**Основы язык программирования Python.**

Внастоящее время общество находится на этапе глобальной информатизации и компьютеризации. Поэтому возрастает потребность в специалистах с высоким уровнем владения информационными компетенциями, отвечающих социальному заказу по подготовке квалифицированных кадров в области программирования, а также обладающих высоким интересом к IT-сфере. Одной из составляющих информационной компетентности является владение языком программирования. Встаёт вопрос о выборе языка программирования, который отвечает современным требованиям к написанию программ. Одним из таких языков является язык программирования **Python.**

Целью представленной методической разработкиявляется изучение школьниками основных приёмов написания программ на современном языке программирования, развитие алгоритмического мышления учащихся, творческих способностей, аналитических и логических компетенций.

Задачи:

* сформировать первоначальные навыки работы с языком программирования Python;
* изучить синтаксис начальных операторов;
* научиться решать задачи ЕГЭ базового уровня, используя язык программирования Python.

**Тема 1. Знакомство с языком программирования Python**

**Аннотация.** Ввод и вывод данных в языке Python. Несложные программы, которые умеют что-то выводить на экран (команда **print**) и считывать информацию с клавиатуры (команда input()).

Язык**Python**разработал голландский программист **Гвидо Ван Россум** (GuidovanRossum) в 1991 году. Гвидо был фанатом британского комедийного сериала «MontyPython’sFlyingCircus», откуда и пришло название языка.

**Преимущества Python**

1. Это интерпретируемый язык программирования:
	* + он не требует отдельного этапа компиляции;
		+ программа на языке Python запускается прямо из исходного кода;
2. Это высокоуровневый язык программирования;
3. Это платформонезависимый язык:
	* программы на Python можно **создавать**на разных операционных системах (Linux, Windows, OS X);
	* программы на Python можно **запускать**на разных
4. Для Python существует огромная библиотека классов на любой вкус.

# Ввод-вывод данных

1. Вывод данных, команда print()
2. Ввод данных, команда input()

Для вывода данных на экран используется команда print().

Внутри круглых скобок пишем, что хотим вывести на экран. Если это текст, то обязательно указываем его внутри кавычек. Кавычки могут быть одинарными или двойными. Только обязательно ставим одинаковые до и после текста.

Например, следующий код:

print('Мы изучаем язык Python')

выведет на экран текст:

Мы изучаем язык Python

**Запомните:** кавычки могут быть и одинарными, и двойными. Следующие две строки делают одно и тоже.

print('Python')

print("Python")

То, что мы пишем в круглых скобках у команды print(), называется **аргументами**или **параметрами**команды.

Команда print() позволяет указывать несколько аргументов, в таком случае их надо отделять запятыми.

Например, следующий код:

print('Скоро я', 'буду программировать', 'на языке', 'Python!')

выведет на экран текст:

Скоро я⎵будупрограммировать⎵наязыке⎵Python!

Обратите внимание, в качестве разделителя при выводе данных между аргументами команды используется **символ пробел**. По умолчанию команда print() добавляет ровно один пробел между всеми своими аргументами. Например, следующая программа:

print('1', '2', '4', '8', '16')

выводит числа 1 2 4 8 16. Числа выведены ровно через один пробел.

**Запомни:** при написании кода, между аргументами команды print() после запятой мы ставим 1 символ пробел. Это общепринятое соглашение в языке Python. Этот символ пробела не влияет на вывод данных.

## Примечания:

**Примечание 1.** Команда print() записывается только маленькими буквами, другое написание недопустимо, так как в Python строчные и заглавные буквы [**различны**](https://ru.wikipedia.org/wiki/%D0%A7%D1%83%D0%B2%D1%81%D1%82%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%BA_%D1%80%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D1%83_%D1%81%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB%D0%BE%D0%B2).

**Примечание 2.** Каждая последующая команда print() выводит указанный текст**с новой строки**. Например, следующая программа:

print('Какой хороший день!')

print('Работать мне не лень!')

выведет на экран две строки:

Какой хороший день!

Работать мне не лень!

**Примечание 3.** Команда print() с пустым списком аргументов просто вставляет новую пустую строку.

# Запуск приложения и первая программа.

1. Запускаем приложение Pythonверсии 3.8 (3.9) и выше



1. Открываем новый файл:



1. Пишем код первой программы



1. Запускаем программу



1. Сохраняем в нужную папку



1. Смотрим результат выполнения



**Решите следующие задачи:**

## Задача 1\_1. Здравствуй, мир!

Напишите программу, которая выводит на экран текст «Здравствуй, мир!» (без кавычек).

## Задача 1\_2.Счастливая последовательность 1

В популярном сериале «Остаться в живых» использовалась последовательность чисел **4 8 15 16 23 42**, которая принесла героям удачу и помогла сорвать джекпот в лотерее. Напишите программу, которая выводит данную последовательность чисел с одним пробелом между ними.

**Примечание.** Текст '4 8 15 16 23 42' не использовать. Воспользуйтесь возможностью команды print() выводить несколько аргументов, указанных через запятую.

## Задача 1\_3. Счастливая последовательность 2

Измените предыдущую программу так, чтобы каждое число последовательности **4 8 15 16 23 42** печаталось на отдельной строке.

**Примечание.** Каждая последующая команда print() выводит указанный текст, начиная с новой строки.

## Задача 1\_4. Звездный треугольник

Напишите программу, которая выводит указанный треугольник, состоящий из звездочек (\*).

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*\*\*

## Ввод данных, команда input

Все предыдущие программы выводили на экран текст, известный в момент написания программного кода. Однако программы могут работать с данными, которые станут известны только во время выполнения программы. Другими словами, программы могут считывать данные, а затем их использовать.

