Python Study (Modules: Collections):

this module implements specialized container data types providing alternatives to python's general purpose built-in containers.

---

collections.namedtuple:

factory function for creating tuple subclasses with named fields.

namedtuple parameters are as follows:

• typename = the name of this type of tuple.
• filed_names = a list of strings to create different named attributes of the tuple. can also be a single string, with names separated by spaces and-or commas.
• defaults = the default value for items in this type of tuple. can either be none or an iterable. if the iterable has less items than there are named attributes, the right-most attributes will receive the values, while the remaining ones will be required. by default, this is none.

---

collections.deque:

deque is an ordered mutable collection, which allows duplicate elements. deque is preferred over list in the cases where we need quicker append and pop operations from both the ends of the container, as deque provides an O(1) time complexity for append and pop operations as compared to list which provides O(n) time complexity.

deque parameters are as follows:

• iterable = an optional iterable with default values to populate the deque on creation.
• maxlen = an optional iterable-dependent integer representing the max length of the deque. length is not limited by default.

deque attributes are as follows:

• maxlen() = a read-only attribute which represents the maximum length of the deque. this can only be set during construction.

deque functions are as follows:

• append(x), where x is an element which is added to the right side of the deque.
• appendleft(x), where x is an element which is added to the left side of the deque.
• clear(), which removes all elements from the deque.
• copy(), which returns a shallow copy of the deque.
• count(x), where x is a value to search for, which returns the number of elements equivalent to x.
• extend(iterable), where iterable is an iterable which has all of its elements appended onto the right side of the deque.

• extendleft(iterable), where iterable is an iterable which has all of its elements appended onto the left side of the deque. note that the last element in iterable will be the left-most once added.
• index(x [, start[, stop]]), where x is a value to search for, where start is an optional starting index for the search, where stop is an optional start-dependent stopping index for the search, which returns the position of x.
• insert(i, x), where i is the index of insertion, where x is the element to insert.
• pop(), which removes and returns an element from the right side of the deque.
• popleft(), which removes and returns an element from the left side of the deque.
• remove(value), where value is a value to remove from the deque. this will remove the first equivalent element found.
• reverse(), which reverses the elements of the deque in place.
• rotate(n), where n is the number of steps to rotate the deque, which shifts the elements of the deque to the right, based on n. if n is negative, the elements will shift to the left instead.
