**Процессы. Ветвление и циклы.**

1. **Понятие линейного алгоритма. Разветвляющегося процесса, циклического процесса.**
2. **Арифметические операции**
3. **Коллекции данных**

Алгоритм - это пошаговый метод решения задачи. Он используется для обработки данных, осуществления расчетов и других связанных с ним компьютерных и математических операций.

Существует несколько типов алгоритмов: линейный, разветвляющийся (алгоритм ветвления), циклический.

**Линейный алгоритм** - это тип алгоритма, в котором последовательность действий не изменяется во время его выполнения, т.е. действия выполняются строго последовательно и один раз.

Например: Дано X. Вычислить Z = Y1 / 2 и Z1 = 1 / Z, Y = X2 + 5.

*Введите X*

*Рассчитайте Y = X2 + 5*

*Рассчитайте Z = Y1 / 2*

*Рассчитайте Z1 = 1 / Z*

*Распечатайте Z1,*

*Конец.*

**Алгоритм ветвления** - это алгоритм, содержащий хотя бы одно условие. Алгоритм ветвления - это последовательность шагов, которая изменяется в зависимости от некоторых условий. Выбирается один из двух / нескольких возможных вариантов.

Например, создайте алгоритм для вычисления функции:

(X + 1, Y>0,)

Z =

(X + Y, Y<0.)

Словесное описание разветвляющегося алгоритма выглядит следующим образом:

*Введите X*

*Если Y>0, то Z = X + 1*

*Если Y<0*

*Распечатайте Z = X + Y*

*Конец.*

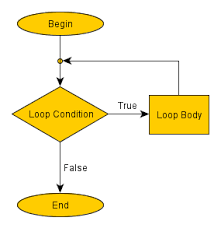
**Циклический алгоритм**

Для обозначения многократных повторяющихся действий используются специальные циклические структуры. Такая структура содержит условие, необходимое для определения количества повторений для определенной последовательности действий.

В основном блоке цикла - теле цикла - выполняются необходимые вычисления. Вспомогательные блоки цикла организуют циклический процесс: устанавливают начальное значение и новые значения данных, проверяют конечное состояние циклического процесса.

Циклический алгоритм позволяет компактно описать большое количество идентичных вычислений по различным данным с целью получения желаемого результата. Существуют циклические структуры с предварительным условием (рис. 1) - циклы с предусловием и циклические структуры с последующим условием (рис. 2) - циклы с постусловием.

Рис. 1. Цикл с предусловием. Тело цикла не может запускаться один раз



ложь

правда

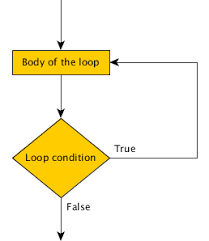
Тело цикла

Условие цикла

Конец

Начало

Рис. 2. Цикл с постусловием. Тело цикла будет выполнено хотя бы один раз.



Условие цикла

Тело цикла

ложь

правда

Чтобы удостовериться в правильности алгоритма, необходимо протестировать его, не дожидаясь исполнения окончательного алгоритма. В этом случае для регистрации выполнения алгоритма указываются шаги алгоритма, аргументы, промежуточные значения, результаты на данном шаге и проверяемые условия.

Пошаговое выполнение дает возможность увидеть, сколько шагов было выполнено до данного момента, и за сколько шагов будет выполнен весь алгоритм. Нумерация шагов алгоритма дает возможность легче ориентироваться в описании алгоритма и в процессе его выполнения. Более того, в любой момент ясно, какой выполняется элемент (команда) алгоритма. Это помогает устранить ошибки.

