Языки программирования:

Низкоуровневые – они представляют собой символьную интерпретацию машинного кода (Асемблер)

Высокоуровневый – используются понятные человеку слова(англ) функции (Python)

Программа переводчик – транслятор.



Переводят исходный код программы на высокоуровневом языке в машинный код, понятный процессору.

Транслятор 2 вида:

* Компилятор – сначала анализирует весь код, затем создает исполняемый файл, после запускает. Если есть ошибки файл не будет создан.
* Интерпретатор – исполняет программу сразу построчно. Если натыкается на ошибку останавливается. (Python)

Также языки программирования классифицируются по **парадигмам**. Парадигма программирования — это совокупность идей и понятий, определяющих стиль написания компьютерных программ. Выделяют следующие парадигмы программирования:

* Императивное программирование
* Процедурное программирование
* Объектно-ориентированное программирование
* Декларативное программирование
* Функциональное программирование
* Логическое программирование

Для **императивного программирования** характерна запись программы в виде последовательно выполняемых инструкций (команд). К нему обычно относят процедурные и объектно-ориентированные языки. В **процедурном программировании** программа состоит из шагов, которые могут объединяться в более крупные структурные единицы - подпрограммы. К таким языкам относятся Basic, C, Pascal. В **объектно-ориентированной** парадигме программы представляются в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы могут наследовать друг друга. Считается, что объектно-ориентированное программирование наиболее абстрактное и приближенное к мышлению человека.

В **декларативном программировании** задаётся спецификация решения задачи, то есть описывается, что представляют собой проблема и ожидаемый результат. К подвидам декларативного программирования также зачастую относят **функциональное** (процесс вычисления трактуется как вычисление значений функций в математическом понимании) и **логическое программирование**(основано на теории и аппарате математической логики с использованием математических принципов резолюций).

Языки можно разделить на 2 группы:

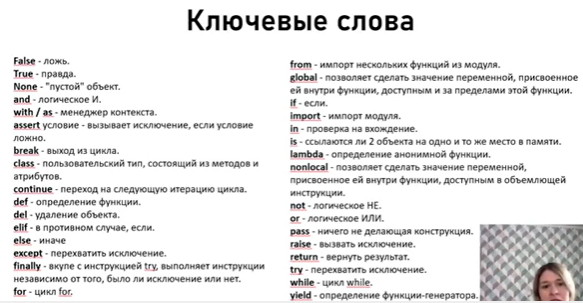
* Императивные (программа состоит из команд/процедур/объектов)
* Декларативные (программа состоит из описаний)

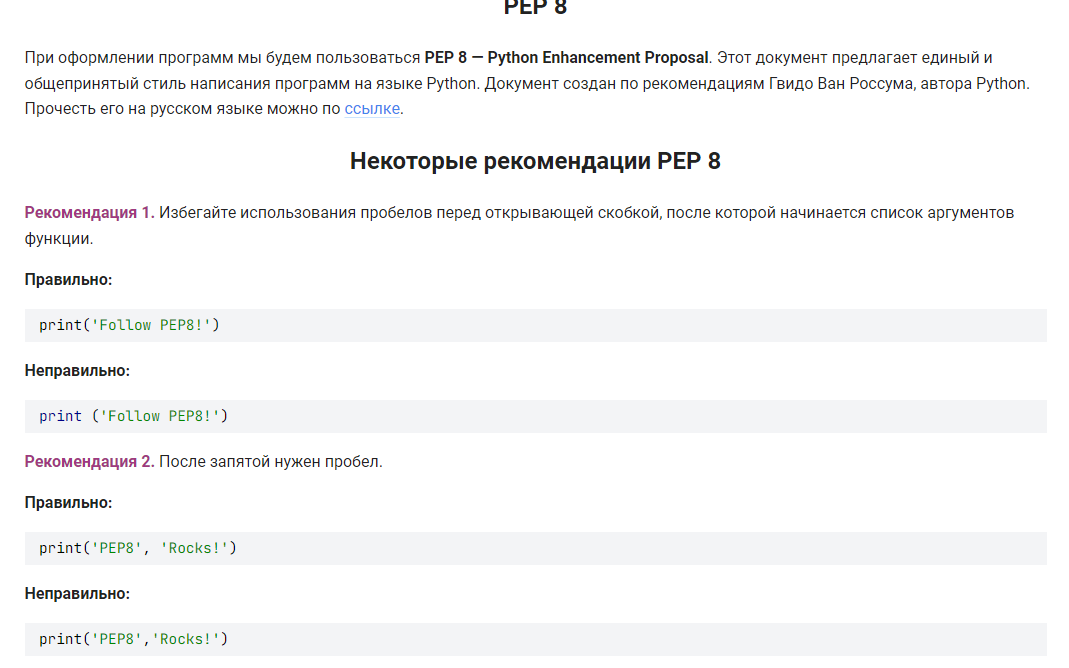
Императивные делятся на:

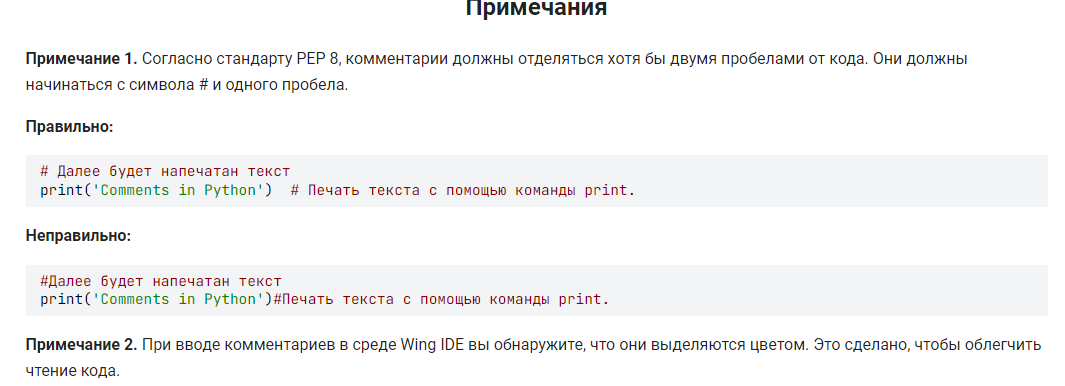
* Процедурные (Паскаль, С)
* Объектно-ориентированные (С++, Java, Python)

ООП (Объектно-ориентированное программирование)

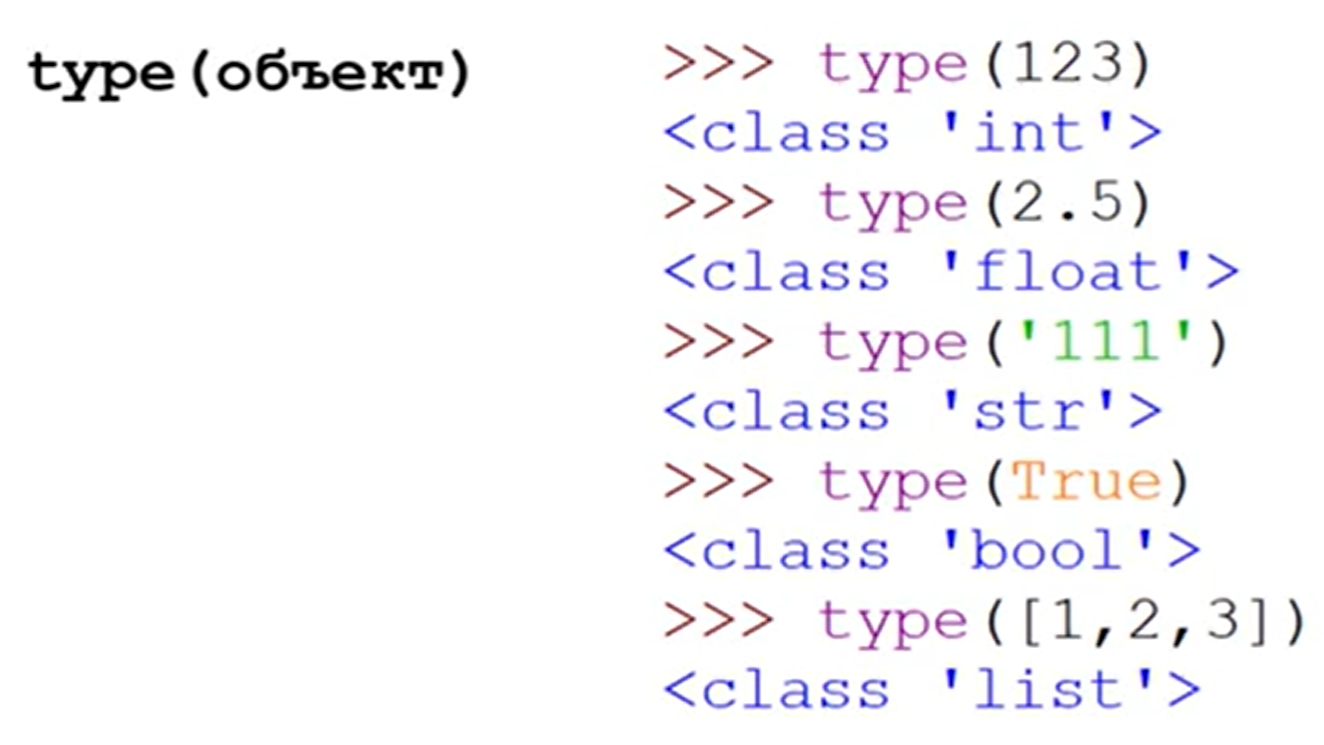
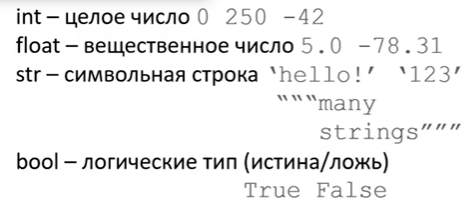
* Все состоит из объектов
* Объект принадлежит определенному классу
* Классы могут наследовать друг друга
* У объекта есть свойства (характеристики) и методы(действия)
* Объекты можно создавать и удалять