Для считывания данных в языке Python используется команда input().

Рассмотрим следующую программу:

print('Как тебя зовут?')

name = input()

print('Привет,', name)

Сначала программа распечатает текст на экран «Как тебя зовут?». Далее программа будет ждать от пользователя ввода данных. Ввод данных реализуется с помощью команды input().

Команда input()всегда пишется с круглыми скобками. Она работает так: когда программа доходит до места, где есть input(), она ждет, пока пользователь введёт текст с клавиатуры (ввод завершается нажатием клавиши Enter). Введенная строка подставляется на место input().

Запомни: Команда print() выводит на экран данные.

Команда input() считывает введенные с клавиатуры данные.

## Примечания:

**Примечание.**Очень часто перед считыванием данных мы печатаем некоторый текст, чтобы пользователь, который вводит эти данные понимал, что именно от него требуется. Например, в программе

print ('Как тебя зовут?')

name = input()

print ('Привет,', name)

мы сначала выведем текст «Как тебя зовут?», а уже потом считаем данные.

Поскольку это достаточно распространённый сценарий, то в языке Python можно выводить текст, передавая его в качестве параметра в команду input(). Предыдущий код можно переписать так:

name = input('Как тебя зовут?')

print ('Привет,', name)

## Задача 1\_5. Приветствие

На вход программе подается строка текста – имя человека. Напишите программу, которая выводит на экран приветствие в виде слова «Привет» (без кавычек), после которого должна стоять запятая и пробел, а затем введенное имя.

**Формат входных данных**
На вход программе подаётся одна строка — имя человека.

**Формат выходных данных**
Программа должна вывести текст в соответствии с условием задачи.

## Необязательные параметры команды print

По умолчанию команда print() принимает несколько аргументов (параметров), выводит их через один **пробел**, после чего **ставит перевод строки**. Это поведение можно изменить, используя необязательные именованные параметры sep (separator, разделитель) и end (окончание).

### Параметр sep

Рассмотрим следующий код:

print('a', 'b', 'c')

print('d', 'e', 'f')

Результатом выполнения такого кода будет:

a b c

d e f

Рассмотрим следующий код:

print('a', 'b', 'c', sep='\*')

print('d', 'e', 'f', sep='\*\*')

Результатом выполнения такого кода будет:

a\*b\*c

d\*\*e\*\*f

При первой печати в качестве строки разделителя между аргументами команды print() установлена строка sep='\*'.

При второй печати в качестве строки разделителя между аргументами команды print() установлена строка sep='\*\*'.

Таким образом, необязательный параметр sep команды print() позволяет установить строку, с помощью которой будут разделены аргументы при печати.

### Параметр end

Если перевод строки делать не нужно или требуется указать специальное окончание, то следует явно указать значение для параметра end.

Рассмотрим следующий код:

print('a', 'b', 'c', end='@')

print('d', 'e', 'f', end='@@')

Результатом выполнения такого кода будет:

a b c@d e f@@

По завершении первой печати вставлена строка @ вместо перевода строки. Аналогично, по завершении второй печати вставлена строка @@.

Параметры sep и end можно использовать вместе. Рассмотрим следующий код:

print('a', 'b', 'c', sep='\*', end='finish')

print('d', 'e', 'f', sep='\*\*', end='^\_\_^')

print ('g', 'h', 'i', sep='+', end='%')

print ('j', 'k', 'l', sep='-', end='#')

print('m', 'n', 'o', sep='/', end='!')

Результатом выполнения такого кода будет:

a\*b\*cfinishd\*\*e\*\*f^\_\_^g+h+i%j-k-l#m/n/o!

## Примечания:

**Примечание 1.** Вызов команды print() с пустыми скобками ставит перевод строки.

**Примечание 2.** Последовательность символов \n называется управляющей последовательностью и задает перевод строки.

**Примечание 3.** Значения по умолчанию у параметров sep и end следующие:

sep=' '# пробел

end='\n'# перевод строки

**Примечание 4.** Чтобы убрать все дополнительные выводимые символы, можно вызывать командуprint() так:

print('a', 'b', 'c', sep='', end='')

**Примечание 5.** Программный код

print('Python')

равнозначенкоду

print('Python', end='\n')

**Примечание 6.** Если после вывода данных нужно более одного перевода строки, то необходимо использовать следующий код:

print('Python', end='\n\n\n')

## Задача 1\_6. I likePython

Напишите программу, которая выводит на экран текст «I\*\*\*like\*\*\*Python» (без кавычек).

**Примечание.** Используйте необязательный параметр sep.

## Переменные

Мы знаем, что команда input () обозначает «подожди, пока пользователь введет что-нибудь с клавиатуры, и запомни то, что он ввёл». Просто так просить «запомнить» довольно бессмысленно: нам ведь потом надо будет как-то сказать компьютеру, чтобы он вспомнил то, что запомнил. Для этого используем **переменные**и пишем такой код:

variable\_name = input ()

print ('Выввелитекст:', variable\_name)

Данный код означает: «Сохрани то, что ввёл пользователь, в памяти, и дальше это место в памяти мы будем называть именем variable\_name». Соответственно, команда print(variable\_name) означает: «Посмотри, что лежит в памяти, под именем variable\_name, и выведи это на экран».

Вот такие «места в памяти» называются переменными. Любая переменная имеет **имя и значение**.

### ****Имя переменной****

1. В имени переменной используйте только латинские буквы a-z, A-Z, цифры и символ нижнего подчеркивания (\_);
2. Имя переменной не может начинаться с цифры;
3. Имя переменной по возможности должно отражать её назначение.

