Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Ширяева Е. В.

Электронное учебное пособие «Основы программирования. Python 3» Темы «Исключения. Словари. Файлы»

Ростов–на–Дону
2017–2026

Содержание

1 Функция assert. Обработка исключений

1.1 Функция assert

Функция assert (утверждать) позволяет прервать выполнение программы, если логическое условие ложно. При этом выводится сообщение, которое указано после запятой. Сообщение может и отсутствовать.

Если условие не выполняется, возбуждается исключительная ситуация, воспринимаемая как неудача при прохождении теста.

Основное назначение функции assert: завершать программу сразу же после обнаружения неверных данных, что позволяет быстро локализовать и исправить ошибки в программе, которые привели к некорректным выходным данным

Условно применение функции assert можно разделить на два класса:

1) проверка корректности входных данных;

2) проверка корректности выходных данных, то есть соответствие тестовым значениям.

Пример 1

```
n = int(input('n (> 0) >>> '))
assert n > 0  # различие
f = 1
for i in range(2, n+1):
    f *= i
print(f)
```

Результат

```
n (> 0) >>> 0
Traceback (most recent call last):
  File "F:/.../_assertExa.py",
line 2, in <module>
    assert n > 0
AssertionError
```

Пользователь может получить не просто информацию о факте ошибки в виде AssertionError (см. последнюю строку в результате ответа), но и комментарий программиста о типе ошибки.

Пример 2

```
n = int(input('n (> 0) >>> '))
assert n > 0, "Input error"
f = 1
for i in range(2, n+1):
    f *= i
print(f)
```

Результат

```
n (> 0) >>> 0
Traceback (most recent call last):
  File "F:/.../_assertExa.py",
line 2, in <module>
    assert n > 0, "Input error"
AssertionError: Input error
```

Пример 3

```
from math import factorial

n = int(input('n (> 0) >>> '))
assert n > 0, "Input error"

# Вычисление факториала <<вручную>>
f = 1
for i in range(2, n+1):
    f *= i

assert f == factorial(n), "Calculation error"
print('Test: Ok!')
print(f)
```

Результат

```
n (> 0) >>> 5
Test: Ok!
120
```

Пример 4

```
from math import factorial

n = int(input('n (> 0) >>> '))
assert n > 0, "Input error"

# Вычисление факториала <<вручную>>
# Внесём ошибку в код
f = 1
for i in range(2, n):  # здесь ошибка
    f *= i

assert f == factorial(n), "Calculation error"
print('Test: Ok!')
print(f)
```

Результат

```
n (> 0) >>> 5
Traceback (most recent call last):
  File "F:/..._assertExa.py",
line 9, in <module>
    assert f == factorial(n), "Calculation error"
AssertionError: Calculation error
```

1.2 Обработка исключений

См. лекцию.

1.3 Задачи

1.1. Дано натуральное число N. Создать текстовый файл, в который записать матрицу $n \times m$ (n, m — случайные целые числа из $[2; N]$) с элементами $a_{ij} \in [-N; N]$.

Указание. Используйте assert для проверки входного данного.

1.2. В текстовом файле, созданном в задании 1.1, найти номер строки с максимальной суммой элементов.

Указание. Используйте исключение для проверки существования файла.

1.3. Даны целые числа k и p. Прочесть из текстового файла, созданного в задании 1.1, матрицу. Вывести на печать a_{kp}^2. Если такого элемента в матрице не существует, вывести об этом сообщение.

Указание. Используйте исключение для проверки.

1.4. Даны целые числа a, b, c, где $c \in [-13; 78]$. Вычисление значение выражения

$$y = \frac{c^2 \cdot \max(a, b)}{\ln a \cdot \min(b, c)}.$$

Указание. Для проверки ввода значения c используйте инструкцию assert. При вычислении y используйте exceptions. Для проверки корректности вычисления значения y используйте инструкцию assert (для сравнения с результатами своих тестовых примеров).

