Государственное автономное нетиповое образовательное учреждение
Свердловской области «Дворец молодёжи»
Центр цифрового образования «IT-Куб»

ОСНОВЫ PYTHON

Учебно – методические материалы по программированию на языке Python для обучающихся программы «Нейронные сети, большие данные и кибергигиена»

Автор:
Бабикова Е. В.

Екатеринбург
2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Согласно анализу наиболее крупных сообществ разработчиков GitHub и Stackoverflow, Python является вторым по популярности языком программирования после Java Script.

Основная причина популярности Python – универсальность. Он применяется в огромном количестве областей и способен справляться с абсолютно разными задачами. Дополняет указанный фактор еще и простота изучения языка, так что армия разработчиков на Python постоянно пополняется новыми и новыми людьми.

Python появился в 1991 году. Создатель Гвидо Ван Россум назвал его в честь популярного комедийного шоу 1980-х гг. Важная цель разработчиков языка Python – сделать его забавным для использования, отсюда и причина выбора названия. Этот язык очень просто изучать, даже если раньше опыта в программировании не было.

Исходя из этого очевидна актуальность предлагаемого материала.

Данное методическое пособие разработано с целью пошаговой проработки материала по программированию на языке Python. Автор не только обобщил опыт, но и попытался представить информацию в интересном виде. Это пособие может применяться при изучении материала по нейронным сетям и анализу больших данных, а также для расширения знаний у обучающегося в области программирования.

Учебно – методические материалы содержат краткие теоретические блоки по основам программирования на языке Python, которые закрепляются решением практических задач разного уровня сложности. В методическом пособии представлены следующие темы: ввод, вывод, арифметика и типы данных; условные конструкции; циклы; строки; списки и словари.

Материалы можно применять как в процессе изучения программы «Нейронные сети, большие данные и кибергигиена», так и для самостоятельного изучения Python.

Учебно – методические материалы разработаны в соответствии с программой «Нейронные сети, большие данные и кибергигиена», модуль 2 «Основы Python» для обучающихся в возрасте 14–17 лет.

1 ВЫВОД, ВВОД, АРИФМЕТИКА И ТИПЫ ДАННЫХ

1.1 Вывод

Для **вывода** значений используется команда print(). Например,

```
print(1)
print(1.2)
print("Hello world!")
print('Hello world!')
```

Чтобы вывести несколько значений, их нужно перечислить через запятую. Например,

```
print("Оценка:", 5)
```

Параметр sep команды print() задает символ, который разделяет значения (по умолчанию пробел). Например, код ниже заменяет пробелы на точки при выводе даты:

```
print(4, 12, 2020, sep=".")
```

Параметр end позволяет указывать, что делать, после вывода строки. По умолчанию происходит переход на новую строку. Однако это действие можно отменить, указав любой другой символ или строку:

```
print(10, end="")
```

Python часто используют как калькулятор. Выражение, которое нужно вычислить, необходимо указать внутри скобок команды print(). После вычисления результат будет выведен на экран. Например,

```
print(1 + 1)
print((17 - 2) * 8)
```

1.2 Ввод

Ввод в Python реализован командой input(). Когда используется данная инструкция, программа останавливает выполнение и ждет, пока пользователь введет текст. После нажатия пользователем на Enter команда input() заберет введенный текст и передаст его программе.

Например, необходимо написать программу, которая приветствует пользователя. Для этого нужно узнать его имя с помощью команды input(). Результат ввода запишем в переменную name:

```
name = input()
```

Теперь нужно вывести строку с приветствием с помощью команды print(). Для этого внутри скобок необходимо написать само приветствие и после запятой имя пользователя:

```
print("Привет", name)
```

При запуске программы, компьютер ждет, когда будет введена строка. После нажатия на Enter, она будет присвоена переменной name. Затем значение переменной будет выведено на экран.

1.3 Арифметические операции

В Python реализовано еще несколько математический действий помимо сложения, вычитания и умножения.

Возведение в степень (**). Например,

```
print(2 ** 16)
```

Деление (/). Например,

```
print(5 / 2)
```

Целая часть от деления (//)). Например,

```
print(5 // 2)
```

Остаток от деления (%). Например,

```
print(5 % 2)
```

Например, необходимо написать программу, которая считывает два числа и затем складывает их. Код может выглядеть следующим образом:

```
num1 = input()
num2 = input()
print(num1 + num2)
```

Код выше ожидает с клавиатуры первое число, после чего сохраняет его в переменную num1. Аналогично программа поступает со вторым числом.

Попробуйте передать ей числа 5 и 6. Программа выведет 56, хотя в реальной жизни 5 + 6 = 11. Это произошло потому, что Python в третьей строчке «сложил» две строки, а не два числа. По умолчанию Python считывает результат, возвращаемый командой input(), как строку – тип данных str.

1.4 Типы данных

Типы данных определяют, что будет находиться в переменной. Наиболее распространенными и простыми типами данных в Python являются:

1. int – **целые числа** – положительные и отрицательные целые числа, а также 0 (например, 4, 687, -45, 0).

2. float – **числа с плавающей точкой** – дробные (вещественные) числа (например, 1.45, -3.789654, 0.00453).

3. str – **строки** – набор символов, заключенных в кавычки (например, "ball", "What is your name?", 'dkfjUUv', '6589').

По умолчанию Python считывает введенные значения как строку. Чтобы превратить ее в число, нужно выполнить **преобразование типов**. Команды int(), float() и str() преобразуют значение к необходимому типу.

Использование команды int() в примере выше, позволит преобразовать строку к целому числу:

```
num1 = int(input())
num2 = int(input())
print(num1 + num2)
```

Если необходимо сложить дробные числа, результат команды input() необходимо преобразовать к типу float:

```
num1 = float(input())
num2 = float(input())
print(num1 + num2)
```

1.5 Задачи

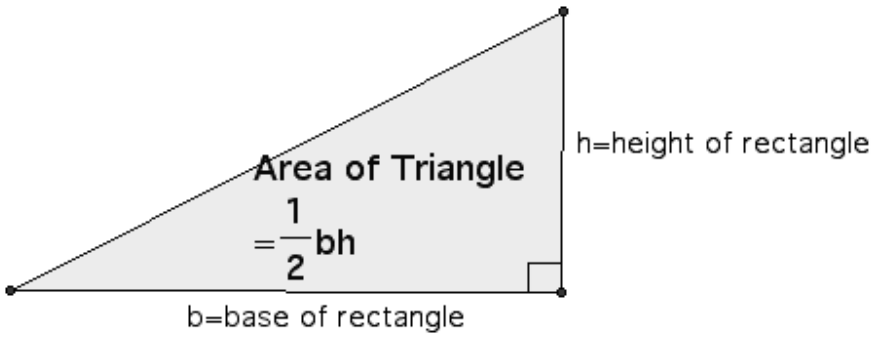
Базовые задачи

Задача №1. Сложение чисел

Напишите программу, которая считывает три числа и выводит их сумму.

Задача №2. Площадь треугольника

Напишите программу, которая считывает длины двух катетов в прямоугольном треугольнике и выводит его площадь по формуле, приведенной на изображении ниже. Каждое число записано в отдельной строке.



Задача №3. Дележка яблок

n школьников делят k яблок поровну, неделящийся остаток остается в корзинке. Сколько яблок достанется каждому школьнику? Сколько яблок останется в корзинке? Программа получает на вход числа n и k и должна вывести искомое количество яблок, которое достанется каждому школьнику и количество яблок, которое останется в корзине. Например, если 6 школьников делят между собой поровну 50 яблок, то каждому достанется по 8 яблок, а в корзине останется 2 яблока.

Дополнительные задачи

Вы можете проверить свое решение задач из этого блока с помощью системы проверки на сайтах Timus и Школа программиста. Для этого необходимо пройти простую процедуру регистрации. Название задач из сборников по олимпиадному программированию содержат название сайта, откуда взята задача, и ее номер в архиве. Помните, зачастую сложность этих задач заключается не в трудности написания кода, а в понимании условия.

Задача №4. Книжный червь (timus.ru, №1638)

Петя купил 100-томник «Советы ветеранов Спортивного Программирования» и решил, что теперь уж точно прервётся череда его неудач. Он приколотил к стене полку и выставил на неё вплотную все тома в порядке возрастания номеров слева направо. Однако Петя не знал, что внутри первого листа одного из томов притаился математический червяк, бесконечно маленький и очень прожорливый. Червяк стал прогрызать себе путь сквозь тома перпендикулярно плоскости листа. Остановился же он лишь когда достиг последнего листа другого тома. На следующий день Петя обнаружил повреждения и заинтересовался, сколько же миллиметров прогрыз червяк.

Входные данные

В первой строке через пробел записаны 4 целых числа: толщина каждого тома (без учёта переплёта), толщина переплёта каждого тома, номер тома, с первого листа которого червяк начал свой путь, и номер тома, на последнем

листе которого он остановился. Обратите внимание, здесь числа разделены пробелом. Вы можете считать одновременно четыре числа используя конструкцию: a, b, c, d = input().split().

Результат

Выведите единственное число — длину пути, прогрызенного червяком.

Пример

Входные данные	Результат
10 1 1 2	2

Задача №5. День программиста (acmp.ru, № 550)

День программиста отмечается в 255-й день года (при этом 1 января считается нулевым днем). Требуется написать программу, которая определит дату (месяц и число григорианского календаря), на которую приходится День программиста в заданном году.

В григорианском календаре високосным является:

- год, номер которого делится нацело на 400
- год, номер которого делится на 4, но не делится на 100.

Входные данные

В единственной строке записано целое число, которое обозначает номер года нашей эры.

Результат

В единственную строку нужно вывести дату Дня программиста в формате DD/MM/YYYY, где DD – число, MM – номер месяца (01 – январь, 02 – февраль, ..., 12 – декабрь), YYYY – год в десятичной записи.