**Запомни:** Python — регистрочувствительный язык. Переменная name и Name — две совершенно разные переменные. Для именования переменных принято использовать стиль **lower\_case\_with\_underscores**(слова из маленьких букв с подчеркиваниями).

### ****Значение переменной****

Значение переменной — сохраненная в ней информация. Это может быть текст, число и т.д.

Знак «=» это **оператор присваивания**. Он присваивает значение, которое находится справа от знака «равно», переменной, которая находится слева от знака «равно».

В нашем случае это то, что поместил в неё пользователь командой input(). Это текстовое значение — строка. То есть переменная сохраняет в себе строковое значение. Говорят, что переменная имеет строковый тип данных.

**Запомни:** интерпретатор ждет, что пользователь что-то введет с клавиатуры ровно столько раз, сколько команд input() встречается в программе. Каждый input() завершается нажатием **Enter** на клавиатуре.

Значения переменной, естественно, можно менять (переприсваивать).

print('Как тебя зовут?')

name = input()

print('Привет,', name)

name = 'Timur'

print('Привет,', name)

Итак, если вы хотите, чтобы у вас была **переменная с каким-то именем и каким-то значением**, нужно написать на отдельной строчке:

<имя переменной> = <значение переменной>



Как только эта команда выполнится, в программе появится переменная с указанным значением.

## ****Множественное присваивание****

В языке Python можно за одну инструкцию присваивания изменять значение сразу нескольких переменных. Делается это так:

name, surname = 'Tim', 'Gev'

print('Имя:', name, 'Фамилия:', surname)

Этот код можно записать и так:

name = 'Tim'

surname = 'Gev'

print('Имя:', name, 'Фамилия:', surname)

Отличие двух способов состоит в том, что множественное присваивание в первом способе присваивает значение двум переменным одновременно.

Если требуется считать текст с клавиатуры и присвоить его в качестве значения переменным, то можно написать так:

name, surname = input(), input()

print('Имя:', name, 'Фамилия:', surname)

Если слева от знака «равно» в множественном присваивании должны стоять через запятую имена переменных, то справа могут стоять произвольные выражения, разделённые запятыми. Главное, чтобы слева и справа от знака присваивания было одинаковое число элементов.

Множественное присваивание удобно использовать, когда нужно обменять значения двух переменных. В Python этоделаетсятак:

name1 = 'Tim'

name2 = 'God'

name1, name2 = name2, name1

**Правильно:**

print('My name', 'is', 'Python', sep='\*\*', end='+')

**Неправильно:**

print('My name', 'is', 'Python', sep = '\*\*', end = '+')

### Однострочные комментарии

Любую строку можно превратить в комментарий, поместив перед ней символ #.

Рассмотрим следующий код:

# Это комментарий в программе на языке Python.

print('Pythonrocks!')

Если запустить эту программу, то получится:

Pythonrocks!

Первая строка при запуске игнорируется. Комментарий, начинающийся с символа #, предназначен только для автора программы и для тех, кто будет читать этот код.

### Комментарий в конце строки

Комментарий

**Правильно:**

# Далее будет напечатан текст

**Урок 2. Работа с целыми числами**

**Аннотация.** Урок посвящен изучению работы с целыми числами. Изучим основные операции, а также научимся преобразовывать строки в числа.

**Целые числа**

Все предыдущие программы, которые мы писали, работали с текстовыми данными. Действительно, команда input() считывает строку текста. Однако во многих случаях нам нужно работать именно с числами. Чтобы в Python создать переменную целого типа данных, нужно опустить кавычки при объявлении переменной. Рассмотрим следующий код:

num1 = 7# num1 - это число

num2 = 10# num2 - это число

num3 = num1 + num2 # num3 - это число

print(num3)

В результате выполнения такой программы будет выведено число 17.

**Запомни:** числа обозначаются без кавычек, а строки — с кавычками.

Рассмотрим следующую программу:

a = 3

b = 2

print(a + b)

print(a - b)

print(a \* b)

print(a / b)

Результатом выполнения такой программы будут числа:

5

1

6

1.5

**Порядок выполнения операций**

В математике существует порядок выполнения операций, определяющий, какие операции должны выполняться раньше других, даже если в выражении они написаны правее. Порядок выполнения операций в Python аналогичен порядку выполнения операций, которые вы изучали на уроках математики.

Рассмотрим следующий код:

num1 = 2 + 3\*4

num2 = (2 + 3)\*4

print(num1)

print(num2)

Результатом выполнения такой программы будут числа 14 и 20. В переменной num1 будет храниться число 14, поскольку первым делом выполняется умножение, а уже затем сложение. В переменной num2 будет храниться число 20, поскольку скобки изменили приоритет выполнения операций.

**Преобразование типов**

**Преобразование строки к целому числу.** Для того, чтобы преобразовать строку к целому числу, мы используем команду int(). Рассмотрим следующий код:

s = '1992'

year = int(s)

Переменная s имеет строковый тип данных. С помощью команды int(), мы преобразовали строку к целому числу и записали результат в переменную с именем year.  **Мы используем новую команду (точнее, функцию) – int()**. Она означает следующее: «Возьми то, что указано в качестве аргумента в скобках, и преврати это в целое число». Таким образом, переменная year имеет целый тип данных, а переменная s – строковый тип данных.

