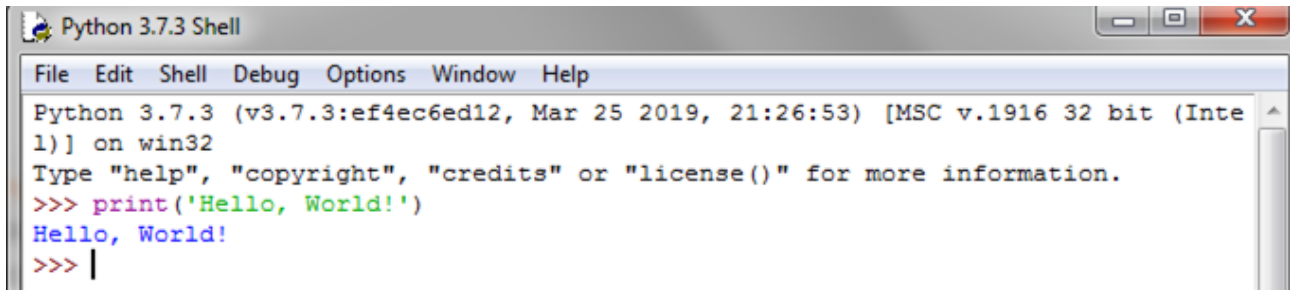


Рис. 7. Первая программа

1.3 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
'''

K4_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
                                     # текстового файла 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
    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

```

```

1 Hello, write your name:
2 Ok. It's cool!
3 How old are you?
4 Age can't be negative or zero!
5 Wow! Your age impressive!
6 How do you rate our program? (Write "like" or "dis")
7 Thx, it was nice to meet you!
8 This function don't work. Try to write "like"
9 What is it?.. Enter "like" or "dis"

```

Рис. 8. text.txt

```

1 Name: qwe
2 Age: 1
3 Review: like

```

Рис. 9. data.txt

```

1 all:
2     python3 K4_1.py

```

Рис. 10. makefile

Листинг 2. K4_2_1.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

K4_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',
                'двадцать 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', 'в
осемьсот', 'девятьсот']

    year_list_hun2 =
    ['', 'сотого', 'двухсотого', 'трёхсотого', 'четырёхсотого', 'пятьсотого', 'шест
ьсотого', '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()

```

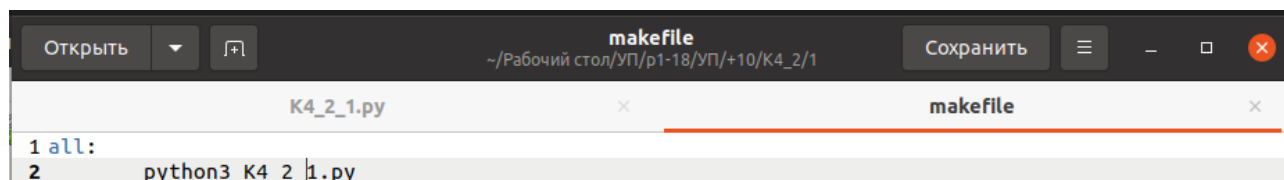


Рис. 11. makefile

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
'''
```

K4_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 список      |}')
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())

```

```

1 Словари в Python - неупорядоченные коллекции произвольных объектов с доступом по ключу. Их иногда
ещё называют ассоциативными массивами или хеш-таблицами.
2
3 Чтобы работать со словарём, его нужно создать. Создать его можно несколькими способами. Во-первых,
с помощью литерала:
4
5 >>> d = {}
6 >>> d
7 {}
8 >>> d = {'dict': 1, 'dictionary': 2}
9 >>> d
10 {'dict': 1, 'dictionary': 2}
11
12
13 Во-вторых, с помощью функции dict:
14
15 >>> d = dict(short='dict', long='dictionary')
16 >>> d
17 {'short': 'dict', 'long': 'dictionary'}
18 >>> d = dict([(1, 1), (2, 4)])
19 >>> d
20 {1: 1, 2: 4}
21
22
23 В-третьих, с помощью метода fromkeys:
24
25 >>> d = dict.fromkeys(['a', 'b'])
26 >>> d
27 {'a': None, 'b': None}
28 >>> d = dict.fromkeys(['a', 'b'], 100)
29 >>> d
30 {'a': 100, 'b': 100}
31
32
33 В-четвертых, с помощью генераторов словарей, которые очень похожи на генераторы списков.
34
35 >>> d = {a: a ** 2 for a in range(7)}
36 >>> d
37 {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}

```

Рис. 12. dict.txt

```
dict_m.txt
~/Рабочий стол/УП/р1-18/УП/+10/К4_2/2
Открыть Сохранить

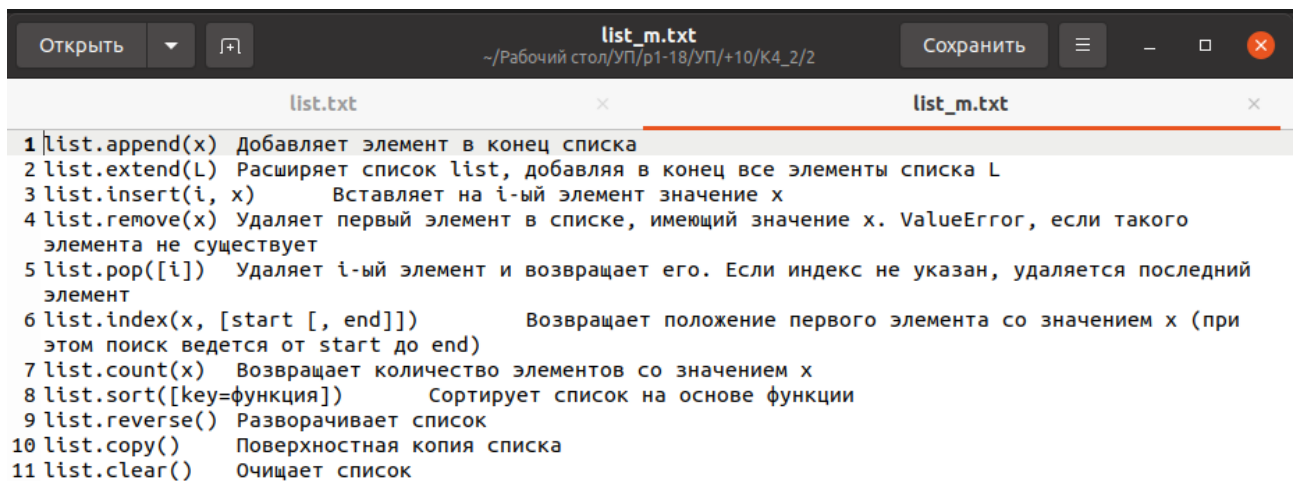
1 dict.clear() - очищает словарь.
2
3 dict.copy() - возвращает копию словаря.
4
5 classmethod dict.fromkeys(seq[, value]) - создает словарь с ключами из seq и значением value (по
  умолчанию None).
6
7 dict.get(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а
  возвращает default (по умолчанию None).
8
9 dict.items() - возвращает пары (ключ, значение).
10
11 dict.keys() - возвращает ключи в словаре.
12
13 dict.pop(key[, default]) - удаляет ключ и возвращает значение. Если ключа нет, возвращает default
  (по умолчанию бросает исключение).
14
15 dict.popitem() - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение
  KeyError. Помните, что словари неупорядочены.
16
17 dict.setdefault(key[, default]) - возвращает значение ключа, но если его нет, не бросает
  исключение, а создает ключ с значением default (по умолчанию None).
18
19 dict.update([other]) - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие
  ключи перезаписываются. Возвращает None (не новый словарь!).
20
21 dict.values() - возвращает значения в словаре.
```

Рис. 13. dict_m.txt

```
list.txt
~/Рабочий стол/УП/р1-18/УП/+10/К4_2/2
Открыть Сохранить

1 Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как
  массив, но типы могут отличаться).
2 Чтобы использовать списки, их нужно создать. Создать список можно несколькими способами. Например,
  можно обработать любой итерируемый объект (например, строку) встроенной функцией list:
3
4 >>> list('список')
5 ['с', 'п', 'и', 'с', 'о', 'к']
6
7
8 Список можно создать и при помощи литерала:
9
10 >>> s = [] # Пустой список
11 >>> l = ['s', 'p', ['isok'], 2]
12 >>> s
13 []
14 >>> l
15 ['s', 'p', ['isok'], 2]
16 Как видно из примера, список может содержать любое количество любых объектов (в том числе и
  вложенные списки), или не содержать ничего.
17
18 И еще один способ создать список - это генераторы списков. Генератор списков - способ построить
  новый список, применяя выражение к каждому элементу последовательности. Генераторы списков очень
  похожи на цикл for.
19
20 >>>
21 >>> c = [c * 3 for c in 'list']
22 >>> c
23 ['lll', 'iii', 'sss', 'ttt']
24
25
26 Возможна и более сложная конструкция генератора списков:
27
28 >>> c = [c * 3 for c in 'list' if c != 'i']
29 >>> c
30 ['lll', 'sss', 'ttt']
31 >>> c = [c + d for c in 'list' if c != 'i' for d in 'spam' if d != 'a']
32 >>> c
33 ['ls', 'lp', 'lm', 'ss', 'sp', 'sm', 'ts', 'tp', 'tm']
34 Но в сложных случаях лучше пользоваться обычным циклом for для генерации списков.
```

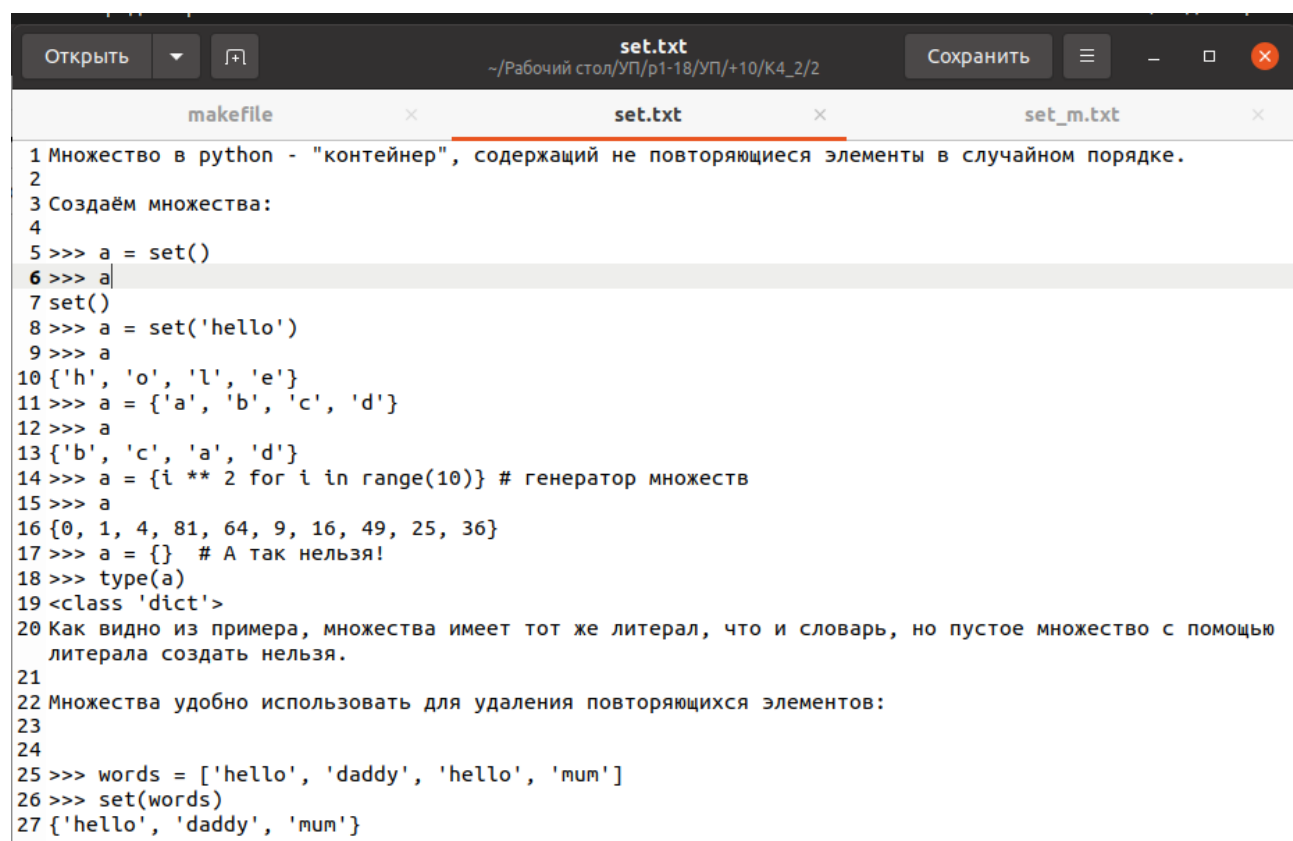
Рис. 14. list.txt



```
list_m.txt
~/Рабочий стол/УП/р1-18/УП/+10/К4_2/2
Сохранить

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

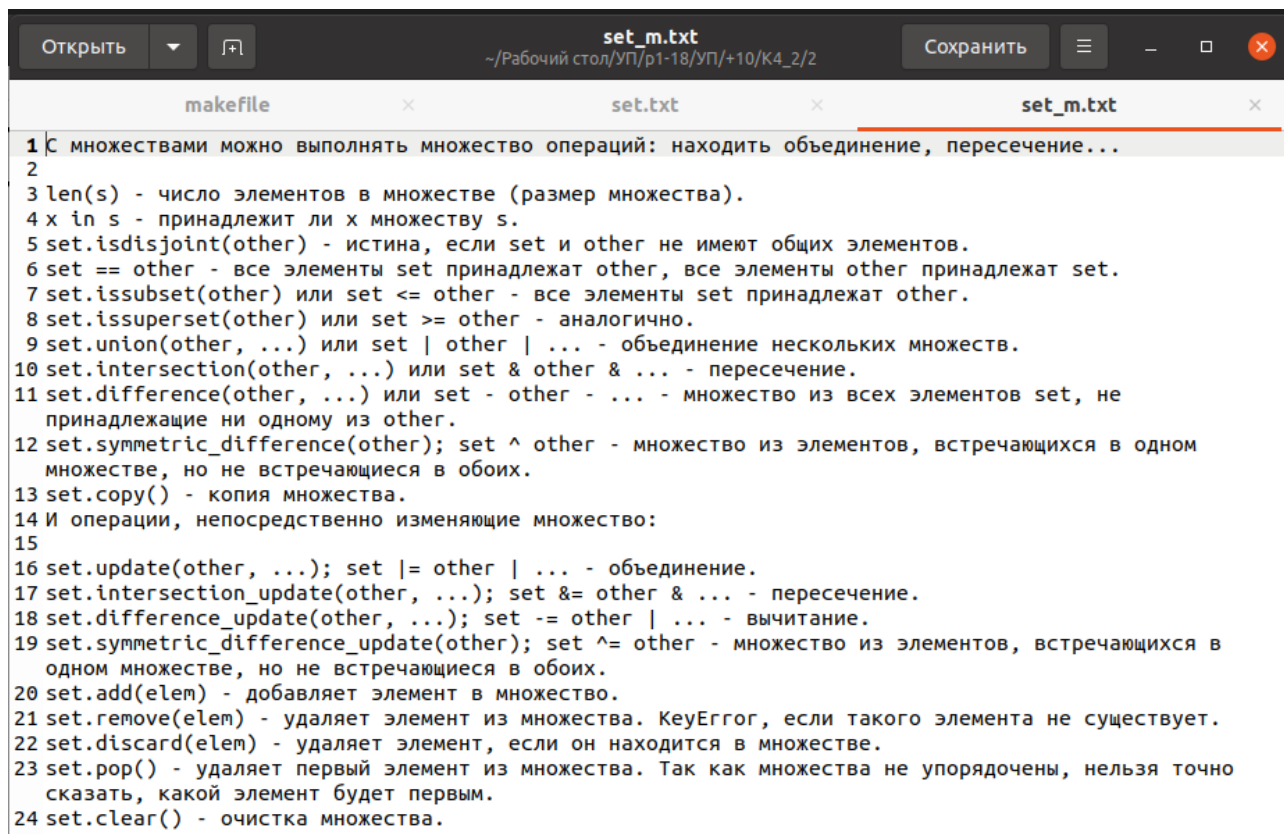
Рис. 15. list_m.txt