**Арифметические операции**

|  |  |  |
| --- | --- | --- |
| **Математический знак** | **Описание** | **Пример** |
| + Сложение | Добавляет значения по обе стороны от оператора. | a + b = 30 |
| - Вычитание | Вычитает правый операнд из левого операнда. | a - b = -10 |
| \* Умножение | Умножает значения по обе стороны от знака | a \* b = 200 |
| / Деление | Делит левый операнд на правый операнд | b / a = 2 |
| % Модуль | Делит левый операнд на правый и возвращает остаток | b % a = 0 |
| \*\* Степень | Выполняет экспоненциальный (показатель степени) расчет по знакам | a\*\*b =10 в степени 20 |
| *//* | Полное деление - деление операндов, где результатом является частное, в котором удаляются цифры после десятичной точки. Но если один из операндов отрицателен, результат получается отрицательным, т.е. округляется от нуля (в сторону отрицательной бесконечности) - | 9//2 = 4 и 9.0//2.0 = **4**.**0**, -11//3 = -4. -11.003 =-**4.0** |

Правило с наивысшим приоритетом по отношению к правилу с наименьшим приоритетом:

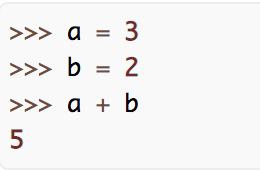
* Круглые скобки всегда имеют приоритет
* Экспоненцирование (возведение в степень)
* Умножение, деление и остаток
* Сложение и вычитание
* Слева направо

Сложение

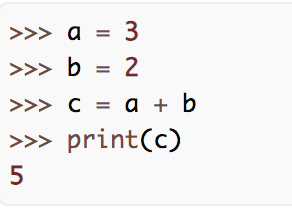
Вы можете сложить непосредственно цифры



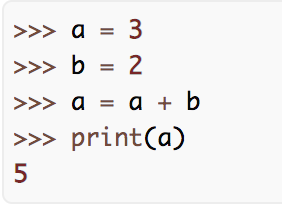
либо переменные, но переменные должны быть сначала инициализированы



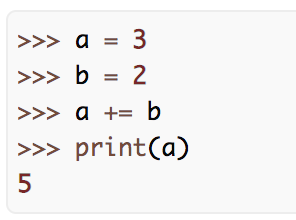
Результат операции сложения может быть присвоен другой переменной.



Вы можете использовать полную или сокращенную версию, полная версия

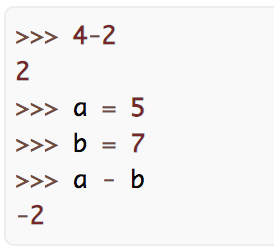


сокращенная версия

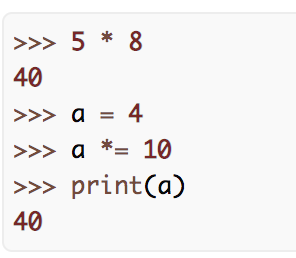


Все вышеперечисленные варианты, применимые для операции сложения, могут применяться ко всем следующим операциям.

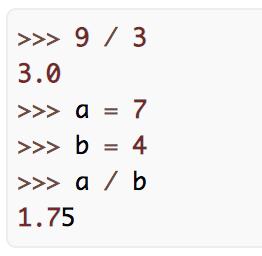
Вычитание



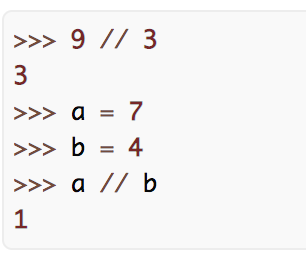
Умножение



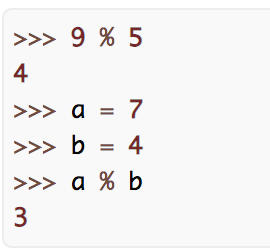
Деление



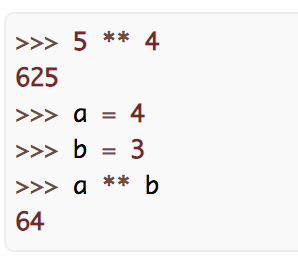
Получение целой части от деления



Получение остатка от деления.