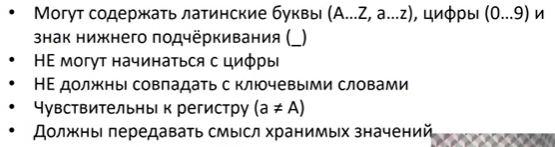
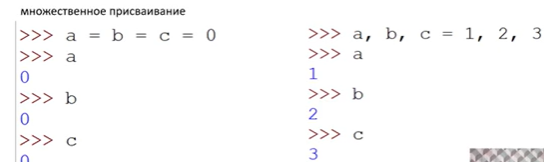


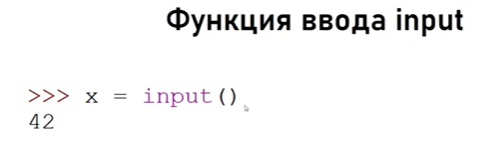


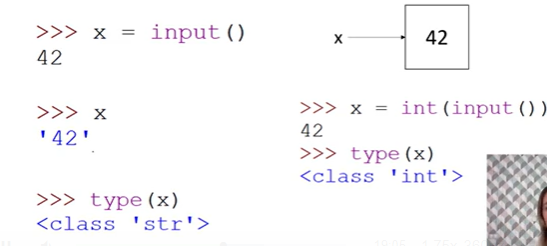
Типы данных. Переменные ввод-вывод.

Типы данных 

С помощью функции **type** мы можем определить, к какому типу относится то или иное значение. Результатом выполнения функции будет строка, начинающаяся со слова class и затем название типа данных.

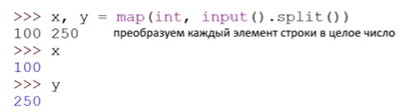
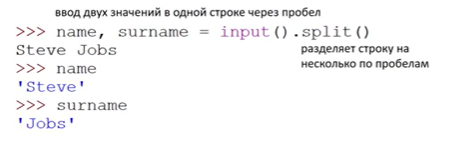


Функция input всегда имеет строковое значение.

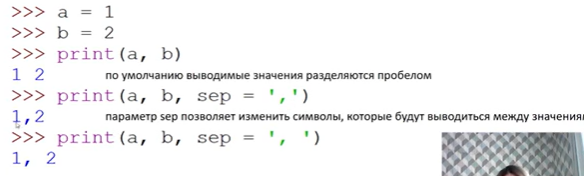
 Преобразование значение при помощи функции type

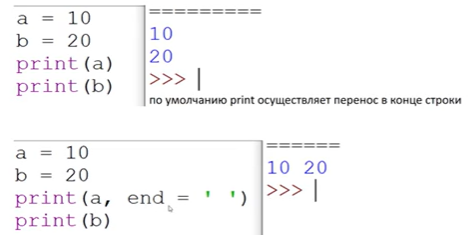
Перенос на новую строку \n

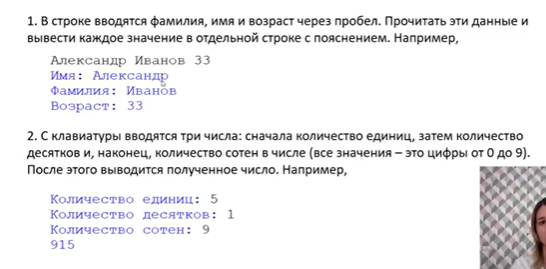
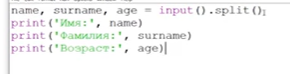
Часто несколько значений вводятся в одной строке и разделяются пробелом. Для того, чтобы присвоить каждое значение отдельной переменной, воспользуйтесь методом **split**. Он вызывается через точку после вызова функции input. Слева от оператора присваивания нужно перечислить через запятую имена переменных в том порядке, в котором необходимо присвоить прочитанные значения.

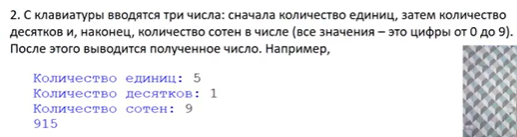


Если значения имеют числовой тип, то для преобразования нам понадобится функция **map**. Первым её параметром мы укажем, в какой тип нужно преобразовать каждое из считанных значений, а вторым параметром – саму считанную строку, разделенную по пробелам.



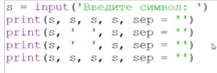


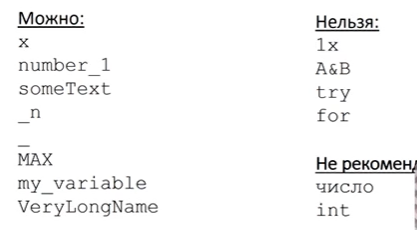
Задача: Решение: 

Задача:  Решение: 

Задача:  Решение: 

Задача: 

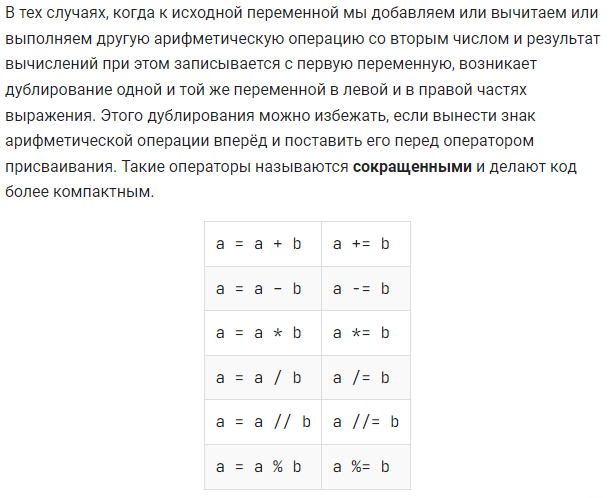
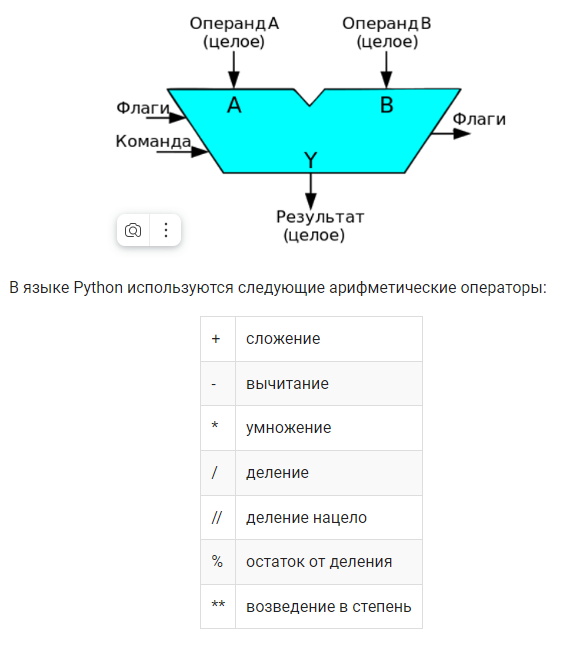
Решение: 

Имена переменных: 

Арифметические операторы

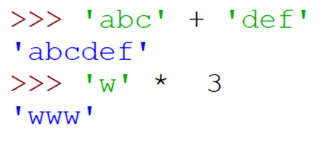
Выполнение арифметических действий – это основное назначение процессора компьютера. Изначально компьютер предназначен именно для выполнения вычислений. И даже если вы редактируете изображение на мониторе, для процессора это превращается в операции сложения, вычитания, умножения или деления, применяемые к двоичным числам.

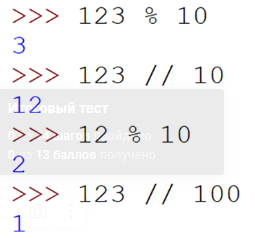
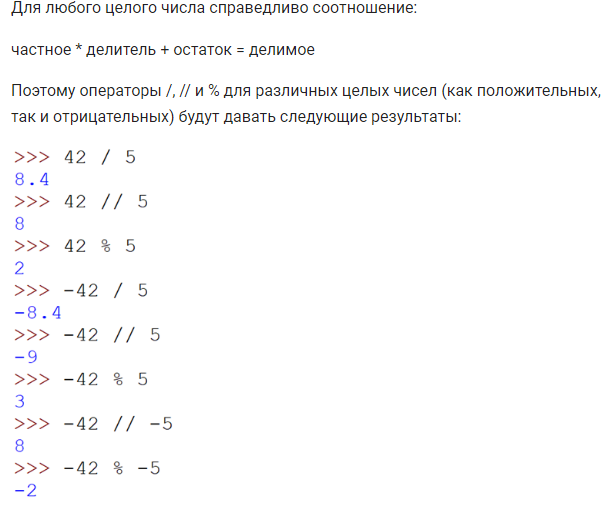
Основной вычислительный блок процессора – это АЛУ, арифметико-логическое устройство. Оно принимает на вход до двух значений операндов, а также код команды, которую необходимо выполнить. После выполнения команды, мы получаем на выходе результат вычислений.