Примеры

Входные данные	Результат
2000	12/09/2000
2009	13/09/2009

Задача №6. Игра (acmp.ru, № 4)

В свободное время одноклассники Вася и Петя любят играть в различные логические игры: морской бой, крестики-нолики, шахматы, шашки и многое другое. Ребята уже испробовали и поиграли во всевозможные классические игры подобного рода, включая компьютерные. Однажды им захотелось сыграть во что-нибудь новое, но ничего подходящего найти не удалось. Тогда Петя придумал следующую игру «Угадайка»: Играют двое участников. Первый загадывает любое трехзначное число, такое что первая и последняя цифры отличаются друг от друга более чем на единицу. Далее загадавший число игрок переворачивает загаданное число, меняя первую и последнюю цифры местами, таким образом получая еще одно число. Затем из максимального из полученных двух чисел вычитается минимальное. Задача второго игрока – угадать по первой цифре полученного в результате вычитания числа само это число. Например, если Вася загадал число 487, то перестановкой первой и последней цифры он получит число 784. После чего ему придется вычесть из 784 число 487, в результате чего получится число 297, которое и должен отгадать Петя по указанной первой цифре «2», взятой из этого числа. Петя успевает лучше Васи по математике, поэтому практически всегда выигрывает в играх такого типа. Но в данном случае Петя схитрил и специально придумал такую игру, в которой он не проиграет Васе в любом случае. Дело в том, что придуманная Петей игра имеет выигрышную стратегию, которая заключается в следующем: искомое число всегда является трехзначным и вторая его цифра всегда равна девяти, а для получения значения последней достаточно отнять от девяти первую, т.е. в рассмотренном выше случае последняя цифра равна 9-2=7. Помогите Пете еще упростить процесс отгадывания числа по заданной его первой цифре, написав соответствующую программу.

Входные данные

В единственной строке задана единственная цифра К, соответствующая первой цифре полученного Васей в результате вычитания наименьшего загаданного Васей значения из наибольшего.

Результат

Нужно вывести значение полученной Васей разности.

Примеры

Входные данные	Результат
5	594
2	297
7	792

2 УСЛОВНЫЕ КОНСТРУКЦИИ

Когда в программе в зависимости от значения выражения, требуется выполнить определенное действие, используется **условная конструкция**.

2.1 Условный оператор

Условный оператор if … else в Python имеет следующий синтаксис:

```
if Условие:
   Блок инструкций 1
else:
   Блок инструкций 2
```

Блок инструкций 1 будет выполнен, если Условие истинно. Если Условие ложно, будет выполнен Блок инструкций 2.

Например, требуется написать код, который по введенному с клавиатуры числу x определит его модуль. Программа должна напечатать значение переменной x, если x > 0 или же величину -x в противном случае. В Python это может выглядеть следующим образом:

```
x = int(input())
if x > 0:
   print(x)
else:
   print(-x)
```

После слова if указывается проверяемое условие (в данной программе – x > 0), завершающееся двоеточием. После этого идет блок инструкций, который будет выполнен, если условие истинно. В этом примере – вывод на экран величины x. Затем идет слово else, также завершающееся двоеточием, и блок инструкций, который будет выполнен, если проверяемое условие неверно, в данном случае будет выведен –x.

2.2 Неполное ветвление

В условной инструкции может отсутствовать слово else и последующий блок. Такая инструкция называется **неполным ветвлением**.

Синтаксис неполного ветвления в Python следующий:

```
if Условие:
    Блок инструкций 1
```

Код, приведенный в примере выше, можно упростить используя неполное ветвление:

```
x = int(input())
if x < 0:
    x = -x
print(x)
```

Здесь переменной x будет присвоено значение -x, но только в том случае, когда x < 0. Инструкция print(x) будет выполнена всегда, независимо от проверяемого условия.

2.3 Логические выражения и операторы

Логический тип данных

Логический тип данных (bool) имеет всего два возможных значения: True (правда, истина) и False (ложь). Он используется при вычислении **логических выражений**. Результатом логических выражений может являться лишь истина или ложь.

Например, выражение 4 > 5 – логическое, так как его результатом является либо правда, либо ложь. Выражение 4 + 5 не является логическим, потому что результатом его выполнения является число.

В программировании False обычно приравнивают к 0, а True – к 1:

```
print(int(False), int(True))
```

Также к истине в программировании относится любое значение кроме отсутствия такового и 0. Например,

```
print(bool(1), bool(0), bool(3.14), bool(" "), bool(""))
```

Операторы сравнения

Для сравнения используются **операторы сравнения**. В Python они имеют следующие обозначения: > – больше, < – меньше, >= – больше или равно, <= – меньше или равно, == – равно, != – не равно.

Логические операторы

Чтобы объединить несколько условий, используются **логические операторы**. Основные логические операторы следующие:

Логическое И (and): True, когда оба имеют значение True.

Логическое ИЛИ (or): True, когда хотя бы один равен True.

Логическое НЕ (not): True, если операнд равен False и наоборот (превращает ложь в правду, а правду – в ложь).

Например, код ниже проверяет, что хотя бы одно из чисел a или b оканчивается на 0:

```
a = int(input())
b = int(input())
if a % 10 == 0 or b % 10 == 0:
    print('YES')
else:
    print('NO')
```

Если необходимо проверить, что число a – положительное, а b – неотрицательное, можно использовать следующий код:

```
a = int(input())
b = int(input())
if a > 0 and not (b < 0):
    print('YES')
```

```
else:
   print('NO')
```

2.4 Каскадные условные конструкции

Программа может предполагать выбор больше, чем из двух путей. В данном случае используются **каскадные условные конструкции**.

Синтаксис каскадных условий в Python имеет вид:

```
if Условие 1:
   Блок инструкций 1
elif Условие 2:
      Блок инструкций 2
else:
   Блок инструкций 3
```

В каскадной условной конструкции if.. elif... else условия проверяются по очереди, выполняется блок, соответствующий первому из истинных условий. Если все проверяемые условия ложны, то выполняется блок else, если он присутствует.

Блок инструкций 1 будет выполнен, если Условие 1 истинно. Если Условие 1 ложно, будет проверяться Условие 2. Если это условие истинно, то выполнится Блок инструкций 2. В случае, если оба условия ложны, будет выполнен Блок инструкций 3.

Например, нужно написать программу, которая позволит узнать по координатам точки (x и y) к какой четверти координатной плоскости она относится. Исходя из поставленной задачи, необходимо проверить четыре условия:

если x > 0 и y > 0, то точка лежит в I четверти;

если x > 0 и y < 0, то точка лежит в IV четверти;

если x < 0 и y > 0, то точка лежит во II четверти;

если x < 0 и y < 0, то точка лежит в III четверти.

Для такого случая оптимальным решением будет использовать каскадную условную конструкцию Python. Программа для определение координатной четверти будет выглядеть следующим образом:

```
x = int(input())
y = int(input())
if x > 0 and y > 0:
    print("Первая четверть")
elif x > 0 and y < 0:
    print("Четвертая четверть")
elif y > 0:
    print("Вторая четверть")
elif y < 0:
    print("Третья четверть")
else:
    print("Начало координат")
```

2.5 Задачи

Базовые задачи

Задача №1. Четность числа

Определите, является ли введенное пользователем число четным. 0 является четным числом. Например,

Входные данные	Результат
0	четное
9	нечетное

Задача №2. День недели

Напишите программу, которая по введенному номеру дня недели (понедельник – первый день недели – 1, вторник – второй день недели – 2 и т.д.) определяет выходной это или будний день. Например,

Входные данные	Результат
1	будний
6	выходной

Задача №3. Траты

Вася получил первую зарплату и захотел это отпраздновать: провести вечер за любимым сериалом и пиццей. Хватит ли Васе денег, если подписка на онлайн-кинотеатр стоит s рублей, пицца – p рублей, а всего он заработал m рублей? На вход подается стоимость подписки на онлайн-кинотеатр (s), стоимость пиццы (p) и зарплата Пети (m), а выводится «Да», если он сможет позволить себе покупку, а иначе – «Нет». Например,

Входные данные	Результат
500 600 2000	Да
700	Нет

800 1000	

Задача №4. Скидка

В доставке из ресторана действует следующая система скидок:

– скидка 3% предоставляется при покупке от 500 рублей;

– скидка 5% предоставляется при покупке от 1000 рублей.

Напишите программу, которая определит, сколько клиент заплатит за покупку согласно стоимости заказа и системы скидок. Например,

Входные данные	Результат
500	485
300	300
1100	1045

Задача №5. Функция знака числа

В математике функция *sign*(*x*) (знак числа) определена так:

sign(*x*) = 1, если *x* > 0,

sign(*x*) = -1, если *x* < 0,

sign(*x*) = 0, если *x* = 0.

Вычислите для числа, введенного пользователем, результат от этой функции. Например,

Входные данные	Результат
-42	-1
2	1
0	0

Дополнительные задачи

Задача №6. Очередь (acmp.ru, № 511)

Студент Василий живет в общежитии. Отделение банка, в котором он производит оплату за проживание, имеет всего две кассы, поэтому почти всегда длинная очередь к ним. Первая касса открывается в 8.00, вторая – в 8.05. Последний клиент будет принят в 20.00. Очередь единая, и очередной клиент обслуживается, как только освобождается одна из касс. На обслуживание одного клиента уходит ровно 10 минут. Василий приходит ровно в 8.00 и видит, сколько человек стоит перед ним. Требуется определить, сколько времени ему придется простоять в очереди, и вообще обслужат ли его сегодня.

Входные данные

На вход подается натурально число K – номер Василия в очереди.

Результат

Выведите строку «NO», если Василий сегодня заплатить уже не успеет, и время его ожидания (в формате X Y, где X – количество целых часов, которые простоит в очереди Василий, и Y – количество минут), если все же успеет заплатить.

Примеры

Входные данные	Результат
1	0 0
20	1 35
235	NO

Задача №7. От перестановки что-то меняется… (acmp.ru, № 970)

Всем известно, что «от перестановки слагаемых сумма не изменяется». Однако, случается и так, что перестановка двух чисел приводит к более интересным последствиям.