Возведение в степень

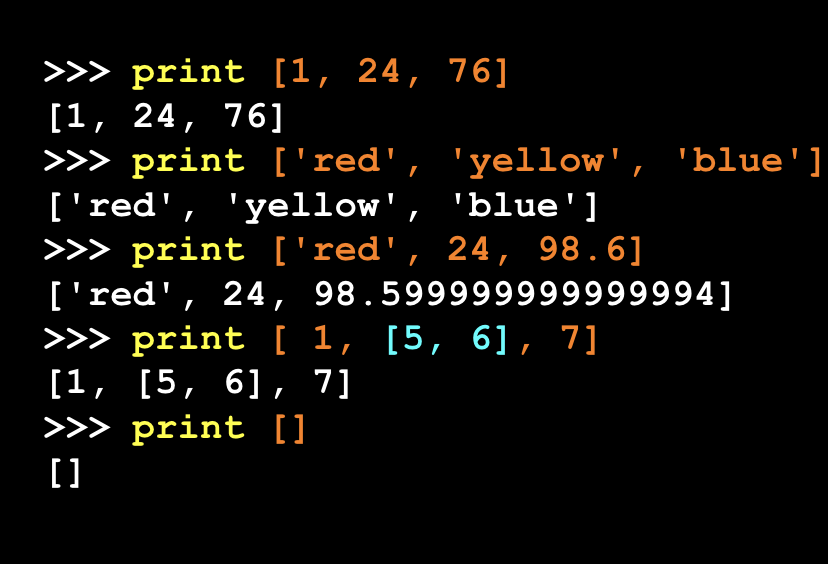


**Коллекции**

Коллекция позволяет нам поместить много значений в одну «переменную».

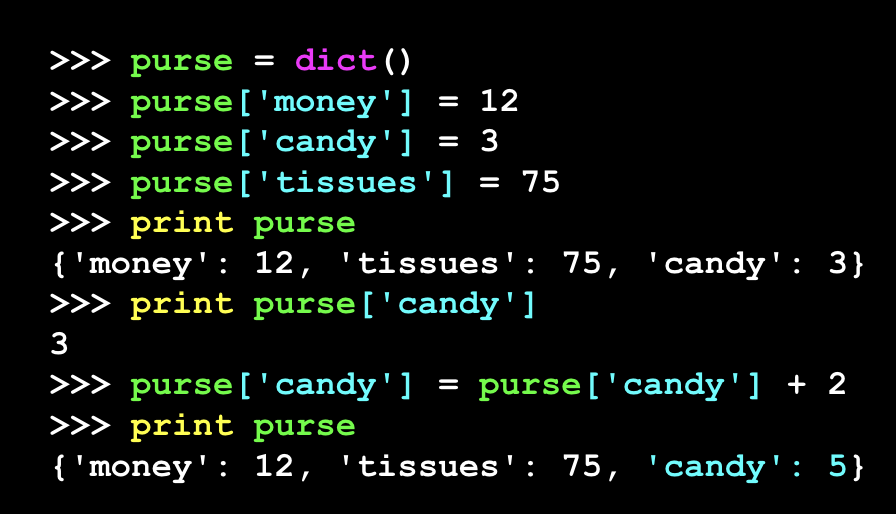
Коллекция хороша тем, что мы можем хранить все множество значений в одном удобном пакете.

* Константы списка заключаются в квадратные скобки, а элементы в списке разделяются запятыми.
* Элементом списка может быть любой объект Python - даже другой список
* Список может быть пустым