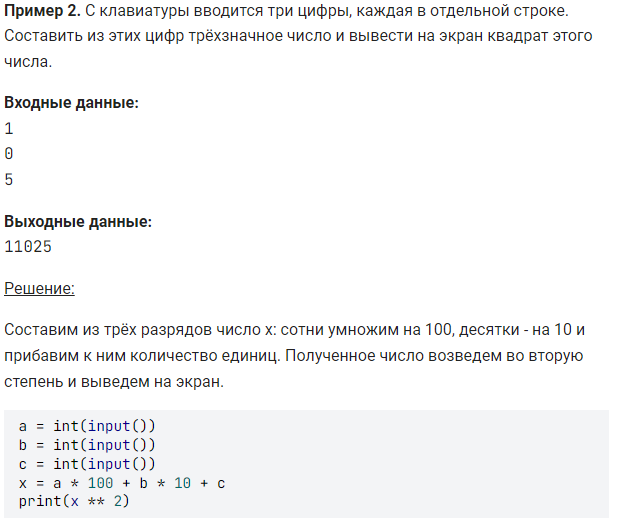
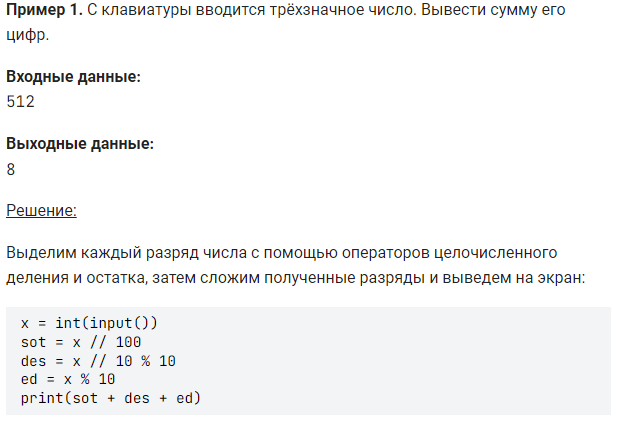
Арифметические операции имеют следующий приоритет выполнения:

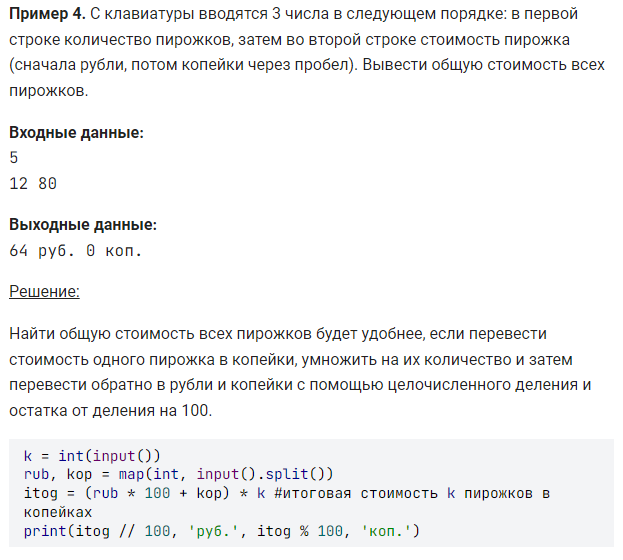
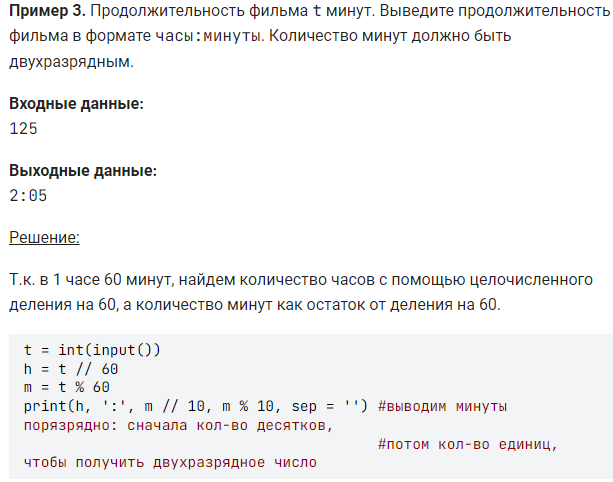
1. Скобки
2. Степень
3. Унарный минус
4. Умножение, деление
5. Сложение, вычитание

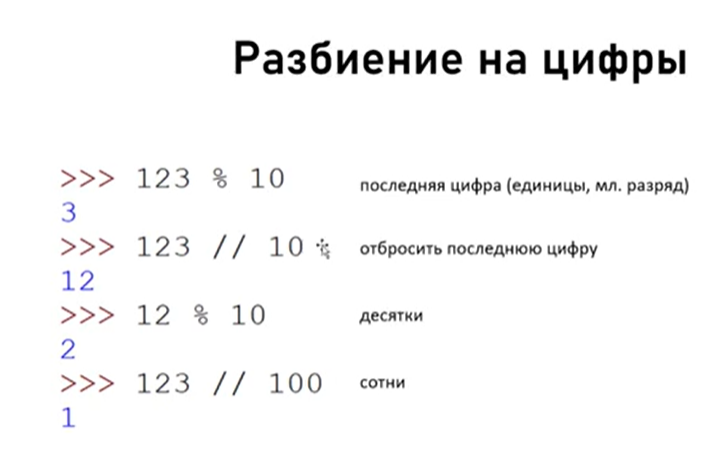
Несмотря на то, что арифметические операторы в основном предназначены для работы с числами, некоторые из них можно использовать и со строками. Оператор сложения выполняет конкатенацию строк, то есть соединяет их в одну строчку. Порядок сложения строк имеет значение, если записать слагаемые наоборот, то и результат получится другой. Также строки можно умножать на целые числа. При умножении на положительное число происходит дублирование строки указанное количество раз. При умножении на 0 или на отрицательное число, всегда получаем пустую строку. 



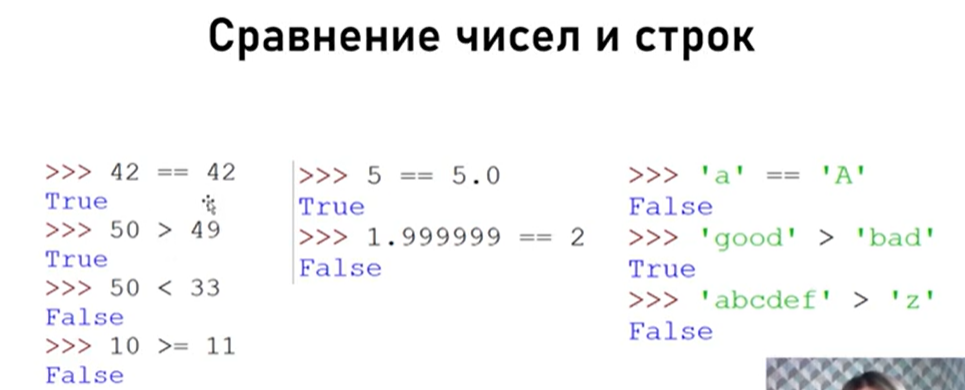
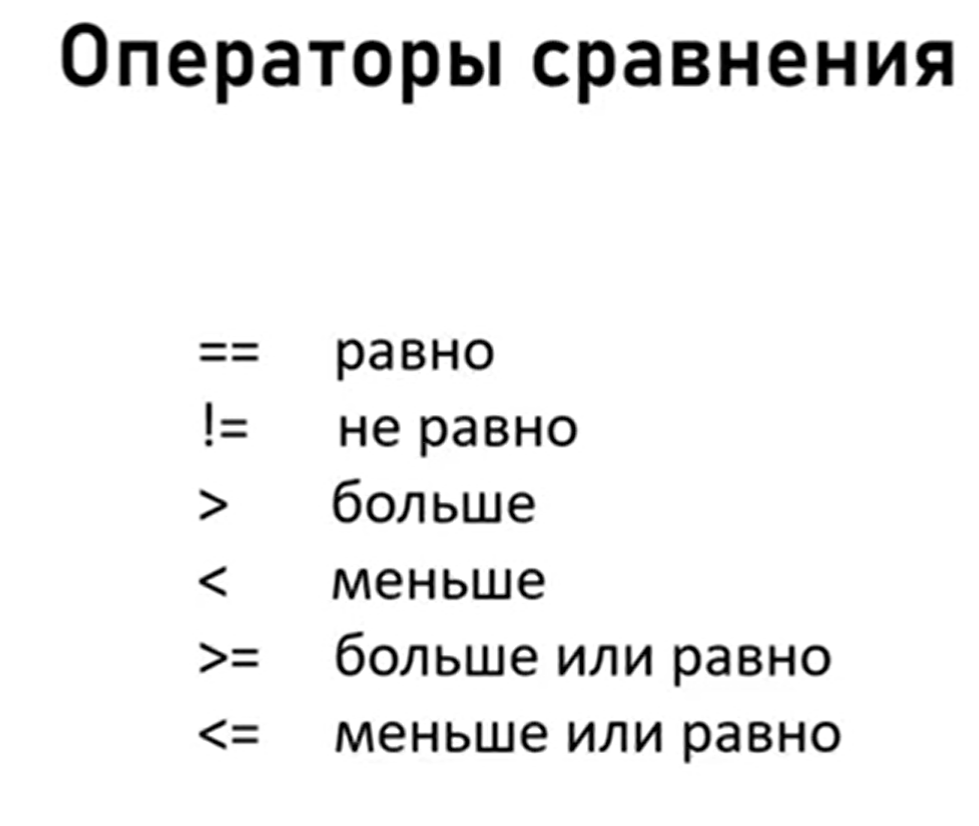
Одним из основных применений целочисленного деления в алгоритмах можно назвать выделение разрядов числа. Например, для любого десятичного числа справедливо утверждение, что остаток от деления на 10 – это его младший разряд, а частное – это все цифры без младшего разряда. Последовательное применение операций деления и деления с остатком позволит выделить все разряды из числа.