```
set.txt
~/Рабочий стол/УП/р1-18/УП/+10/К4_2/2
Сохранить

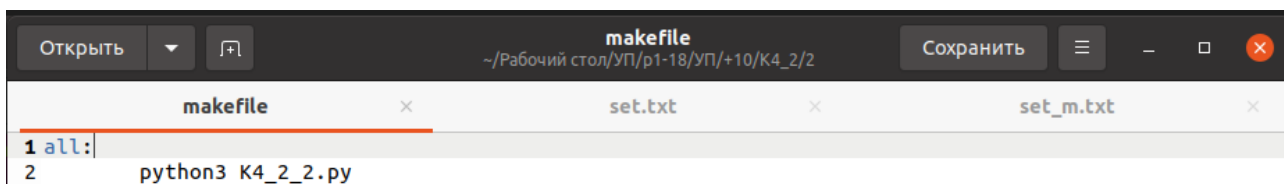
makefile  set.txt  set_m.txt
1 Множество в python - "контейнер", содержащий не повторяющиеся элементы в случайном порядке.
2
3 Создаём множества:
4
5 >>> a = set()
6 >>> a
7 set()
8 >>> a = set('hello')
9 >>> a
10 {'h', 'o', 'l', 'e'}
11 >>> a = {'a', 'b', 'c', 'd'}
12 >>> a
13 {'b', 'c', 'a', 'd'}
14 >>> a = {i ** 2 for i in range(10)} # генератор множеств
15 >>> a
16 {0, 1, 4, 81, 64, 9, 16, 49, 25, 36}
17 >>> a = {} # А так нельзя!
18 >>> type(a)
19 <class 'dict'>
20 Как видно из примера, множества имеет тот же литерал, что и словарь, но пустое множество с помощью
  литерала создать нельзя.
21
22 Множества удобно использовать для удаления повторяющихся элементов:
23
24
25 >>> words = ['hello', 'daddy', 'hello', 'mum']
26 >>> set(words)
27 {'hello', 'daddy', 'mum'}
```

Рис. 16. set.txt



```
1 С множествами можно выполнять множество операций: находить объединение, пересечение...
2
3 len(s) - число элементов в множестве (размер множества).
4 x in s - принадлежит ли x множеству s.
5 set.isdisjoint(other) - истина, если set и other не имеют общих элементов.
6 set == other - все элементы set принадлежат other, все элементы other принадлежат set.
7 set.issubset(other) или set <= other - все элементы set принадлежат other.
8 set.issuperset(other) или set >= other - аналогично.
9 set.union(other, ...) или set | other | ... - объединение нескольких множеств.
10 set.intersection(other, ...) или set & other & ... - пересечение.
11 set.difference(other, ...) или set - other - ... - множество из всех элементов set, не
    принадлежащие ни одному из other.
12 set.symmetric_difference(other); set ^ other - множество из элементов, встречающихся в одном
    множестве, но не встречающиеся в обоих.
13 set.copy() - копия множества.
14 И операции, непосредственно изменяющие множество:
15
16 set.update(other, ...); set |= other | ... - объединение.
17 set.intersection_update(other, ...); set &= other & ... - пересечение.
18 set.difference_update(other, ...); set -= other | ... - вычитание.
19 set.symmetric_difference_update(other); set ^= other - множество из элементов, встречающихся в
    одном множестве, но не встречающиеся в обоих.
20 set.add(elem) - добавляет элемент в множество.
21 set.remove(elem) - удаляет элемент из множества. KeyError, если такого элемента не существует.
22 set.discard(elem) - удаляет элемент, если он находится в множестве.
23 set.pop() - удаляет первый элемент из множества. Так как множества не упорядочены, нельзя точно
    сказать, какой элемент будет первым.
24 set.clear() - очистка множества.
```

Рис. 17. set_m.txt



```
1 all:
2     python3 K4_2_2.py
```

Рис. 18. makefile

1.4 Техника работы с циклическими программами, цикл while

Приложения: K5_1.py K5_1_2.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
'''
K5_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("а) 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)

```

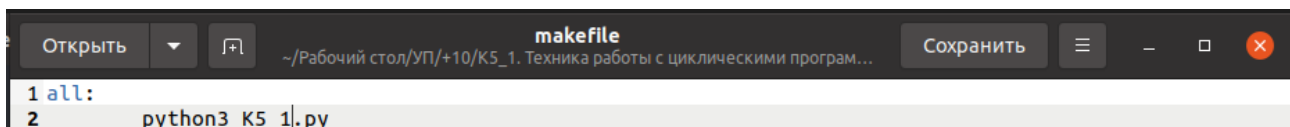


Рис. 19. Makefile

Листинг 5. K5_1_2.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K5_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")

```

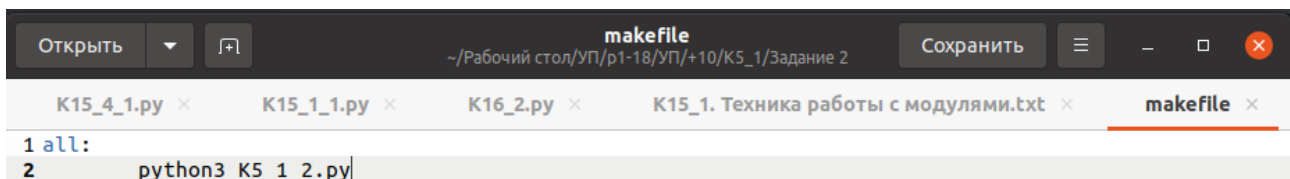


Рис. 20. makefile

Листинг 6. K5_2_1.py makefile

```

#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
K5_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)}")

```

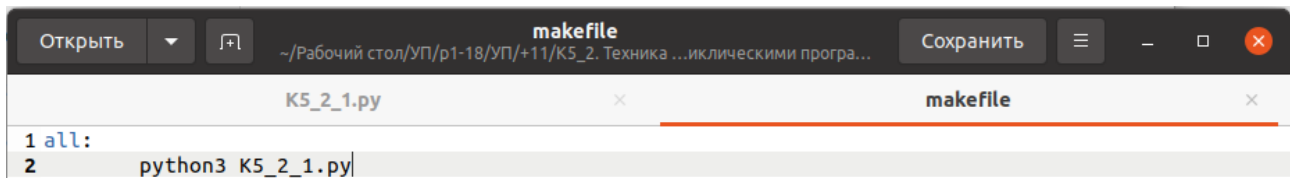


Рис. 21. makefile

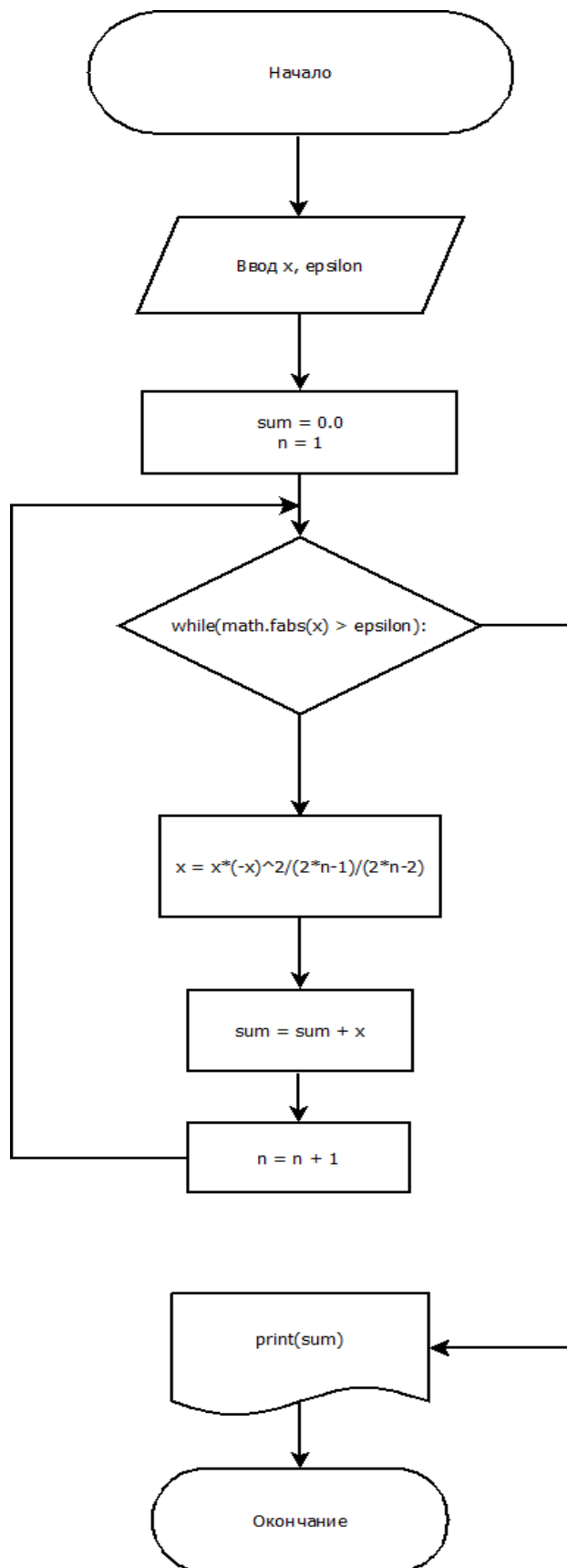


Рис. 22. makefile

Листинг 7. K5_2_2.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K5_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)
```