Напишем программу, которая считывает два целых числа и выводит на экран их сумму. Следующий код тут не сработает:

num1 = input()

num2 = input()

print(num1 + num2)

Python считает, что в переменных num1 и num2 находится **текст**, поскольку команда input() по умолчанию считывает именно **текст**. Для того, чтобы явно указать, что требуется работать с переменными целого типа, надо написать так:

num1 = int(input())

num2 = int(input())

print(num1 + num2)

**Запомни:** для того, чтобы считать одно целое число, мы пишем следующий код:

num = int(input())

**Преобразование целого числа к строке.** Для того, чтобы преобразовать целое число в строку, мы используем команду str(). Рассмотрим следующий код:

num = 17

s = str(17)

Переменная num имеет целый тип данных. С помощью команды str() мы преобразовали целое число в строку и записали результат в переменную s. Таким образом, переменная num имеет целочисленный тип данных, а переменная s – строковый тип данных.

**Примечания:**

**Примечание 1.** При работе с целочисленными переменными часто нужно осуществлять присвоение значению переменной целой математической формулы:

<имя переменной> = <математическая формула>



**Примечание 2.** Сокращение int происходит от английского **integer** – целый.

## Задача 2\_1. Три последовательных числа

Напишите программу вывода на экран трех последовательно идущих чисел, каждое на отдельной строке. Первое число вводит пользователь, остальные числа вычисляются в программе.

**Формат входных данных**
На вход программе подается одно целое число. **8**

**Формат выходных данных**
Программа должна вывести три последовательно идущих числа в соответствии с условием задачи.

8

9

10

## Задача 2\_2. Сумма трёх чисел

Напишите программу, которая считывает три целых числа и выводит на экран их сумму. Каждое число записано в отдельной строке.

**Формат входных данных**
На вход программе подаётся три целых числа, каждое на отдельной строке.

9

11

2

**Формат выходных данных**
Программа должна вывести сумму введенных чисел.

22

## Задача 2\_3. Куб

Напишите программу, вычисляющую объём куба и площадь его полной поверхности, по введённому значению длины ребра.

**Формат входных данных**
На вход программе подается одно целое число – длина ребра. **25**

**Формат выходных данных**
Программа должна вывести текст и числа в соответствии с условием задачи.

**Объем = 15625**

**Площадь полной поверхности = 3750**

**Примечание.** Объём куба и площадь полной поверхности можно вычислить по формулам *V*=*a*3,*S*=6*a*2.  Операция \*\* обозначает возведение в степень (Например а2 запишем, как а \*\* 2)

## Задача 2\_4. Следующее и предыдущее

Напишите программу, которая считывает целое число, после чего на экран выводится следующее и предыдущее целое число с пояснительным текстом.

**Формат входных данных**
На вход программе подаётся целое число.**20**

**Формат выходных данных**
Программа должна вывести текст согласно условию задачи.

**Следующее за числом 20 число: 21**

**Для числа 20 предыдущее число: 19**

## Задача 2\_5. ****Стоимость покупки****

Напишите программу, которая считает стоимость **трех компьютеров**, состоящих из монитора, системного блока, клавиатуры и мыши.

**Формат входных данных**
На вход программе подаётся четыре целых числа, каждое на отдельной строке. В первой строке — стоимость монитора, во второй строке — стоимость системного блока, в третьей строке — стоимость клавиатуры и в четвертой строке — стоимость мыши.

**Формат выходных данных**
Программа должна вывести одно число – стоимость покупки (трех компьютеров).

**Sample Input 1:**

9900

55600

3999

2990

**Sample Output 1:**

217467

**Sample Input 2:**

15700

80550

12050

5890

## Задача 2\_6. ****Арифметические операции****

Напишите программу, в которой вычисляется сумма, разность и произведение двух целых чисел, введенных с клавиатуры.

**Формат входных данных**
На вход программе подаётся два целых числа, каждое на отдельной строке.

**Формат выходных данных**
Программа должна вывести сумму, разность и произведение введённых чисел, каждое на отдельной строке.

**Sample Input 1:**

2

7

**Sample Output 1:**

2 + 7 = 9

2 - 7 = -5

2 \* 7 = 14

**Sample Input 2:**

5

8

**SampleOutput 2:**

5 + 8 = 13

5 - 8 = -3

5 \* 8 = 40

## Задача 2\_7. Арифметическая прогрессия

**Арифметической прогрессией** называется последовательность чисел a\_1, a\_2, ..., a\_n каждое из которых, начиная с a\_2, получается из предыдущего прибавлением к нему одного и того же постоянного числа d(разность прогрессии), то есть:

a\_n=a\_{n−1}+d

Если известен первый член прогрессии и её разность, то n-ый член арифметической прогрессии находится по формуле: **a\_n=a\_1+d(n-1)**

**Входные данные**
На вход программе подаётся три целых числа: **a\_1**, **d** и **n**, каждое на отдельной строке.

**Выходные данные**
Программа должна вывести n-ый член арифметической прогрессии.

**Sample Input 1:**

1

1

10

**Sample Output 1:**

10

**Sample Input 2:**

-1

1

2

**SampleOutput 2:**

0

# Урок 3. Работа с целыми числами

**Аннотация.** Урок посвящен дополнительным операциям при работе с целыми числами. Изучим дополнительные операции, а также научимся обрабатывать цифры целого числа.

## Дополнительные операции

Мы познакомились с 4 основными математическими операциями в языке Python: сложение, вычитание, умножение и деление . Добавив ещё 3 операции.

|  |  |
| --- | --- |
| **Оператор** | **Описание** |
| \*\* | Возведение в степень |
| % | Остаток от деления |
| // | Целочисленное деление |

### Возведение в степень

Оператор возведения в степень a \*\* n возводит число *a* в степень *n*. Рассмотрим работу данного оператора на примерах:

print(2 \*\* 0)

print(2 \*\* 1)

print(2 \*\* 2)

print(2 \*\* 3)

print(2 \*\* (-1))

Результатом выполнения такой программы будет:

1

2

4

8

0.5

**Обратите внимание:** оператор возведения в степень \*\* может возводить не только в положительную степень, но и в отрицательную. Аналогично, основание степени *a* также может быть числом отрицательным.