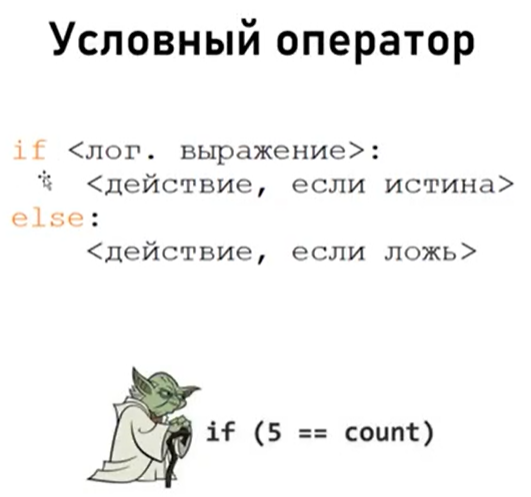


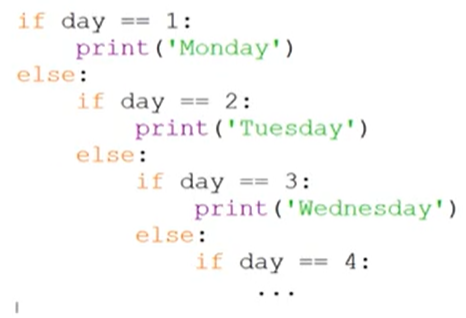
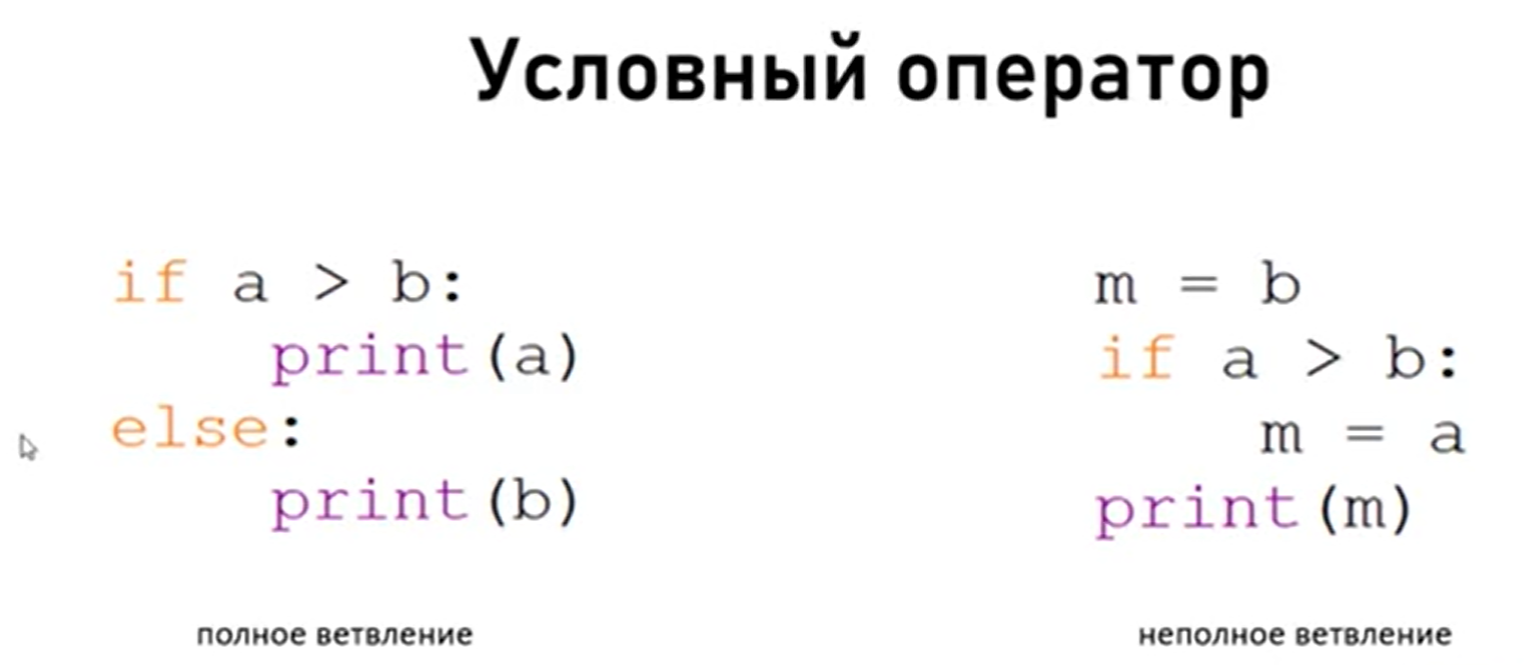


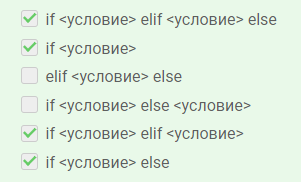
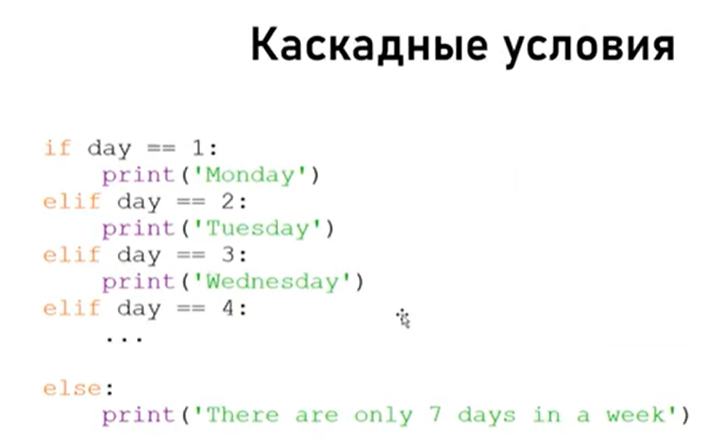


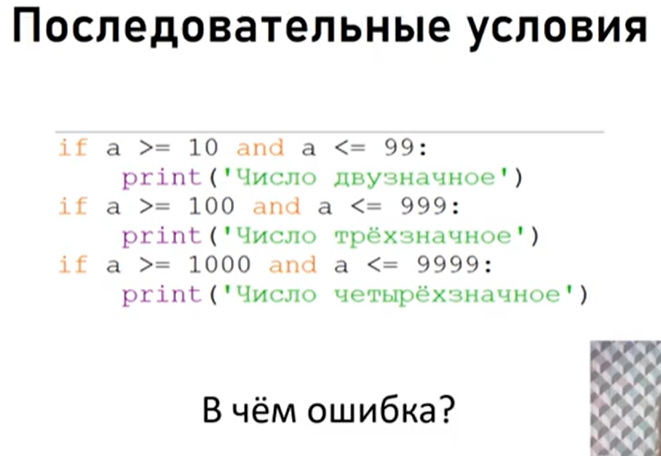
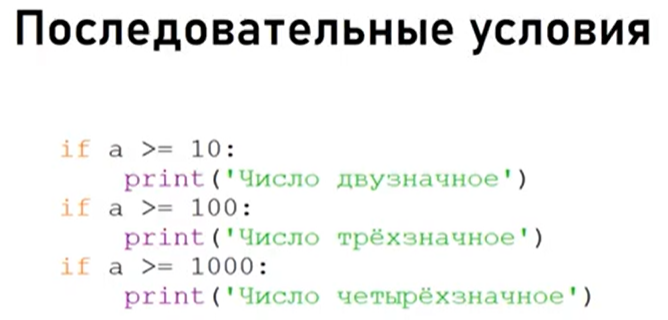
Условия

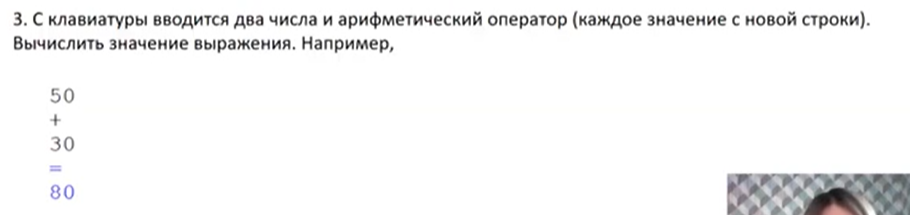
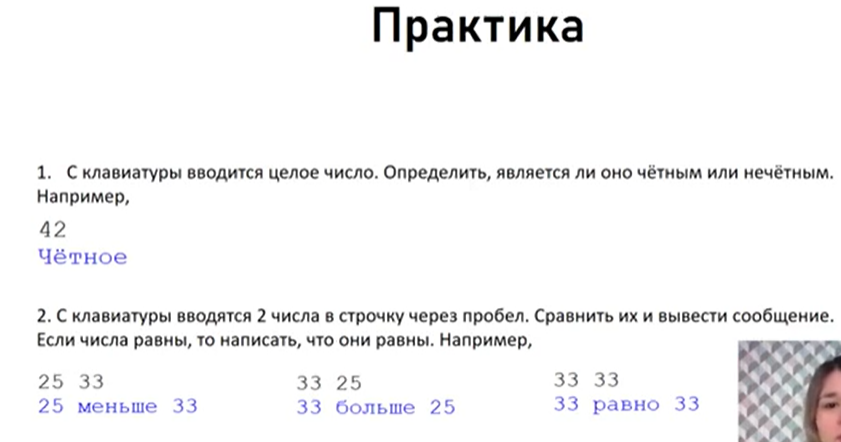




