[Списки в Python](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html" \l "id13)

Большинство программ работает не с отдельными переменными, а с набором переменных. Например, программа может обрабатывать информацию об учащихся класса, считывая список учащихся с клавиатуры или из файла, при этом изменение количества учащихся в классе не должно требовать модификации исходного кода программы.

Раньше мы сталкивались с задачей обработки элементов последовательности, например, вычисляя наибольший элемент последовательности. Но при этом мы не сохраняли всю последовательность в памяти компьютера, однако, во многих задачах нужно именно сохранять всю последовательность, например, если бы нам требовалось вывести все элементы последовательности в возрастающем порядке («отсортировать последовательность»).

Для хранения таких данных можно использовать структуру данных, называемую в Питоне список (в большинстве же языков программирования используется другой термин — «массив»). Список представляет собой последовательность элементов, пронумерованных от 0. Список можно задать перечислением элементов в квадратных скобках, например, список можно задать так:

primes = [2, 3, 5, 7, 11, 13]

Rainbow = ['Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Indigo', 'Violet']

В списке primes — 6 элементов: .. code-block:: python

*>>> primes = [2, 3, 5, 7, 11, 13]*

*>>> print(primes[0])*

*2*

*>>> print(primes[1])*

*3*

*>>> print(primes[2])*

*5*

*>>> print(len(primes))*

*6*

Список rainbow состоит из 7 элементов, каждый из которых является строкой.

Также как и символы строки, элементы списка можно индексировать отрицательными числами с конца, например,primes[-1] == 13, primes[-6] == 2.

Длину списка, то есть количество элементов в нем, можно узнать при помощи функции len, например, len(A) == 6.

Рассмотрим несколько способов создания и считывания списков. Пустой, т.е. не имеющий элементов список, можно создать следующим образом:

A = []

Для добавления элементов в конец списка используется метод append. Если программа получает на вход количество элементов в списке n, а потом n элементов списка по одному в отдельной строке, то организовать считывание списка можно так:

A = []

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

A.append(int(input())

В этом примере создается пустой список, далее считывается количество элементов в списке, затем по одному считываются элементы списка и добавляются в его конец.

Для списков целиком определены следующие операции: конкатенация списков (добавление одного списка в конец другого) и повторение списков (умножение списка на число). Например:

A = [1, 2, 3]

B = [4, 5]

C = A + B

D = B \* 3

В результате список C будет равен [1, 2, 3, 4, 5], а список D будет равен [4, 5, 4, 5, 4, 5]. Это позволяет по-другому организовать процесс считывания списков: сначала считать размер списка и создать список из нужного числа элементов, затем организовать цикл по переменной i начиная с числа 0 и внутри цикла считывается i-й элемент списка:

A = [0] \* int(input())

**for** i **in** range(len(A)):

A[i] = int(input())

Вывести элементы списка A можно одной инструкцией print(A), при этом будут выведены квадратные скобки вокруг элементов списка и запятые между элементами списка. Такой вывод неудобен, чаще требуется просто вывести все элементы списка в одну строку или по одному элементу в строке. Приведем два примера, также отличающиеся организацией цикла:

**for** i **in** range(len(A)):

**print**(A[i])

Здесь в цикле меняется индекс элемента i, затем выводится элемент списка с индексом i.

**for** elem **in** A:

**print**(elem, end = ' ')

В этом примере элементы списка выводятся в одну строку, разделенные пробелом, при этом в цикле меняется не индекс элемента списка, а само значение переменной. Например, в цикле for elem in ['red', 'green', 'blue'] переменная elemбудет последовательно принимать значения 'red', 'green', 'blue'.