**Запомни:** в отличие от многих языков программирования, в языке Python для возведения в степень используется оператор \*\*, а не ^.

### Целочисленное деление

Для положительных чисел оператор целочисленного деления ведёт себя как обычное деление, за исключением того, что он отбрасывает десятичную часть результата. Рассмотрим работу данного оператора на примерах:

print(10 // 3)

print(10 // 4)

print(10 // 5)

print(10 // 6)

print(10 // 12)

Результатом выполнения такой программы будет:

3

2

2

1

0

При делении отрицательных чисел необходимо помнить, что результат целочисленного деления не превосходит частное. Другими словами, округление берётся в меньшую сторону (число -4−4 меньше, чем число -3−3).

Результатом работы следующей программы:

print(10 // 3)

print(-10 // 3)

будут числа:

3 # округление в меньшую сторону

-4 # округление в меньшую сторону

### Деление с остатком

Оператор деления с остатком возвращает остаток от деления двух целых чисел. Рассмотрим работу данного оператора на примерах:

print(10 % 3)

print(10 % 4)

print(10 % 5)

print(10 % 6)

print(10 % 12)

print(10 % 20)

Результатом выполнения такой программы будет:

1

2

0

4

10

10

## Примечания:

**Примечание 1.** Оператор нахождения остатка очень полезен при решении многих задач. Например, число делится на *n* нацело тогда и только тогда, когда остаток от деления на *n* равен 0.

**Примечание 2.** Операторы // и % имеют такой же приоритет, как и операторы умножения и обычного деления.

**Примечание 3.** Наивысший приоритет имеет оператор возведения в степень \*\*.

**Примечание 5.** Обратите внимание: результатом деления n % m при n < m является число n. Например, 5 % 9 = 5, 3 % 13 = 3 и т.д.

## Задача 3\_1. ****Расстояние в метрах****

Напишите программу, которая находит полное число метров по заданному числу сантиметров.

**Формат входных данных**
На вход программе подаётся натуральное число – количество сантиметров.

**Формат выходных данных**
Программа должна вывести одно число – полное число метров.

**Sample Input 1:**

345

**Sample Output 1:**

3

**Sample Input 2:**

100

**SampleOutput 2:**

1

## Задача 3\_2. Мандарины

*n* школьников делят *k* мандаринов поровну, неделящийся остаток остается в корзине. Сколько целых мандаринов достанется каждому школьнику? Сколько целых мандаринов останется в корзине?

**Формат входных данных**
На вход программе подаётся два целых числа: количество школьников и количество мандаринов, каждое на отдельной строке.

**Формат выходных данных**
Программа должна вывести два числа: количество мандаринов, которое достанется каждому школьнику, и количество мандаринов, которое останется в корзине, каждое на отдельной строке.

**Sample Input 1:**

3

6

**Sample Output 1:**

2

0

**Sample Input 2:**

12

6

**SampleOutput 2:**

0

6

## Задача 3\_3. Пересчет временного интервала

Напишите программу для пересчёта величины временного интервала, заданного в минутах, в величину, выраженную в часах и минутах.

**Формат входных данных**
На вход программе подаётся целое число – количество минут.

**Формат выходных данных**
Программа должна вывести текст в соответствии с условием задачи.

**Sample Input 1:**

150

**Sample Output 1:**

150 мин - это 2 час 30 минут.

**Sample Input 2:**

50

**Sample Output 2:**

50 мин - это 0 час 50 минут.

**Обработка цифр числа**

При помощи операции нахождения остатка и целочисленного деления можно достаточно несложно вычислить любую цифру числа.

Рассмотрим программу получения цифр **двузначного числа**:

num = 17

a = num % 10

b = num // 10

print(a)

print(b)

Результатом выполнения программы будут два числа:

7

1

То есть сначала мы вывели последнюю цифру числа, а затем первую цифру.

**Запомни:** последняя цифра числа определяется всегда как **остаток от деления** числа на 10 (а % 10). Чтобы отщепить последнюю цифру от числа, необходимо разделить его **нацело**на 10 (а // 10).

Рассмотрим программу получения цифр **трёхзначного числа**:

num = 754

a = num % 10

b = (num % 100) // 10

c = num // 100

print(a)

print(b)

print(c)

Результатом выполнения программы будут три числа:

4

5

7

То есть сначала мы вывели последнюю цифру числа, затем среднюю цифру, а затем первую цифру.

**Алгоритм получения цифр***n***-значного числа**

Несложно понять, по какому алгоритму можно найти каждую цифру *n*-значного числа num:

* Последняя цифра: (num % 101) // 100;
* Предпоследняя цифра: (num % 102) // 101;
* Предпредпоследняя цифра: (num % 103) // 102;
* .....
* Вторая цифра: (num % 10n-1) // 10n-2;
* Первая цифра: (num % 10n) // 10n-1.