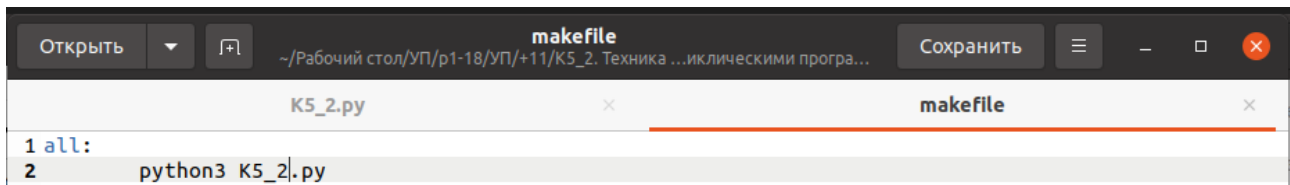


Рис. 23. makefile

Листинг 8. K5_2_3.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K5_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) )
```

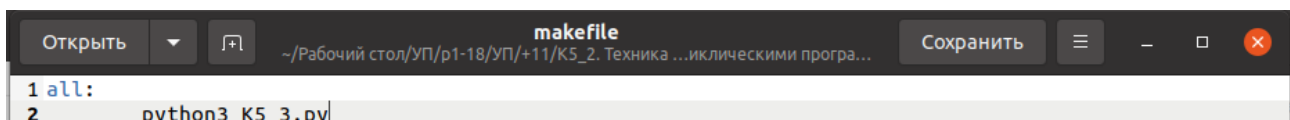


Рис. 24. makefile

Листинг 9. K5_2_4.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K5_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()
```

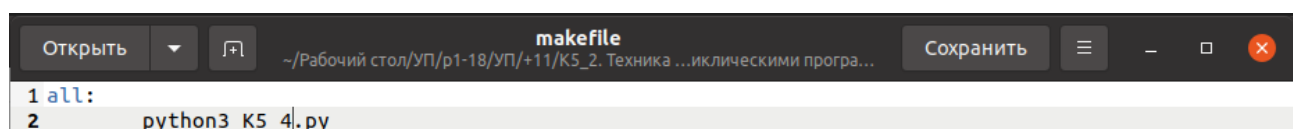


Рис. 25. makefile

Листинг 10. K5_2_5.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K5_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()
```

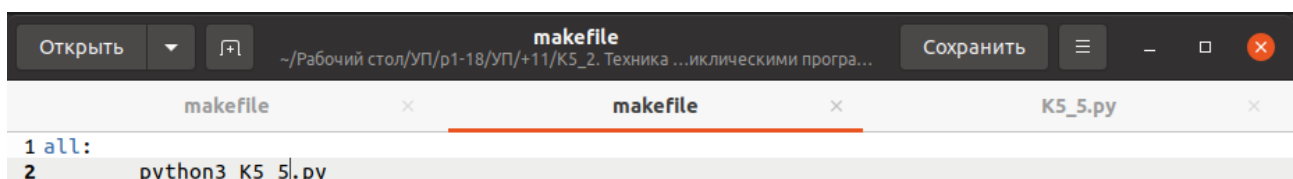


Рис. 26. makefile

1.5 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
'''
K6_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
знаков 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 дроби:")
    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()

```

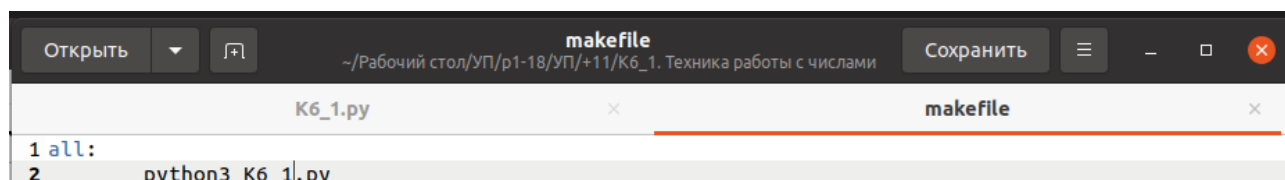


Рис. 27. makefile

Листинг 12. K6_2_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

K6_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()

```

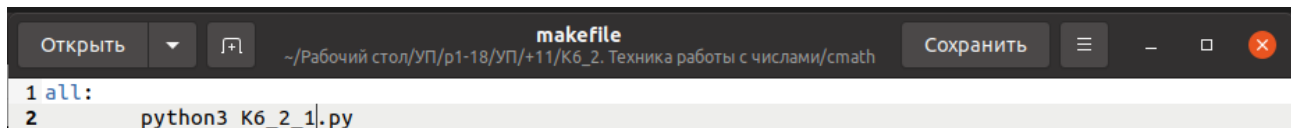


Рис. 28. makefile

Листинг 13. K6_2_2.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K6_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()

```

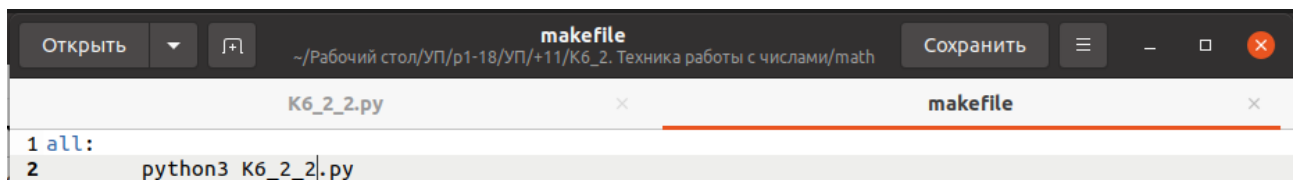


Рис. 29. makefile

1.6 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
'''
K7_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')
```

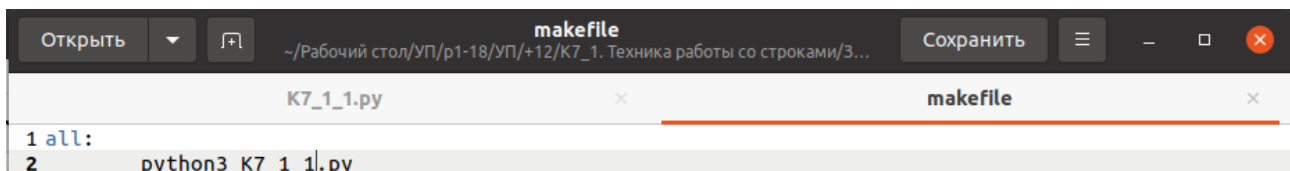


Рис. 30. makefile

Листинг 15. K7_1_2.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K7_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 слодиться 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='_')
```

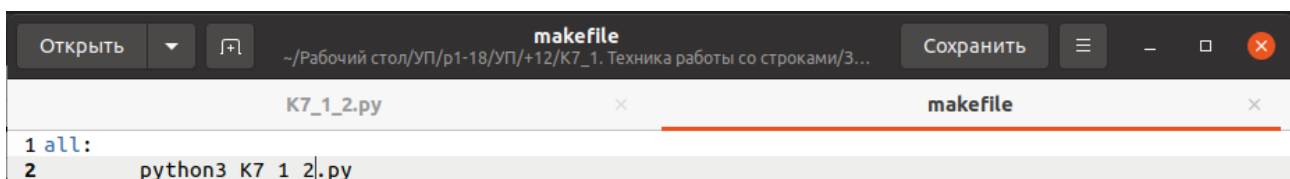


Рис. 31. makefile

Листинг 16. K7_1_3.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K7_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=' ')
```

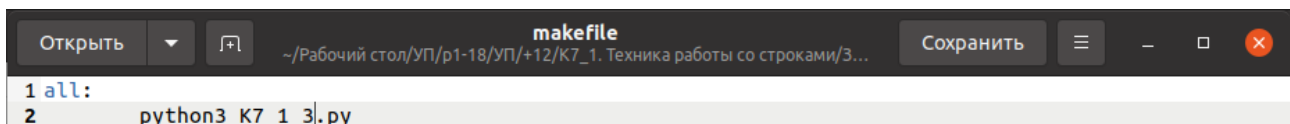


Рис. 32. makefile

Листинг 17. K7_1_4.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K7_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')

```

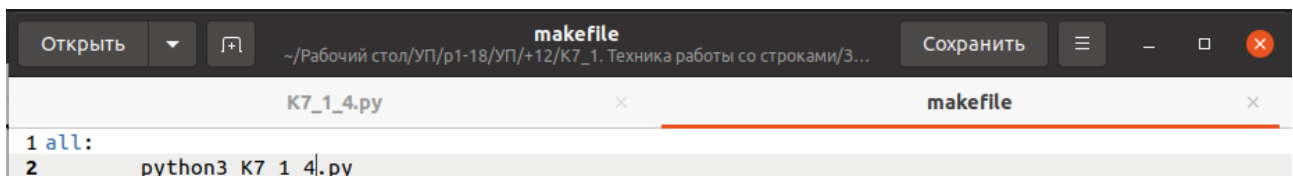


Рис. 33. makefile

Листинг 18. K7_2.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K7_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 шестнадцатичного 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("
      Template(templ_string).substitute("
      name=name, error=hex(errno)")
      )")
print(")")
print(")")
print(
    Template(templ_string).substitute(
        name=name, error=hex(errno)
    )
)
print("-----")
print()

```

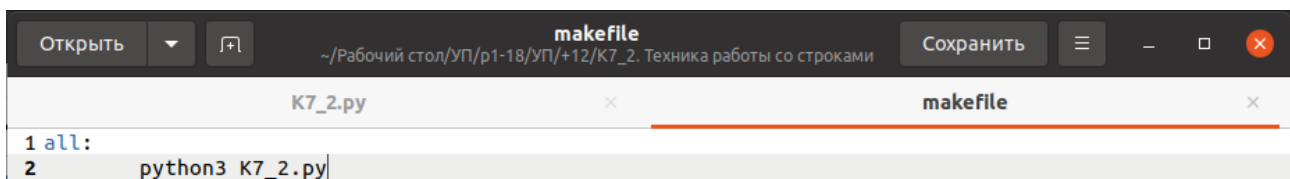


Рис. 34. makefile

1.7 Техника работы со списками

Приложения: K8_1_1.py, K8_1_2.py, K8_1_3.py, K8_2_1, K8_2_2, K8_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)
```

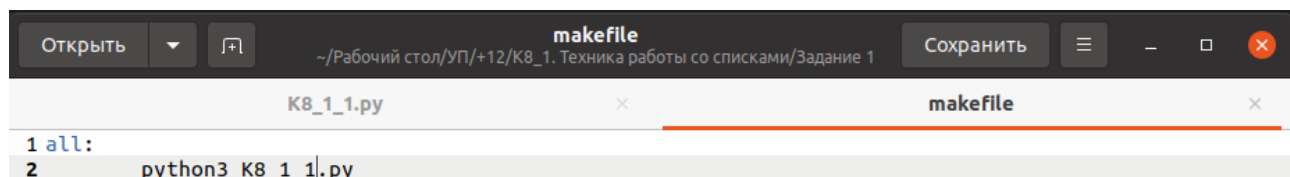


Рис. 35. makefile

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)

```

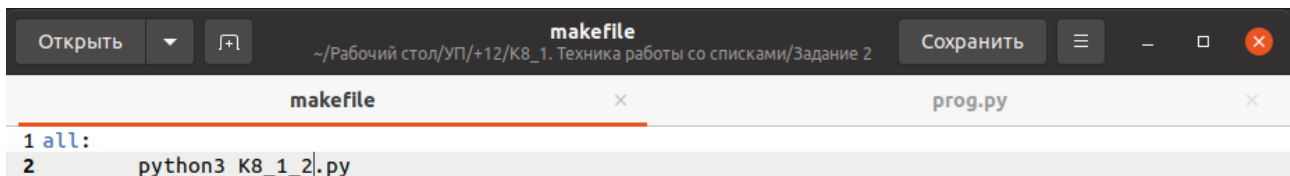


Рис. 36. makefile

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.В.)
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)
```

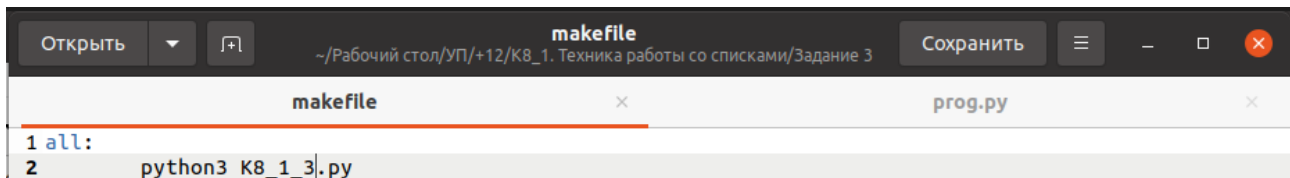


Рис. 37. makefile

Листинг 22. K8_2_1.py makefile

```
#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
K8_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)
```