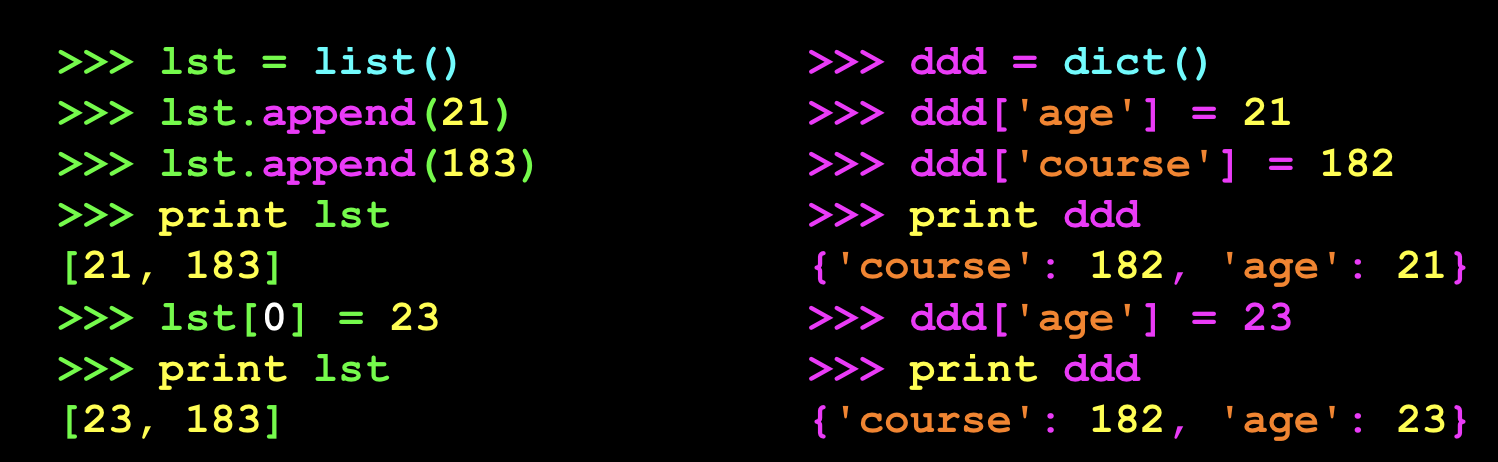


Словари

* Индексация элементов списков производится с учетом их позиции в списке
* В словарях, как и в дамских сумочках, порядка нет
* Поэтому элементы, которые мы помещаем в словарь, индексируются с помощью «тега поиска»

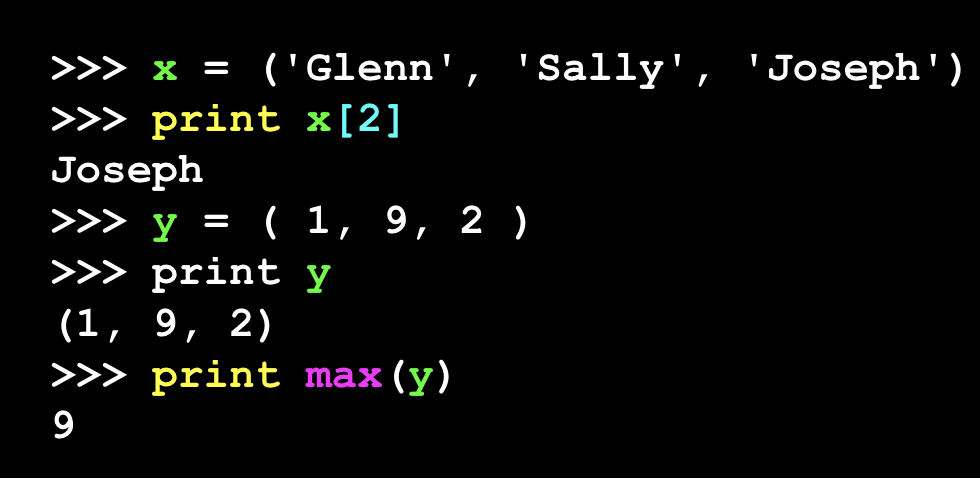


Словари похожи на списки, за исключением того, что для поиска значений они используют ключи вместо цифр



Кортежи

Кортежи - это другой вид последовательности, которая функционирует во многом как список - они содержат элементы, которые индексируются, начиная с ( )



В отличие от списка, после создания кортежа вы не можете изменять его содержимое - подобно строке