Пример диалога с пользователем и вывода ответа программы в случае «благоприятных» входных данных:

Результат

```
a = ...
b = ...
c = ...
Test 1. Ok!
Test 2. Ok!
Результат для введённых данных = ...
```

1.5. В программе от пользователя требуется ввести только гласную букву. Создайте и используйте собственное исключение для контроля за входным данным.

Результат 1

```
Input vowel letter >>> u
:))
```

Результат 2

```
Input vowel letter >>> t
:((
```

---

2 Словари

2.1 Примеры

Пример 2.1. Создание пустого словаря.

Создание пустых словарей

```
dict1 = {}
dict2 = dict()
```

Пример 2.2. Создание непустого словаря.

Создание непустых словарей

```
d0 = {'Никита': 100, 'Анна': 56}
d1 = dict(home = 'Машенька', work = 'Мария Сергеевна')
d2 = dict([(1, 100), (2, 4)])
d3 = dict.fromkeys(['cat', 'dog'])
d4 = dict.fromkeys(['cat', 'dog'], 100)
d5 = {a: a ** 2 for a in range(5)}
```

Пример 2.3. Вывод всего словаря и отдельного элемента

Печать

```
dict0 = {'Никита': 'биология', 'Анна': 'информатика'}

print(dict0)          # вывод всего словаря

print(dict0['Анна'])  # вывод значения по ключу Анна
```

Результат работы программы

```
{'Никита': 'биология', 'Анна': 'информатика'}
информатика
```

Пример 2.4 (замена значения в словаре)**.** Выполним замену предмета у элемента словаря с ключом 'Иван'.

Замена значения в словаре

```
dict1 = {'Иван': 'физкультура', 'Анна': 'история',
         'Алина': 'информатика'}
dict1['Иван'] = 'физика'
print(dict1)
```

Результат работы программы

```
{'Иван': 'физика', 'Анна': 'история', 'Алина': 'информатика'}
```

Попытка обратиться к элементу по несуществующему ключу порождает исключение KeyError.

...

См. лекции.

2.2 Задачи

2.1. Дан русско-английский словарь (например, «Дни недели»). Выполнить задания.

1) Составить англо-русский словарь.

2) Удалить элемент в словаре по ключу.

3) Удалить элемент в словаре по значению.

4) Добавить новый элемент в словарь.

2.2 (лидер и аутсайдер)**.** Имеется таблица, содержащая логины и баллы нескольких игроков некоторой игры.

Выведите имя игрока с наибольшим количеством баллов (лидера). Удалите игрока с наименьшим числом баллов (аутсайдера).

Пример работы программы

```
Имеется таблица игроков
{'Hamster': 135, 'Cat': 200, 'Funtik': 111}
--------------------------
Игрок Cat имеет наибольшее число баллов
--------------------------
Список после удаления аутсайдера: {'Hamster': 135, 'Cat': 200}
```

2.3 (итоги чемпионата)**.** Написать программу подведения итогов чемпионата по некоторому виду спорта. Каждая команда представляет одну страну.

Для каждой страны задана информация:

<Страна-участница>: <КЗ> <КС> <КБ>,

где <КЗ> <КС> <КБ> — количество золотых, серебряных и бронзовых медалей, соответственно.

Программа должна напечатать

1) список стран-участниц с набранной суммой баллов.

2) список стран-участниц, занимающих I, II и III места по сумме баллов.

Сумма баллов считается по правилу: за золотую медаль команда получает 7 баллов, за серебряную медаль — 6 баллов, за бронзовую — 5 баллов. Для подсчёта суммы баллов создать функцию.

2.4. Чемпионат по игре «Морской бой» проводится по следующим правилам:

1. Каждый участник регистрируется на сайте игры под некоторым именем. Имена всех участников разные. Каждый игрок может прислать организаторам файл в формате bmp со своим логотипом. Наличие логотипа желательно, но необязательно.