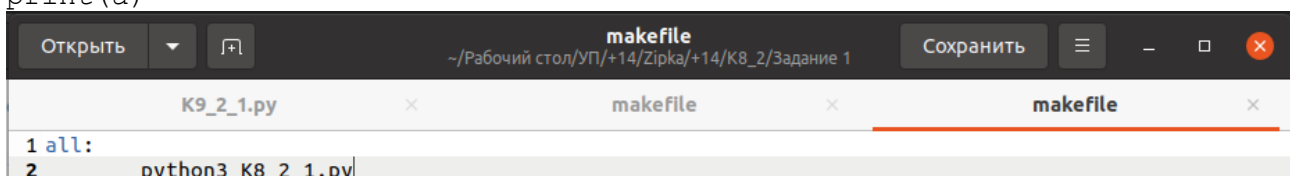


Рис. 38. makefile

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 вывода 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)

```

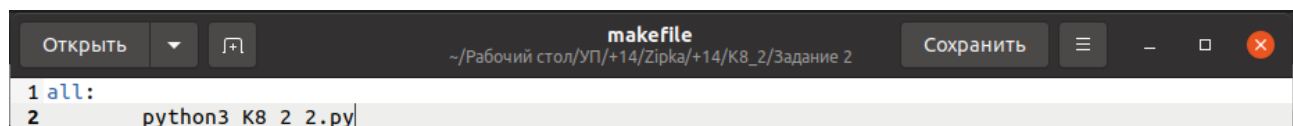


Рис. 39. makefile

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)
```

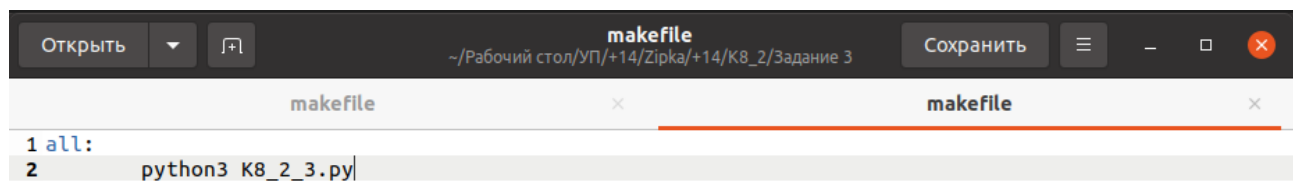


Рис. 40. makefile

1.8 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
'''
K9_1. Техника работы с циклом for и генераторами списков

Задание 1. (Л.В.) 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-----

```

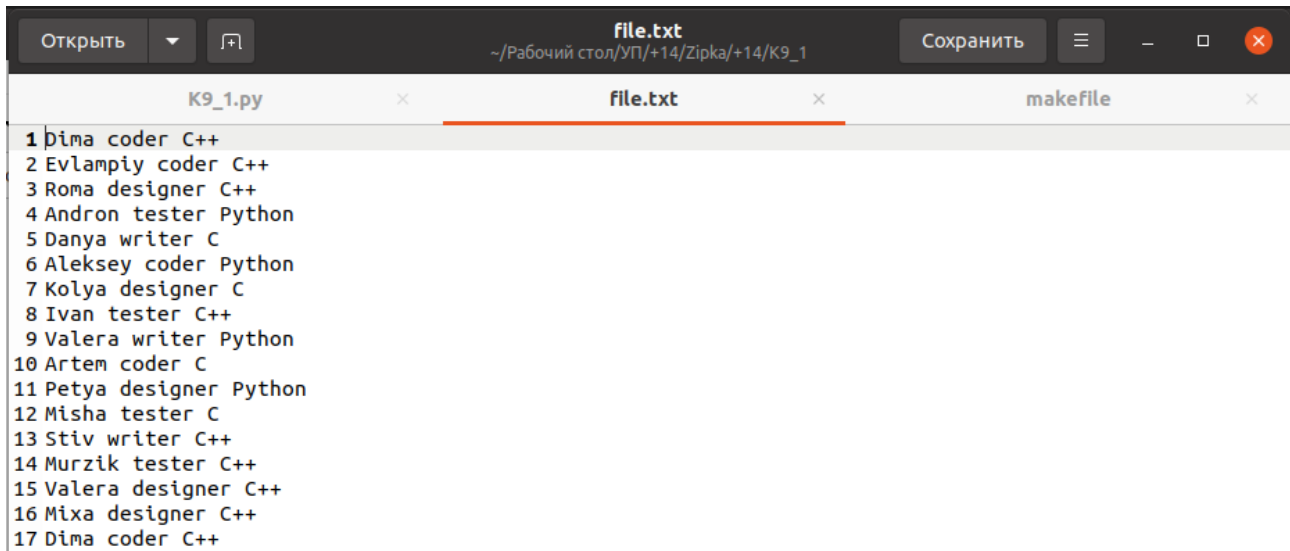


Рис. 41. file.txt

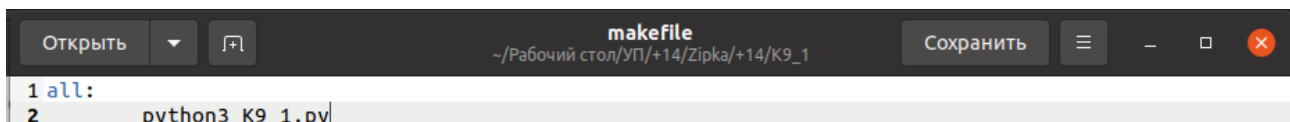


Рис. 42. makefile

Листинг 26. K9_2_1.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

K9_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))
```

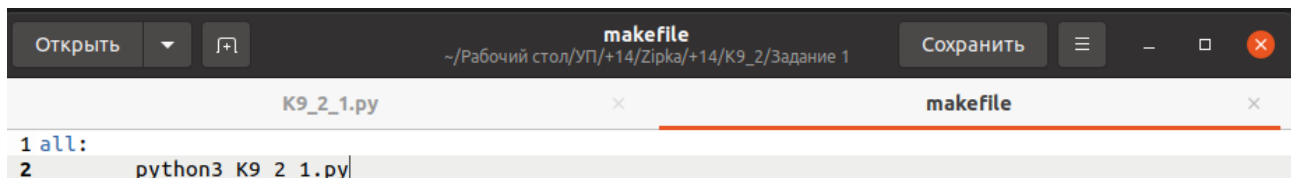


Рис. 43. makefile

Листинг 27. K9_2_2.py makefile

```

#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
K9_2. Техника работы с циклом for и генераторами списков

Задание 2. Array57. Дан целочисленный массив А размера N. Переписать в
новый целочисленный массив В
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))

```

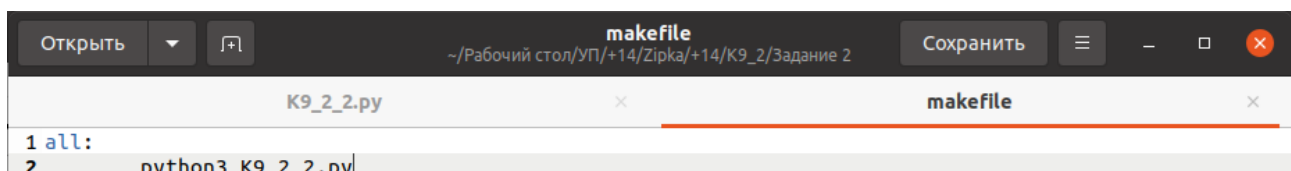


Рис. 44. makefile

Листинг 28. K9_2_3.py makefile

```

#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
K9_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)
```

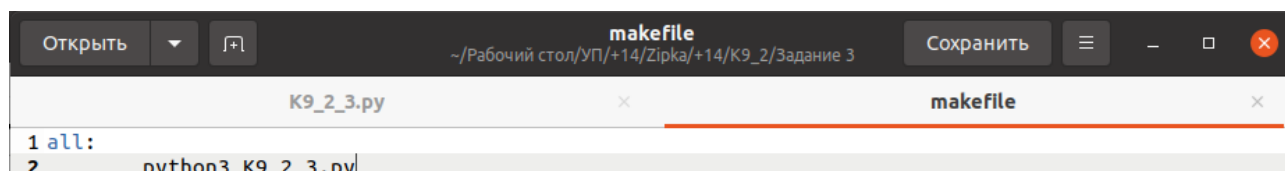


Рис. 45. makefile

Листинг 29. K9_2_4.py makefile

```
#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
```

K9_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]
```

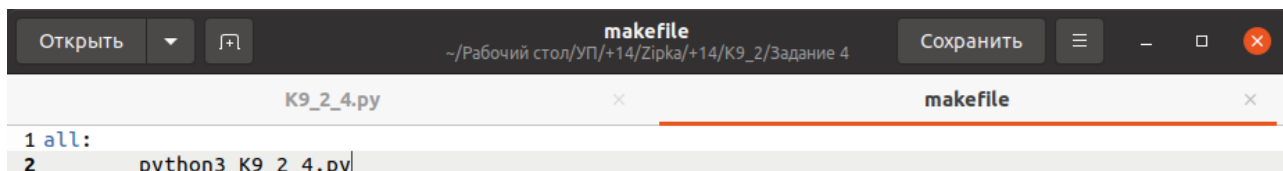


Рис. 46. makefile

Листинг 30. K9_2_5.py makefile

```
#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
K9_2. Техника работы с циклом for и генераторами списков

Задание 5. Matrix56. Дана матрица размера М 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
'''
```

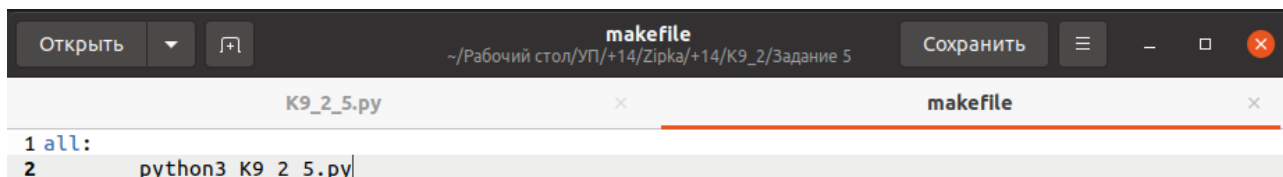


Рис. 47. makefile

Листинг 31. K9_2_6.py makefile

```
#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
K9_2. Техника работы с циклом for и генераторами списков

Задание 6. Matrix88. Дана квадратная матрица порядка М. 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])
```

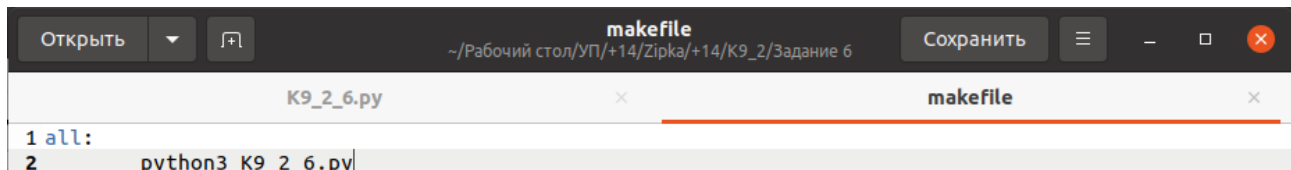


Рис. 48. makefile

1.9 Техника работы с функциями

Приложения: K10_1_2.py, K10_1_3.py, K10_1_4.py, K10_2_1.py, K10_2_2.py, K10_2_3.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)
```

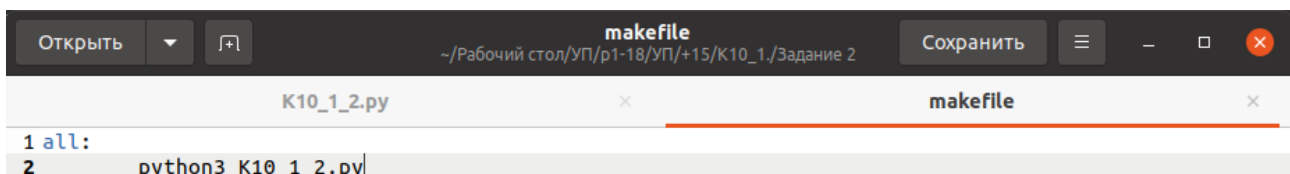


Рис. 49. makefile

Листинг 33. K10_1_3.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K10_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)

```

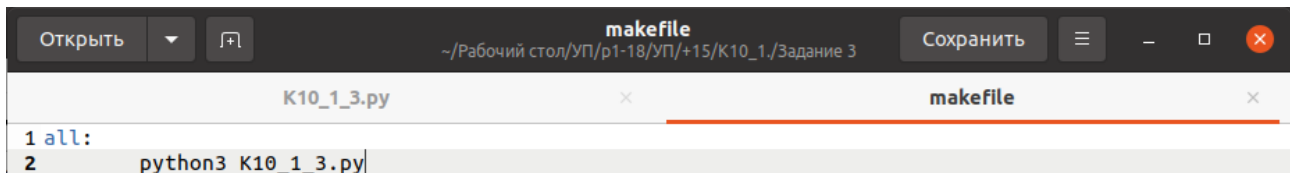


Рис. 50. makefile

Листинг 34. K10_1_4.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K10_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)

