

## Файлы, чтение и обработка.

Во всех задачах ввод-вывод осуществляется только посредством чтения-записи файлов. Файл со входными данными называется `input.txt`, с выходными — `output.txt`.

Во входных данных всех задач можно рассчитывать на то, что **все** строки файла, включая последнюю, заканчиваются переносом строки (символом `'\n'`).

Словом во всех задачах называется последовательность прописных или строчных латинских букв, ограниченная слева и справа символом не-буквой или началом (концом) строки.

Если в файле есть русские буквы (и вообще любой символ с кодом, большим 127), то необходимо при открытии файла указывать кодировку:

```
data = open('input.txt', 'r', encoding='utf-8')
```

### Открытие файла

Для каждого файла, с которым необходимо производить операции ввода-вывода, нужно связать специальный объект - поток. Открытие файла осуществляется функцией `open`, которой нужно передать два параметра. Первый параметр (можно также использовать именованный параметр `file`) имеет значение типа `str`, в котором записано имя открываемого файла. Второй параметр (можно также использовать именованный параметр `mode`) — это значение типа `str`, которое равно `'r'`, если файл открывается для чтения данных (`read`), `'w'`, если на запись (`write`), при этом содержимое файла очищается, и `'a'` — для добавления данных в конец файла (`append`). Если второй параметр не задан, то считается, что файл открывается в режиме чтения.

Функция `open` возвращает ссылку на файловый объект, которую нужно записать в переменную, чтобы потом через данный объект использовать методы ввода-вывода. Например:

```
input = open('input.txt', 'r')
output = open('output.txt', 'w')
```

### Чтение данных из файла

Для файла, открытого на чтение данных, можно вызывать следующие методы, позволяющие читать из него данные. Метод `readline()` считывает одну строку из файла (до символа конца строки `'\n'`, возвращается считанная строка вместе с символом `'\n'`. Если считывание не было успешно (достигнут конец файла), то возвращается пустая строка. Для удаления символа `'\n'` из конца файла удобно использовать метод строки `rstrip()`. Например: `s = s.rstrip()`.

Метод `readlines()` считывает все строки из файла и возвращает список из всех считанных строк (одна строка — один элемент списка). При этом символы `\n` остаются в концах строк.

Метод `read()` считывает всё содержимое из файла и возвращает строку, которая может содержать символы `'\n'`. Если методу `read` передать целочисленный параметр, то будет считано не более заданного количества символов. Например, считывать файл побайтово можно при помощи метода `read(1)`.

### Вывод данных в файл

Данные выводятся в файл при помощи метода `write`, которому в качестве параметра передается одна строка. Этот метод не выводит символ конца строки `'\n'` (как это делает функция `print` при стандартном выводе), поэтому для перехода на новую строку в файле необходимо явно вывести символ `'\n'`.

Также можно выводить данные в файл при помощи функции `print`, если передать ей еще один именованный параметр `file`, равный ссылке на открытый файл. Например:

```
output = open('output.txt', 'w')
print(a, b, c, file = output)
```

### Заккрытие файла

После окончания работы с файлом необходимо закрыть его при помощи метода `close()`. Вот так:

```
f.close()
```

Лучше же всего открывать файлы при помощи следующей конструкции, которая гарантирует автоматическое закрытие файла (или файлов), открытых в её заголовке.

```
with open('input.txt') as fin:
    <блок операторов с отступом>
```

Более того, в указанном заголовке можно открывать сразу несколько файлов:

```
with open('input.txt') as fin, open('output.txt', 'w') as fout:
    <блок операторов с отступом>
```

## Пример

Следующая программа считывает все содержимое файла `input.txt`, записывает его в переменную `s`, а затем выводит ее в файл `output.txt`.

```
with open('input.txt', 'r') as fin:
    s = fin.read()
with open('output.txt', 'w') as fout:
    fout.write(s)
```

Так можно читать файл построчно:

```
with open('input.txt', 'r') as fin, open('output.txt', 'w') as fout:
    line = fin.readline()
    while line != '':
        fout.write(line)
        line = fin.readline()
```

Существует и другой способ обработать файл построчно, более удобный:

```
with open('input.txt', 'r') as fin, open('output.txt', 'w') as fout:
    for line in fin:
        fout.write(line)
```

А вот аналогичная программа, но читающая данные уже посимвольно:

```
with open('input.txt', 'r') as fin, open('output.txt', 'w') as fout:
    c = fin.read(1)
    while c != '':
        fout.write(c)
        c = fin.read(1)
```

Термин АСГТ-строка означает строку произвольной длины, состоящую только из заглавных символов А, С, G и T.

### A. Количество строк с символами

Дан файл. Вывести количество строк, где есть хотя бы один символ, отличный от переноса строки.

Input	Output
this is a  file	4

### B. Самая короткая строка

Дан файл с непустыми АСГТ-строками разной длины. Вывести самую короткую строку и её длину.

Input	Output
ACGT TGCAT AAAAAA TT	TT 2

### C. Первая встреченная длинная строка

Дан файл с непустыми АСГТ-строками. Вывести самую длинную строку. Если строк наибольшей длины несколько, вывести ту, которая встретилась раньше.

Input	Output
TT TGTT ACGT TGT TT	TGTT

### D. Наименьший триплет

Дан файл со строками, содержащими только заглавные латинские буквы и переносы строк.

Вывести наименьший в лексикографическом порядке триплет из латинских букв, содержащийся в данном файле. Триплеты могут переноситься на следующую строку (см. пример).

Input	Output
SGBG B SH YT	BGB

Е. *Наименьшая длинная строка*

Дан файл с непустыми ACGT-строками. Вывести самую длинную строку и её длину. Если строк наибольшей длины несколько, вывести наименьшую в лексикографическом порядке.