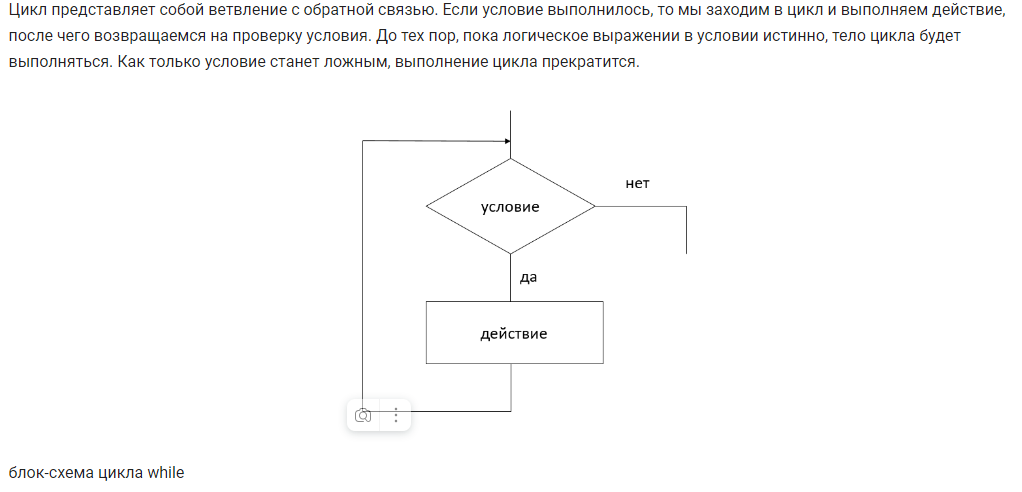
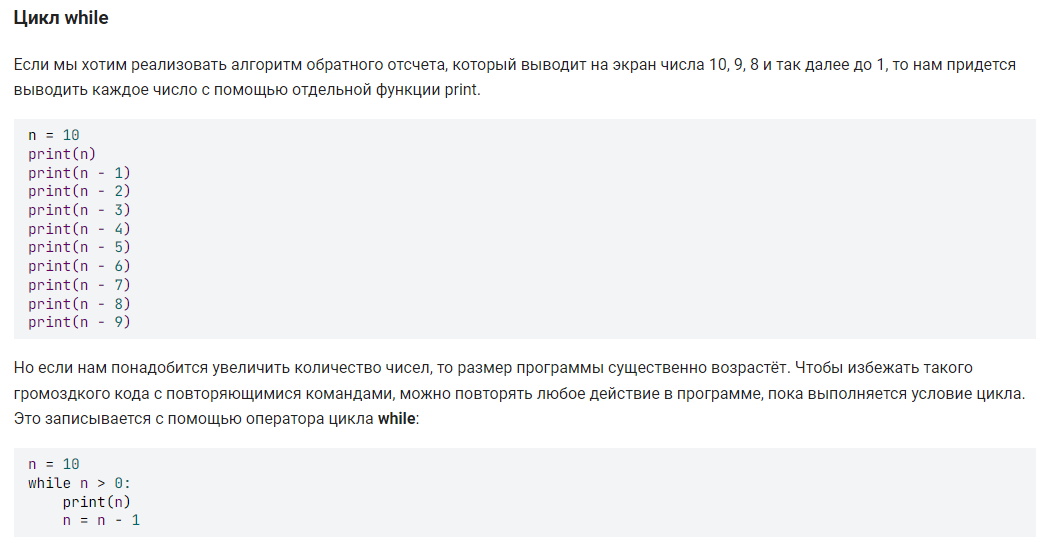