```

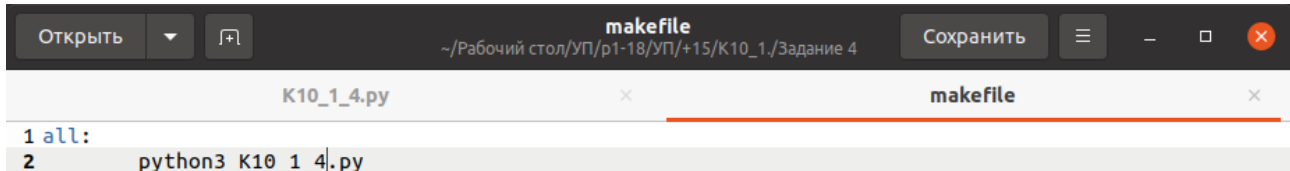


Рис. 51. makefile

Листинг 35. K10_2_2.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K10_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()

```

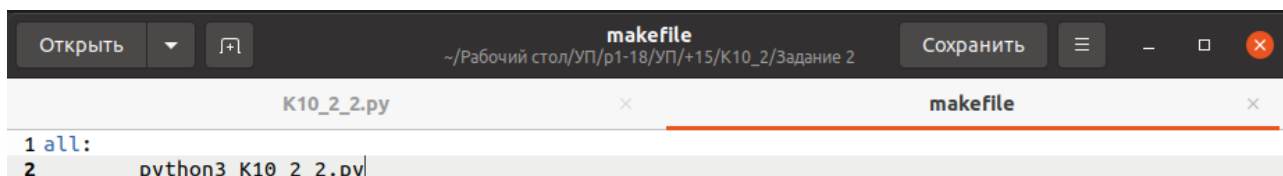


Рис. 52. makefile

Листинг 36. K10_1_4.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K10_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)

```

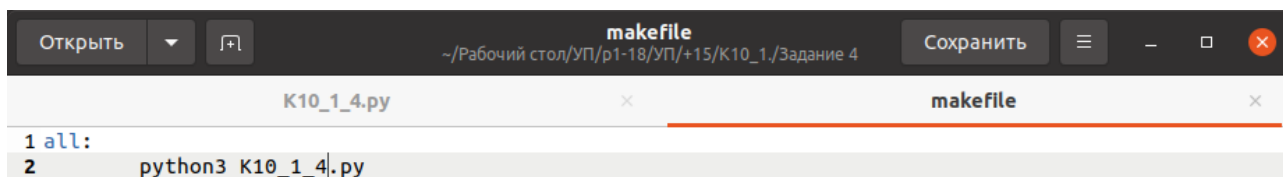


Рис. 53. makefile

Листинг 37. K10_2_2.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K10_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()
```

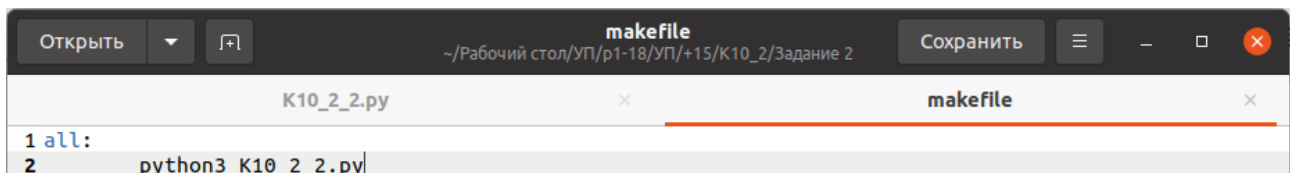


Рис. 54. makefile

Листинг 38. K10_2_3.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K10_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))
```

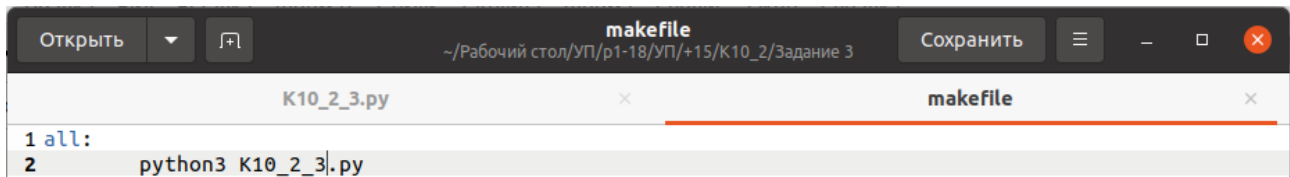


Рис. 55. makefile

Листинг 39. K10_2_4.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K10_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)
```

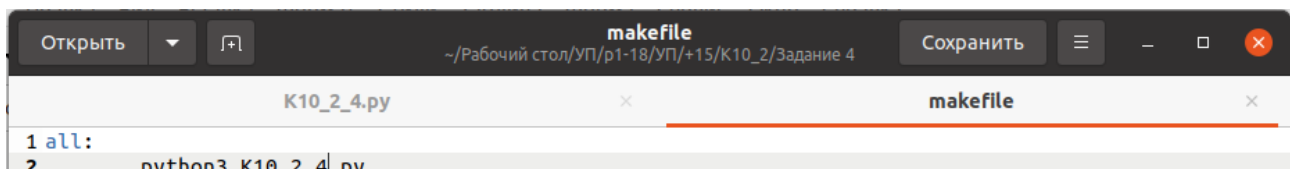


Рис. 56. makefile

1.10 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

'''

K11_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()
```

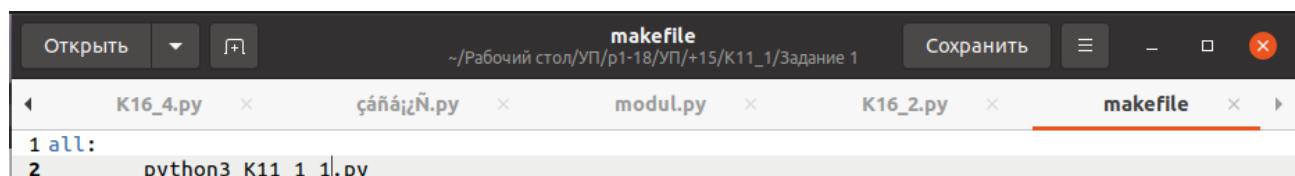


Рис. 57. makefile

Листинг 41. K11_1_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

K11_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 число M — количество запросов к файлам. В последних

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

любое количество запросов.

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

каждого запроса должна будет возвращать ОК 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')

```

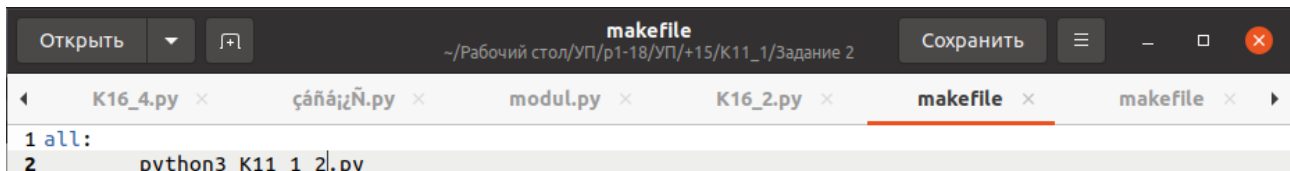


Рис. 58. makefile

Листинг 42. K11_1_3.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

K11_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

```

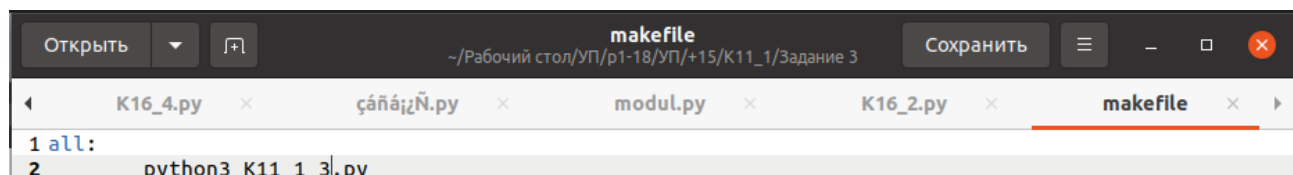


Рис. 59. makefile

Листинг 43. K11_2_1.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''

```

K11_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]

```

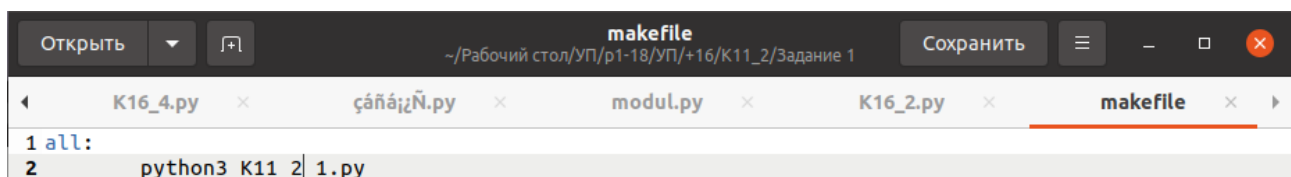


Рис. 60. makefile

Листинг 44. K11_2_2.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K11_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])

```

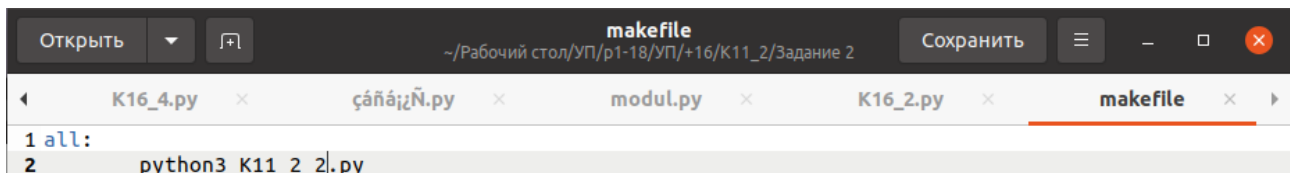


Рис. 61. makefile

Листинг 45. K11_2_3.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K11_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])
```

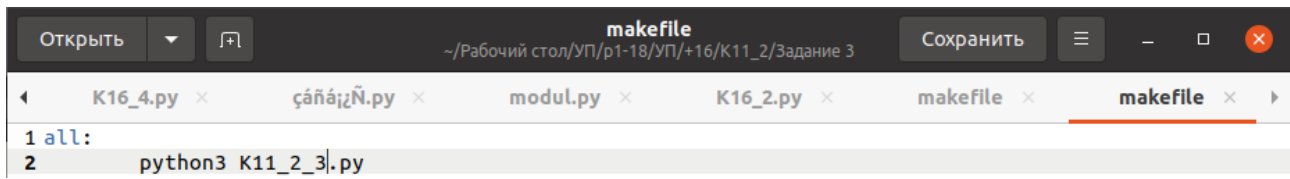


Рис. 62. makefile

1.11 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
'''
K12_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())))
```

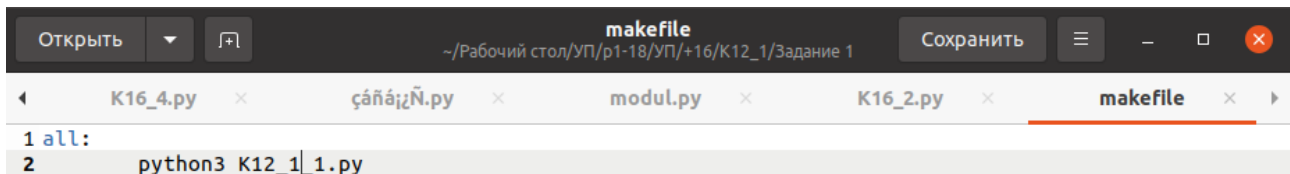


Рис. 63. makefile

Листинг 47. K12_1_2.py makefile

```
#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
K12_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()))))
```

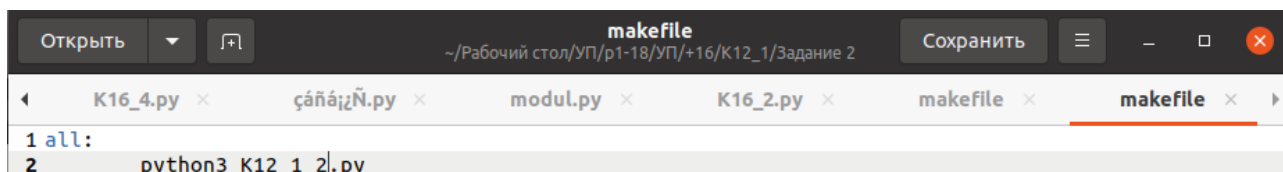


Рис. 64. makefile

Листинг 48. K12_1_3.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

K12_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))
```

