

Lists change in place: `append()`, `sort()` and friends edit the list and return `None`, so don't assign their result. Each section shows the list it starts from.

CREATE

```
[1, 'a', 'a'] → [1, 'a', 'a']  
any mix of types; duplicates allowed
```

```
list("abc") → ['a', 'b', 'c']  
any iterable to a list
```

```
list(range(4)) → [0, 1, 2, 3]
```

```
[0] * 3 → [0, 0, 0]  
[0] * 3 for _ in range(2)  
→ [[0, 0, 0], [0, 0, 0]]  
a grid: a new row each time
```

INDEX AND SLICE

```
xs = [10, 20, 30, 40, 50]
```

```
xs[0] → 10  
first
```

```
xs[-1] → 50  
last
```

```
xs[1:4] → [20, 30, 40]  
start included, stop excluded
```

```
xs[-2:] → [40, 50]  
last two
```

```
xs[::-2] → [10, 30, 50]  
every second item
```

```
xs[::-1] → [50, 40, 30, 20, 10]  
a reversed copy
```

```
xs[3:99] → [40, 50]  
slices never raise, indexes do
```

```
len(xs) → 5
```

```
xs[0] = 99  
replace one item
```

```
xs[1:3] = [0]  
xs  
→ [99, 0, 40, 50]  
replace a slice; the lengths can differ
```

ADD

```
xs = [1, 2]
```

```
xs.append(3) one item on the end
```

```
xs.extend([4, 5])  
each item of an iterable
```

```
xs.insert(0, 0) before index 0
```

```
xs → [0, 1, 2, 3, 4, 5]  
after the three lines above
```

```
xs += [6]  
same as extend: changes xs in place
```

```
[1, 2] + [3] → [1, 2, 3]  
+ builds a new list
```

REMOVE

```
xs = [1, 2, 3, 2, 5]
```

```
xs.remove(2) by value, first match only
```

```
xs.pop() → 5  
removes and returns the last
```

```
xs.pop(0) → 1  
or the item at an index
```

```
del xs[0]  
by index; del xs[1:3] cuts a slice
```

```
xs → [2]  
after the four lines above
```

```
xs.clear() empty it
```

```
xs.remove(9) → raises ValueError  
test with in first
```

SEARCH AND COUNT

```
xs = [3, 1, 2, 1]
```

```
2 in xs → True
```

```
xs.index(1) → 1  
position of the first match
```

```
xs.index(9) → raises ValueError  
lists have no find(): check in first
```

```
xs.count(1) → 2
```

```
min(xs), max(xs), sum(xs) → (1, 3, 7)
```

```
any(x > 2 for x in xs) → True  
at least one
```

```
all(x > 0 for x in xs) → True  
every one
```

SORT AND REVERSE

```
xs = [3, 1, 2]  
pairs = [("c", 2), ("a", 1)]
```

```
sorted(xs) → [1, 2, 3]  
a new list; xs is unchanged
```

```
sorted(xs, reverse=True) → [3, 2, 1]
```

```
xs.sort() in place, returns None
```

```
xs.reverse() in place too
```

```
xs → [3, 2, 1]  
sorted, then reversed
```

```
list(reversed(xs)) → [1, 2, 3]  
reversed() gives an iterator
```

```
sorted(["bb", "a", "ccc"], key=len)  
→ ['a', 'bb', 'ccc']  
key: sort by what the function returns
```

```
sorted(pairs, key=lambda p: p[1])  
→ [('a', 1), ('c', 2)]  
sort records by one field
```

COPY

```
import copy  
a = [1, [2, 3]]
```

```
a.copy() → [1, [2, 3]]  
shallow copy; so are a[:] and list(a)
```

```
b = a.copy()  
b[1].append(4)  
a  
→ [1, [2, 3, 4]]  
a shallow copy still shares the inner lists
```

```
copy.deepcopy(a) copies nested lists too
```

LOOP

```
xs, ys = ["a", "b"], [1, 2]
```

```
list(enumerate(xs))  
→ [(0, 'a'), (1, 'b')]  
for i, x in enumerate(xs): the index too
```

```
list(enumerate(xs, 1))  
→ [(1, 'a'), (2, 'b')]  
count from 1
```

```
list(zip(xs, ys)) → [('a', 1), ('b', 2)]  
pairs; stops at the shorter list
```

```
dict(zip(xs, ys)) → {'a': 1, 'b': 2}
```

COMPREHENSIONS

```
nums = [3, -1, 4]  
grid = [[1, 2], [3, 4]]
```

```
[n * 2 for n in nums] → [6, -2, 8]
```

```
[n for n in nums if n > 0] → [3, 4]  
filter: the if goes at the end
```

```
[n if n > 0 else 0 for n in nums]  
→ [3, 0, 4]  
choose a value: if/else goes first
```

```
[x for row in grid for x in row]  
→ [1, 2, 3, 4]  
flatten: loops in nested-for order
```

```
list(map(str, nums)) → ['3', '-1', '4']  
convert every item
```

HANDY PATTERNS

```
xs = [1, 2, 3, 4, 5]
```

```
sum(xs) / len(xs) → 3.0  
the average
```

```
[xs[i:i + 2] for i in range(0, len(xs), 2)]  
→ [[1, 2], [3, 4], [5]]  
chunks of 2
```

```
list(dict.fromkeys([3, 1, 3, 2]))  
→ [3, 1, 2]  
drop duplicates, keep the order
```

```
xs[0], xs[-1] = xs[-1], xs[0]  
swap two items, no temp variable
```

WATCH OUT

```
xs = [3, 1, 2]  
a, ys = [1, 2], [1, 2]
```

```
xs = xs.sort()  
sort() returns None, so now xs is None: use sorted(xs)
```

```
b = a  
b.append(3)  
a  
→ [1, 2, 3]  
b = a is not a copy: both names see the change
```

```
grid = ([0] * 2) * 2  
grid[0][0] = 9  
grid  
→ [[9, 0], [9, 0]]  
* repeats one inner list: use a comprehension
```

```
for x in ys:  
    ys.remove(x)  
changing a list while looping over it skips items
```

```
ys.append([4, 5])  
adds the list as one item: extend() adds each
```

```
a[len(a)] → raises IndexError  
the last index is len(a) - 1
```

```
sorted([3, "a"]) → raises TypeError  
mixed types can't be ordered
```

```
def add(x, items=[]):  
    items.append(x)  
the default list is shared by every call: use None
```