Пусть, например, заданы три числа: a_1, a_2, a_3. Рассмотрим равенство $a_1 + a_2 = a_3$. Оно может быть неверным (например, если $a_1 = 1$, $a_2 = 4$, $a_3 = 3$), однако может стать верным, если поменять некоторые числа местами (например, если поменять местами a_2 и a_3, оно обратится в равенство 1 + 3 = 4).

Ваша задача – по заданным трем числам определить: можно ли их переставить так, чтобы сумма первых двух равнялась третьему.

Входные данные

На вход подается три целых числа: a_1, a_2, a_3. Обратите внимание, здесь числа разделены пробелом. Вы можете считать одновременно все три числа используя такую конструкцию: a1, a2, a3 = input().split().

Результат

Выведите слово «YES», если заданные числа можно переставить так, чтобы сумма первых двух равнялась третьему. В противном случае выведите в слово «NO».

Примеры

Входные данные	Результат
3 5 2	YES
2 2 5	NO
2 2 4	YES

Задача №8. Арифметика (acmp.ru, № 8)

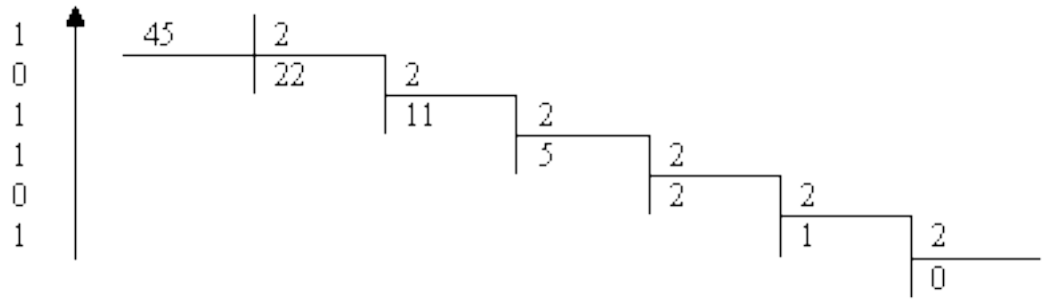
В прошлом году Вася пошел в школу и научился считать. В этом году он изучил таблицу умножения и теперь умеет перемножать любые числа от 1 до 10 без ошибок. Друг Петя рассказал ему про системы счисления, отличные от десятичной. В частности, про двоичную, восьмеричную и даже шестнадцатеричную. Теперь Вася без труда (но уже с помощью листка и

ручки) может перемножать числа от 1 до 10 и в этих системах, используя перевод из нестандартной системы в десятичную и обратно из десятичной. Например, если Васе нужно перемножить числа 101 и 1001 в двоичной системе, то он сначала эти числа переводит в десятичное представление следующим образом:

$(101)_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 4 + 0 + 1 = 5$

$(1001)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 0 + 0 + 1 = 9$

После чего перемножение чисел 5 и 9 Вася с легкостью производит в десятичной системе счисления в уме и получает число 45. Далее производится перевод из десятичной системы счисления в двоичную. Для этого Вася делит число 45 на 2 (порядок системы счисления), запоминая остатки от деления, до тех пор, пока в результате не останется число 0:



Ответ составляется из полученных остатков от деления путем их записи в обратном порядке. Таким образом Вася получает результат: $(101)_2 \cdot (1001)_2 = (101101)_2$. Но теперь Вася изучает таблицу умножения чисел от 1 до 100 в десятичной системе счисления, а поскольку запомнить такую таблицу очень сложно, то Васе придется очень долго ее зубрить. Составьте для Васи программу, которая поможет ему проверять свои знания.

Входные данные

На вход подаются три натуральных числа A, B и C. Обратите внимание, здесь числа разделены пробелом. Вы можете считать одновременно все три числа используя такую конструкцию: A, B, C = input().split().

Результат

Выведите слово «YES», если A · B = C. В противном случае выведите в слово «NO».

Примеры

Входные данные	Результат
8 54 432	YES
16 19 777	NO

3 ЦИКЛЫ

Циклы позволяют организовывать повторное выполнение участков кода. В Python представлено два вида циклов: for и while.

3.1 Цикл for

Цикл for используется для повторения последовательности действий заданное количество раз.

Структуру цикла for можно представить в общем виде следующим образом:

for Переменная in Значения:

Инструкция 1

Инструкция 2

После ключевого слова for указывается Переменная, а после слова in – множество Значений, которые эта Переменная будет поочередно принимать. Цикл будет выполняться столько раз, сколько Значений указано.

Переменную называют **счетчиком**. Ее значение меняется с каждым проходом цикла (Переменная подсчитывает количество итераций цикла).

Функция range()

Множество значений часто задается с помощью функции range(). Она создает список чисел, значения которых поочередно будет принимать переменная. Функция range() может иметь разное количество параметров.

range(b) создает список чисел от 0 до b-1 (не включая b) с шагом 1. Цикл будет выполняться b раз.

Например, код, который возводит в квадрат числа от 0 до 5 (не включая 5) может выглядеть следующим образом:

```
for i in range(5):
print(i ** 2)
```

```
print('Конец цикла')
```

В этом примере числа будут генерироваться от 0 до 4 (5 не включается). Всего будет создан ряд из пяти чисел: 0, 1, 2, 3, 4. Переменная-счетчик i будет поочередно принимать значения из указанного диапазона от 0 до 4. Цикл повторится пять раз.

range(a, b) создает список чисел от a до b-1 с шагом 1.

Например, чтобы просуммировать значения чисел от 1 до 5, включая 5, можно воспользоваться следующей программой:

```
sum = 0
for i in range(1, 5 + 1):
sum += i
print(sum)
```

range(a, b, d) помимо начала (a) и конца (b) диапазона чисел, задает шаг изменения переменной i (d) – на сколько она будет изменяться.

Например, код, который выводит все нечетные числа от 1 до 9 (включительно) будет выглядеть следующим образом:

```
for i in range(1, 10, 2):
print(i)
```

3.2 Задачи по циклу for

Базовые задачи

Задача №1. Четные числа

Вывести все четные числа от 0 до n. Число n вводит пользователь. При решении задачи используйте функцию range() с тремя параметрами. Например,

Входные данные	Результат
5	0 2 4

Задача №2. От A до B

Пользователь вводит два числа A и B. Выведите все числа от A до B. В этой задаче используйте функцию range() с двумя параметрами. Например,

Входные данные	Результат
2 6	2 3 4 5 6

Задача №3. Сумма квадратов

Пользователь вводит число n. Вычислите квадраты чисел от 1 до n, а на экран выведите сумму этих квадратов. В этой задаче используйте функцию range() с двумя параметрами. Например,

Входные данные	Результат
5	55

Задача №4. Подсчет нулей

Пользователь вводит количество чисел (n), а затем ровно n целых чисел. Посчитайте сколько нулей среди введенных чисел и выведите это количество. Вам нужно подсчитать количество чисел, равных нулю, а не количество нулей в числах. Например,

Входные данные	Результат
3 5 0 100	1

Дополнительные задачи

Задача №5. Пингвины (timus.ru, № 1585)

Программист Денис с детства мечтал побывать в Антарктиде, но почему-то регулярных рейсов туда нет. Поэтому Денис все лето изучал Антарктиду с помощью соседнего кинотеатра. Теперь он знает, что в Антарктиде водится несколько видов пингвинов:

- Императорские пингвины (Emperor Penguins) — любители петь;
- Малые пингвины (Little Penguins) — любители потанцевать;
- Пингвины Макарони (Macaroni Penguins) — любители сёрфинга.

К сожалению, в мультфильмах не было сказано, какой вид пингвинов самый многочисленный. Денис решил выяснить это: он посмотрел эти мультфильмы еще раз, и каждый раз, когда видел пингвина, записывал в блокнот название его вида. Сейчас Денис дал вам блокнот с просьбой выяснить, какой вид пингвинов самый многочисленный.

Входные данные

В первой строке записано целое число n – количество записей в блокноте. В каждой из следующих n строк записано по одному виду пингвинов. Среди видов встречаются только «Emperor Penguin», «Little Penguin» и «Macaroni Penguin».

Результат

Выведите самый популярный вид пингвинов.

Пример

Входные данные	Результат
7 Emperor Penguin Macaroni Penguin Little Penguin Emperor Penguin Macaroni Penguin Macaroni Penguin Little Penguin	Macaroni Penguin

Задача №6. Уравнение (acmp.ru, № 10)

Вася в школе изучил квадратные уравнения и понял, как они легко решаются путем вычисления дискриминанта. Но Петя поведал ему о методе решения кубических уравнений вида $A \cdot x^3 + B \cdot x^2 + A \cdot x + D = 0$. На факультативе по математике Васе задали решить около ста уравнений как раз такого вида. Но, к сожалению, Вася забыл формулы, о которых рассказывал ему Петя. Но Васе было известно, что все корни уравнений – целые числа и находятся на отрезке [-100, 100]. Поэтому у Васи есть шанс найти их методом перебора, но для этого ему придется затратить уйму времени, т.к. возможно необходимо будет осуществить перебор нескольких тысяч значений. Помогите Васе написать программу, которая поможет ему найти корни кубических уравнений!

Входные данные

В единственной строке записаны 4 числа: A, B, C и D – целые коэффициенты кубического уравнения. Обратите внимание, здесь числа разделены пробелом. Вы можете считать одновременно все четыре числа используя такую конструкцию: A, B, C, D = input().split().

Результат

Выведите через пробел в порядке возрастания все корни заданного кубического уравнения. Кратные корни следует выводить только один раз.

Примеры

Входные данные	Результат
1 -3 0 0	0 3
3 -15 18 0	0 2 3
1 -7 -33 135	-5 3 9

Задача №7. Карточки

Для настольной игры используются карточки с целочисленными номерами от 1 до N. Одна из них потерялась. Необходимо найти ее, зная номера оставшихся карточек.

