Python - P42

class int([X]) / class int(x, base=10)
return integer object constructed from number / string X
return 0 if no argument given
return x.__int__() if x define __int__()
return x.__trunc__() if x define __trunc__()
for floating point number, truncate towards 0
if x not number / base given
then x must be string / bytes / bytearray instance
representing integer literal in radix base
optionally preceded by +/- with no space in between
surrounded by whitespace
base : default 10, allowed value 0, 2-36
base-n literal consist of digit 0 to n-1
with a-z/A-Z having value 10-35
-2, -8, -16 can optionally prefixed with 0b/0B, 0o/0O, 0x/0X
base 0 : interpret exactly as code literal
actual base -2/-8/-10/-16

iter (object [, sentinel]), return iterator object
first argument interpreted differently depending on presence of second argument
without sentinel, object must be collection object
support iteration protocol, __iter__()
or support sequence protocol, __getitem__() with integer argument starting at 0
raise TypeError if not support either protocol
if sentinel given, object must be callable object
iterator created call object with no argument
for each call to __next__()
raise StopIteration if value returned equal to sentinel

next (iterator [, default]), retrieve next item from iterator
by calling __next__()
if default given, returned if iterator exhausted
otherwise raise StopIteration

Python - P43

class list ([iterable]), actually mutable sequence type rather than function
list may be constructed in several ways:
using pair of square bracket to denote empty list: []
using square bracket, separating items with comma: [1, 2, 3]
using list comprehension: eg [x for x in iterable]
using type constructor: list() / list(iterable)

sort(): sort list in place, using only LT (<) comparison
if any comparison operation fail, entire sort operation will fail
list will likely be left in partially modified state
accept two arguments only be passed by key word
sort(*, key = None, reverse = False)
key: specify function used to extract comparison key from each list element
key corresponding to each item calculated once
used for entire sorting process
reverse: if True, list element sorted as if each comparison reversed
modify sequence in place for economy of space
does not return sorted sequence
guaranteed to be stable, not to change relative order of element compare equal

class tuple ([iterable]), immutable sequence
tuple may be constructed in several ways:
using pair of parentheses to denote empty tuple: ()
using trailing comma for singleton tuple: eg a, / (a,)
separating items with comma: eg a, b, c / (a, b, c)
using type constructor: tuple() / tuple(iterable)
(注意: comma actually make tuple, not parentheses, parentheses optional
except empty tuple / avoid syntactic ambiguity

map (function, iterable, ...), return iterator that apply function to every element of iterable
if additional iterable argument passed
function must take that many arguments
and applyed to items from all iterables in parallel
iterator stop when shortest iterable exhausted

Python - P44

class range(stop), immutable sequence
class range(start, stop[, step]), argument to range constructor must be integer
either built-in int or any object implement __index__() special method
if start omitted, default to 0
if step omitted, default to 1
if step is 0, raise ValueError
if step positive, range r determined by r[i] = start + step * i
where i >= 0 and r[i] < stop
if step negative, range r determined by r[i] = start + step * i
where i >= 0 and r[i] > stop
range object empty if r[0] not meet value constraint
range do support negative index
range containing absolute value larger than sys.maxsize is permitted
but features (such as len()) may raise OverflowError

common sequence operation, supported by most sequence types, both mutable and immutable
basic sequence type: list, tuple, range

operation	result
x in s	True if one item of s equal to x, else False
x not in s	False if one item of s equal to x, else True
s + t	concatenation of s and t
s * n / n * s	equivalent to adding s to itself n times
s[i]	i-th item of s, 0-based
s[i:j]	slice of s from i to j
s[i:j:k]	slice of s from i to j with step k
len(s)	length of s
min(s)	smallest item of s
max(s)	largest item of s
s.index(x[, i[, j]])	index of first occurence of x in s at/after index i and before j
s.count(x)	total number of occurence of x in s

where s, t are sequences of same type, n, i, j, k are integers
x is arbitrary object meet type/value restriction imposed by s

Python - P45

common sequence operations

sequence of same type also support comparison

in particular tuple / list compared lexicographically by comparing corresponding element

in / not in operation used only for simple containment testing in general case

specialised sequence (str / bytes / bytearray) also use for subsequence testing

当 n * s / s * n 中的 n 为负数时，会视为 n = 0

yield empty sequence of same type as s

特别注意，n * s / s * n 时，其中的元素并非复制，而是重复引用 (reference)

如 lst = [[]] * 3

lst[0].append(1) lst[1].append(2) lst[2].append(3)

⟶ [[1,2,3],[1,2,3],[1,2,3]]

即实际上 lst[0], lst[1], lst[2] 指向同一个 []

而 lst = [[] for _ in range(3)] 进行同样的操作

⟶ [[1],[2],[3]]

concatenating immutable sequence always result in new object

repeating concatenation have quadratic runtime (也叫 cn^2) cost in total sequence length

switch to alternative to get linear runtime (也叫 cn) cost

str object: build list first and use str.join() at the end

or write to io.StringIO instance and retrieve value when complete

bytes object: similarly use bytes.join() / io.BytesIO

or do in-place concatenation with bytearray object

bytearray mutable and have efficient overallocation mechanism

tuple object: extend list instead

some sequence types (如 range) only support item sequence follow specific pattern

not support sequence concatenation / repetition

对于 s[i], s[i:j], s[i:j:k]

当索引 i / j 小于 0 时，relative to end of sequence s

len(s) + i / len(s) + j substituted

在 slice of s 中，如果 i / j 大于 len(s) 时，视为 len(s)

当 i omitted / None 时视为 0，当 j omitted / None 时视为 len(s)

index 当 x 不在 s 中时抛出 ValueError

not all implementation support passing additional argument i / j

roughly equivalent to s[i:j].index(x), only without copying any data

with returned index relative to start of sequence rather than start of slice