Внутри одного списка могут быть любые объекты (и даже вперемешку), поэтому такая конструкция как список списков вполне осмысленна (аналог двумерного массива). Обращаться к элементам внутри такого списка нужно так A[i][j] , где j - индекс внутри внутреннего списка, i - индекс внутри внешнего списка. Но обратите внимание на следующую вещь:

A = [[0] \* 10]\*10 *# вроде бы это обычный список списков 10х10 состоящий из 0*

A[0][0] = 1 *# меняем элемент с индексом 0 в списке с индексом 0*

**print**(A[1][0]) *# печатаем элемент с индексом 0 в списке с индексом 1*

Что вывела программа? Как можно это объяснить? Попробуйте напечатать A целиком.

[Методы split и join](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id14)

Выше мы рассмотрели пример считывания списка, когда каждый элемент расположен на отдельной строке. Иногда бывает удобно задать все элементы списка при помощи одной строки. В такой случае используется метод split, определённый в строковом типе:

A = input().split()

Если при запуске этой программы ввести строку 1 2 3, то список A будет равен ['1', '2', '3']. Обратите внимание, что список будет состоять из строк, а не из чисел. Если хочется получить список именно из чисел, то можно затем элементы списка по одному преобразовать в числа:

**for** i **in** range(len(A)):

A[i] = int(A[i])

Используя функции языка map и list то же самое можно сделать в одну строку:

A = list(map(int, input().split()))

Объяснений, как работает этот пример, пока не будет. Если нужно считать список действительных чисел, то нужно заменить тип int на тип float.

У метода split есть необязательный параметр, который определяет, какая строка будет использоваться в качестве разделителя между элементами списка. Например, вызов метода split('.') для строки вернет список, полученный разрезанием этой строки по символам '.'.

Используя «обратные» методы можно вывести список при помощи однострочной команды. Для этого используется метод строки join. У этого метода один параметр: список строк. В результате создаётся строка, полученная соединением элементов списка (которые переданы в качестве параметра) в одну строку, при этом между элементами списка вставляется разделитель, равный той строке, к которой применяется метод. Например, программа

A = ['red', 'green', 'blue']

**print**(' '.join(A))

**print**(''.join(A))

**print**('\*\*\*'.join(A))

выведет строки red green blue, redgreenblue и red\*\*\*green\*\*\*blue. Обратите внимание, что join является методом **строки**, а не списка.

Если же список состоит из чисел, то придется использовать еще и функцию map. То есть вывести элементы списка чисел, разделяя их пробелами, можно так:

**print**(' '.join(map(str, A)))

[Срезы списков](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id15)

Со списками, так же как и со строками, можно делать срезы. А именно:

|  |  |
| --- | --- |
| A[i:j] | срез из j-i элементов A[i], A[i+1], ..., A[j-1]. |
| A[i:j:-1] | срез из i-j элементов A[i], A[i-1], ..., A[j+1] (то есть меняется порядок элементов). |
| A[i:j:k] | срез с шагом k: A[i], A[i+k], A[i+2\*k],... . Если значение k меньше 0, то элементы идут в противоположном порядке. |

Каждое из чисел i или j может отсутствовать, что означает «начало строки» или «конец строки».

Списки, в отличие от строк, являются изменяемыми объектами: можно отдельному элементу списка присвоить новое значение. Но можно менять и целиком срезы. Например:

A = [1, 2, 3, 4, 5]

A[2:4] = [7, 8, 9]

Получится список, у которого вместо двух элементов среза A[2:4] вставлен новый список уже из трех элементов. Теперь список стал равен [1, 2, 7, 8, 9, 5].

A = [1, 2, 4, 5, 6, 7]

A[::-2] = [10, 20, 30, 40]

Получится список [40, 2, 30, 4, 20, 6, 10]. Здесь A[::-2] — это список из элементов A[-1], A[-3], A[-5], A[-7], которым присваиваются значения 10, 20, 30, 40 соответственно.

Если **не непрерывному** срезу (то есть срезу с шагом k, отличному от 1), присвоить новое значение, то количество элементов в старом и новом срезе обязательно должно совпадать, в противном случае произойдет ошибка ValueError.

Обратите внимание, A[i] — это **элемент** списка, а не срез!