Дано число N – количество карточек в игре, номера оставшихся карточек в количестве N - 1 (неповторяющиеся числа от 1 до N). Программа должна вывести номер потерянной карточки.

Массивами и аналогичными структурами данных пользоваться нельзя. Например,

Входные данные	Результат
5 3 5 2 1	4

3.3 Цикл while

Цикл while выполняется, пока проверяемое условие истинно. Условие записывается до тела цикла и проверяется до выполнения тела цикла. Цикл while используется, когда невозможно определить точное значение количества проходов (итераций) исполнения цикла.

Синтаксис цикла while выглядит так:

while условие:

блок инструкций

Например, программа, которая печатает на экран квадраты всех целых чисел от 1 до 5 может выглядеть следующим образом:

```
i = 1
while i <= 5:
        print(i ** 2)
   i = i + 1
```

В первой строке объявляется переменная i, которой присваивается значение 1. В цикле проверяется условие i <= 5 (необходимо вывести квадраты всех целых чисел до 5 включительно).

Данный цикл выполнится 5 раз:

1. Сначала переменная i = 1, так как ей присвоили это значение в первой строке. В заголовке цикла проверяется условие i <= 5. В данный момент оно вернет истину. Поэтому переменная возводится в квадрат и результат выводится на экран. Затем значение i увеличивается на единицу.

2. Теперь i = 2. Условие возвращает True, цикл выполняется второй раз. В последней строке тела цикла значение i увеличивается на единицу.

3. i = 3, условие верно. i снова увеличивается на единицу.

4. i = 4, условие верно, i увеличивается на единицу.

5. i = 5, условие верно, i увеличивается на единицу.

После пятой итерации i = 6. Снова проверяется условие i <= 5, но в этот раз оно возвращает False. Поэтому цикл больше не выполняется, достигнуто условие остановки цикла.

Зацикливание

Если условие в заголовке while никогда не возвращает False, то цикл будет повторяться бесконечно, пока его работу не прервут с помощью специального оператора или извне. Это называется **зацикливанием**.

Если переменная i не будет изменять значение внутри тела цикла, то код выше бесконечно будет выводить квадрат единицы. Поэтому условие после ключевого слова while называют **условием остановки цикла** – гарантией того, что зацикливания не произойдет, способом контроля цикла while.

3.4 Инструкции управления циклом

Контроль над поведением циклов выполняется не только с помощью условия остановки, но и с использованием специальных **инструкций управления циклом**.

Инструкция else

Инструкция else указывает, какие операции выполнить после окончания цикла, когда проверяемое условие станет неверно, например,

```
i = 1
while i <= 5:
        print(i)
        i += 1
  else:
        print('Цикл окончен, i =', i)
```

Инструкция break

Инструкция break позволяет прекратить выполнение цикла и выйти из него при достижении условия, указанного в блоке if. При этом ветка else исполняться не будет, потому что цикл завершился, а else относится к циклу.

Например, код, приведенный ниже, позволяет считывает положительные числа, пока пользователь не введет число ноль. Если в программу будет передано отрицательное число, то работа цикла прекратиться благодаря использованию инструкции break.

```
a = int(input())
while a != 0:
        if a < 0:
                print('Встретилось отрицательное число', a)
                break
        a = int(input())
    else:
    print('Ни одного отрицательного числа не встретилось')
```

Оператор break и ключевое слово else можно использовать в цикле for. В этом случае код, предложенный выше, будет выглядеть следующим образом:

```
n = int(input())
for i in range(n):
        a = int(input())
        if a < 0:
                print('Встретилось отрицательное число', a)
            break
else:
        print('Ни одного отрицательного числа не встретилось')
```

Инструкция continue

Если в цикле используется команда continue, то пропускаются все оставшиеся инструкции до конца цикла, и исполнение цикла продолжается со следующей итерации.

Например, код, который подсчитывает количество неотрицательных чисел может выглядеть следующим образом:

```
n = int(input())
count = 0
for i in range(n):
    a = int(input())
    if a < 0:
        print('Встретилось отрицательное число', a)
        continue
    count += 1
print(count)
```

Если в программу выше будет передано отрицательное число, то на экран выведется соответствующее сообщение и выполнение этой итерации будет завершено благодаря инструкции continue, а к переменной count единица не добавится. Затем выполнится следующий проход цикла.

Инструкциями continue и break не следует злоупотреблять. Они усложняют читаемость кода, поэтому, если это возможно, их лучше избегать.

3.5 Задачи по циклу while

Базовые задачи

Задача №1. Квадраты до N

Пользователь вводит число N. Необходимо написать программу, которая напечатает квадраты натуральных чисел, не превосходящие N. При решении задачи используйте цикл while. Например,

Входные данные	Результат
50	1 4 9 16 25 36 49

Задача №2. Наименьший делитель

Пользователь вводит целое число, большее двух. Выведите его наименьший натуральный делитель, отличный от 1. Используйте цикл while. Например,

Входные данные	Результат
16	3

Задача №3. Пробежка

В первый день спортсмен пробежал x километров, а затем он каждый день увеличивал пробег на 10% от предыдущего значения. По данному числу y определите номер дня, на который пробег спортсмена составит более y километров. Используйте цикл while. Например,

Входные данные	Результат
10 20	9

Дополнительные задачи

При решении этих задач вы можете выбирать цикл на свое усмотрение.

Задача №4. Эния (acmp.ru, №195)

Неспокойно сейчас на стапелях шестого дока межгалактического порта планеты Торна. Всего через месяц закончится реконструкция малого броненесущего корвета “Эния”. И снова этому боевому кораблю и его доблестной команде предстоят тяжелые бои за контроль над плутониевыми рудниками Сибелиуса. Работа не прекращается ни на секунду, лазерные сварочные аппараты работают круглые сутки. От непрерывной работы плавятся шарниры роботов-ремонтников. Но задержаться нельзя ни на секунду.

И вот в этой суматохе обнаруживается, что термозащитные панели корвета вновь требуют срочной обработки сульфидом тория. Известно, что на обработку одного квадратного метра панели требуется 1 нанограмм сульфида. Всего необходимо обработать N прямоугольных панелей размером A на B метров. Вам необходимо как можно скорее подсчитать, сколько всего сульфида необходимо на обработку всех панелей “Энии”. И не забудьте, что панели требуют обработки с обеих сторон.

Входные данные

На вход подается 3 целых положительных числа N, A, B. Обратите внимание, числа разделены пробелом. Одновременно считать три числа можно с помощью конструкции: N, A, B = input().split().

Результат

Нужно вывести единственное число – вес необходимого для обработки сульфида тория в нанограммах.

Примеры

Входные данные	Результат
5 2 6	60
14 23 5	3220

Задача №5. Конечные автоматы (acmp.ru, №35)

Однажды известный профессор обнаружил описания k конечных автоматов. По его мнению, нетривиальность конечного автомата, имеющего n состояний и m переходов, можно описать целым числом $d = 19m + (n + 239) \cdot (n + 366) / 2$. Чем больше d, тем больший интерес для науки представляет изучение его свойств.

Помогите профессору вычислить нетривиальность имеющихся у него автоматов.

Входные данные

На вход подается целое число k – количество конечных автоматов. Следующие k строк содержат по два целых числа n_i и m_i – число состояний и переходов i-го автомата. Обратите внимание, здесь числа разделены пробелом. Вы можете считать одновременно два числа используя такую конструкцию: n, m = input().split().

Результат

Вывод состоит из k строк. На i-й строке выведите одно число – нетривиальность i-го автомата.

Примеры

Входные данные	Результат
4 2 0 13 20 5 23 18 6	44344 48134 45699 49458
2 15 20 1000 26000	48767 1340237

Задача №6. Числа Фибоначчи (acmp.ru, №147)

Последовательностью Фибоначчи называется последовательность чисел a_0, a_1, ..., a_n, ..., где $a_0 = 0$, $a_1 = 1$, $a_k = a_k - 1 + a_k - 2$ ($k > 1$). Требуется найти n-е число Фибоначчи.

Входные данные

На вход подается целое число n.

Результат

Необходимо вывести n-е число Фибоначчи.

Пример

Входные данные	Результат
7	13

Задача №7. Перепись (acmp.ru, № 131)

В доме живет *N* жильцов. Однажды решили провести перепись всех жильцов данного дома и составили список, в котором указали возраст и пол каждого жильца. Требуется найти номер самого старшего жителя мужского пола.

Входные данные

На вход в первой строке подается натуральное число N – количество жильцов. В последующих N строках располагается информация о всех жильцах: каждая строка содержит два целых числа: V и S – возраст и пол человека. Мужскому полу соответствует значение S = 1, а женскому – S = 0. Обратите внимание, здесь числа разделены пробелом. Вы можете считать одновременно два числа используя такую конструкцию: v, s = input().split().

Результат

Программа должна вернуть номер самого старшего мужчины в списке. Если таких жильцов несколько, то следует вывести наименьший номер. Если жильцов мужского пола нет, то выведите -1.