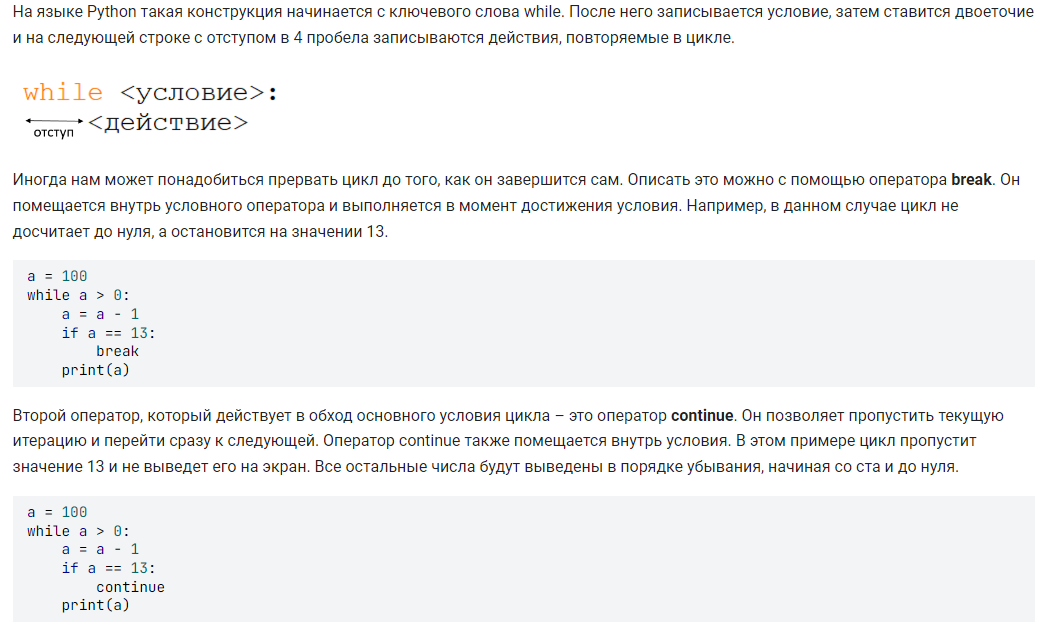


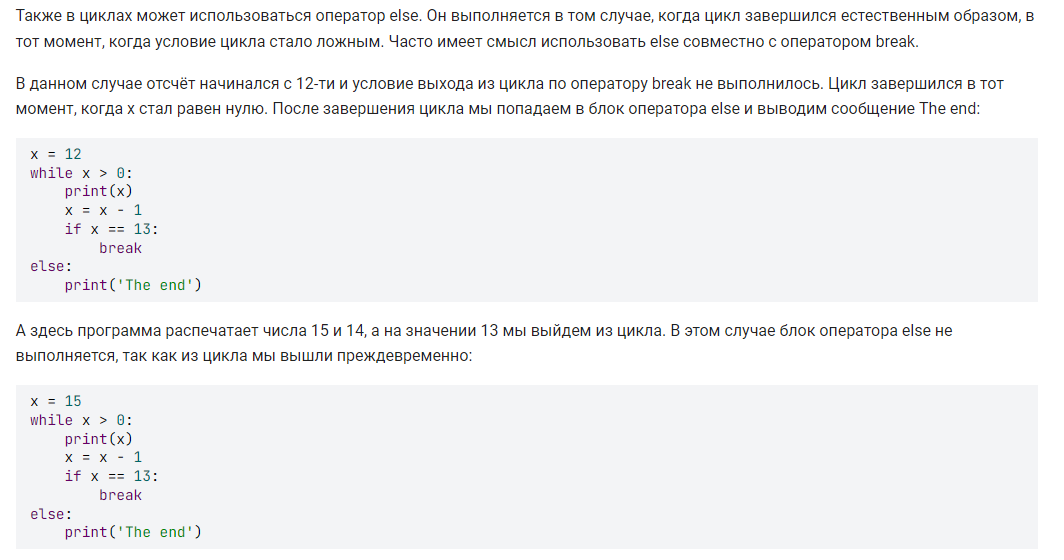


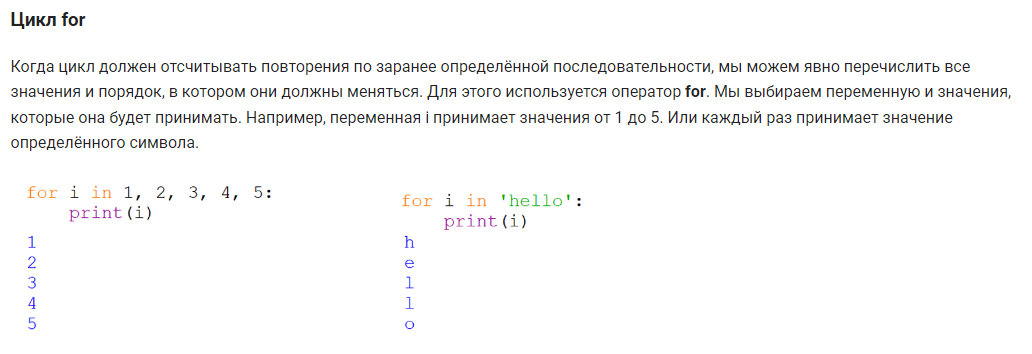


Циклы



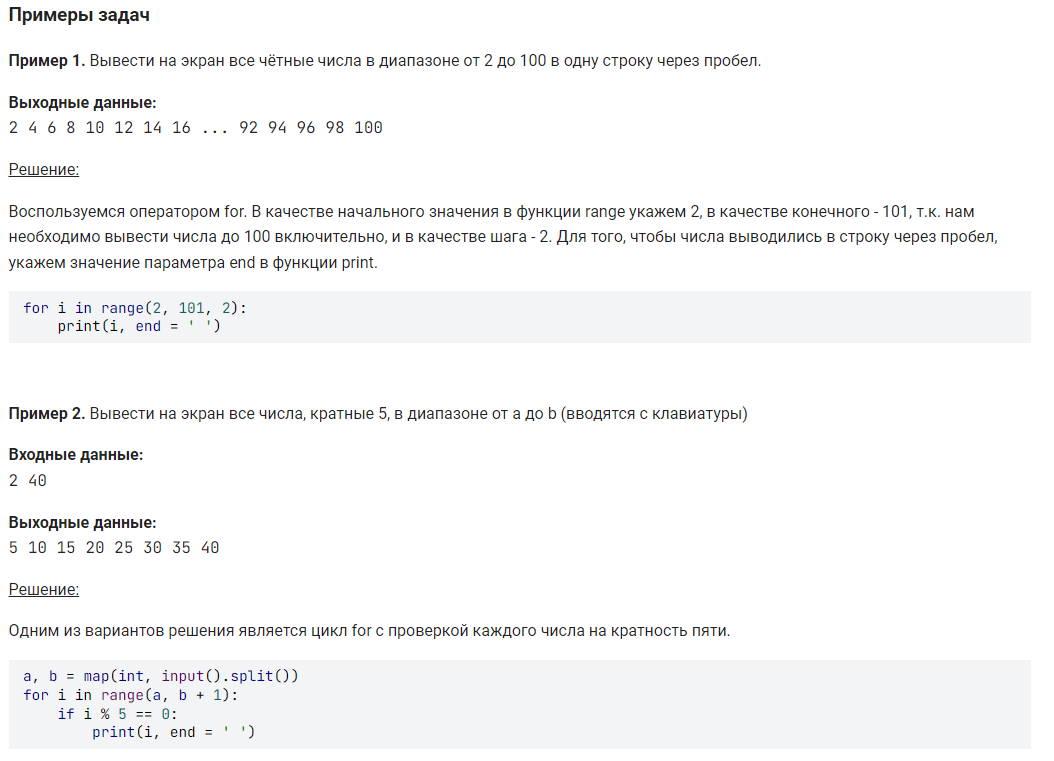


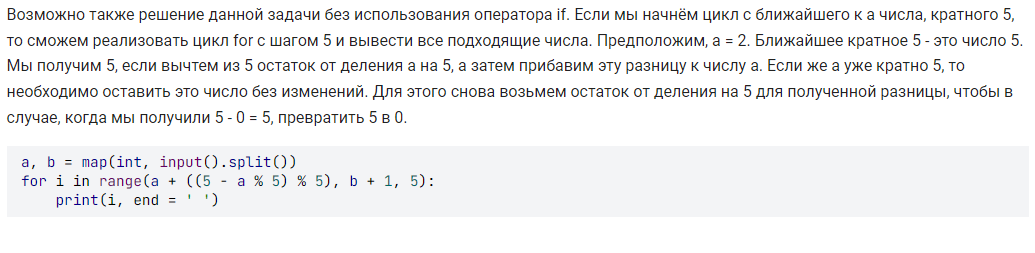






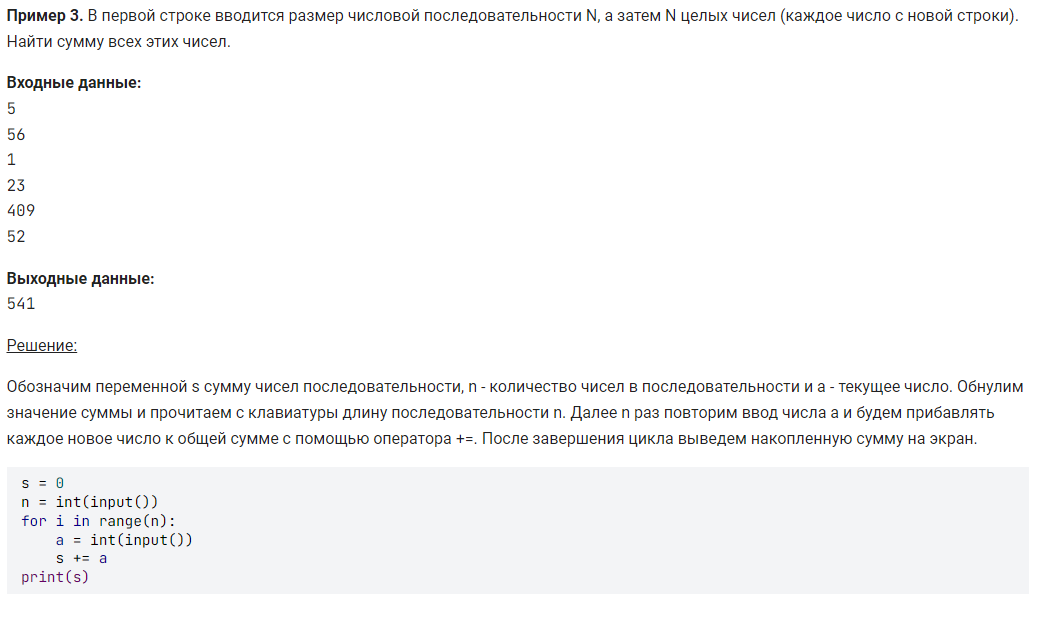
Какой параметр функции range является обязательным? Stop

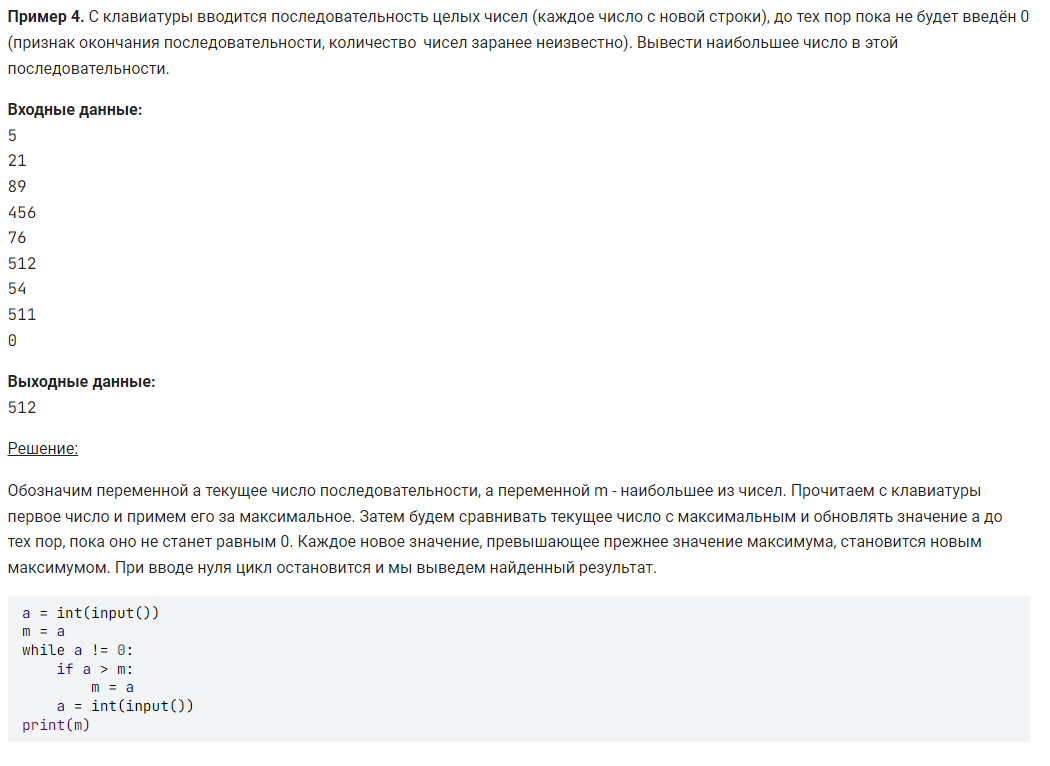




Какие виды циклов существуют в Python?