2. Чемпионат проводится в течение определённого времени. Во время проведения чемпионата зарегистрированные участники имеют право играть любое количество раз. По окончании каждой игры количество набранных очков участника заносится в протокол. Каждая строка протокола описывает одну игру и имеет формат:

<Количество баллов> <Имя игрока>

<Количество баллов> — целое положительное число, не превышающее 100 миллионов; <Имя игрока> — строка, не содержащая пробелов.

Строки в протоколе игры расположены в порядке проведения игр.

Если игрок имеет логотип, то в каталоге с протоколом также лежит файл с его именем.

3. Окончательный результат участника — это лучший для участника результат.

4. Более высокое место в соревнованиях занимает участник, набравший наибольшее количество очков. При равенстве результатов более высокое место занимает участник, раньше показавший лучший результат.

Спонсор чемпионата предоставил призы для награждения K лучших игроков ($K \leqslant 10$). Если участников окажется меньше K, призами награждаются все.

Написать программу, которая по данным протокола определяет K лучших игроков и занятые ими места. Программа дополнительно должна генерировать новый файл, содержащий имена игроков, приславших свои логотипы.

Описание входных данных. Количество призов и имя текстового файла с протоколом игр.

Описание выходных данных. Программа должна вывести имена и результаты K лучших игроков в порядке занятых мест по форме, приведённой ниже в примере. Если игроков окажется меньше K, нужно вывести данные обо всех игроках.

Пример входных данных:

```
Количество призов: 3
Имя файла: 26mar2020
```

Пример выходных данных:

```
Количество строк протокола: 51
Первые десять записей
59485 Жужа
97523 Пончик
85665 КотБатон
83647 МистерИкс
107032 Пончик
95715 Жилетик
94449 КотБатон
96517 КотБатон
95715 МистерИкс
12675 Капитан

Победители
1. Пончик (107032)
2. КотБатон (96517)
3. Жилетик (95715)
```

Задания для команды студентов (3 команды в подгруппе):

1. Написать программу, демонстрируя работу со словарями. Списки можно добавлять только в крайних случаях и, возможно, только на первом этапе чтения данных их файла.
2. Сделать не менее трёх файлов протокола для тестирования программы.
3. Сделать несколько логотипов для игроков чемпионата.

2.5. Чемпионат по игре «Черепашьи бега» проводится по следующим правилам:

1. Каждый участник регистрируется на сайте игры под некоторым именем. Имена всех участников разные.

2. Чемпионат проводится в течение определённого времени. Во время проведения чемпионата зарегистрированные участники имеют право играть любое количество раз. По окончании каждой игры количество набранных очков участника заносится в протокол. Каждая строка протокола описывает одну игру и имеет формат:

<Количество баллов> <Имя игрока>

<Количество баллов> — целое положительное число, не превышающее 100 миллионов; <Имя игрока> — строка, не содержащая пробелов.

Строки в протоколе игры расположены в порядке проведения игр.

3. Окончательный результат участника — это лучший для участника результат.

4. Более высокое место в соревнованиях занимает участник, набравший наибольшее количество очков. При равенстве результатов более высокое место занимает участник, раньше показавший лучший результат.

Написать программу, которая по данным протокола определяет
— лучший результат (выводится имя лучшего игрока и его балл)
— худший результат (выводится имя худшего игрока и его балл)
— средний балл всех игр
— имя игрока, игравшего чаще всех (если таких игроков несколько, то вывести все имена)

Описание входных данных. Количество строк протокола и сами строки протокола.

Пример входных данных:

```
Количество игроков: 5
94449 КотБатон
95715 МистерИкс
107032 Пончик
95715 Жилетик
96517 КотБатон
```

Пример выходных данных:

```
Лучший результат у игрока: Пончик (107032)
Худший результат у игрока: КотБатон (94449)
Средний балл всех игр: ...
Имя игрока, игравшего чаще всех: КотБатон
```

Задания для команды студентов (3 команды в подгруппе):

1. Написать программу, демонстрируя работу со словарями. Списки можно добавлять только в крайних случаях и, возможно, только на первом этапе чтения данных их файла.

2. Сделать не менее трёх файлов протокола для тестирования программы.