[Генерация списков](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id16)

В питоне существует специальная синтаксическая конструкция, позволяющая создавать заполненные списки по определенным правилам. Создаваемые списки могут быть разными, содержание конструкции немного отличаться, поэтому такие конструкции называют генераторами списков (англ. - List comprehensions). Их удобство заключается в более короткой записи, чем если создавать список обычным способом. Расскажем вкратце об этой конструкции.

Например, надо создать список, заполненный натуральными числами до определенного числа. "Классический" способ будет выглядеть так:

a = []

**for** i **in** range(1,10):

a.append(i)

С помощью генераторов можно сделать это одной строкой:

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

Пример генерации списка квадратов четных натуральных чисел

a = [i\*\*2 **for** i **in** range(10) **if** i % 2 == 0]

Таким образом, генератору можно передавать следующую информацию:

1. Что делаем (возводим в квадрат).
2. Что берем (элемент i).
3. Откуда берем (из range(10), но можно сюда передать список или даже строку).
4. Условие (в генератор попадают только числа i, для которых выполнено i % 2 == 0 ).

Пример изменения типа всех элементов списка с помощью генератора:

a = ['12', '4', '151']

b = [int(i) **for** i **in** a]

[Операции со списками](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id17)

Со списками можно легко делать много разных операций.

| **Операция** | **Действие** |
| --- | --- |
| x in A | Проверить, содержится ли элемент в списке. Возвращает True или False. |
| x not in A | То же самое, что not(x in A). |
| min(A) | Наименьший элемент списка. Элементы списка могут быть числами или строками, для строк сравнение элементов проводится в лексикографическом порядке. |
| max(A) | Наибольший элемент списка. |
| sum(A) | Сумма элементов списка, элементы обязательно должны быть числами. |
| A.index(x) | Индекс первого вхождения элемента x в список, при его отсутствии генерирует исключение ValueError. |
| A.count(x) | Количество вхождений элемента x в список. |
| A.append(x) | Добавить в конец списка A элемент x. |
| A.insert(i, x) | Вставить в список A элемент x на позицию с индексом i. Элементы списка A, которые до вставки имели индексы i и больше сдвигаются вправо. |
| A.extend(B) | Добавить в конец списка A содержимое списка B. |
| A.pop() | Удалить из списка последний элемент, возвращается значение удаленного элемента. |
| A.pop(i) | Удалить из списка элемент с индексом i, возвращается значение удаленного элемента. Все элементы, стоящие правее удаленного, сдвигаются влево. |

[Упражнение №1. Однострочники.](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id18)

Каждая из задач должна быть решена в одну строку. Список чисел A уже введён.

1. Выведите элементы списка с чётными индексами.

| **Ввод** | **Вывод** |
| --- | --- |
| 1 2 3 4 5 | 1 3 5 |

1. Найдите наибольший элемент в списке. Выведите значение элемента и его индекс.

| **Ввод** | **Вывод** |
| --- | --- |
| 1 2 3 2 1 | 3 2 |

1. Выведите список в обратном порядке.

| **Ввод** | **Вывод** |
| --- | --- |
| 1 2 3 4 5 | 5 4 3 2 1 |

[Упражнение №2. Задачи посложнее.](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id19)

1. Переставьте соседние элементы в списке. Задача решается в три строки.

| **Ввод** | **Вывод** |
| --- | --- |
| 1 2 3 4 5 | 2 1 4 3 5 |

1. Выполните циклический сдвиг элементов списка вправо. Решите задачу в две строки.

| **Ввод** | **Вывод** |
| --- | --- |
| 1 2 3 4 5 | 5 1 2 3 4 |

1. Выведите элементы, которые встречаются в списке только один раз. Элементы нужно выводить в том порядке, в котором они встречаются в списке.

| **Ввод** | **Вывод** |
| --- | --- |
| 1 2 2 3 3 3 | 1 |

1. *В этой задаче****нельзя****модицифицировать список, использовать вспомогательные списки, строки, срезы.*
2. Определите, какое число в этом списке встречается чаще всего. Если таких чисел несколько, выведите любое из них.

| **Ввод** | **Вывод** |
| --- | --- |
| 1 2 3 2 3 3 | 3 |

1. *В этой задаче также****нельзя****модицифицировать список, использовать вспомогательные списки, строки, срезы.*

[Работа с текстовыми файлами в Python](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id20)

До этого для ввода информации мы использовали исключительно клавиатуру. При этом в большинстве случаев данные, считываемые программой, **уже** хранятся на носителе информации в виде **файлов**.