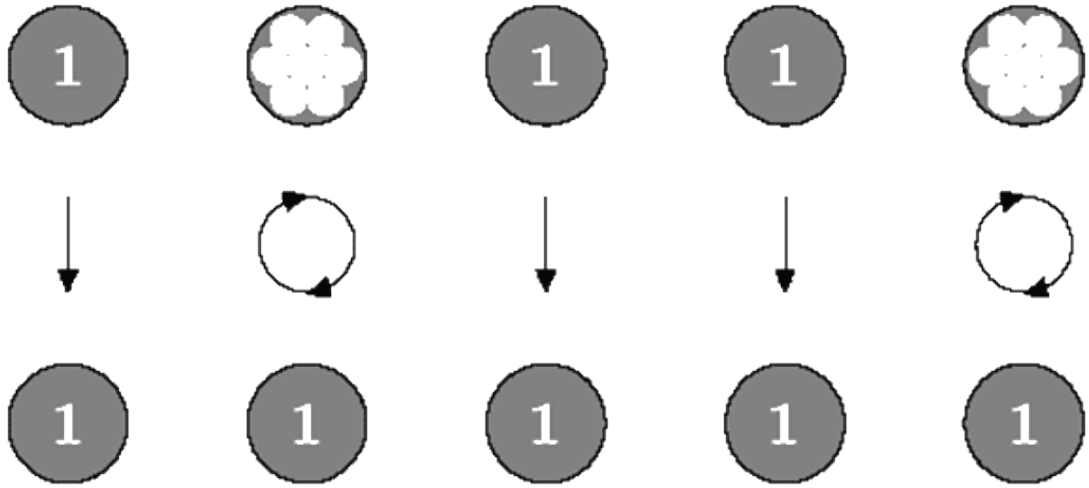
Примеры

Входные данные	Результат
4 25 1 70 1 100 0 3 1	2

2 25 0 25 1	2

Задача №8. Монетки (acmp.ru, №106)

На столе лежат *n* монеток. Некоторые из них лежат вверх решкой, а некоторые – гербом. Определите минимальное число монеток, которые нужно перевернуть, чтобы все монетки были повернуты вверх одной и той же стороной.



Входные данные

В первой строке записано натуральное число N – число монеток. В каждой из последующих N строк содержится одно целое число – 1 если монетка лежит решкой вверх и 0 если вверх гербом.

Результат

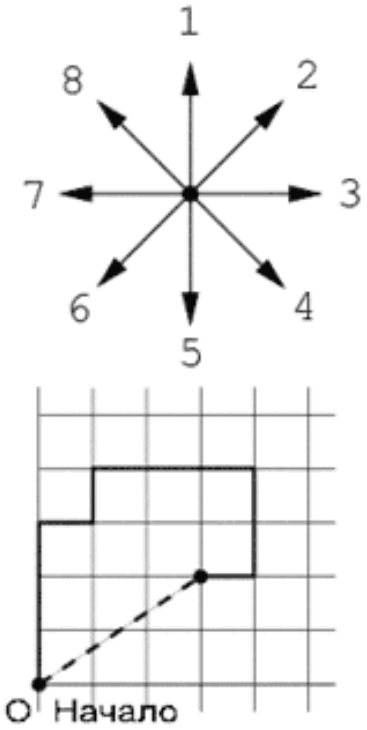
Выведите минимальное количество монет, которые нужно перевернуть.

Пример

Входные данные	Результат
5 1 0 1 1 0	2

Задача №9. Клад (acmp.ru, №207)

Найти закопанный пиратами клад просто: всё, что для этого нужно – это карта. Как известно, пираты обычно рисуют карты от руки и описывают алгоритм действий. Большая часть таких действий просто сводится к прохождению какого-то количества шагов в одном из восьми направлений (1 – север, 2 – северо-восток, 3 – восток, 4 – юго-восток, 5 – юг, 6 – юго-запад, 7 – запад, 8 – северо-запад). Длина шага в любом направлении равна 1.



Путешествие по такому пути обычно является прекрасным способом посмотреть окрестности, однако в наше время постоянной спешки ни у кого нет времени на это. Поэтому кладоискатели хотят идти напрямую в точку, где зарыт клад. Например, вместо того, чтобы проходить три шага на север, один шаг на восток, один шаг на север, три шага на восток, два шага на юг и один шаг на запад, можно пройти напрямую, использовав около 3.6 шага.

Вам необходимо узнать точку, в которой находится клад согласно указаниям пиратов.

Входные данные

Первая строка содержит число N – число указаний. Последующие N строк содержат сами указания – номер направления и количество шагов. Числа разделены пробелами. Вы можете считать одновременно два числа используя такую конструкцию: n, k = input().split().

Результат

Выведите координаты X и Y точки (два вещественных числа, разделённые пробелом), где зарыт клад, считая, что ось *OX* направлена на восток, а ось *OY* – на север. В начале кладоискатель должен стоять в начале координат. Координаты необходимо вывести с точностью 10^{-3}.

Примеры

Входные данные	Результат
6 1 3 3 1 1 1 3 3 5 2 7 1	3.000 2.000
1 8 10	-7.071 7.071

Задача №10. Штрафное время (timus.ru, №1636)

Команду ZZZ вновь постигла неудача. На очередном соревновании в городе Екатеринозаводске участники выступили блестяще, первыми решив все 10 предложенных задач ещё до заморозки монитора. Однако в итоговом протоколе команда ZZZ оказалась лишь на втором месте, проиграв команде QXX по штрафному времени. Участник команды QXX предположил, что это произошло из-за неаккуратности участников и применяемой ими техники грязной отладки. Однако капитан ZZZ заявил, что во всём виновата хитрая командная тактика, поскольку даже если бы команда сдавала все задачи с первой попытки, она всё равно заняла бы только второе место. Выясните, кто из них прав.

Входные данные

В первой строке через пробел записаны числа T_1 и T_2 — штрафное время в минутах команды QXX и команды ZZZ. Во второй строке через пробел записаны 10 чисел — i-е число обозначает количество штрафных попыток, сделанных командой ZZZ по i-й задаче. Напомним, что каждая штрафная попытка прибавляет к штрафному времени 20 минут. Чтобы считать их одновременно два числа, используйте конструкцию: qxx, zzz = input().split().

Результат

Если штрафные попытки не повлияли на итоговое место команды ZZZ, выведите «No chance.». Иначе выведите «Dirty debug :(». При равенстве штрафного времени команды сортируются по алфавиту, а значит, команда ZZZ в этом случае всё равно оказалась бы на втором месте.

Примеры

Входные данные	Результат
290 420 0 0 0 2 1 0 2 0 1 0	No chance.
300 719 0 0 0 0 0 0 21 0 0 0	Dirty debug :(

4 СТРОКИ

Строка в Python – это последовательность символов, заключенных в одинарные или двойные кавычки. Она считывается функцией input(). Например, 'Hello, world!', '1', '123', ' '.

В Python нет отдельного символьного типа – типа данных, объектами которого являются одиночные символы. Аналогом символа можно считать строку, состоящую из одного элемента, например, '1'.

Чтобы узнать длину строки существует функция len(). Например,

```
s = '123'
print(len(s))
```

Строки можно дублировать (умножать на число) и складывать (дописывать к одной строке другую – **конкатенация**). Например,

```
s1 = 'Hello'
s2 = 'World'
print(s1 * 2)
print(s1 + s2)
```

4.1 Срезы

Срез (slice) – извлечение из строки одного символа или некоторого фрагмента строки (последовательности символов).

Срез из одного символа

Из строки можно извлечь один символ.

Для этого используется конструкция: s[i] – это срез, состоящий из одного символа, который имеет номер i. При этом считается, что нумерация начинается с числа 0.

Например, если s = 'Hello', то s[0] == 'H', s[1] == 'e', s[2] == 'l', s[3] == 'l', s[4] == 'o'. i – номер элемента – называется **индексом**.

Если указать отрицательное значение индекса, то номер будет отсчитываться с конца, начиная с номера -1.

Например, s[-1] == 'o', s[-2] == 'l', s[-3] == 'l', s[-4] == 'e', s[-5] == 'H'.

Таким образом, чтобы получить определенный символ из строки, нужно обратиться к нему по положительному или отрицательному номеру (индексу):

Строка s	H	e	l	l	o
Положительный индекс	s[0]	s[1]	s[2]	s[3]	s[4]
Отрицательный индекс	s[-5]	s[-4]	s[-3]	s[-2]	s[-1]

Подстрока

Можно извлекать последовательность символов – **подстроку**. Для этого используется конструкция s[a, b], где a – индекс, с которого начинается нужная подстрока, а b – индекс, которым она заканчивается, при этом b не включен в диапазон.

Например, извлечем из строки 'Hello' подстроку 'Hell':

```
s = 'Hello'
print(s[0:4])
```

Или можно записать иначе, используя отрицательный индекс:

```
s = 'Hello'
print(s[0:-1])
```

Если срез начинается с 0, параметр a = 0 можно опустить. Тогда код выше будет выглядеть следующим образом:

```
s = 'Hello'
print(s[:4])
```

Если нужно взять срез с первого элемента и до конца, можно это записать так:

```
s = 'Hello'
print(s[1:5])
```

Если срез идет до конца строки, параметр b можно опустить. Например,

```
s = 'Hello'
```

```
print(s[1:])
```

Срез с шагом

Последний параметр среза – шаг. Аналогично функции range(): s[a:b:d].

Например, код, выполняющий извлечение из строки каждого второго символа, может выглядеть так:

```
s = '123456789'
print(s[0:10:2])
```

Пример выше можно записать проще, опустив параметры a и b, так как здесь они имеют значения начала и конца строки, совпадающие со стандартными:

```
s = '123456789'
print(s[::2])
```

Чтобы перевернуть строку, нужно использовать отрицательное значение шага. Например,

```
s = '123456789'
print(s[::-1])
```

Любые операции среза со строкой создают новые строки и никогда не меняют исходную строку. В Python строки являются неизменяемыми. Можно присвоить срез новой строке. Например,

```
s = '123456789'
s1 = s[::-1]
print(s, s1)
```

4.2 Методы строк

Метод find()

Метод find() находит в строке указанную подстроку. Функция возвращает индекс первого вхождения искомой подстроки. Если же подстрока не найдена, то метод возвращает значение -1. Например,

```
s = 'Hello'
print(s.find('e'))
print(s.find('ll'))
print(s.find('L'))
```

Если вызвать метод find() с тремя параметрами s.find(T, a, b), то поиск будет осуществляться в срезе s[a:b].