3 Файлы

Работа с файлом делится на три этапа:

- Открытие файла
- Выполнение операций (запись, чтение)
- Закрытие файла

3.1 Открытие файла

Открыть файл, значит указать имя файла и способ доступа к данным.

Синтаксис функции open()

```
myFile = open(имя_файла[, режим_доступа])
```

myFile — имя переменной, через которую будем работать с файлом,

имя_файла — строка, содержащая имя файла с расширением (путь к файлу может быть относительным или абсолютным),

режим_доступа — строка, в которой указывается для чего открывается файл (чтение, запись, добавление информации и т.д. Режим по умолчанию — чтение).

Функция open() при вызове создаёт объект типа «файл», с которым дальше производится работа.

О новом файловом объекте можно получить следующую информацию:

Метод	Возвращаемый результат
<file>.closed	True, если файл был закрыт.
<file>.mode	режим доступа к файлу.
<file>.name	имя файла.

здесь <file> — имя файловой переменной

Пример 3.1. Вывод имени открытого файла и режима доступа к нему.

Пример использования методов файлового объекта

```
myF = open("File_No1.txt")
print("Имя файла:", myF.name)
print("Режим доступа:", myF.mode)
```

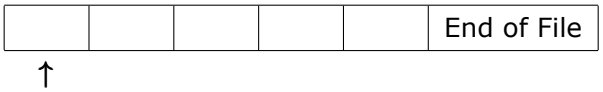
Результат

```
Имя файла: File_No1.txt
Режим доступа: r
```

3.1.1 Режимы доступа к файлу

r — открывает файл только для чтения (режим по умолчанию).

↑: в начале файла.



Файл отсутствует ⇒ ошибка (No such file or directory).

rb — открывает файл для чтения в двоичном формате;

r+ — открывает на чтение и запись;

rb+ — открывает на чтение и запись в двоичном формате;

Пример 3.2. Открытие существующего файла.

Исходный файл

```
Мне нравится Python!
И точка!
```

Открытие существующего файла (r)

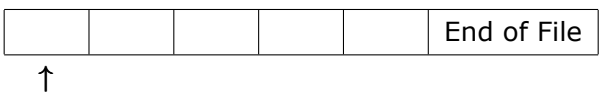
```
myF = open("FileNo40.txt", "r")
print(myF.read())   # чтение из файла: ok
myF.close()
```

Результат работы программы

```
Мне нравится Python!
И точка!
```

w — открывает файл только для записи.

↑: в начале файла.



Файл существует ⇒ стирается содержимое.

Файл отсутствует ⇒ создаётся новый файл с указанным именем.

wb — открывает файл для записи в двоичном формате;

w+ — открывает на чтение и запись;

wb+ — открывает на чтение и запись в двоичном формате;

Пример 3.3. Открытие существующего файла для записи приведёт к перезаписи файла.

Исходный файл

```
Весна
Цветы
```

Запись в файл (w)

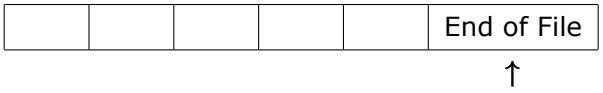
```
myF = open("FileNo40.txt", "w")
myF.write("Зима")   # запись в файл: ok
myF.write("\nСнег\n")
myF.close()
```

Содержимое изменённого файла

```
Зима
Снег
```

a — открывает файл для добавления.

↑: в конце файла.



Файл отсутствует ⇒ создаётся новый файл с указанным именем.

ab — открывает файл для записи в двоичном формате;

a+ — открывает на чтение и запись;

ab+ — открывает на чтение и запись в двоичном формате;

3.2 Закрытие файла

Синтаксис метода close()

```
<имя файлового объекта>.close()
```

Работать с файлом (читать, записывать) после закрытия нельзя.

Python автоматически закрывает файл, если файловый объект, к которому он привязан, присваивается другому файлу.