Для каждого файла, с которым необходимо производить операции ввода-вывода, нужно создать специальный объект – поток. Именно с потоками работают программы — использование такого дополнительного слоя **абстракции** позволяет прозрачно работать не только с текстовыми файлами, но и, например, с архивами.

[Открытие файла](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id21)

Открытие файла осуществляется функцией open, которой нужно передать два параметра. Первый параметр — строка, задающая имя открываемого файла. Второй параметр — строка, указывающая режим октрытия файла.

Существует три режима открытия файлов:

| **Режим** | **Описание** |
| --- | --- |
| "r" (read) | Файл открывается для чтения данных. |
| "w" (write) | Файл открываетсяна запись, при этом содержимое файла очищается. |
| "a" (append) | Файл открывается для добавления данных в конец файла. |

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 работать с этим файлом. Например:

file\_input = open('input.txt', 'r')

file\_output = open('output.txt', 'w')

Здесь открыто два файла (один на чтение, другой на запись) и создано два связанных с ними объекта.

[Чтение данных из файла](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id22)

Для файла, открытого на чтение данных, можно использовать несколько методов, позвозволяющих считывать данные. Мы рассмотрим три из них: readline, readlines, read.

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 все 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](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id23)

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.

Выводить данные в файл можно и при помощи print, если передать функции еще один именованный параметр file. Например:

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

**print**(a, b, c, file=output) *# через print*

output.write("Some string") *# через write*

[Закрытие файла](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id24)

После окончания работы с файлом необходимо закрыть его при помощи метода close().

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

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

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

s = input.read()

output.write(s)

input.close()

output.close()

А вот аналогичная программа, но читающая данные посимвольно:

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

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

c = input.read(1)

**while** len(c) > 0:

output.write(c)

c = input.read(1)

input.close()

output.close()

Так же работать с файлами можно при помощи конструкции with ... as :

**with** open('file\_name.txt', 'r') **as** f:

**for** line **in** f:

**print**(line)

[Цикл упражнений к практической работе](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id25)

[Упражнение №3](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id26)

Вывести список в следующем порядке: первое число, последнее, второе, предпоследнее и так далее все числа.

| **Ввод** | **Вывод** |
| --- | --- |
| 1 2 3 4 5 | 1 5 2 4 3 |

| **Ввод** | **Вывод** |
| --- | --- |
| 1 2 3 4 | 1 4 2 3 |

A[::2], A[1::2] = A[:(len(A) + 1)//2], A[(len(A) + 1)//2:][::-1]

**print**(A)

[Упражнение №4](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id27)

N кузнечиков стоят в ряд. Для каждого кузнечика задана числовая характеристика — длина его прыжка. Если длина прыжка кузнечика равна l, то он за один прыжок перепрыгивает через l других кузнечиков.

Каждую секунду последний кузнечик прыгает к началу ряда, перепрыгивает через столько кузнечиков, чему равна длина его прыжка, и становится между двумя другими кузнечиками.

В первой строке входных данных задана расстановка кузнечиков (длины их прыжков). Во второй строке входных данных задано число секунд t. Опеределите и выведите на экран расстановку кузнечиков через t секунд. Все длины прыжков — натуральные числа, меньшие, чем число кузнечиков в ряду.

Решите задачу в четыре строки.

| **Ввод** | **Вывод** |
| --- | --- |
| 1 2 3 4 2 | 4 1 2 2 3 |
| 2 |  |

[Упражнение №5](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id28)

Назовем последовательность чисел последовательностью k-боначчи, если каждый элемент этой последовательности является суммой k предыдущих членов последовательности. В частности, последовательность 2-боначчи является последовательностью Фибоначчи.