Например, чтобы найти если букву l в срезе s[:3] можно воспользоваться следующим кодом:

```
s = 'Hello'
print(s.find('l', 0, 3)) # вернёт 2
```

Если указать только два параметра s.find(T, a), то поиск будет осуществляться в срезе s[a:]. Код, который ищет символ e в срезе s[2:] может выглядеть так:

```
s = 'Hello'
print(s.find('e', 2))
```

Метод replace()

Метод replace() заменяет все вхождения одной строки на другую. Формат: s.replace(old, new) – заменить в строке s все подстроки old на подстроку new. Например,

```
s = 'Hello'
print(s.replace('l', 'L'))
```

Если методу replace задать еще один параметр: s.replace(old, new, count), то заменены будут не все вхождения, а только не больше, чем первые count из них. Например, чтобы заменить в строке 'AbrAkAdabra' только первые две строчные буквы a можно использовать следующий код:

```
print('Abrakadabra'.replace('a', 'A', 2))
```

Метод count()

Метод count() подсчитывает, сколько раз в строке встретилась другая строка. Например,

print('Abracadabra'.count('a'))

При указании трех параметров s.count(T, a, b), будет выполнен подсчет числа вхождений строки T в срез s[a:b].

4.3 Задачи

Базовые задачи

Задача №1. Abrakadabra

Пользователь вводит строку. Вам необходимо вывести на экран следующее:

1. Третий символ строки.
2. Предпоследний символ строки.
3. Первые пять символов строки.
4. Всю строку, кроме последних двух символов.
5. Все символы с четными индексами.
6. Все символы с нечетными индексами.
7. Все символы в обратном порядке.
8. Все символы строки через один в обратном порядке, начиная с последнего.
9. Длину строки.

Пример строки:

Входные данные	Результат
Abrakadabra	r r Abrak Abrakadab Arkdba baaar arbadakarbA abdkrA 11

Задача №2. Перестановка

Пользователь вводит строку, состоящую только из двух слов, разделенных пробелом. Переставьте эти слова местами. Результат запишите в строку и выведите получившуюся строку. Например,

Входные данные	Результат

Hello world	world Hello

Задача №3. Количество слов

Пользователь вводит строку, состоящую из слов, разделенных пробелами. Определите, сколько в ней слов. Используйте для решения задачи метод count(). Например,

Входные данные	Результат
Hello world	2

Задача №4. Автозамена

Наступил Новый год, а Вася все еще ставит в документе прошедший, 2020, год. Помогите автоматизировать процесс замены номера старого года на новый. Используйте для решения метод replace(). Например,

Входные данные	Результат
В 2020 году я буду все делать вовремя!	В 2021 году я буду все делать вовремя!

Дополнительные задачи

Задача №5. Быки и коровы (acmp.ru, №13)

Петя и Вася часто играют в различные логические игры. Недавно Петя поведал Васе о новой игре «Быки и коровы» и теперь они играют в эту игру сутками. Суть игры очень проста: Петя загадывает четырехзначное число, состоящее из различных цифр. Вася отгадывает задуманное Петей число, перебирая возможные варианты. Каждый раз Вася предлагает вариант своего числа, а Петя делает Васе подсказку: сообщает количество быков и коров, после чего Вася с учетом подсказки продолжает отгадывание числа до тех пор, пока не отгадает. Быки – это количество цифр в предложенном Васей числе, совпадающих по значению и стоящих в правильной позиции в задуманном Петей числе. Коровы – количество цифр, совпадающих по значению, но находящихся в неверной позиции. Например, если Петя задумал число 5671, а Вася предложил вариант 7251, то число быков равно 1 (только цифра 1 на своем месте), а число коров равно 2 (только цифры 7 и 5 не на своих местах). Петя силен в математике, но даже он может ошибаться. Помогите Пете написать программу, которая бы по загаданному Петей и предложенному Васей числам сообщала количество быков и коров.

Входные данные

В единственной строке записано два четырехзначных натуральных числа A и B через пробел, где A – загаданное Петей число, а B – предложенный Васей вариант. Вы можете считать одновременно два числа используя метод split(): A, B = input().split().

Результат

Вывести два числа через пробел — количество быков и коров.

Примеры

Входные данные	Результат
5671 7251	1 2
1234 1234	4 0

2034 6234	2 1

Задача №6. Стрелки (acmp.ru, №44)

Задана последовательность, состоящая только из символов ‘>’, ‘<’ и ‘-‘. Требуется найти количество стрел, которые спрятаны в этой последовательности. Стрелы – это подстроки вида ‘>>-->’ и ‘<--<<’.

Входные данные

В строке записана строка, состоящая из символов ‘>’, ‘<’ и ‘-‘ (без пробелов).

Результат

Нужно вывести искомое количество стрелок.

Пример

Входные данные	Результат
<<<<>>--><--<<--<<>>>--><<<<<	4

5 СПИСКИ

В Python аналогом массивов являются **списки.** Они могут состоять из элементов разных типов.

Список – это последовательность элементов, пронумерованных от 0, как символы в строке. Его можно задать перечислением элементов списка в квадратных скобках через запятую. Например,

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

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

Results = [5, 2, 4.5, 'Good']

Обращаться к элементам списка можно с помощью положительного и отрицательного индекса:

Primes	2	3	5	7	11	13
i > 0	Primes [0]	Primes [1]	Primes [2]	Primes [3]	Primes [4]	Primes [5]
i < 0	Primes [-6]	Primes [-5]	Primes [-4]	Primes [-3]	Primes [-2]	Primes [-1]

Чтобы вычислить длину списка, достаточно использовать функцию len(), аналогично строкам, например,

len(Primes)

В отличие от строк, элементы списков изменять можно. Например,

print(Rainbow[0])

Rainbow[0] = 'красный'

print(Rainbow[0])

5.1 Вывод элементов списка

Выводить список поэлементно можно с помощью циклов, обращаясь к элементам по индексу:

for i in range(len(Rainbow)):

```
    print(Rainbow[i])
```

Здесь переменная i будет пробегать по всему списку вплоть до его конца, что указано в функции range(). Внутри нее указан только один параметр – сколько элементов нужно перебрать – количество элементов в списке (или длина списка).

Или перебирать сами элементы списка. Например,

```
for color in Rainbow:
    print(color)
```

В этом случае в цикле изменяется не индекс элемента списка, а само значение переменной. Из примера выше, переменная color в цикле последовательно принимает значения 'red', 'green', 'blue' и т. д.

Если необходимо, чтобы элементы списка выводились через пробел, можно использовать параметр end функции print(). Например,

```
for i in range(len(Rainbow)):
    print(Rainbow[i], end = ' ')
```

5.2 Методы списков и строк

Считывание списков, удаление элементов из них и многие другие операции возможны с помощью **методов списков.**

Метод append()

Первый способ: создать пустой список и последовательно добавлять в конец списка элементы с помощью метода append().

Например, нужно создать список из количества элементов, равного n. Число n задается пользователем и вводится с клавиатуры. Тогда код будет выглядеть следующим образом:

```
l = []
n = int(input())
for i in range(n):
```

```
        new_element = int(input())
        l.append(new_element)
print(l)
```

Метод split()

Метод split() разделяет строку по указанному символу-разделителю (по умолчанию – пробел) на список строк.

Например, пользователь вводит числа в строку через пробел, а нам нужно посчитать их сумму. С использованием метода split() программа для решения поставленной задачи может выглядеть следующим образом:

```
l = input()
numbers = l.split()
result = 0
for i in range(len(numbers)):
        numbers[i] = int(numbers[i])
        result = result + numbers[i]
print(result)
```

Метод split() можно указать разделитель между элементами будущего списка. Например, вызов метода split('.') вернет список, полученный разрезанием исходной строки по символу '.':

```
ip = '192.168.0.1'.split('.')
print(ip)
```

Чтобы одновременно считать с клавиатуры заранее известное число строк, можно воспользоваться следующей конструкцией:

```
a, b = input().split()
print(int(a) * int(b))
```

Метод join()

Метод join() вставляет между элементами списка указанную строку или символ. Его можно использовать для вывода списка в столбец или строку. Например,

```
color = ['red', 'green', 'blue']
print(' '.join(color))
print(''.join(color))
print('\n'.join(color))
```

Этот способ работает только со списками из строк. Чтобы применить его к числам, нужно привести каждое число к строковому типу данных в цикле.

Методы удаления элементов из списка

Метод remove() удаляет из списка указанный элемент, встретившийся в первую очередь. Например,

```
color = ['red', 'green', 'red', 'blue']
color.remove('red')
print(color)
```

Метод pop() удаляет последний элемент в списке, если ему не передан никакой аргумент. Например,

```
color = ['red', 'green', 'red', 'blue']
color.pop()
print(color)
```

Чтобы удалить конкретный элемент, методу pop() нужно передать его индекс. При этом данный метод возвращает тот элемент, который был удален. Например,

```
color = ['red', 'green', 'red', 'blue']
print(color.pop(2))
print(color)
```

Чтобы очистить список, можно воспользоваться методом clear(). Например,

```
color = ['red', 'green', 'red', 'blue']
color.clear()
print(color)
```

Метод count()

Метод count() позволяет определить, сколько раз встретился указанный элемент в списке. Например,

```
color = ['red', 'green', 'red', 'blue']
print(color.count('red'))
```

Метод index()

Чтобы найти, под каким номером находится элемент, можно воспользоваться методом index(). При этом, он вернет индекс первого вхождения элемента в список. Например,

```
color = ['red', 'green', 'red', 'blue']
print(color.index('red'))
```

5.3 Срезы

В списках, также, как и в строках, можно использовать срезы. Списки, в отличии от строк, являются изменяемыми объектами: можно отдельному элементу списка присвоить новое значение. Но можно менять и целиком срезы. Например,

```
nums = [1, 2, 3, 4, 5]
nums[2:4] = [7, 8, 9]
print(nums)
```

5.4 Генераторы списков

Генераторы списков позволяют заполнять список по некоторой формуле. Общий вид этой конструкции следующий:

[выражение for переменная in последовательность]

где последовательность — последовательность значений, который принимает переменная, выражение — некоторое выражение, которым будут заполнены элементы списка.

Например, заполнение списка квадратами целых чисел от 0 до 5 может выглядеть следующим образом:

```
sq = [i**2 for i in range(6)]
print(sq)
```

В цикле for переменная i пробегает значения от 0 до 6, затем ее значение возводится в квадрат и добавляется в пустой список sq.