**Примеры решения задач**

**Задача 1.** Напишите программу, определяющую число десятков и единиц в двузначном числе.

**Решение.** Число единиц – это последняя цифра числа, число десятков – первая цифра. Чтобы получить последнюю цифру любого числа, нужно найти остаток от деления числа на 10. Чтобы найти первую цифру двузначного числа, нужно поделить число нацело на 10. Программа, решающая поставленную задачу, может иметь следующий вид:

num = int(input())

last\_digit = num % 10

first\_digit = num // 10

print('Числодесятков =', first\_digit)

print('Числоединиц =', last\_digit)

**Задача 2.** Напишите программу, в которой рассчитывается сумма цифр двузначного числа.

**Решение.** Программа, решающая поставленную задачу, может иметь следующий вид:

num = int(input())

last\_digit = num % 10

first\_digit = num // 10

print('Суммацифр =', last\_digit + first\_digit)

**Задача 3.** Напишите программу, которая печатает число, образованное при перестановке цифр двузначного числа.

**Решение.** Программа, решающая поставленную задачу, может иметь следующий вид:

num = int(input())

last\_digit = num % 10

first\_digit = num // 10

print('Искомоечисло =', last\_digit \* 10 + first\_digit)

**Задача 4.** Напишите программу, в которую вводится трёхзначное число и которая выводит на экран его цифры (через запятую).

**Решение.** Программа, решающая поставленную задачу, может иметь следующий вид:

num = int(input())

digit3 = num % 10

digit2 = (num // 10) % 10

digit1 = num // 100

print(digit1, digit2, digit3, sep=',')

## Задача 3\_4. ****Трехзначное число****

Напишите программу, в которой рассчитывается сумма и произведение цифр положительного трёхзначного числа.

**Формат входных данных**
На вход программе подаётся положительное трёхзначное число.

**Формат выходных данных**
Программа должна вывести два числа с поясняющим текстом: сумма цифр и произведение цифр.

**Sample Input 1:**

123

**Sample Output 1:**

Сумма цифр = 6

Произведение цифр = 6

**Sample Input 2:**

333

**Sample Output 2:**

Сумма цифр = 9

Произведение цифр = 27

## Задача 3\_5. ****Перестановка цифр****

Дано трехзначное положительное число ***abc***, в котором все цифры различны. Напишите программу, которая выводит шесть чисел, образованных при перестановке цифр заданного числа.

**Формат входных данных**
На вход программе подаётся положительное трёхзначное целое число, все цифры которого различны.

**Формат выходных данных**
Программа должна вывести шесть чисел, образованных при перестановке цифр заданного числа в следующем порядке: abc, \, acb, \, bac, \, bca, \, cab, \, cba

**Sample Input 1:**

123

**Sample Output 1:**

123

132

213

231

312

321

## Задача 3\_6. ****Четырёхзначное число****

Напишите программу для нахождения цифр четырёхзначного числа.

**Формат входных данных**
На вход программе подаётся положительное четырёхзначное целое число.

**Формат выходных данных**
Программа должна вывести текст в соответствии с условием задачи.

**Sample Input 1:**

3281

**Sample Output 1:**

Цифра в позиции тысяч равна 3

Цифра в позиции сотен равна 2

Цифра в позиции десятков равна 8

Цифра в позиции единиц равна 1

**SampleInput 2:**

1000

**SampleOutput 2:**

Цифра в позиции тысяч равна 1

Цифра в позиции сотен равна 0

Цифра в позиции десятков равна 0

Цифра в позиции единиц равна 0

**Тема 4. Числовые типы данных**

1. Целочисленный тип данных int
2. Числа с плавающей точкой float
3. Встроенные функции max(), min(), abs()
4. Решение задач

**Аннотация.** Числовые типы данных. Вспомним особенности работы с целыми числами, научимся работать с числами с плавающей точкой. Изучим три встроенные функции для работы с числами max, min, abs.

**Целочисленный тип данных**

Целые числа в Python представлены типом данных int (сокращение int происходит от слова integer). Для определения целого числа типа int используется последовательность цифр от 0 до 9.

Явно указанное численное значение в коде программы называется **целочисленным литералом**. Когда Python встречает целочисленный литерал, он создает объект типа int, хранящий указанное значение.

n = 17# целочисленный литерал

m = 7# целочисленный литерал

Целочисленный тип данных int используют не только потому, что он встречается в реальном мире, но и потому, что он естественным образом возникает при создании большинства программ.

**Преобразование строки в целое число**

Для преобразования строки в целое число, мы используем команду int():

num = int(input()) # преобразование считанной строки в целое число

Для преобразования строки в целое число не обязательно использовать команду input().

Следующий код преобразует строку 12345 в целое число:

n = int('12345') # преобразование строки в целое число

Если строка не является числом, то при преобразовании возникнет ошибка.

**Целочисленные операторы**

Язык Python предоставляет четыре основных арифметических операторов для работы с целыми числами (+, **−**, \*, /), а также три дополнительных (% для остатка, // для целочисленного деления и \*\* для возведения в степень).

Следующая программа демонстрирует все целочисленные операторы:

a = 13

b = 7

total = a + b

diff = a - b

prod = a \* b

div1 = a / b

div2 = a // b

mod = a % b

exp = a \*\* b

print(a, '+', b, '=', total)

print(a, '-', b, '=', diff)

print(a, '\*', b, '=', prod)

print(a, '/', b, '=', div1)

print(a, '//', b, '=', div2)

print(a, '%', b, '=', mod)

print(a, '\*\*', b, '=', exp)

В результате работы такой программы будет выведено:

13 + 7 = 20

13 - 7 = 6

13 \* 7 = 91

13 / 7 = 1.8571428571428572

13 // 7 = 1

13 % 7 = 6

13 \*\* 7 = 62748517

При обычном делении (/) получается число не являющееся целым. Деление на ноль приводит к ошибке.