Более формально, i-й элемент последовательности ki равен 1, если 0≤i≤k-1, и равен сумме k предыдущих членов последовательности ki−1+ki−2+…+ki−k при i≥k.

Даны два числа k и n (k≥2, n≥0). Вычислите n-й член последовательности k-боначчи kn.

Решите задачу в пять строк.

| **Ввод** | **Вывод** |
| --- | --- |
| 3 6 | 17 |
| 100 0 | 1 |

[Упражнение №6](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id29)

В списке — нечетное число элементов, при этом все элементы различны. Найдите медиану списка: элемент, который стоял бы ровно посередине списка, если список отсортировать.

При решении этой задачи нельзя модифицировать данный список (в том числе и сортировать его), использовать вспомогательные списки.

Программа получает на вход нечетное число N, в следующей строке заданы N элементов списка через пробел.

Программа должна вывести единственное число — значение медианного элемента в списке.

| **Ввод** | **Вывод** |
| --- | --- |
| 7 | 4 |
| 6 1 9 2 3 4 8 |  |

[Упражнение №7](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id30)

Вася хочет узнать, какую оценку он получит в четверти по информатике. Учитель придерживается следующей системы: вычисляется среднее арифметическое всех оценок в журнале, и ставится ближайшая целая оценка, не превосходящая среднего арифметического.

При этом если у школьника есть двойка, а следующая за ней оценка – не двойка, то двойка считается закрытой, и при вычислении среднего арифметического не учитывается.

Вводится десять натуральных чисел от 2 до 5 через пробел – оценки Васи.

Выведите натуральное число (от 2 до 5) – его четвертную оценку.

| **Ввод** | **Вывод** |
| --- | --- |
| 2 5 2 5 2 5 2 5 2 5 | 5 |
| 2 2 2 2 2 2 2 2 2 5 | 2 |
| 5 5 5 5 5 5 5 5 5 2 | 4 |

[Упражнение №8](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id31)

Для изучения пассажиропотока в метро было записано время входа и время выхода в метро каждого пассажира. На основании этих данных определите, сколько пассажиров было в метро в некоторый заданный момент времени T.

Программа получает на вход число пассажиров N. Далее в N строчках записано время входа Ai и время выхода Biкаждого пассажира (Ai<Bi). Время задается в минутах от начала работы метрополитена.

В следующей строке дано время T.

Выведите одно число: количество пассажиров в момент времени T. Если какой-то пассажир в момент T входит или выходит, то его тоже необходимо посчитать.

| **Ввод** | **Вывод** |
| --- | --- |
| 4 | 3 |
| 3 12 |  |
| 8 9 |  |
| 5 10 |  |
| 10 12 |  |
| 10 |  |

[Упражнение №9](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id32)

Не без вашей помощи в метро посчитали количество пассажиров в каждый час работы метро. Теперь вас просят по этим данным найти «час пик»: такие k подряд идущих часов, что суммарное число пассажиров в эти часы максимальное.

Первая строка входных данных содержит количество часов в сутках, в течение которых работает метрополитен N (N≥1). Вторая строка содержит N неотрицательных чисел, записанных через пробел. В третьей строке записана продолжительность часа пик k (1≤k≤N).

Найдите k подряд идущих часов работы метрополитена с максимальным суммарным числом пассажиров и выведите суммарное число пассажиров за эти часы.

| **Ввод** | **Вывод** |
| --- | --- |
| 7 | 12 |
| 3 2 5 4 3 2 4 |  |
| 3 |  |

[Упражнение №10](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id33)

По данному числ N выведите первые N+1 строку [треугольника Паскаля](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA_%D0%9F%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D1%8F). Числа в строке разделяйте одним пробелом.

