你可能不知道的30个Python语言的特点技巧

从我开始学习Python时我就决定维护一个经常使用的“窍门”列表。不论何时当我看到一段让我觉得“酷，这样也行！”的代码时（在一个例子中、在StackOverflow、在开源码软件中，等等），我会尝试它直到理解它，然后把它添加到列表中。这篇文章是清理过列表的一部分。如果你是一个有经验的Python程序员，尽管你可能已经知道一些，但你仍能发现一些你不知道的。如果你是一个正在学习Python的C、C++或Java程序员，或者刚开始学习编程，那么你会像我一样发现它们中的很多非常有用。

每个窍门或语言特性只能通过实例来验证，无需过多解释。虽然我已尽力使例子清晰，但它们中的一些仍会看起来有些复杂，这取决于你的熟悉程度。所以如果看过例子后还不清楚的话，标题能够提供足够的信息让你通过Google获取详细的内容。

列表按难度排序，常用的语言特征和技巧放在前面。

1.1   分拆

>>> a, b, c = 1, 2, 3

>>> a, b, c

(1, 2, 3)

>>> a, b, c = [1, 2, 3]

>>> a, b, c

(1, 2, 3)

>>> a, b, c = (2 \* i + 1 for i in range(3))

>>> a, b, c

(1, 3, 5)

>>> a, (b, c), d = [1, (2, 3), 4]

>>> a

1

>>> b

2

>>> c

3

>>> d

4

1.2   交换变量分拆

>>> a, b = 1, 2

>>> a, b = b, a

>>> a, b

(2, 1)

1.3   拓展分拆 (Python 3下适用)

>>> a, \*b, c = [1, 2, 3, 4, 5]

>>> a

1

>>> b

[2, 3, 4]

>>> c

5

1.4   负索引

>>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

>>> a[-1]

10

>>> a[-3]

8

1.5   列表切片 (a[start:end])

>>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

>>> a[2:8]

[2, 3, 4, 5, 6, 7]

1.6   使用负索引的列表切片

>>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

>>> a[-4:-2]

[7, 8]

1.7   带步进值的列表切片 (a[start:end:step])

>>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

>>> a[::2]

[0, 2, 4, 6, 8, 10]

>>> a[::3]

[0, 3, 6, 9]

>>> a[2:8:2]

[2, 4, 6]

1.8   负步进值得列表切片

>>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

>>> a[::-1]

[10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

>>> a[::-2]

[10, 8, 6, 4, 2, 0]

1.9   列表切片赋值

>>> a = [1, 2, 3, 4, 5]

>>> a[2:3] = [0, 0]

>>> a

[1, 2, 0, 0, 4, 5]

>>> a[1:1] = [8, 9]

>>> a

[1, 8, 9, 2, 0, 0, 4, 5]

>>> a[1:-1] = []

>>> a

[1, 5]

1.10   命名切片 (slice(start, end, step))

>>> a = [0, 1, 2, 3, 4, 5]

>>> LASTTHREE = slice(-3, None)

>>> LASTTHREE

slice(-3, None, None)

>>> a[LASTTHREE]

[3, 4, 5]

1.11   zip打包解包列表和倍数

>>> a = [1, 2, 3]

>>> b = ['a', 'b', 'c']

>>> z = zip(a, b)

>>> z

[(1, 'a'), (2, 'b'), (3, 'c')]

>>> zip(\*z)

[(1, 2, 3), ('a', 'b', 'c')]

1.12   使用zip合并相邻的列表项

>>> a = [1, 2, 3, 4, 5, 6]

>>> zip(\*([iter(a)] \* 2))

[(1, 2), (3, 4), (5, 6)]

>>> group\_adjacent = lambda a, k: zip(\*([iter(a)] \* k))

>>> group\_adjacent(a, 3)

[(1, 2, 3), (4, 5, 6)]

>>> group\_adjacent(a, 2)

[(1, 2), (3, 4), (5, 6)]

>>> group\_adjacent(a, 1)

[(1,), (2,), (3,), (4,), (5,), (6,)]

>>> zip(a[::2], a[1::2])

[(1, 2), (3, 4), (5, 6)]

>>> zip(a[::3], a[1::3], a[2::3])

[(1, 2, 3), (4, 5, 6)]

>>> group\_adjacent = lambda a, k: zip(\*(a[i::k] for i in range(k)))

>>> group\_adjacent(a, 3)

[(1, 2, 3), (4, 5, 6)]

>>> group\_adjacent(a, 2)

[(1, 2), (3, 4), (5, 6)]

>>> group\_adjacent(a, 1)

[(1,), (2,), (3,), (4,), (5,), (6,)]

1.13  使用zip和iterators生成滑动窗口 (n -grams)

>>> from itertools import islice

>>> def n\_grams(a, n):

...     z = (islice(a, i, None) for i in range(n))

...     return zip(\*z)

...

>>> a = [1, 2, 3, 4, 5, 6]

>>> n\_grams(a, 3)

[(1, 2, 3), (2, 3, 4), (3, 4, 5), (4, 5, 6)]

>>> n\_grams(a, 2)

[(1, 2), (2, 3), (3, 4), (4, 5), (5, 6)]

>>> n\_grams(a, 4)

[(1, 2, 3, 4), (2, 3, 4, 5), (3, 4, 5, 6)]

1.14   使用zip反转字典

>>> m = {'a': 1, 'b': 2, 'c': 3, 'd': 4}

>>> m.items()

[('a', 1), ('c', 3), ('b', 2), ('d', 4)]

>>> zip(m.values(), m.keys())

[(1, 'a'), (3, 'c'), (2, 'b'), (4, 'd')]

>>> mi = dict(zip(m.values(), m.keys()))