**Длинная арифметика**

Отличительной особенностью языка Python является неограниченность целочисленного типа данных. По факту, размер числа зависит только от наличия свободной памяти на компьютере. Это отличает Python от таких языков как C++, C, C#, Java где переменные целого типа данных имеют ограничение.

atom = 10 \*\* 80# количество атомов во вселенной

print('Количество атомов =', atom)

Результатом выполнения программы будет число с 81 цифрой:

Количество атомов = 100000000000000000000000000000000000000000000000000000000000000000000000000000000

**Символ разделитель**

Для удобного чтения чисел можно использовать символ подчеркивания:

num1 = 25\_000\_000

num2 = 25000000

print(num1)

print(num2)

Результатом выполнения такого кода будет:

25000000

25000000

**Числа с плавающей точкой**

Наравне с целыми числами в Python есть возможность работы с дробными (вещественными) числами.  Дробные (вещественные) числа в информатике называют **числами с плавающей точкой**.

Для представления чисел с плавающей точкой в Python используется тип данных **float**.

e = 2.71828# литерал с плавающей точкой

pi = 3.1415# литерал с плавающей точкой

В отличие от математики, где разделителем является запятая, в информатике используется точка.

**Преобразование строки к числу с плавающей точкой**

Для преобразования строки к числу с плавающей точкой мы используем команду float():

num = float(input()) # преобразование считанной строки в число с плавающей точкой

Для преобразования строки к числу с плавающей точкой необязательно использовать команду input().

Следующий код преобразует строку 1.2345  к числу с плавающей точкой:

n = float('1.2345') # преобразование строки к числу с плавающей точкой

Если строка не является числом, то при преобразовании возникнет ошибка.

**Арифметические операторы**

Язык Python предоставляет четыре основных арифметических оператора для работы с числами с плавающей точкой (+, −, \*, /) и один дополнительный (\*\* для возведения в степень).

Следующая программа демонстрирует все целочисленные операторы:

a = 13.5

b = 2.0

total = a + b

diff = a - b

prod = a \* b

div = a / b

exp = a \*\* b

print(a, '+', b, '=', total)

print(a, '-', b, '=', diff)

print(a, '\*', b, '=', prod)

print(a, '/', b, '=', div)

print(a, '\*\*', b, '=', exp)

В результате работы такой программы будет выведено:

13.5 + 2.0 = 15.5

13.5 - 2.0 = 11.5

13.5 \* 2.0 = 27.0

13.5 / 2.0 = 6.75

13.5 \*\* 2.0 = 182.25

Деление на ноль приводит к ошибке.

**Преобразование между int и float**

**Неявное преобразование.** Любое целое число (тип int) можно использовать там, где ожидается число с плавающей точкой (тип float), поскольку при необходимости Python автоматически преобразует целые числа в числа с плавающей точкой.

**Явное преобразование.** Число с плавающей точкой нельзя неявно преобразовать в целое число. Для такого преобразования необходимо использовать явное преобразование с помощью команды int().

num1 = 17.89

num2 = -13.56

num3 = int(num1)

num4 = int(num2)

print(num3)

print(num4)

Результатом выполнения такого кода будет:

17

-13

Обратите внимание, что преобразование чисел с плавающей точкой в целое производится с округлением в сторону нуля, то есть int(1.7) = 1, int(-1.7) = -1.

Не путайте операцию преобразования и округления. Для округления чисел с плавающей точкой используются дополнительные команды. О них расскажем позже.

**Задача 4\_1. Площадь треугольника**

Напишите программу, которая считывает длины двух катетов в прямоугольном треугольнике и выводит его площадь.



**Формат входных данных**
На вход программе подаётся два числа с плавающей точкой – длины катетов, каждое на отдельной строке.

**Формат выходных данных**
Программа должна вывести одно число – площадь треугольника.

**Sample Input 1:**

443

390

**Sample Output 1:**

86385.0

**Sample Input 2:**

32.2

25.5

**Sample Output 2:**

410.55

## Задача 4\_2.****Две старушки****

Две старушки идут навстречу друг другу с постоянными скоростями V\_1​ и V\_2​ км/ч. Определите, через какое время старушки встретятся, если расстояние между ними равно S км.



**Формат входных данных**
На вход программе подаются три числа с плавающей точкой S, V\_1, V\_2 каждое на отдельной строке.

**Формат выходных данных**
Программа должна вывести одно число в соответствии с условием задачи.

**Примечание**
Это очень быстрые старушки.

**Sample Input 1:**

60.0

15.5

14.5

**Sample Output 1:**

2.0

**Sample Input 2:**

100.0

60.75

19.25

**Sample Output 2:**

1.25

## https://ucarecdn.com/4df00cbb-37df-4955-9f01-4da08fd44cad/Задача 4\_3. Обратное число

Напишите программу, которая считывает с клавиатуры одно число и выводит [обратное](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE)ему. Если при этом введённое с клавиатуры число – ноль, то вывести «Обратного числа не существует» (без кавычек).

**Формат входных данных**
На вход программе подается одно действительное число.

**Формат выходных данных**
Программа должна вывести действительное число обратное данному, либо текст в соответствии с условием задачи.

**Sample Input 1:**

2.5

**Sample Output 1:**

0.4

**Sample Input 2:**

-55.6

**SampleOutput 2:**

-0.017985611510791366

## Задача 4\_4. Первая цифра после точки

Дано положительное действительное число. Выведите его первую цифру после десятичной точки.

**Формат входных данных**
На вход программе подается положительное действительное число.

**Формат выходных данных**
Программа должна вывести цифру в соответствии с условием задачи.

**Sample Input 1:**

3384390.4339

**Sample Output 1:**

4

**Sample Input 2:**

1.5

**SampleOutput 2:**

5

## Задача 4\_5. Дробная часть

Дано положительное действительное число. Выведите его дробную часть.