Считывание списка с клавиатуры и приведение типов можно также реализовать с помощью генераторов:

```
n = int(input())
l = [int(input()) for i in range(n)]
print(l)
```

Приведенный выше код вначале спросит у пользователя количество элементов в списке. Затем с помощью цикла внутри генератора список l будет заполняться ровно n-раз значениями с клавиатуры, которые тут же приводятся к целочисленному типу.

Когда пользователь вводит значения через пробел, то преобразовать их к целым числам можно также в одну строку с помощью генератора:

```
l = [int(elem) for elem in input().split()]
print(l)
```

5.5 Задачи

Базовые задачи

Задача №1. Четные индексы и элементы

Пользователь вводит последовательность элементов списка через пробел. Выведите все элементы списка с четными индексами. Например,

Входные данные	Результат
1 2 2 3 3 3 4	1 2 3 4

А теперь выведите все четные элементы. Должно получиться нечто подобное:

Входные данные	Результат
1 2 2 3 3 3 4	2 2 4

Попробуйте перебирать список в этой задаче двумя способами: по индексам и по элементам.

Задача №2. Больше предыдущего

Пользователь вводит последовательность чисел через пробел. Выведите все элементы списка, которые больше предыдущего элемента. Например,

Входные данные	Результат
1 5 2 4 3	5 4

Задача №3. Первый положительный

Список вводится с клавиатуры. Вам нужно найти индекс и значение первого положительного элемента списка и вывести эти числа на экран. Например,

Входные данные	Результат
0 -1 2 -3 4 5 -6	2 2

Задача №4. Модуль числа

Пользователь вводит список, содержащий положительные и отрицательные числа. Вам необходимо заменить все элементы списка на противоположные по знаку. Результат выведите на экран. Например,

Входные данные	Результат
0 -1 2 -3 4 5 -6	0 1 -2 3 -4 -5 6

Задача №5. Соседи одного знака

Пользователь вводит список чисел через пробел. Если в нем у двух соседних элементов один знак, выведите эти числа. Если соседних элементов одного знака нет — не выводите ничего. Например,

Входные данные	Результат
-1 2 3 -1 -2	2 3 -1 -2

Задача №6. Неугодный студент

Не для кого ни секрет, что из института часто отчисляют студентов за неуспеваемость. Прошла очередная сессия и деканату нужно убрать из списка учеников студента с определенным именем. Помогите сотрудникам решить эту задачу. В первой строке пользователь вводит список фамилий. А во второй – фамилию отчисленного студента. Удалите его из списка, а результат выведите на экран. Например,

Входные данные	Результат
Ivanov Sidorov Petrov Prohorov Petrov	Ivanov Sidorov Prohorov

Задача №7. Оценки Васи

Петя решил подвести итоги четверти и посчитать, сколько он получил пятерок, четверок, троек и двоек. Пользователь вводит список цифр через пробел. В первой строке вам необходимо вывести количество пятерок, четверок, троек и двоек через пробел. Во второй – средний балл Васи. Например,

Входные данные	Результат
5 5 5 5 3 4 5 4 4	5 3 1 0 4.444444444444445

Задача №8. Подмена

Петя очень умный школьник, и поэтому уверен, что оценки в жизни не главное! И, чтобы доказать это всем, он придумал план: заменить свои плохие оценки на хорошие. А чтобы было не очень заметно, Петя решил заменить двойки на трояки. Да вот одна проблема: оценок много, а времени мало. Помогите мальчику автоматизировать процесс. Обратите внимание, числа вводятся и выводятся в строку, при этом между элементами стоит пробел. Например,

Входные данные	Результат
5 2 4 2 3	5 3 4 3 3

Дополнительные задачи

Задача №9. Статистика (acmp.ru, №5)

Вася не любит английский язык, но каждый раз старается получить хотя бы четверку за четверть, чтобы оставаться ударником. В текущей четверти Вася заметил следующую закономерность: по нечетным дням месяца он получал тройки, а по четным – четверки. Так же он помнит, в какие дни он получал эти оценки. Поэтому он выписал на бумажке все эти дни для того, чтобы оценить, сколько у него троек и сколько четверок. Помогите Васе это сделать, расположив четные и нечетные числа в разных строчках. Вася может рассчитывать на оценку 4, если четверок не меньше, чем троек.

Входные данные

В первой строке записано единственное число n – количество элементов целочисленного массива. Вторая строка содержит n чисел, представляющих заданный массив. Все элементы массива разделены пробелом.

Результат

В первую строку нужно вывести числа, которые соответствуют дням месяцев, в которые Вася получил тройки, а во второй строке соответственно расположить числа месяца, в которые Вася получил четверки. В третьей строке нужно вывести «YES», если Вася может рассчитывать на четверку и «NO» в противном случае. В каждой строчке числа следует выводить в том же порядке, в котором они идут во входных данных. При выводе, числа отделяются пробелом.

Примеры

Входные данные	Результат
5 4 16 19 31 2	19 31 4 16 2 YES
8 29 4 7 12 15 17 24 1	29 7 15 17 1 4 12 24 NO

Задача №10. Стоимость проезда (acmp.ru, №952)

Цена проезда в автобусах нашего города — один рубль. Однако, не все так просто — каждый взрослый пассажир имеет право провезти бесплатно не более одного ребенка. Это значит, что взрослый пассажир, который провозит с собой k ($k > 0$) детей, платит всего k рублей: за один билет для себя и за (k - 1) билетов для своих детей. Также взрослый может ехать без детей, в этом случае он платит всего один рубль. Известно, что дети не могут проезжать в автобусе без сопровождения взрослых. Помогите посчитать минимальную и максимальную стоимость проезда в рублях, которую могли заплатить пассажиры автобуса.

Входные данные

Вход содержит два целых числа n и m – количество взрослых и количество детей в автобусе, соответственно.

Результат

Выведите через пробел два числа – минимальную и максимальную возможную стоимость проезда, если поездка возможна, в противном случае следует вывести «Impossible».

Примеры

Входные данные	Результат
1 2	2 2
0 5	Impossible
2 2	2 3

Задача №11. SMS-спам (timus.ru, №1567)

Студент Петя решил открыть свой бизнес — он предлагает арендаторам офисов в только что открывшемся небоскрёбе Призма услуги SMS-рекламы. Услуга заключается в том, что заказчик придумывает речёвку про свою фирму, а Петя со своего сотового телефона рассылает ее как SMS-сообщение тысячам жителей Екатеринбурга, используя купленную у пиратов базу телефонных номеров горожан. Стоимость каждой речёвки определяется как

сумма стоимостей каждого символа в ней, а стоимость символов Петя определяет по незатейливой схеме: за каждое свое нажатие на кнопку телефона он берёт по 1 рублю.

Петин телефон не поддерживает T9 и имеет только английскую раскладку:

1 abc	2 def	3 ghi
4 jkl	5 mno	6 pqr
7 stu	8 vwx	9 yz
	0 .,!	# _

Символом «_» в таблице обозначен пробел. Например, чтобы набрать букву «a», надо нажать один раз на «1», букву «k» — два раза на «4», «!» — три раза на «0» и т.д.

Чтобы узнать, какой гонорар он должен получить за рекламную речёвку, которую в данный момент рассылает, Пете необходимо посчитать её стоимость по этому простому алгоритму. А поскольку Петя очень занятой и вообще не умеет считать, так как учится на философском факультете, вы, как его самый лучший друг, готовы ему помочь.

Входные данные

В единственной строке записана рекламная речёвка, состоящая из слов, пробелов, запятых, точек и восклицательных знаков. Все слова состоят из строчных английских букв.

Результат

Выведите единственное число — стоимость речёвки в петином понимании.

Пример

Входные значения	Результат
pokupaite gvozdi tolko v kompanii gvozdederov i tovarischi!	114

Задача №12. Руины титанов: сокрытый вход (timus.ru, №1910)

Сорен и Альба — известные и влиятельные маги. В своё время именно они основали магическую гильдию, которая объединила магов Северных земель и позволила им получить свободу и независимость от Южной империи. И несмотря на свой почтенный возраст, они всё ещё активно участвуют во всех важных событиях. Когда исследователи Осинских ущелий обнаружили там следы построек древних титанов, сотворителей этого мира, Сорен и Альба были одними из первых магов, кто прибыл к месту находки. Ведь титаны были самыми могущественными магами, и почти любое изучение их следов давало мощный толчок к развитию современной магии и помогало понять природу магии в целом.

Когда Сорен и Альба добрались до руин титанов, то они увидели лишь немного выступающую из скалы глухую стену, разделённую на небольшие одинаковые секции.

— Где же вход? — спросил Сорен.

— Очевидно, замаскирован — ответил Альба — не думал же ты, что титаны встретят тебя открытыми вратами. Они всегда достаточно серьёзно относились к собственной безопасности. Но я почти уверен, что где-то среди этих секций есть настоящая дверь. Причём во всех известных нам постройках титанов, насколько я помню, двери всегда шириной ровно в три таких секции.

— Это всё, конечно, хорошо, но как нам их найти? Стена длинная, а заклинание рассеивания подобных иллюзий потребует большого количества магических сил, применить его ко всей стене нам явно не удастся.

— Я чувствую, что хоть вдоль всей стены и действует заклинание маскировки, сила его весьма неравномерна. И я почти уверен, что сильнее всего оно именно там, где находятся двери. Я уже запустил заклинание

сканирования магических полей, и скоро у нас будет информация о том, с какой силой поле действует на каждой из секций. Останется лишь выбрать из них три подряд идущие с максимальной суммарной силой поля, и применить к ним заклинание рассеивания.

Входные данные

В первой строке дано единственное целое число n — количество секций стены. Во второй строке через пробел записаны n положительных целых чисел a_i — сила магического поля на каждой из секций.

Результат

Выдайте через пробел два числа — максимальную суммарную силу поля возле трёх подряд идущих секций стены и номер средней из них. Гарантируется, что ответ всегда однозначен.

