Tuples are immutable A tuple1 is a sequence of values much like a list. The values stored in a tuple can be any type, and they are indexed by integers. The important difference is that tuples are immutable. Tuples are also comparable and hashable so we can sort lists of them and use tuples as key values in Python dictionaries.

The comparison operators work with tuples and other sequences. Python starts by comparing the first element from each sequence. If they are equal, it goes on to the next element, and so on, until it finds elements that differ. Subsequent elements are not considered (even if they are really big)

The sort function works the same way. It sorts primarily by first element, but in the case of a tie, it sorts by second element, and so on. This feature lends itself to a pattern called DSU for Decorate a sequence by building a list of tuples with one or more sort keys preceding the elements from the sequence, Sort the list of tuples using the Python built-in sort, and Undecorate by extracting the sorted elements of the sequence.

The first loop builds a list of tuples, where each tuple is a word preceded by its length. sort compares the first element, length, first, and only considers the second element to break ties. The keyword argument reverse=True tells sort to go in decreasing order. The second loop traverses the list of tuples and builds a list of words in descending order of length. The four-character words are sorted in reverse alphabetical order, so “what” appears before “soft” in the following list.

One of the unique syntactic features of the Python language is the ability to have a tuple on the left side of an assignment statement. This allows you to assign more than one variable at a time when the left side is a sequence. In this example we have a two-element list (which is a sequence) and assign the first and second elements of the sequence to the variables x and y in a single statement.

Dictionaries have a method called items that returns a list of tuples, where each tuple is a key-value pair: >>> d = {'a':10, 'b':1, 'c':22} >>> t = list(d.items()) >>> print(t) [('b', 1), ('a', 10), ('c', 22)] As you should expect from a dictionary, the items are in no particular order. However, since the list of tuples is a list, and tuples are comparable, we can now sort the list of tuples. Converting a dictionary to a list of tuples is a way for us to output the contents of a dictionary sorted by key: >>> d = {'a':10, 'b':1, 'c':22} >>> t = list(d.items()) >>> t [('b', 1), ('a', 10), ('c', 22)] >>> t.sort() >>> t [('a', 10), ('b', 1), ('c', 22)] The new list is sorted in ascending alphabetical order by the key value.

comparable A type where one value can be checked to see if it is greater than, less than, or equal to another value of the same type. Types which are comparable can be put in a list and sorted. data structure A collection of related values, often organized in lists, dictionaries, tuples, etc. DSU Abbreviation of “decorate-sort-undecorate”, a pattern that involves building a list of tuples, sorting, and extracting part of the result. gather The operation of assembling a variable-length argument tuple. hashable A type that has a hash function. Immutable types like integers, floats, and strings are hashable; mutable types like lists and dictionaries are not. scatter The operation of treating a sequence as a list of arguments. shape (of a data structure) A summary of the type, size, and composition of a data structure. singleton A list (or other sequence) with a single element. tuple An immutable sequence of elements. tuple assignment An assignment with a sequence on the right side and a tuple of variables on the left. The right side is evaluated and then its elements are assigned to the variables on the left.