リストとは

可変のシーケンス型オブジェクト。
要素として任意の型のオブジェクトを格納することができる。
リストの要素は順番を持つ。

スライス、結合が可能。

角括弧で作成する。

リストのキューイング

shift

shift はスライスを使わなくても、pop() メソッドの引数を 0 に指定することで簡単に実現できる。
r = a.pop(0)

unshift

unshift はスライスで実現できるが、挿入したい要素が 1 個の場合は insert(0, x) が使える。

リストの先頭に要素 1 を挿入。
a.insert(0, 1)
a[:0] = [1]

リストの先頭にリストを拡張挿入。
a[:0] = [1, 2]

push

リストの末尾に要素 5 を挿入。
a.append(5)

リストの末尾にリストを拡張挿入。
a.extend([5, 6])

pop

r = a.pop()

リストのコピー

Python では変数に代入をしても、オブジェクトへの参照がコピーされるだけなので、すべての変数が同じオブジェクトを指す。

```
a = [0, 1, 2, 3]

# a を破壊したいので、破壊用に変数 b にコピー
b = a
b[2:4] = [5, 6]

# print(a) shows [0, 1, 5, 6]
```

これだと、b を破壊すると a も破壊される（a と b が参照しているオブジェクトは同じだから）。
リストをコピーするには、スライスの全要素参照を使う。

```
b = a[:]
```

a[:] は、a と同じ値の要素を持つ、別のリストオブジェクトを返す。

リストのクリア

```
a[:] = []
```

リスト全てを表すスライスを左辺、空リストを右辺。

リストのソート

- list.sort()
- sortedlist = sorted(list)

key オプション

sort()、sorted() には key オプションを指定できる。key オプションに指定できるのは、元のリストからどう比較値を抜き出すかを定義した関数。

文字列のリストを大文字小文字を意識せずに比較したい：

```
str_list.sort(key=str.lower)
```

入れ子リストのソート：
特に何も指定しなければ、各要素シーケンスの 0 番目の値から順番に比較が行われる。シーケンスの任意の要素順でソートしたい場合は下記のように、比較したい順を並び替えた要素を返す関数を key 引数に指定する。
並び替える要素として、元のシーケンスの要素個分を返す必要はない。比較したい要素だけを並び替えて返せばよい。

```
t_list = [('key1', 5), ('key2', 10), ('key3', 7), ('key4', 8), ('key5', 2)]
sorted_list = sorted(t_list, key=lambda x:(x[1], x[0]))
```

key に指定した関数にリストの値が与えられ、その返り値同士が比較される。
Python 3 では比較可能でないオブジェクト同士を比較すると、例外が発生する。

reverse オプション

降順でソートする。

```
a_list.sort(reverse=True)
```

ディクショナリーとリストの相互変換

ディクショナリーをリストに展開する

リスト内包表記を使う。

```
pair_list = [x for x in d.items()]
```

「キーと値のタプル」のリストができる。
[(key0, value0),(key1, value1),(key2, value2),...] みたいな感じ。

キーと値のタプルのリストをディクショナリーにする

ディクショナリーでも内包表記が可能に。上のように展開したキーと値のタプルのリストを、元のディクショナリーに戻す。

```
d = {x[0]:x[1] for x in pair_list}
```

内包表記は set でも使えるとのこと。

キーのリストと値のリストからディクショナリーを作る

キーのリスト [key0, key1, key2, ...] と値のリスト [value0, value1, value2, ...] からディクショナリーを作る。
zip() を使う。

```
d = {x:y for x,y in zip(keys, values)}
```

キーと値がならんだリストをディクショナリーにする

```
a = [key0, value0, key1, value1, key2, value2,...]
```

これは、一度キーだけのタプル、値だけのタプルを作って、上記の内包表記で処理する。

```
keytup = a[0::2]
valuetup = a[1::2]
d = {x:y for x,y in zip(keytup, valuetup)}
```

一行版

```
d = {x:y for x,y in zip(a[0::2], a[1::2])
```