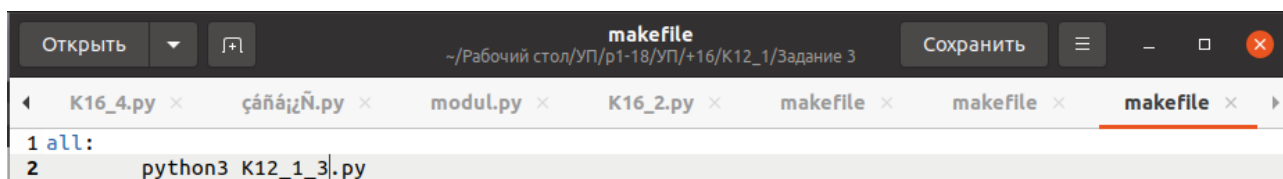


Рис. 65. makefile

Листинг 49. K12_1_4.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

K12_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))
```

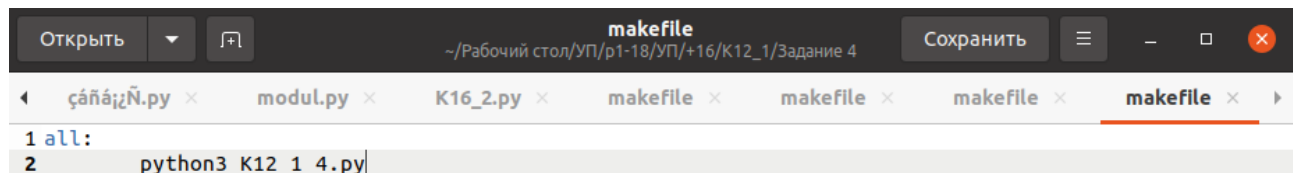


Рис. 66. makefile

Листинг 50. K12_1_5.py makefile

```
#Выполнили: Груздев Роман, Герасимов Дмитрий
```

```
#Группа: П1-18
```

```
'''
```

```
K12_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')
```

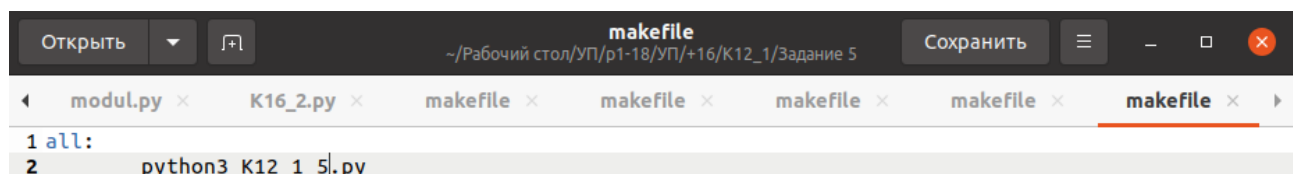


Рис. 67. makefile

Листинг 51. K12_1_5.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

```

#Группа: П1-18
'''
K12_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):
    str = input().lower()
    words.add(str)

errors = set()
l = int(input())
for i in range(l):
    str = input().lower().split()
    for j in str:
        if(j not in words and j not in errors):
            errors.add(j)
print('\n'.join(errors))

```

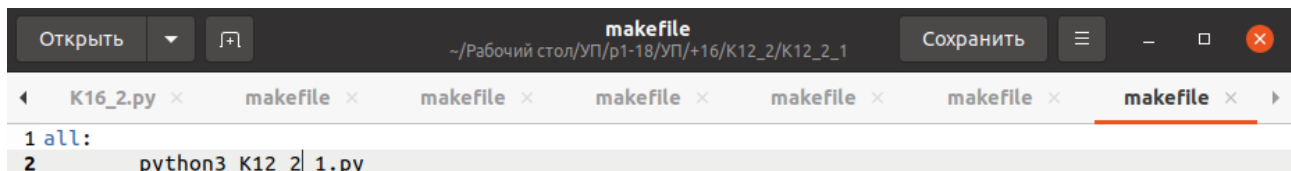


Рис. 68. makefile

Листинг 52. K12_1_5.py makefile disc.txt session.txt out.txt

```

#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
K12_2. Техника работы с множествами

Задание 2. (Л.В.) 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()

```

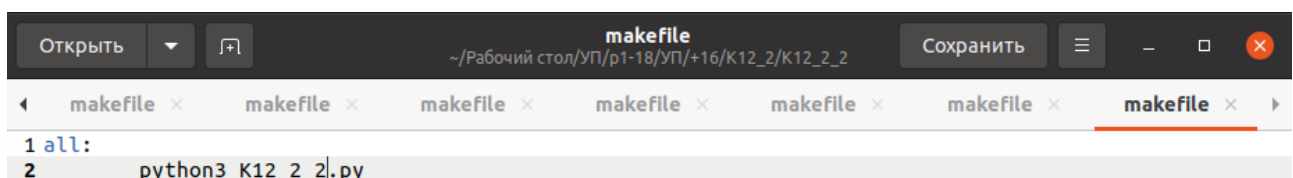


Рис. 69. makefile

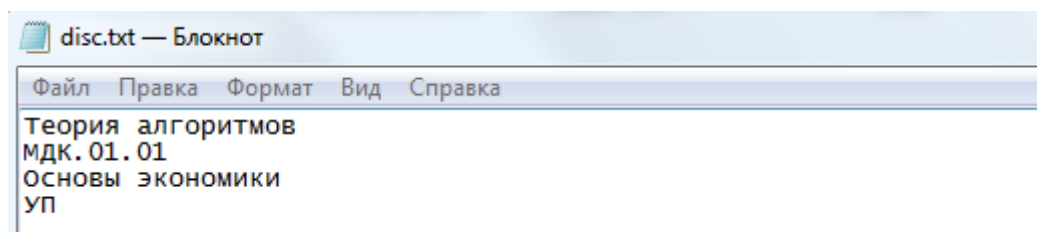


Рис. 70. disc.txt

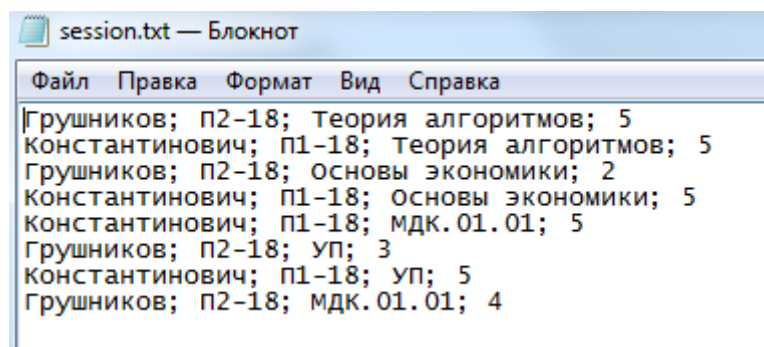


Рис. 71. session.txt

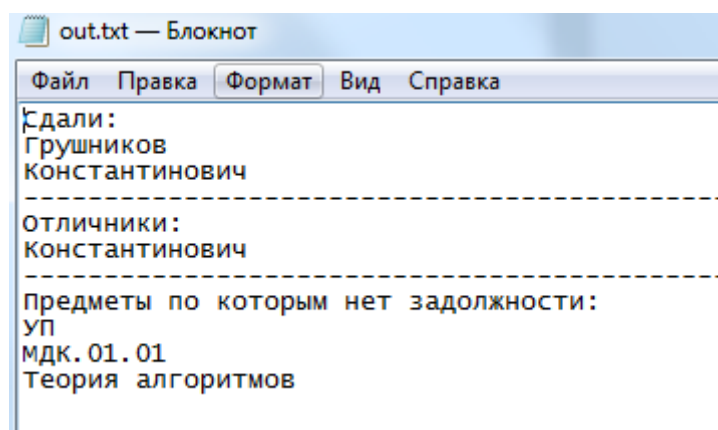


Рис. 72. out.txt

1.12 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
'''
K13_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)]))
```

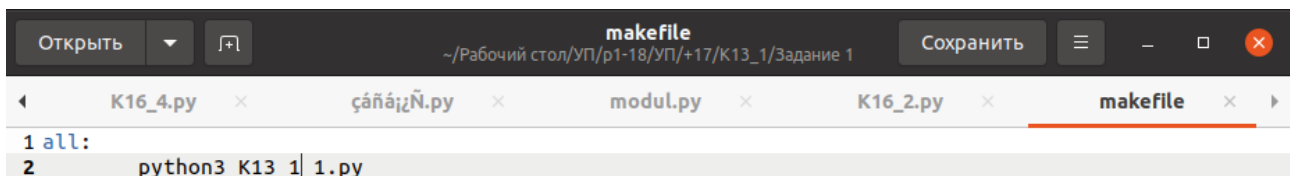


Рис. 73. makefile

Листинг 54. K13_1_2.py makefile

```
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K13_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')
```

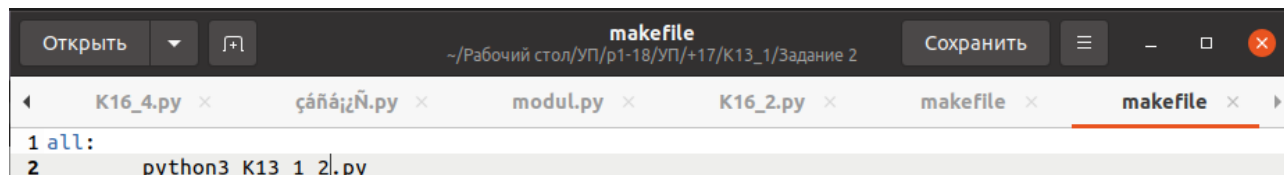


Рис. 74. makefile

Листинг 55. K13_1_3.py makefile input.txt

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

K13_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. Прочитать 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("Сортированы 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
обнаружено")

'''
for i in sorted(d_hyp.values()):
    print(i)
'''

'''
fout = open("output.txt", 'w')
fout.write('\n'.join(tup))
fout.close()
'''

```

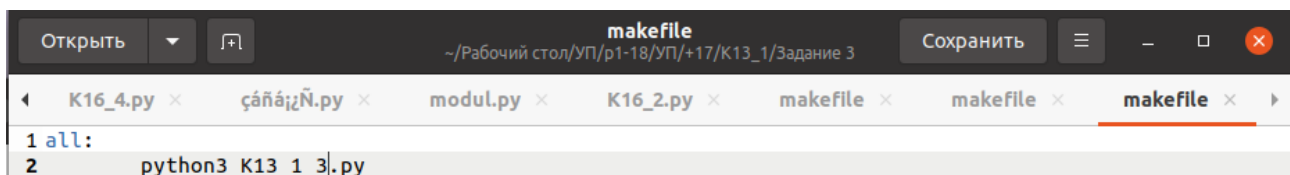


Рис. 75. makefile

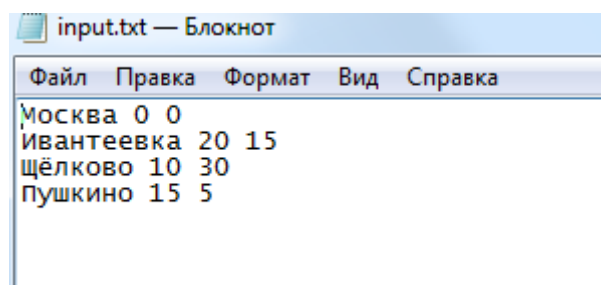


Рис. 76. input.txt

Листинг 56. K13_2.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''

```

K13_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("Именованные кортежи")

#Подключаем модуль namedtuple
print("#Подключаем модуль namedtuple")
print("from collections import namedtuple")
from collections import namedtuple
print()
#Создаем именованный кортеж
print("#Создаем именованный 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 распаковать, как обычный кортеж
print("#Можно распаковать, 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'))

```

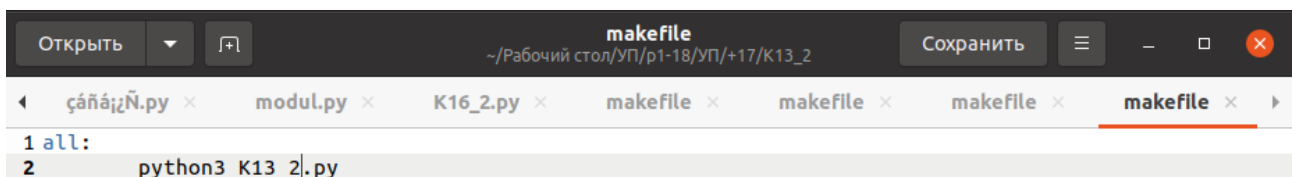


Рис. 77. makefile

1.13 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
'''
K14_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!")
```

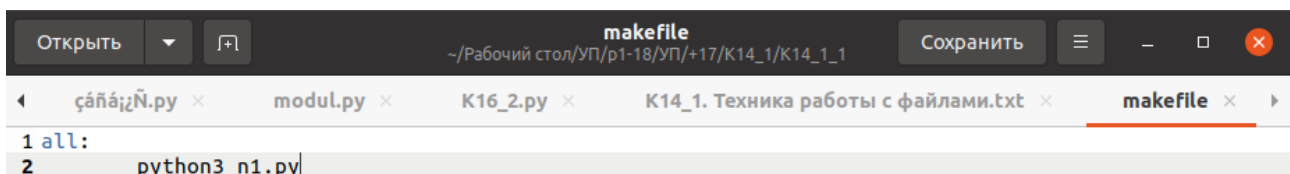


Рис. 78. makefile

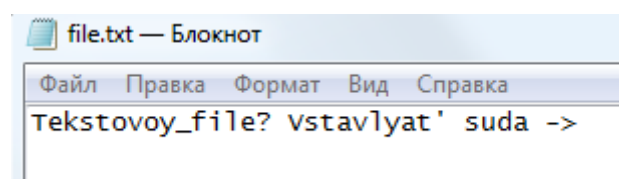


Рис. 79. file.txt

Листинг 58. K14_1_2.py makefile file.txt

```
#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
K14_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")
str = 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((str)+'\n')
        else:
            fout.write(i[j])

fout.close()
```