Хорошая практика: вручную закрывать файл командой close()

```
myF = open("File_No_2.txt")
print("Файл закрыт:", myF.closed)   # Файл закрыт: False
myF.close()
print("Файл закрыт:", myF.closed)   # Файл закрыт: True
```

3.3 Чтение из файла

Синтаксис метода read()

```
<имя файлового объекта>.read([число компонентов])
```

Результат — строка.

Если число компонентов файла не указано, то читается весь файл целиком.

Чтение всего файла построчно в цикле

```
for line in <имя файлового объекта>:
    line  # line -  строка в файле
```

Компонентами текстового файла являются символы.

Пример 3.4. Пусть имеется текстовый файл FileNo1.txt.

Дан файл FileNo1.txt

```
213 456 7
stroka
```

Чтение всего файла

```
myF = open("FileNo1.txt", "r")

a = myF.read()
print(a)

myF.close()
```

Результат

```
213 456 7
stroka
```

Пример 3.5. Пусть имеется текстовый файл FileNo1.txt.

Дан файл FileNo1.txt

```
213 456 7
stroka
```

Чтение фрагментов файла

```
myF = open("FileNo1.txt", "r")
a = myF.read(1)
print(a)

a = myF.read(10)
print(a)

myF.close()
```

Результат

```
2
13 456 7
s
```

Пример 3.6. Пусть имеется текстовый файл FileNo1.txt.

Дан файл FileNo1.txt

```
213 456 7
stroka
```

Чтение всего файла построчно

```
myF = open("FileNo1.txt")
for line in myF:
    print(line)

myF.close()
```

Результат

```
213 456 7

stroka
```

Пример 3.7. Дан текстовый файл FileNo2.txt, содержащий вещественные числа:

FileNo2.txt

```
1.0
-0.7
6.3
0.0
5.2
-11.8
```

Чтение числовых данных из текстового файла

```
myF = open("FileNo2.txt")

# сохранили данные из файла в строку
_str = myF.read()

myF.close()

print('Дан файл:')
print(_str)

_num = list(map(float, _str.split('\n')))
print('Представление данных из файла в виде списка:')
print(_num)
```

Результат работы программы

```
Дан файл:
1.0
-0.7
6.3
0.0
5.2
-11.8
Представление данных из файла в виде списка:
[1.0, -0.7, 6.3, 0.0, 5.2, -11.8]
```

3.4 Запись в файл

Синтаксис метода write()

```
<файловый объект>.write(<данные>)
```

Метод write() записывает <данные> в <файловый объект>.

Возвращаемый результат: количество добавленных компонентов.

Метод write() не добавляет символ переноса строки ('\n') в конец файла.

Пример 1

```
myF = open("FileNo3.txt", "w")

myF.write("Мне нравится Python!")
myF.write("И точка!")

myF.close()
```

Результат: текстовый файл с одной строкой

```
Мне нравитсяPython!И точка!
```

Пример 2

```
myF = open("FileNo4.txt", "w")

myF.write("Мне нравится Python!")
myF.write("\nИ точка!\n")

myF.close()
```

Результат: файл с тремя строкам (третья строка пустая)

```
Мне нравится Python!
И точка!

```

3.5 Методы readline() и readlines()

Используются для больших файлов.

Синтаксис метода readline()

```
<файловый объект>.readline([<кол-во компонентов>])
```

<файловый объект>.readline() — чтение текущей строки

<файловый объект>.readline(<кол-во компонентов>) — чтение из текущей строки заданного количества компонентов.

Синтаксис метода readlines()

```
<файловый объект>.readlines()
```

Прочитать все строки.

Возвращаемый результат: список строк.

Пример 3.8. Дан текстовый файл FileNo50.txt.