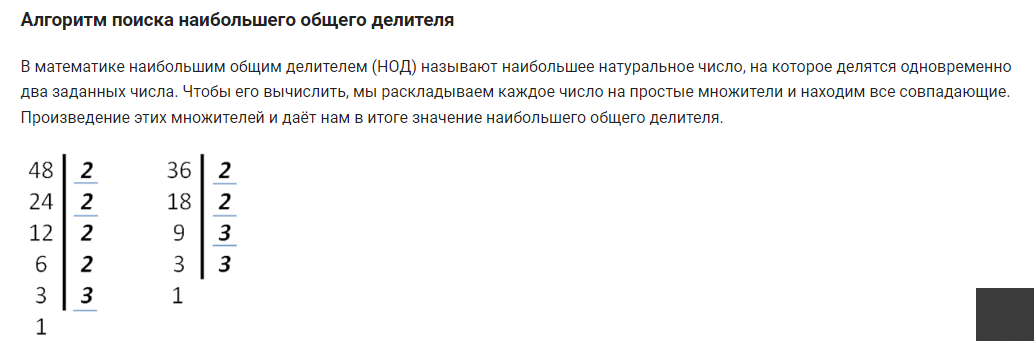
С параметром и с предусловием

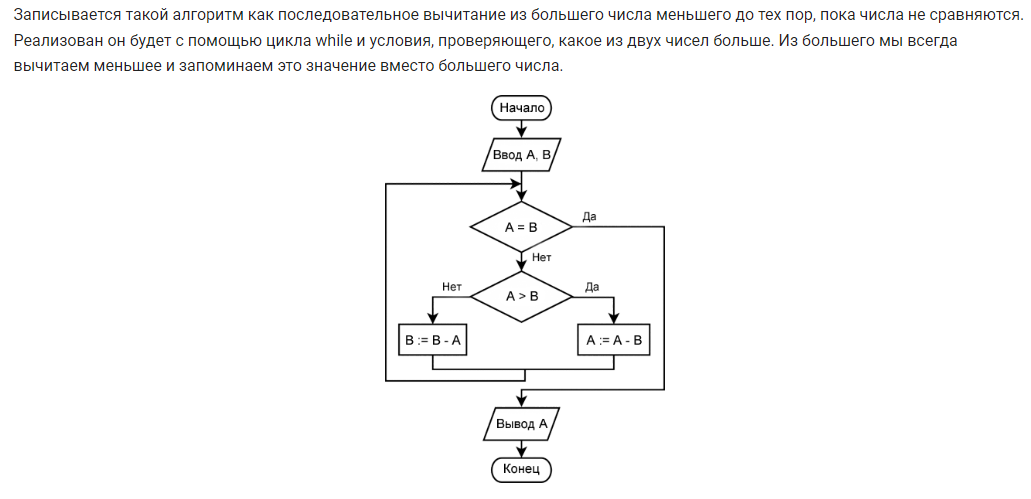


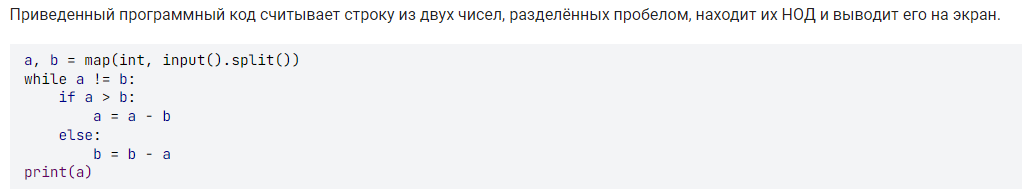


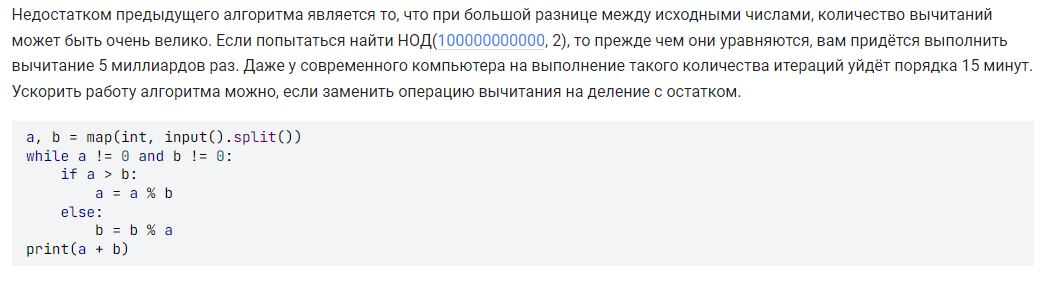
Алгоритмы

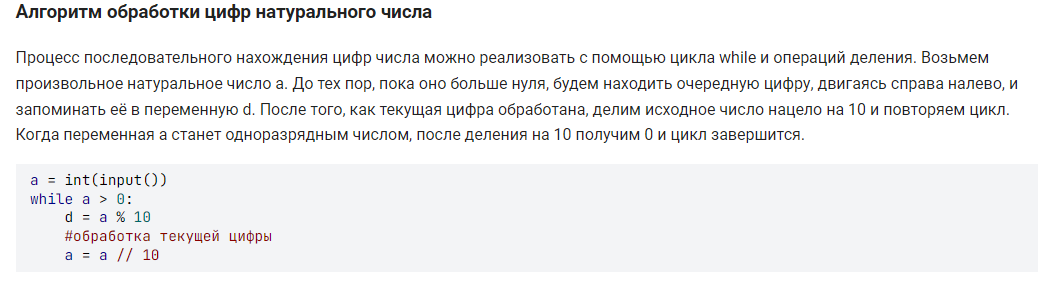
Алгоритмом называют некоторую последовательность действий, которая преобразует входные данные в результат. В большинстве программ мы сначала вводим данные, затем происходят определенные преобразования, после которых мы получаем нужный нам результат и выводим его.

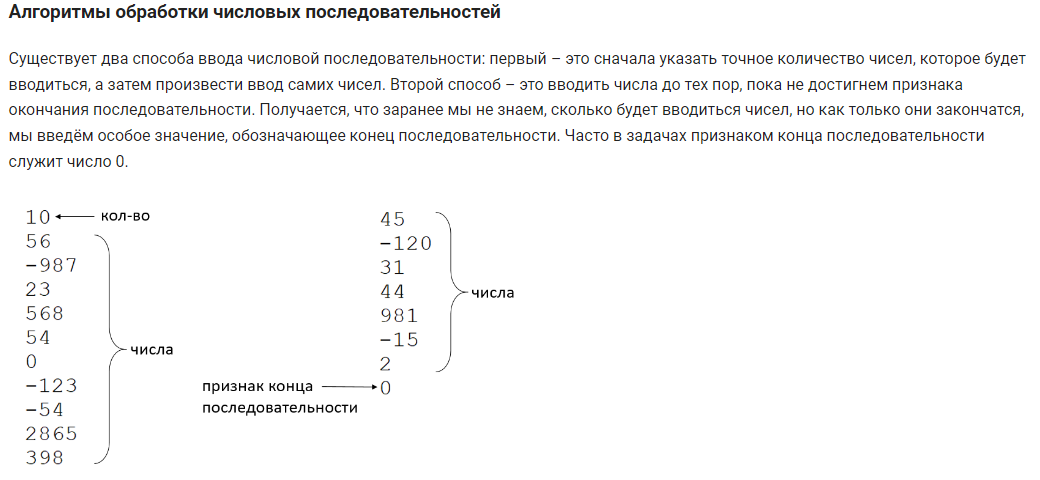


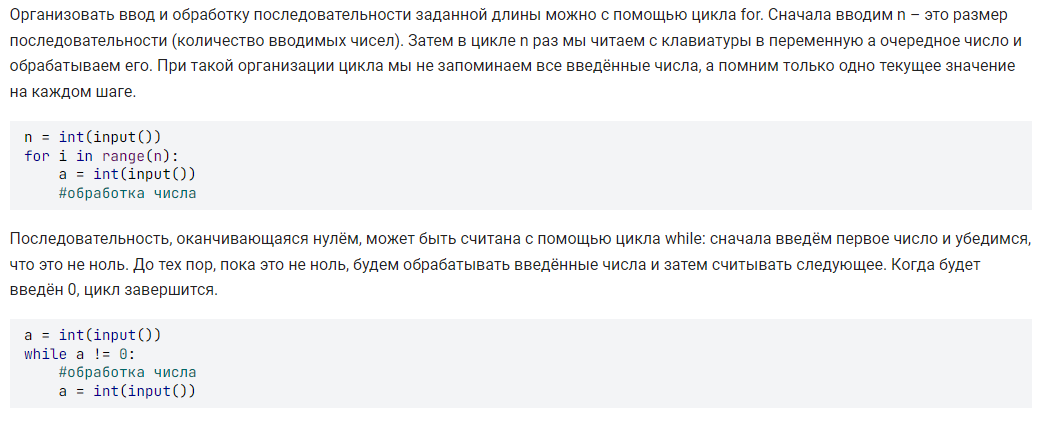


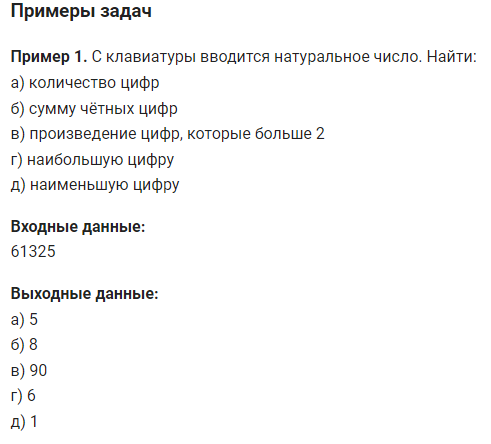
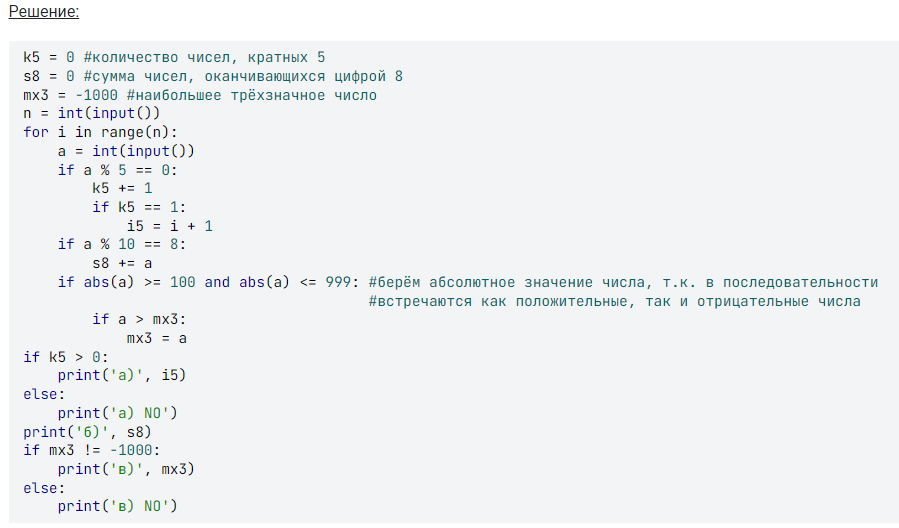
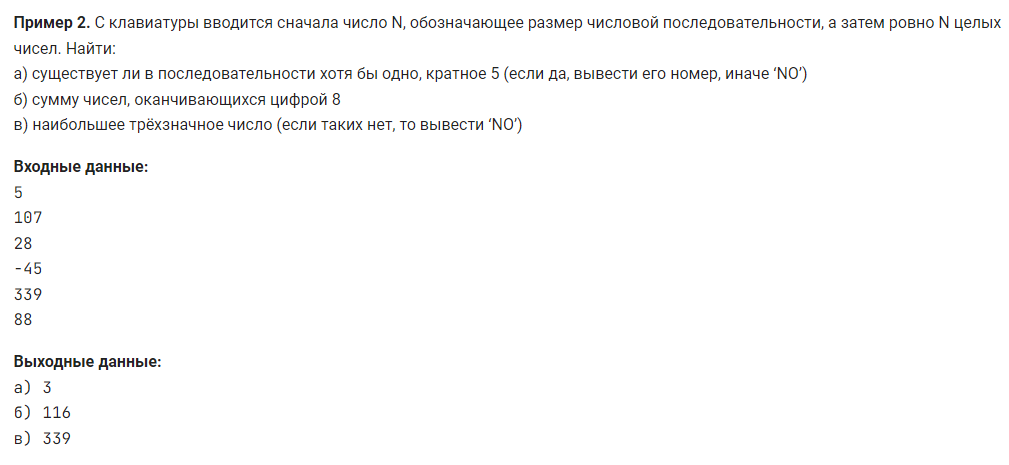
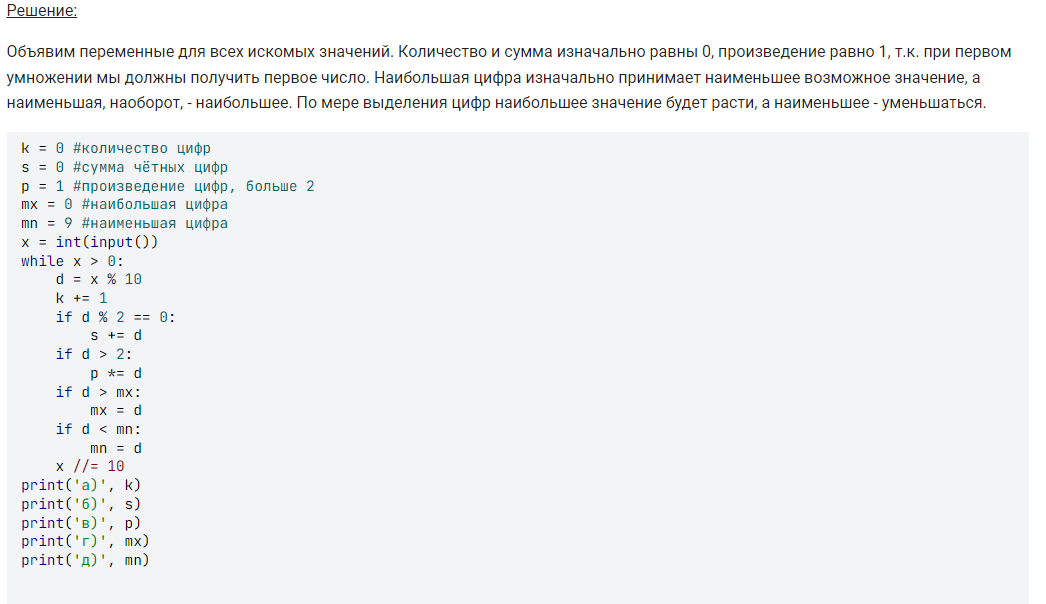


