A. Making lists:
   \[ L = [ \ ] \] (the *empty* list, which is the list with no elements)
   \[ L = ["abc", "de", "fghij", 1, [2, 3] ] \]: this list has 5 elements: three strings, one integer and one list.
   \[ L = L1 + L2 \], where \( L1 \) and \( L2 \) are lists. This concatenates \( L1 \) and \( L2 \) into a new list \( L \).
   \[ L = L1 * 3 \], where \( L1 \) is a list. This makes a new list \( L \), which is the concatenation of \( L1 \) 3 times, as in \( L1 + L1 + L1 \).

B. Indexing:
   \[ L[0] \]: the first element in list \( L \)
   \[ L[1] \]: the second element in list \( L \)
   \[ L[2:5] \]: a *slice* of list \( L \), which is a new list consisting of the elements at positions 2, 3, and 4 (but not 5) of \( L \).

C. Changing the contents of the list, without changing the list itself:
   \[ L[i] = a \] changes the value of the \( i \)th entry of \( L \) to \( a \)
   \[ L.append(x) \]: adds \( x \) to the end of the list \( L \)
   \[ L.extend(L1) \]: where \( L1 \) is a list. This adds all the entries of \( L1 \) onto \( L \)

   \[ L.sort() \]: *sorts*, or arranges in order, the entries of \( L \)
   \[ L.sort(compare) \]: again, this sorts the entries of \( L \), using compare as a function to compare two entries. \( compare(a, b) \)
   should return -1 if \( a < b \), 0 if \( a == b \), and 1 if \( a > b \)

   \[ L.reverse() \]: reverses the order of the entries of \( L \)
   \[ del L[i] \]: deletes the \( i \)th element of \( L \)
   \[ L[i:j] = [ ] \] deletes the index \( i \) through \( j \) slice of \( L \)

D. Other stuff
   \[ len(L) \]: the *length*, or number of entries, of \( L \)
   \[ for x in L \]: iterates a loop over all entries of \( L \)
   \[ x in L \]: returns \( True \) if \( L \) has an entry whose value is \( x \)
   \[ L.index(v) \]: returns the index of the first entry of \( L \) that equals \( v \); crashes if \( L \) does not contain \( v \)