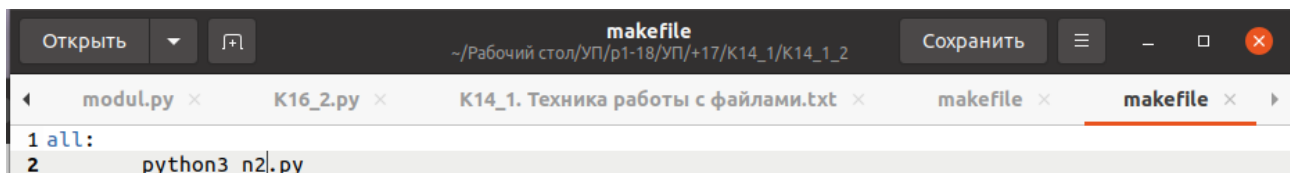


Рис. 80. makefile

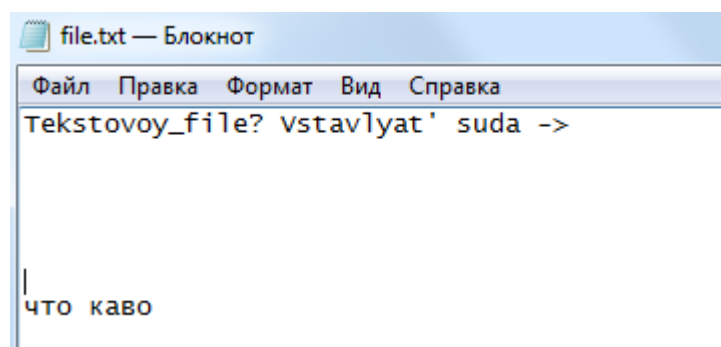


Рис. 81. file.txt

Листинг 59. K14_1_3.py makefile file.txt file2.txt

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

K14_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()
```

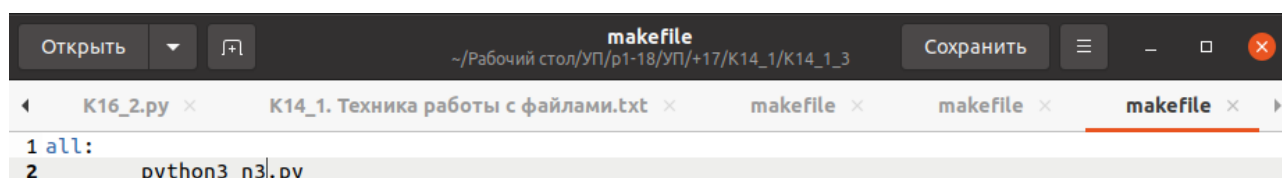


Рис. 82. makefile

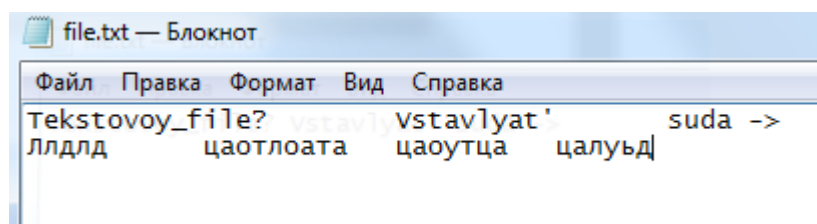


Рис. 83. file.txt

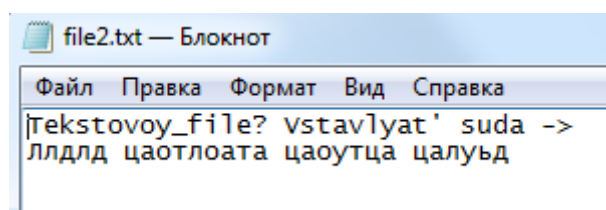


Рис. 84. File2.txt

Листинг 60. K14_1_4.py makefile file.txt

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

K14_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)
```

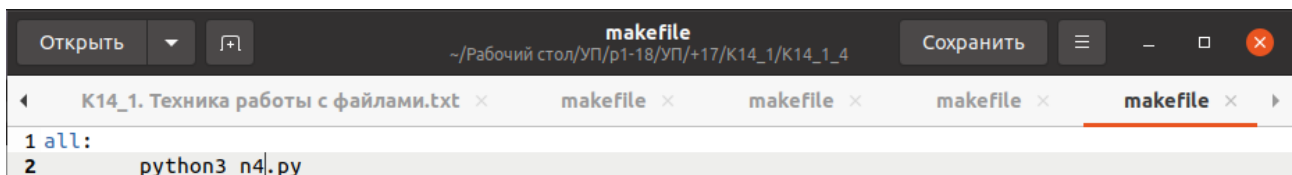


Рис. 85. makefile

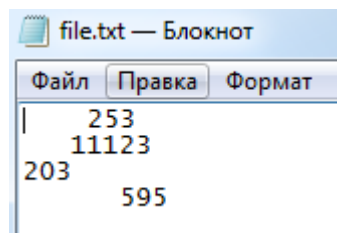


Рис. 86. file.txt

Листинг 61. K14_1_5.py makefile file.txt file2.txt

```
#Выполнили: Груздев Роман, Герасимов Дмитрий  
#Группа: П1-18  
'''  
  
K14_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()
```

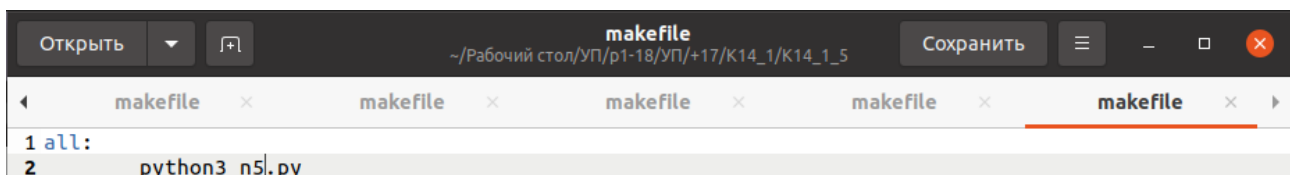


Рис. 87. makefile

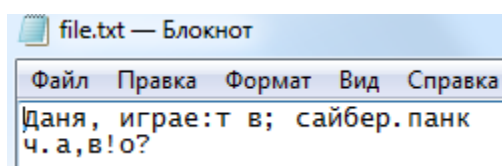


Рис. 88. file.txt

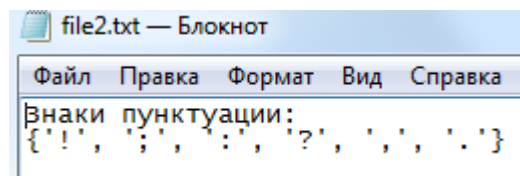


Рис. 89. file2.txt

Листинг 62. K14_2.py makefile input.txt

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

K14_2. Техника работы с файлами

Задание 1. (Л.В.)

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=', ')

```

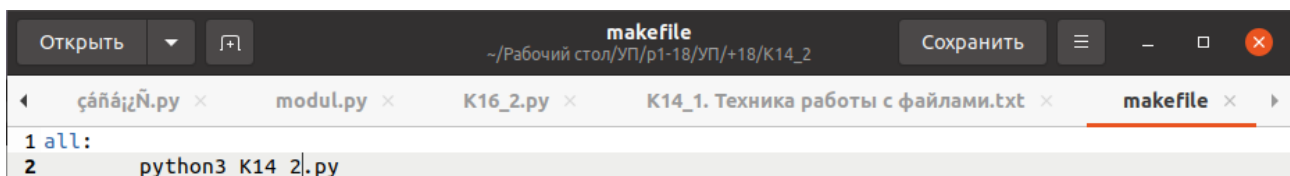


Рис. 90. makefile

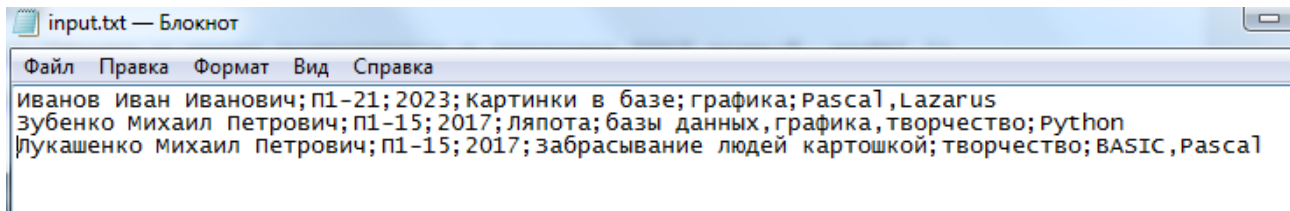


Рис. 91. input.txt

1.14 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
```

```
'''
K15_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)

```

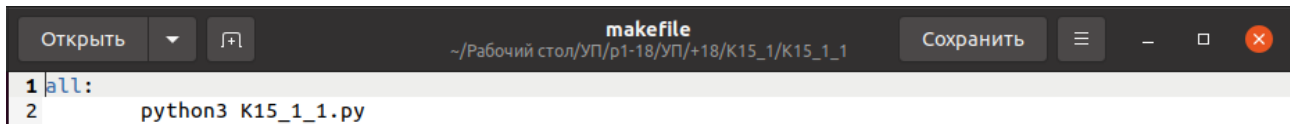


Рис. 92. makefile

Листинг 64. K15_1_2.py makefile text1.txt

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18

```

```
'''
```

K15_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)

```

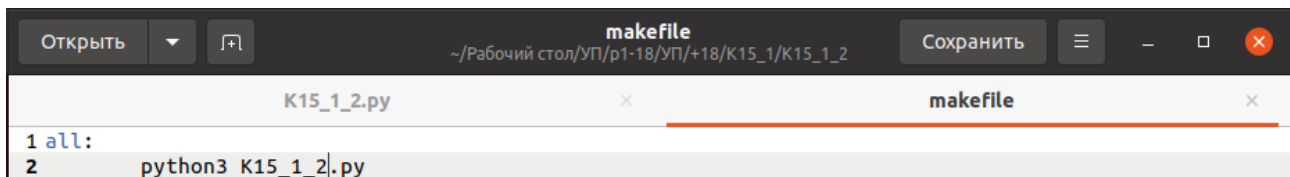


Рис. 92. makefile

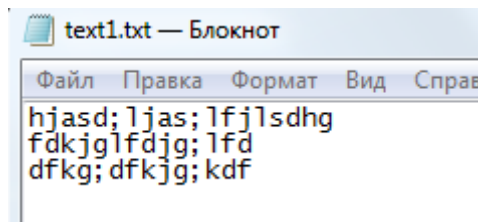


Рис. 93. text1.txt

Листинг 64. K15_2_1.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18

```

'''
K15_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()

```

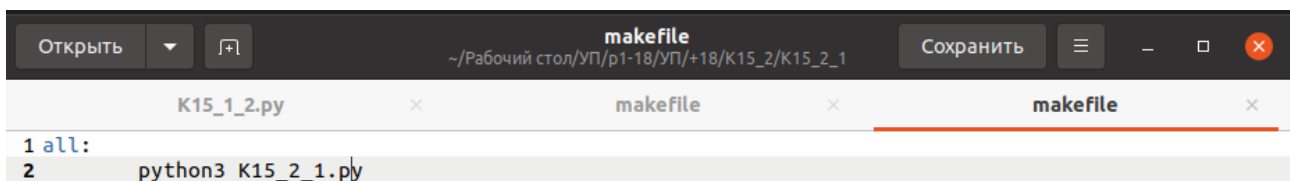


Рис. 94. makefile

Листинг 65. K15_2_2.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18

```

'''
K15_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()))

```

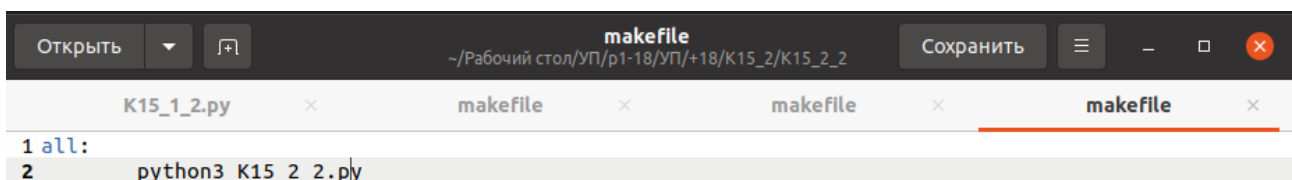


Рис. 95. makefile

Листинг 66. K15_3_1.py makefile

```

#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''

