Combos
Combos allow you to assign special functionality to combinations of key presses.
The two default behaviors are:
* Chords: match keys in random order, all pressed within 50ms.
* Sequences: match keys in order, pressed within 1s of one another.

You can define combos to listen to any valid KMK key, even internal or
functional keys, like HoldTap. When using internal KMK keys, be aware that the
order of modules matters.

The result of a combo is another key being pressed/released; if the desired
action isn't covered by KMK keys: create your own with `make_key` and attach
corresponding handlers.

Combos may overlap, i.e. share match keys amongst each other.

Keycodes
New Keycode	Description
`KC.LEADER`	a dummy / convenience key for leader key sequences

Custom Combo Behavior
Optional arguments that customize individual combos:
* `fast_reset`: If True, allows tapping every key (default for sequences);
   if False, allows holding at least one key and tapping the rest to repeatedly
   activate the combo (default for chords).
* `per_key_timeout`: If True, reset timeout on every key press (default for
  sequences).
* `timeout`: Set the time window within which the match has to happen in ms.
* `match_coord`: If True, matches key position in the matrix.

Example Code
```python
from kmk.keys import KC, make_key
from kmk.modules.combos import Combos, Chord, Sequence
combos = Combos()
keyboard.modules.append(combos)

make_key(
    names=('MYKEY',),
    on_press=lambda *args: print('I pressed MYKEY'),
)

combos.combos = [
    Chord((KC.A, KC.B), KC.LSFT),
    Chord((KC.A, KC.B, KC.C), KC.LALT),
    Chord((0, 1), KC.ESC, match_coord=True),
    Chord((8, 9, 10), KC.MO(4), match_coord=True),
    Sequence((KC.LEADER, KC.A, KC.B), KC.C),
    Sequence((KC.E, KC.F), KC.MYKEY, timeout=500, per_key_timeout=False, fast_reset=False)
]
```