FileNo50.txt

```
Первая строка
Вторая строка
Третья строка
```

Демонстрация метода readline()

```
myF = open("FileNo50.txt")

line1 = myF.readline()    # чтение I строки из файла
print(line1, end = "")

# чтение 5 символов из II строки файла
line11 = myF.readline(5)
print(line11, end = "")

myF.close()
```

Результат работы программы

```
Первая строка
Втора
```

Демонстрация метода readlines()

```
myF = open("FileNo50.txt")

lines = myF.readlines()    # чтение всех строк из файла
print(lines[1], end = "")  # Вторая строка

print(lines, end = "")     # Список всех строк

myF.close()
```

Результат работы программы

```
Вторая строка
['Первая строка\n', 'Вторая строка\n', 'Третья строка']
```

3.6 Методы writelines()

Синтаксис метода writelines()

```
<файловый объект>.writelines(<lines>)
```

Возвращаемый результат: None

<lines> — любой объект, поддерживающий итерирование и производящий строки.

Пример writelines()

```
myF = open("FileNo7.txt", "w")
myF.writelines(["cat", "dog", "mouse"])
myF.close()
```

Результат: файл с одной строкой

```
catdogmouse
```

Метод не добавляет разделители строк автоматически.

Пример writelines()

```
myF = open("FileNo7.txt", "w")

myF.writelines(["cat", "dog", "mouse"])
myF.writelines("racoon")  # в код добавлена строка

myF.close()
```

Результат: файл с одной строкой

```
catdogmouseracoon
```

Пример ошибочного оператора: myF.writelines(345)

```
TypeError: 'int' object is not iterable
```

Пример ошибочного оператора: myF.writelines([345, 34])

```
TypeError: write() argument must be str, not int
```

3.7 Методы tell() и seek()

tell() текущая позиция в файле.

seek() изменить текущую позицию.

Для использования этих методов файл должен быть открыт.

Синтаксис метода tell()

```
<файловый объект>.tell()
```

Синтаксис метода seek()

```
<файловый объект>.seek(offset, [from])
```

offset — количество позиций, на которое нужно сдвинуться.

from — позиция, с которой начинается отсчёт: 0 — начало файла, 1 — текущая позиция, 2 — конец файла.

Демонстрация tell()

```
myF = open("FileNo1.txt")

print('Было прочитано:', myF.read(7))
print('Указатель находится на позиции:', myF.tell())

my_file.close()
```

Дан файл со строкой:

Мне нравится программирование

Результат работы

```
Было прочитано: Мне нра
Указатель находится на позиции: 7
```

3.8 Метод truncate()

file.truncate([<арг>])

уменьшает размер файл.

Если аргумент указан, то файл обрезается до <арг> байт, если нет — до текущей позиции.

3.9 Задачи

Указание. При выполнении заданий продемонстрируйте как можно больше методов для работы с файлами.

3.1. Дан текстовый файл, содержащий строки. Найти:

а) число строк;

б) номер самой короткой строки;

в) число строк файла, начинающихся и заканчивающихся одинаковыми символами.

Программа должна вывести на экран три числа — каждое в отдельной строке.

3.2. Дан текстовый файл. Вставить в начало каждой строки ее номер и записать преобразованные строки в новый файл. В конце нового файла выдать общее количество строк и количество пустых строк.

3.3. Дан текстовый файл, в котором хранятся целые числа. Найти в каждой строке файла максимальное число и записать его в новый текстовый файл (все числа должны записываться в одну строку через пробел).

На экран выдать содержимое нового текстового файла и количество строк в исходном файле.

3.4. Дан текстовый файл, содержащий только целые числа, в каждой строке может быть несколько чисел, которые разделяются пробелами. Дописать в его конец следующие данные: число строк, число символов в каждой строке, число элементов в каждой строке.

3.5. Написать программу, которая бы проверяла на равенство два текстовых файла. Результат проверки («файлы совпадают» или «файлы не совпадают») записать в новый файл.

3.6. Даны два текстовых файла. Записать в третий файл только те строки, которые есть и в первом, и во втором файле.

3.7. Дан текстовый файл, содержащий программу на языке Паскаль. Считать, что каждый оператор программы занимает не более одной строки файла. Проверить программу на соответствие числа открывающих и закрывающих круглых скобок.