```

K15_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")
```

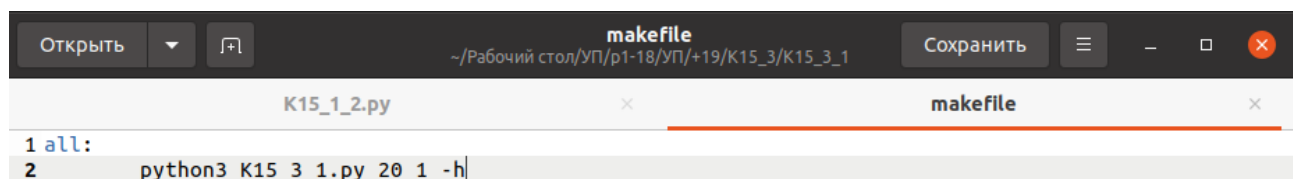


Рис. 96. makefile

Листинг 67. K15_3_2.py makefile

```
#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
```

K15_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-ispolzueмой-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())
```

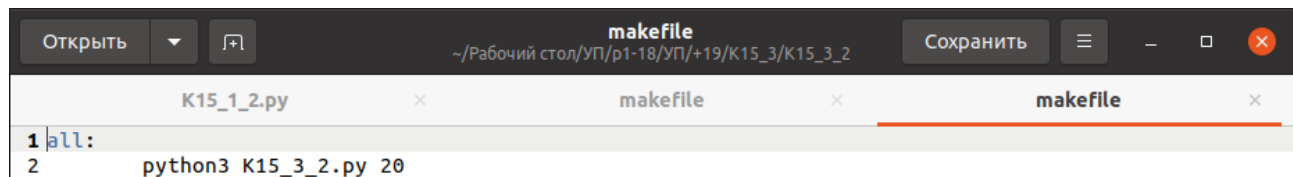


Рис. 97. makefile

Листинг 68. K15_3_3.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

```
'''
```

K15_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-svedeniya-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)
```

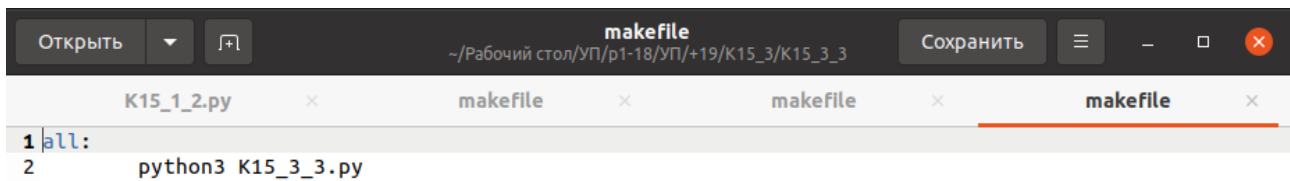


Рис. 98. makefile

Листинг 69. K15_3_4.py makefile

```
#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
```

K15_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)

```

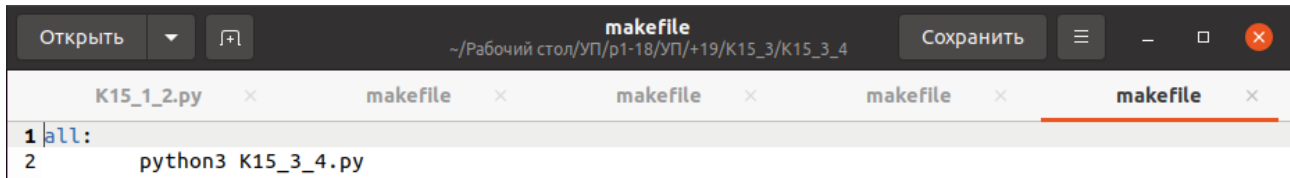


Рис. 99. makefile

Листинг 70. K15_3_5.py makefile

#Выполнили: Груздев Роман, Герасимов Дмитрий

#Группа: П1-18

'''

K15_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
```

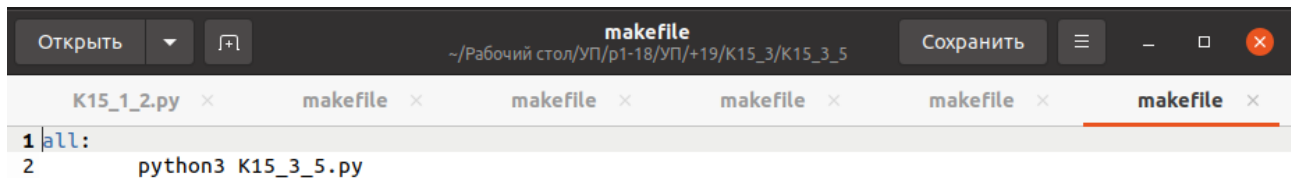


Рис. 100. makefile

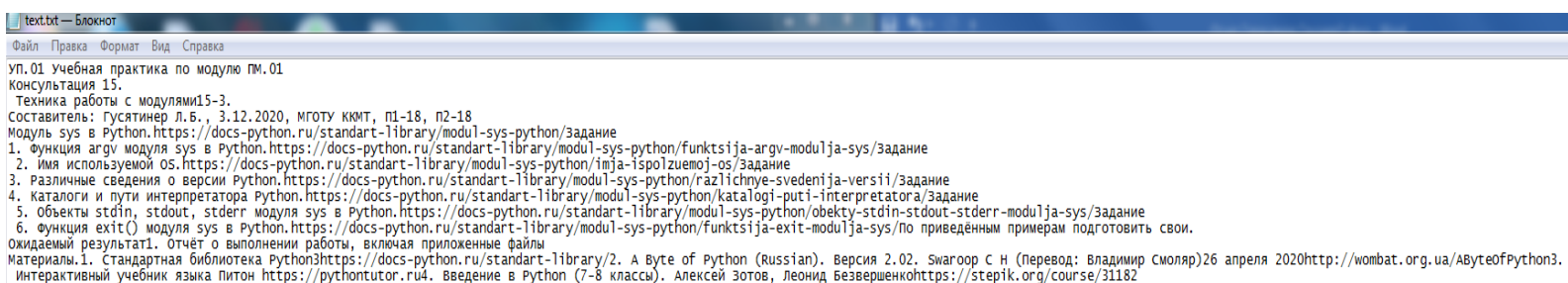


Рис. 101. text.txt

Листинг 71. K15_3_6.py makefile

```
#Выполнили: Груздев Роман, Герасимов Дмитрий
#Группа: П1-18
'''
K15_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)

```

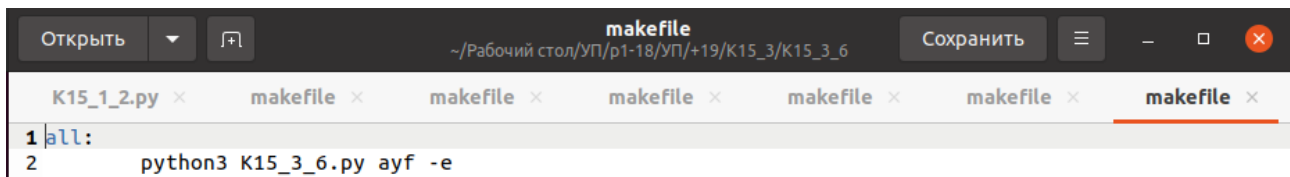


Рис. 102. makefile

Листинг 72. K15_4.py makefile

#Выполнили: Герасимов Дмитрий, Груздев Роман

#Группа: П1-18

'''

K15_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 отсутствие папки с таким
названием

os.mkdir("Балдеж")

```

#Задание 3. Изменение директории
os.chdir("../Балдеж")

if not os.path.isdir("Новая папка"): # Проверка на отсутствие 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()
```

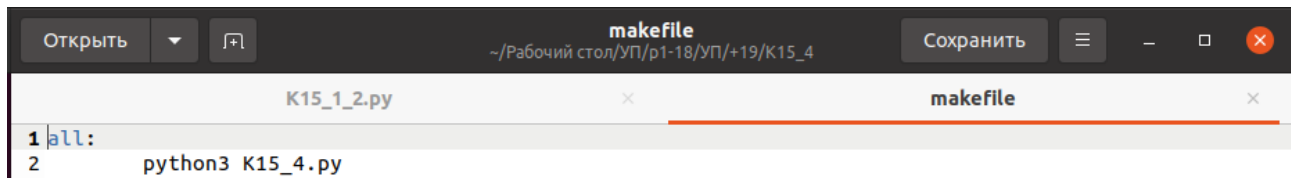


Рис. 103. makefile

1.15 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
'''
K16_1. Техника работы с классами.

Задание 1. Создание класса
Задание 2. Создание объекта
Задание 3. Функция init
Задание 4. Методы объектов
Задание 5. Параметр self
Задание 6. Изменение свойств объекта
Задание 7. Удалить свойства объекта
Задание 8. Удаление объектов
'''

#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

```

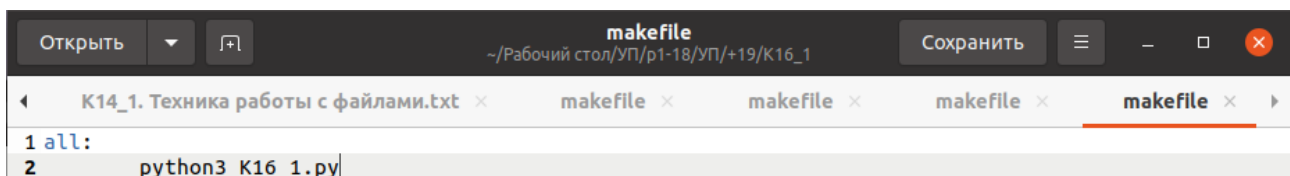


Рис. 104. makefile

Листинг 74. K16_2.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''

```

K16_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

```

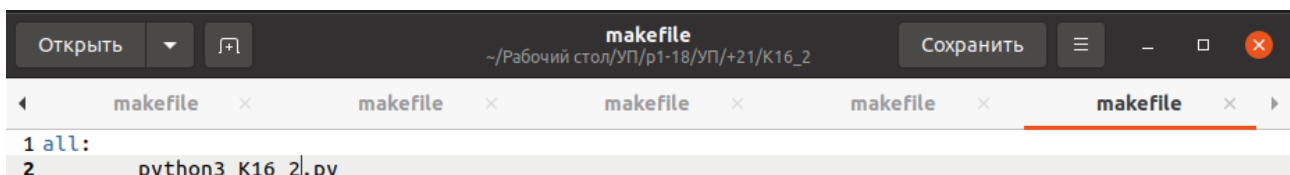


Рис. 105. makefile

Листинг 75. K16_3.py makefile

```

#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K16_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
```

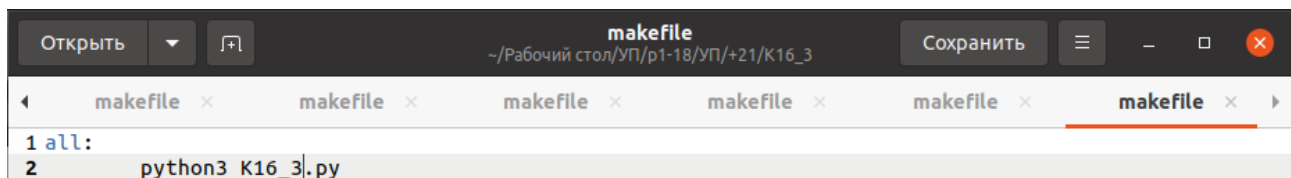


Рис. 106. makefile

Листинг 76. K16_4.py modul.py makefile

```
-----K16_4.py-----
#Выполнили: Герасимов Дмитрий, Груздев Роман
#Группа: П1-18
'''
K16_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
'''
K16_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.health = 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.health += 10
        self.num_subj += 1
        self.subjects.append('shield')

    def printSubjectsHero(self):
        print("Damage:", self.damage)
        print("Health:", self.health)
        print("Subjects hero:", end=' ')
        for i in range(self.num_subj):
            print(self.subjects[i], end = ' ')
        print()

    def SubjectsHero(self):
        return [self.damage, self.health, self.subjects]

    def __del__(self):
        class_name = self.__class__.__name__
```

```
print(f"Класс \"{class_name}\" удален")
```

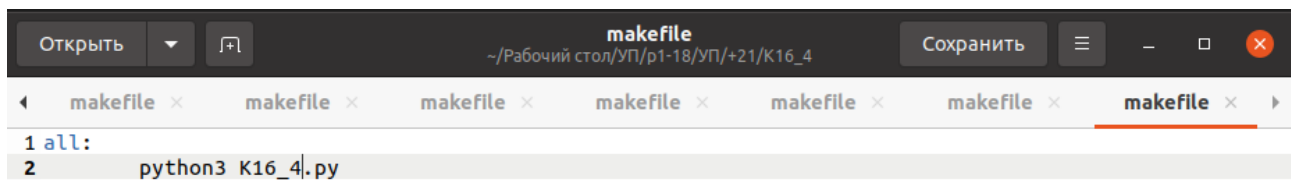


Рис. 107. makefile