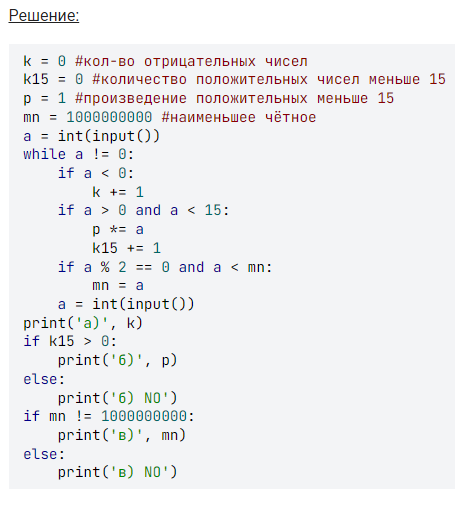
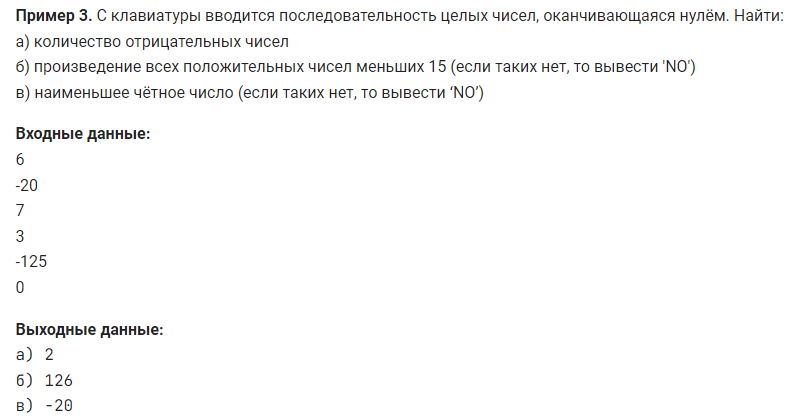




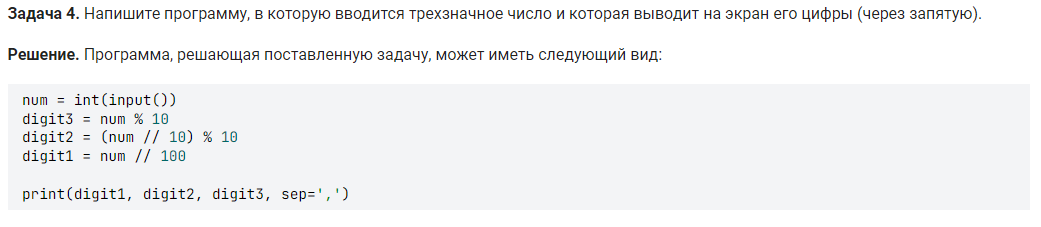




1







**Тема: Условный оператор**

Программы должны уметь выполнять разные действия в зависимости от введенных данных. Для принятия решения программа проверяет, истинно или ложно определенное условие.

В Python существует несколько способов проверки, и в каждом случае возможны два исхода: истина (True) или ложь (False).

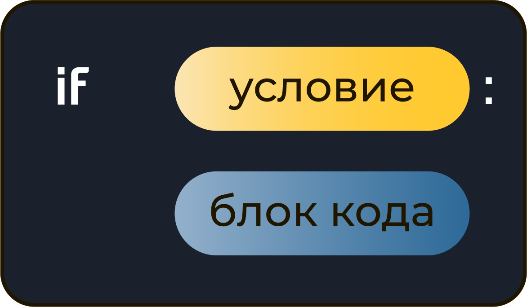
Проверка условий и принятие решений по результатам этой проверки называется **ветвлением**(branching). Программа таким способом выбирает, по какой из возможных ветвей ей двигаться дальше.

В Python проверка условия осуществляется при помощи ключевого слова if.

Двоеточие (:) в конце строки с инструкцией if сообщает интерпретатору Python, что дальше находится **блок команд**. В блок команд входят все строки с отступом под строкой с инструкцией if, вплоть до следующей строки без отступа.

Если условие истинно, выполняется весь расположенный ниже блок. В предыдущем примере блок инструкций составляет третья и четвертая строки программы.

**Блоком кода** называют объединенные друг с другом строки. Они всегда связаны с определенной частью программы (например, с инструкцией if). В Python блоки кода формируются при помощи **отступов**.



**Отступы**

В некоторых языках программирования отступы — дело личного вкуса, и можно вообще обходиться без них. Однако в Python они – неотъемлемая часть кода. Именно отступ сообщает интерпретатору Python, где начинается и где заканчивается блок кода.

**Отступ**— небольшое смещение строки кода вправо. В начале такой строки находятся пробелы, и поэтому она на несколько символов отстоит от левого края.

Некоторым инструкциям в Python (например, инструкции if) именно блок кода сообщает, какие действия следует предпринять. После if блок кода информирует интерпретатор Python, как действовать, если условие истинно, и как — если оно ложно.

По соглашению PEP 8, для отступа блоков кода используются **4 пробела**. Если в среде VS Code или Wing IDE нажать на клавишу Enter после if, она автоматически выставит **4 пробела**.

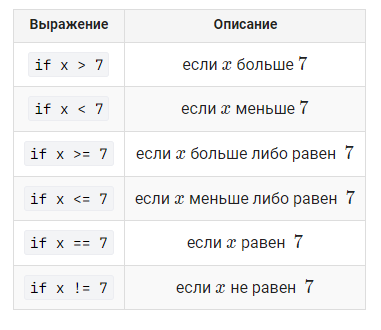
**Операторы сравнения**

Можно заметить, что в проверке условия мы использовали двойное равенство (==), вместо ожидаемого одиночного (=). Не стоит путать **оператор присваивания** (=) с **условным оператором** (==).

Оператор присваивания (=) присваивает переменным значения

Путаница с операторами == и = является одной из самых распространенных ошибок в программировании. Эти символы используются не только в Python, и каждый день множество программистов используют их неправильно.

В Python существует 6 основных операторов сравнения.



**Транзитивность**

Операция равенства является [транзитивной](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C). Это означает, что если a == b и b == c, то из этого следует, что a == c. Именно поэтому предыдущий код, проверяющий равенство трех переменных, работает, как полагается.

Из курса математики вам могут быть знакомы другие примеры транзитивных операций:

* **Отношение порядка:** если 𝑎>𝑏*a*>*b* и 𝑏>𝑐*b*>*c*, то 𝑎>𝑐*a*>*c*;
* **Параллельность прямых:** если 𝑎∥𝑏*a*∥*b* и 𝑏 ∥𝑐*b* ∥*c*, то 𝑎∥𝑐*a*∥*c*;
* **Делимость:** если 𝑎*a* делится на 𝑏*b* и 𝑏*b* делится на 𝑐*c*, то 𝑎*a* делится на 𝑐*c*.

Наглядно транзитивность отношения порядка можно понять на таком примере: если сосед слева старше вас (𝑎>𝑏)(*a*>*b*), а вы старше соседа справа (𝑏>𝑐)(*b*>*c*), то сосед слева точно старше соседа справа (𝑎>𝑐)(*a*>*c*).

Операция неравенства (!=), в отличие от операции равенства (==), является [нетранзитивной](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D1%82%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C" \t "_blank). То есть из того, что a != b и b != c вовсе не следует, что a != c. Действительно, если вас зовут не так, как соседа слева и не так, как соседа справа, то нет гарантии, что у обоих соседей не окажутся одинаковые имена.

Таким образом, следующий код вовсе **не проверяет** тот факт, что все три переменные различны:

if a != b != c:

print('числа не равны')

else:

print('числа равны')