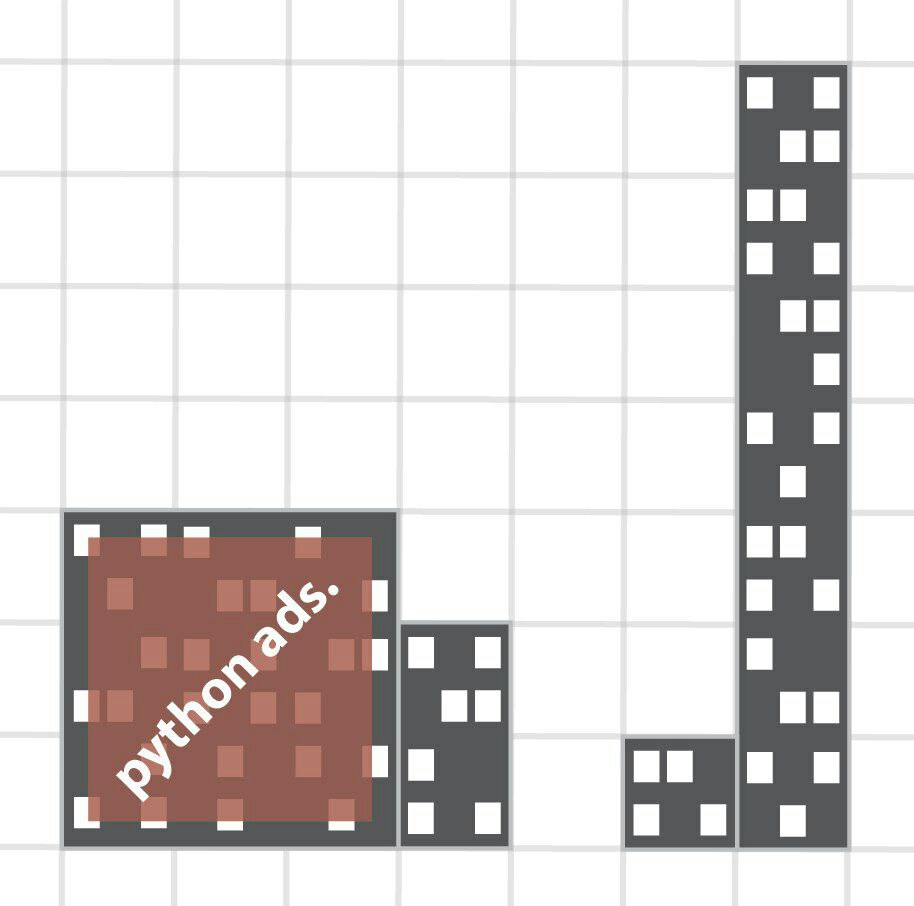
| **Ввод** | **Вывод** |
| --- | --- |
| 4 | 1 |
|  | 1 1 |
|  | 1 2 1 |
|  | 1 3 3 1 |
|  | 1 4 6 4 1 |

[Упражнение №11](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id34)

По данному числ N выведите сумму квадратов натуральных чисел, меньших N, дающих при делении на 3 остаток 1. Решите данную программу в 2 строки

| **Ввод** | **Вывод** |
| --- | --- |
| 7 | 17 |

[Упражнение №12 (небоскребы)](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab5.html#id35)



На улице вплотную друг к другу расположено N небоскребов различной высоты. Высота небоскребa может принимать целые неотрицательные значения (в т.ч. 0). Ширина каждого небоскреба равна 1.

Рекламодатель хочет повесит на стенах небоскребов одно прямоугольное объявление и при этом хочет, чтобы площадь этого объявления была максимальной. Необходимо найти максимальную площадь такого объявления и его высоту.

Высоты небоскребов находятся в файле input.txt , по одному небоскребу в каждой строке. Вывод необходимо сделать в файл output.txt, в первой строке площадь, во второй высота.

| **Ввод** | **Вывод** |
| --- | --- |
| 7 | 8 |
| 1 | 2 |
| 1 |  |
| 0 |  |
| 2 |  |
| 3 |  |
| 2 |  |
| 6 |  |