Указание. Результат работы программы выдать на экран в виде: сначала пронумерованные строки программного кода из исходного файла, затем сообщения о номерах строк, содержащих ошибки.

Результат работы программы в случае ошибок

```
 1: program TestFiles;
 2: procedure PrintFile(var FT: Text);
 3: var s: string;
 4: begin
 5:   ResetFT);
 6:   while not Eof(FT) do
 7:   begin
 8:     Readln(FT, s;
 9:     Writeln(s);
10:   end;
11:   Close(FT);
12: end;
13: var F0 : Text;
14: begin
15:   Assign(F0, 'TestTT.txt'(;
16:   PrintFile(F0);
17: end.
Error in 5
Error in 8
Error in 15
File not is correctly
```

3.8. Дан текстовый файл, содержащий программу на языке Паскаль. Считать, что каждый оператор программы занимает не более одной строки файла.

Требуется: а) проверить программу на соответствие числа открывающих и закрывающих круглых скобок; б) рассмотреть случай неверной расстановки скобок, например, `Writeln)(`.

Указание. Результат работы программы выдать на экран в виде: сначала пронумерованные строки программного кода из исходного файла, затем сообщения о номерах строк, содержащих ошибки.

Результат работы программы в случае ошибок

```
 1: program TestFiles;
 2: procedure PrintFile(var FT: Text);
 3: var s: string;
 4: begin
 5:   ResetFT);
 6:   while not Eof)FT( do
 7:   begin
 8:     Readln(FT, s;
 9:     Writeln(s);
10:   end;
11:   Close((FT);
12: end;
13: var F0 : Text;
14: begin
15:   Assign(F0, 'TestTT.txt';
16:   PrintFile(F0);
17: end.
Error in 5
Error in 6
Error in 8
Error in 11
Error in 15
File not is correctly
```

3.9. Дан текстовый файл f1.txt, содержащий операторы, записанные на языке Python. Порядок строк в файле перепутан. Пример файла см. ниже.

Файл f1.txt

```
    for i in range(s):
            print(i)
        if i != 'k':
def f(s):
```

Считать, что каждый оператор программы занимает не более одной строки файла. Отступ в начале каждой строки разный.

Написать программу, создающую новый файл с верным порядком строк.

Файл f2.txt

```
def f(s):
    for i in range(s):
        if i != 'k':
            print(i)
```

Указания. 1) Используйте словарь с элементами:

число пробелов в строке : строка

2) Работа со списком строк запрещена, но можно использовать список ключей словаря.

3) Придумайте свой вариант входного файла, содержащий большее количество строк.

3.10. Дан текстовый файл, содержащий описание функций на языке Python. Пример файла приведён ниже.

Файл zf2.py

```
def s(x: int, y: int) -> int:
    '''
    s(x, y)
    Функция возвращает сумму целых чисел x и y.
    '''
    return x + y

def d(x: int) -> int:
    '''
    d(x)
    Функция возвращает произведение целого числа x на 2.
    '''
    return 2 * x
```

Написать программу, формирующую файл с документацией для подобного модуля.

Файл zf2doc.txt (пример выходного файла)

```
Имя файла: zf2.py
Количество функций: 2

Описания функций
1. s(x, y)
Функция возвращает сумму целых чисел x и y.

Аннотация типов функции s():
{'x': <class 'int'>, 'y': <class 'int'>, 'return': <class 'int'>}

2. d(x)
Функция возвращает произведение целого числа x на 2.

Аннотация типов функции d():
{'x': <class 'int'>, 'return': <class 'int'>}
```

Указания. 1) Добавьте в исходный файл библиотеки описания любых двух новых функций: одну для работы со списками, вторую — для работы со словарями.

2) Используйте аннотации в функциях. Атрибут `__annotations__` даёт доступ к использованным в функции аннотациям

```
<имя функции>.__annotations__
```

Результат предоставляется в виде словаря, где ключами являются атрибуты, а значениями — аннотации. Возвращаемое функцией значение хранится в записи с ключом return (см. пример выше).