Пример

Входные значения	Результат
6 1 4 4 4 1 1	12 3

Задача №13. Азартный Шрек (acmp.ru, №648)

Как-то раз Шрек решил посетить казино. Не будучи заядлым любителем азартных игр, Шрек обнаружил, что он не знает правил ни одной из игр, доступных в казино. Недолго думая, Шрек решил все-таки поиграть. Его взор привлекла игра с довольно незамысловатыми правилами.

На игровом столе лежат *N* карточек. На каждой карточке написано целое положительное число. Игра проходит между игроком и крупье. Карточки лежат на столе числами вниз. Игра заключается в том, что игрок открывает ровно *N*/2 карточек. Сумма всех чисел, написанных на карточках открытых игроком, называется “суммой игрока”. Следующим ходом крупье открывает оставшиеся *N*/2 карточек. Сумма всех чисел, написанных на карточках открытых крупье, называется “суммой крупье”. Выигрыш игрока определяется разностью чисел между “суммой игрока” и “суммой крупье”.

Очевидно, что полученная разность может быть отрицательным числом. Это свидетельствует о том, что игрок проиграл и должен казино соответствующую сумму.

Все бы ничего, но Шрек обладает способностью видеть надписи сквозь бумагу любой плотности. Ваша задача определить максимальную сумму выигрыша, которую может получить Шрек с учетом того, что он видит все числа, написанные на карточках.

Входные данные

Первая строка содержит одно четное натуральное число N. Вторая строка входного файла содержит ровно N чисел A_i – числа, написанные на игральных карточках. Все числа в строке разделяются одиночными пробелами, A_i – число, написанное на i-й карточке. Карточки нумеруются последовательно, начиная с единицы.

Результат

Единственная строка должна содержать ровно одно целое число – максимальный выигрыш, который может получить Шрек с учетом своей уникальной способности видеть числа, написанные на карточках.

Примеры

Входные значения	Результат
2 1 3	2
4 3 1 8 100	104

6 СЛОВАРИ

Структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, а по произвольному, называется **словарем** или **ассоциативным массивом**. Словарь – неупорядоченная структура данных. Последовательность расположения пар не важна. Поэтому доступ к элементам словаря по их индексу нельзя, ведь самого порядка нет.

Каждый элемент словаря состоит из двух элементов: **ключа** и **значения**. Синтаксис словаря в Python описывается такой схемой:

{ключ: значение, ключ: значение, ...}

Например, словарь, в котором будут лежать страны и их столицы может выглядеть следующим образом:

```
Capitals = {'Russia': 'Moscow',
            'Ukraine': 'Kiev',
            'Italy': 'Rome',
            'France': 'Paris'}
print(Capitals)
```

В этом случае индексом будет являться название страны, а значением – название столицы этой страны.

В словаре доступ к значениям осуществляется по **ключам**, которые заключаются в квадратные скобки (по аналогии с индексами списков). В рассматриваемом примере, ключом являются названия стран. Поэтому чтобы получить доступ к данным, нужно в квадратных скобках записать страну. Например, Capitals['Italy'].

Ключом может быть произвольный неизменяемый тип данных, например, целые и действительные числа и строки.

Значением – данными, которые идентифицируются ключом – в этом примере является столица. Например, для ключа 'Italy' значением является 'Rome'.

6.1 Создание словарей

Пустой словарь

Пустой словарь можно создать при помощи функции dict() или пустой пары фигурных скобок {}. Например,

```
Persons = {}
Cars = dict()
```

Для создания словаря с некоторым набором начальных значений можно использовать следующие конструкции:

```
Capitals = {'Russia': 'Moscow', 'Ukraine': 'Kiev', 'Italy': 'Rome'}
Capitals = dict(Russia = 'Moscow', Ukraine = 'Kiev', Italy = 'Rome')
```

Функция zip()

Функция zip() объединяет элементы из нескольких списков, строк и других итерируемых объектов.

Например, создать словарь из двух списков можно следующим образом:

```
Names = ["Ivanov I.I.", "Petrov P.P", "Babikova E. V."]
Jobs = ["Director", "Administrator", "Teacher"]
Catalog = zip(Jobs, Names)
Catalog = dict(Catalog)
```

6.2 Работа с элементами словарей

Получение элемента

Основная операция – получение значения элемента по ключу. Эта операция записывается так же, как и для списков: A[key]. Если элемента с заданным ключом нет в словаре, то возникает ошибка KeyError. Например, можно попробовать узнать имя директора и дворника из созданного выше списка:

```
print(Catalog["Director"])
```

print(Catalog["Wiper"])

Имя дворника не вывелось, потому что такого ключа и, соответственно, значения в словаре нет.

Метод get() позволяет определить значение по ключу: A.get(key). Если элемента с ключом key нет в словаре, то возвращается значение None.

В форме записи с двумя аргументами A.get(key, val) метод возвращает значение val, если элемент с ключом key отсутствует в словаре. Например,

print(Catalog.get("Director"))

print(Catalog.get("Wiper", "такого работника нет"))

Добавление элемента и изменение значения

Добавление элемента в словарь происходит следующим образом:

Catalog["Wiper"] = "Metelkin O. A."

Для обновления значения нужно просто указать ключ и новое значение:

Catalog["Wiper"] = "Snegoff I. M."

Удаление элементов

Чтобы удалить значение, можно воспользоваться конструкцией del:

del Catalog["Wiper"]

Удаление можно выполнить с помощью метода pop(). Например,

Catalog.pop("Administrator")

Получение ключей, значений и всех элементов

Если необходимо получить все ключи в словаре, пригодится метод keys(), а если все значение – values(). Например,

print(Catalog.keys(), Catalog.values())

Чтобы получить все пары "ключ, значение" в словаре, существует метод items(). Например,

print(Catalog.items())

Циклы со словарями

При использовании цикла for со словарями, переменная счетчик будет перебирать ключи из всего словаря и организован следующим образом:

```
for key in Catalog:
    print(key, Catalog[key])
```

Чтобы организовать цикл так, чтобы в переменной key был ключ элемента, а в переменной val, было его значение можно так, можно использовать метод items(). Например,

```
for key, val in Catalog.items():
    print(key, value)
```

Чтобы быстро проверить, есть ли среди всех значений в словаре определенное значение, можно воспользоваться такой конструкцией:

```
val in A.values()
```

Например, проверим, есть ли в нашем справочнике значение "Ivanov. I.I":

```
print("Ivanov I.I." in Catalog.values())
```

Также мы можем проверить, есть ли в словаре какой-то определенный ключ. Например,

```
print("Wiper" in Catalog.keys())
```

6.3 Задачи

По данной теме всего 7 задач. Они все являются базовыми.

Задача №1. Результаты четверти

Четверть подходит к концу и Пете интересно узнать, есть ли у него двойки в электронном дневнике. Журнал ведется в форме словаря и выглядит так: предмет (ключ) – итоговая оценка (значение).

Если среди оценок будет двойка, выведите сообщение "Готовься к худшему.". А если оценки только положительные, выведите – "В этот раз пронесло!".

Задача №2. Самый старший и самый младший

В программе задается два списка. В одном – имена учеников в классе. В другом – их возраста. Создайте из этих списков словарь и выведите на экран, имя самого младшего и самого старшего учеников из класса.

Задача №3. Поход в кино

Каждый сеанс в кинотеатре имеет свою стоимость. А у Пети есть всего 200 рублей. Выведите те сеансы, которые мальчик сможет посетить.

Руководство кинотеатра вводит с клавиатуры список сеансов через пробел, а затем – список цен. В итоге формируется словарь в следующем виде: сеанс – цена. Вам нужно найти те варианты, которые подходят Пете. Например,

Входные данные	Результат
Matrix Inception Lion_King 1+1 Wall-I 100 500 200 300 700	Matrix Lion_King

Задача №4. Словарь синонимов

Пользователь заполняет с клавиатуры словарь синонимов из n пар синонимов: термин – синоним. А затем вводит слово, для которого нужно найти синоним. Выведите искомый синоним на экран. Например,

Входные данные	Результат
3 Hello Hi Bye Goodbye List Array Goodbye	Bye

Задача №6. Словарь повторов

В единственной строке записан текст. Для каждого слова из данного текста подсчитайте, сколько раз оно встречалось в этом тексте ранее.

Результат оформите в виде словаря: слово – число повторов в тексте. Выведите полученный словарь на экран. Например,

Входные данные	Результат
one two one four three	four: 1 two: 1 one: 2 three: 1

Обратите внимание на вывод – в нем нет служебных символов, а между словом и числом стоит двоеточие.

Задача №7. Выборы в США

В США проходят выборы. А вам необходимо подвести их итоги.

В первой строке задается количество записей. В каждую следующую строку с клавиатуры вводится фамилия кандидата и сколько голосов за него отдали в одном из штатов. Определите, число отданных голосов для каждого участника. Участников нужно выводить в алфавитном порядке. Например,

Входные данные	Результат
5 McCain 10 McCain 5 Obama 9 Obama 8 McCain 1	McCain 16 Obama 17

ЗАКЛЮЧЕНИЕ

Учебно – методические материалы, разработаны преподавателем с учётом конкретных задач, возрастных особенностей обучающихся. Варианты демонстрационных программ, терминология, задачи, учебная и техническая литература так же представлены в пособии.

Сегодня Python является одним из самых популярных языков программирования. Основными причинами этого являются универсальность и простота.

Исходя из вышесказанного программирование на Python на сегодняшний день очень востребовано. Поэтому обучение по данному методическому пособию – первый и важный шаг в изучении языка Python.

СПИСОК ЛИТЕРАТУРЫ

1. Архив задач Школы программиста. URL: https://acmp.ru/index.asp?main=tasks.
2. Архив задач Timus Online Judge. URL: https://timus.online/problemset.aspx.
3. Интерактивный учебник языка Питон. URL: https://pythontutor.ru.
4. Шапошникова С. Курс «Python. Введение в программирование». URL: https://younglinux.info/python/course.