Input	Output
TT GCGT TGCA CCTC AAA	CCTC 4

Ф. *Все комплементарные дополнения*

Дан файл с ACGT-строками. Вывести все строки, но пары символов C и G, а также A и T поменяны местами, после чего строки развёрнуты задом наперёд.

Input	Output
ACGT GTTA	ACGT ТААС

Г. *ACGT-статистика.*

Дан файл, содержащий ACGT-последовательности. Для каждой строки вывести 4 числа: количество символов A, C, G и T соответственно.

Input	Output
ACGTGTTA GACGCGAA	2 1 2 3 3 2 3 0

Н. *Вычисление GC-содержания.*

Дан файл, содержащий ACGT-последовательности. Вывести строки с максимальным суммарным количеством букв C и G в том порядке, в котором они были указаны во входном файле.

Input	Output
ACGCATG TGACTG AGCTGCA CGTC	ACGCATG AGCTGCA

Комментарий к тесту: указанные строки содержат в сумме 4 символа C и G, а остальные меньше.

И. *Вычисление относительного GC-содержания.*

Дан файл, содержащий ACGT-последовательности. Вывести строки с максимальным отношением суммарного количества букв C и G к длине строки (т.н. GC-content).

Input	Output
ACGCATG AGCTGCA TGACTG CGTC	CGTC
GCT AAAAAC ACGAGC TC	GCT ACGAGC

Комментарий к первому тесту: указанная строка содержит в сумме 3 символа C и G, длина строки 4 символа, таким образом отношение равно 0.75. Оно максимальное среди указанных строк.

J. *FASTA-файлы. Заголовки.*

В биоинформатике принято использовать специфический формат хранения информации о белках. Он называется FASTA-формат. Выглядит это примерно так:

```
>MCHU - Calmodulin - Human, rabbit, bovine, rat, and chicken
ADQLTEEQIAEFKEAFSLFDKGDGTITTKELGTVMRSLGQNPTEAELQDMINEVDADGNGTID
FPEFLTMMARKMKDTSDEEEIREAFRVFDKDGNGYISAAELRHVMTNLGEKLTDEEVDEMIREA
DIDGDGQVNYEEFVQMMТАК
```

```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYITIKDFLG
LLILILLLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFPLIAGX
IENY
```

Более формально: файл состоит из произвольного количества блоков. Каждый блок начинается со строки-описания (description line), первый символ которой знак “больше” (>). Далее в блоке записана строка, содержащая только заглавные латинские буквы, возможно разделённая на несколько частей (каждая часть записывается отдельной строкой).

Дан файл в FASTA-формате. Вывести строки-описания без символа >.

Input	Output
>first protein ACGT GTТА	first protein second protein
>second protein TGAC GCGAA	

K. *FASTA-файлы. Строки и их длины.*

Дан файл в FASTA-формате. Вывести строки и их длины в том порядке, в котором они указаны в FASTA-файле.

Input	Output
>first protein ACGT GTТА	ACGTGTТА 8 TGACGCGAA 9
>second protein TGAC GCGAA	

L. *FASTA-файлы. Короткая и длинная строки.*

Дан файл в FASTA-формате. Вывести самую длинную и самую короткую строки с их названиями. Если самая длинная (короткая) не одна, выбрать ту, которая появилась в файле раньше.

Input	Output
>first protein ACGT ATG	>third protein TGACTG
>second protein AGCT GCA	>first protein ACGTATG
>third protein TGA CT G	
>fourth protein AC GT AT C	

M. *FASTA-файлы. Все длинные строки.*

Дан файл в FASTA-формате. Вывести все самые длинные строки с их названиями в том порядке, в котором они были перечислены во входном файле.

Input	Output
>first protein ACGT ATG	>first protein ACGTATG
>second protein AGCT GCA	>second protein AGCTGCA
>third protein TGA CT G	>fourth protein ACGTATC
>fourth protein AC GT AT C	

N. *FASTA-файлы. Переформатирование.*

Дан файл в FASTA-формате. К заголовку каждого блока добавлена запись одного натурального числа  $N$ , отделённая от самого заголовка пробелом.

Требуется вывести тот же файл в FASTA-формате, убрав запись числа  $N$  с отделяющим пробелом, а строку в каждом блоке вывести, “нарезав” по  $N$  символов в каждой части (см. пример).

Input	Output
>first protein 3 AFNQIJRG HJQE JEQ J	>first protein AFN QIJ RGH JQE JEQ J
>second protein 2 AGCT GCA	>second protein AG CT GC
>third protein 50 TGA CT G	A
>fourth protein 1 CG TC	>third protein TGACTG >fourth protein C G T C

O. *FASTA-файлы. Близкие строки.*

Расстоянием Хэмминга, определённым для двух строк одинаковой длины называется количество разных символов, имеющих одинаковые индексы в обеих строках. Например расстояние Хэмминга между строками  $s_1 = \text{'ACCGAGT'}$  и  $s_2 = \text{'ACAGAGG'}$  равно 2, так как  $s_1[2] \neq s_2[2]$  и  $s_1[6] \neq s_2[6]$ , а остальные символы попарно равны.

Дан файл в FASTA-формате, содержащий строки одинаковой длины. Найдите две строки, расстояние Хэмминга между которыми минимальное.

Программа должна вывести минимальное расстояние, а в следующих двух строках соответствующие строки входного файла. Если таких строк несколько — вывести две любые из них. Гарантируется, что входной файл содержит не менее двух строк в FASTA-формате.

Input	Output
>first protein ABCD EFGHI JKZ	1 ZBCDEFGHIJKZ ZBCDEOGHIJKZ
>second protein ZBCDEFGHIJKZ	
>third protein ZBCDE OGHIJKZ	
>forth protein ZBCDEOGHIJ KX	