**Формат входных данных**
На вход программе подается положительное действительное число.

**Формат выходных данных**
Программа должна вывести дробную часть числа в соответствии с условием задачи.

**Sample Input 1:**

44.45

**Sample Output 1:**

0.45

**Sample Input 2:**

39483.2

**SampleOutput 2:**

0.2

## Урок 5. Встроенные функции

Python включает много заранее определенных функций. Программист не видит их реализацию, она скрыта. Достаточно знать, как эти функции называются и что они делают.

Мы уже сталкивались с встроенными функциями:

* print() — вывести на экран;
* input() — считать с клавиатуры;
* int() — преобразовать к целому числу;
* float() — преобразовать к числу с плавающей точкой.

Рассмотрим три новых встроенных функции, которые используются достаточно часто при работе с числами.

### Функции min() и max()

Для определения соответственно минимального или максимального значения используются функции min() и max(). Аргументов у этих функций может быть любое количество, главное, чтобы они все были одного типа.

Например, результатом выполнения следующего кода:

a = max(3, 8, -3, 12, 9)

b = min(3, 8, -3, 12, 9)

c = max(3.14, 2.17, 9.8)

print(a)

print(b)

print(c)

будет:

12

-3

9.8

### Функция abs()

Модулем ([абсолютной величиной](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D0%BE%D0%BB%D1%8E%D1%82%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0)) положительного числа называется само число, модулем отрицательного числа называется противоположное ему число, модуль нуля – нуль. Модуль числа a обозначается |a|.

Для нахождения модуля (абсолютной величины) числа в Python используется функция abs().

Например, результатом выполнения следующего кода:

print(abs(10))

print(abs(-7))

print(abs(0))

print(abs(-17.67))

будет:

10

7

0

17.67

Обратите внимание, все три функции max(), min(), abs() работают как с целыми числами, так и с числами с плавающей точкой.

**Задача 5\_1. Наибольшее и наименьшее**

Напишите программу, которая находит наименьшее и наибольшее из пяти чисел.

**Формат входных данных**
На вход программе подается пять целых чисел, каждое на отдельной строке.

**Формат выходных данных**
Программа должна вывести наименьшее и наибольшее число с поясняющей надписью.

**Sample Input 1:**

1

2

3

4

5

**Sample Output 1:**

Наименьшее число = 1

Наибольшее число = 5

**Sample Input 2:**

454

453

32

8

6769

**Sample Output 2:**

Наименьшее число = 8

Наибольшее число = 6769

## Задача 5\_2. Сортировка трёх

Напишите программу, которая упорядочивает три числа **от большего к меньшему**.

**Формат входных данных**
На вход программе подается три целых числа, каждое на отдельной строке.

**Формат выходных данных**
Программа должна вывести три числа, каждое на отдельной строке, упорядоченных от большего к меньшему.

**Sample Input 1:**

132

129

135

**Sample Output 1:**

135

132

129

**Sample Input 2:**

150

160

156

**Sample Output 2:**

160

156

150

## Задача 5\_3. Интересное число

Назовем число интересным, если в нем разность максимальной и минимальной цифры равняется средней по величине цифре. Напишите программу, которая определяет интересное число или нет. Если число интересное, следует вывести – «Число интересное» иначе «Число неинтересное».

**Формат входных данных**
На вход программе подается целое трехзначное число.

**Формат выходных данных**
Программа должна вывести текст в соответствии с условием задачи.

**Sample Input 1:**

945

**Sample Output 1:**

Числоинтересное

**Sample Input 2:**

123

**Sample Output 2:**

Числоинтересное

**Sample Input 3:**

896

**Sample Output 3:**

Числонеинтересное

## Задача 5\_4. ****Абсолютная сумма****

Даны пятьчисел a\_1, a\_2, a\_3, a\_4, a\_5. Напишите программу, которая вычисляет сумму их модулей |a\_1| + |a\_2| +|a\_3| +|a\_4| + |a\_5|.

**Формат входных данных**
На вход программе подается пять действительных чисел a\_1, a\_2, a\_3, a\_4, a\_5 каждое на отдельной строке.

**Формат выходных данных**
Программа должна вывести одно число – сумму модулей введенных чисел.

**Sample Input 1:**

5.4

33

-1232

-3.889

6

**Sample Output 1:**

1280.289

**Sample Input 2:**

0

-776.4

0

343

55.24

**Sample Output 2:**

1174.64

## Задача 5\_5.Манхэттенское расстояние

Прогуливаясь по Манхэттену, вы не можете попасть из точки А в точку Б по кратчайшему пути. Если только вы не умеете проходить сквозь стены, вам обязательно придется идти вдоль его параллельно-перпендикулярных улиц.

На плоскости [манхэттенское расстояние](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B5_%D0%B3%D0%BE%D1%80%D0%BE%D0%B4%D1%81%D0%BA%D0%B8%D1%85_%D0%BA%D0%B2%D0%B0%D1%80%D1%82%D0%B0%D0%BB%D0%BE%D0%B2) между двумя точками (*p*1​;*p*2​) и (*q*1​;*q*2)​ определяется так ∣*p*1​−*q*1​∣+∣*p*2​−*q*2​∣.

Напишите программу, определяющую манхэттенское расстояние между двумя точками, координаты которых заданы.



**Формат входных данных**
На вход программе подается четыре целых числа, каждое на отдельной строке

**Формат выходных данных**
Программа должна вывести одно число – манхэттенское расстояние.

**Sample Input:**

10

15

21

13

**Sample Output:**

13