>>> mi

{1: 'a', 2: 'b', 3: 'c', 4: 'd'}

1.15   摊平列表:

>>> a = [[1, 2], [3, 4], [5, 6]]

>>> list(itertools.chain.from\_iterable(a))

[1, 2, 3, 4, 5, 6]

>>> sum(a, [])

[1, 2, 3, 4, 5, 6]

>>> [x for l in a for x in l]

[1, 2, 3, 4, 5, 6]

>>> a = [[[1, 2], [3, 4]], [[5, 6], [7, 8]]]

>>> [x for l1 in a for l2 in l1 for x in l2]

[1, 2, 3, 4, 5, 6, 7, 8]

>>> a = [1, 2, [3, 4], [[5, 6], [7, 8]]]

>>> flatten = lambda x: [y for l in x for y in flatten(l)] if type(x) is list else [x]

>>> flatten(a)

[1, 2, 3, 4, 5, 6, 7, 8]

注意: 根据Python的文档，itertools.chain.from\_iterable是首选。

1.16   生成器表达式

>>> g = (x \*\* 2 for x in xrange(10))

>>> next(g)

0

>>> next(g)

1

>>> next(g)

4

>>> next(g)

9

>>> sum(x \*\* 3 for x in xrange(10))

2025

>>> sum(x \*\* 3 for x in xrange(10) if x % 3 == 1)

408

1.17   迭代字典

>>> m = {x: x \*\* 2 for x in range(5)}

>>> m

{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}

>>> m = {x: 'A' + str(x) for x in range(10)}

>>> m

{0: 'A0', 1: 'A1', 2: 'A2', 3: 'A3', 4: 'A4', 5: 'A5', 6: 'A6', 7: 'A7', 8: 'A8', 9: 'A9'}

1.18   通过迭代字典反转字典

>>> m = {'a': 1, 'b': 2, 'c': 3, 'd': 4}

>>> m

{'d': 4, 'a': 1, 'b': 2, 'c': 3}

>>> {v: k for k, v in m.items()}

{1: 'a', 2: 'b', 3: 'c', 4: 'd'}

1.19   命名序列 (collections.namedtuple)

>>> Point = collections.namedtuple('Point', ['x', 'y'])

>>> p = Point(x=1.0, y=2.0)

>>> p

Point(x=1.0, y=2.0)

>>> p.x

1.0

>>> p.y

2.0

1.20   命名列表的继承:

>>> class Point(collections.namedtuple('PointBase', ['x', 'y'])):

...     \_\_slots\_\_ = ()

...     def \_\_add\_\_(self, other):

...             return Point(x=self.x + other.x, y=self.y + other.y)

...

>>> p = Point(x=1.0, y=2.0)

>>> q = Point(x=2.0, y=3.0)

>>> p + q

Point(x=3.0, y=5.0)

1.21   集合及集合操作

>>> A = {1, 2, 3, 3}

>>> A

set([1, 2, 3])

>>> B = {3, 4, 5, 6, 7}

>>> B

set([3, 4, 5, 6, 7])

>>> A | B

set([1, 2, 3, 4, 5, 6, 7])

>>> A & B

set([3])

>>> A - B

set([1, 2])

>>> B - A

set([4, 5, 6, 7])

>>> A ^ B

set([1, 2, 4, 5, 6, 7])

>>> (A ^ B) == ((A - B) | (B - A))

True

1.22   多重集及其操作 (collections.Counter)

>>> A = collections.Counter([1, 2, 2])

>>> B = collections.Counter([2, 2, 3])

>>> A

Counter({2: 2, 1: 1})

>>> B

Counter({2: 2, 3: 1})

>>> A | B

Counter({2: 2, 1: 1, 3: 1})

>>> A & B

Counter({2: 2})

>>> A + B

Counter({2: 4, 1: 1, 3: 1})

>>> A - B

Counter({1: 1})

>>> B - A

Counter({3: 1})

1.23   迭代中最常见的元素 (collections.Counter)

>>> A = collections.Counter([1, 1, 2, 2, 3, 3, 3, 3, 4, 5, 6, 7])

>>> A

Counter({3: 4, 1: 2, 2: 2, 4: 1, 5: 1, 6: 1, 7: 1})

>>> A.most\_common(1)

[(3, 4)]

>>> A.most\_common(3)

[(3, 4), (1, 2), (2, 2)]

1.24   双端队列 (collections.deque)

>>> Q = collections.deque()

>>> Q.append(1)

>>> Q.appendleft(2)

>>> Q.extend([3, 4])

>>> Q.extendleft([5, 6])

>>> Q

deque([6, 5, 2, 1, 3, 4])

>>> Q.pop()

4

>>> Q.popleft()

6

>>> Q

deque([5, 2, 1, 3])

>>> Q.rotate(3)

>>> Q

deque([2, 1, 3, 5])

>>> Q.rotate(-3)

>>> Q

deque([5, 2, 1, 3])

1.25   有最大长度的双端队列 (collections.deque)

>>> last\_three = collections.deque(maxlen=3)

>>> for i in xrange(10):

...     last\_three.append(i)

...     print ', '.join(str(x) for x in last\_three)

...

0

0, 1

0, 1, 2

1, 2, 3

2, 3, 4

3, 4, 5

4, 5, 6

5, 6, 7

6, 7, 8

7, 8, 9

1.26   字典排序 (collections.OrderedDict)

>>> m = dict((str(x), x) for x in range(10))

>>> print ', '.join(m.keys())

1, 0, 3, 2, 5, 4, 7, 6, 9, 8

>>> m = collections.OrderedDict((str(x), x) for x in range(10))

>>> print ', '.join(m.keys())

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

>>> m = collections.OrderedDict((str(x), x) for x in range(10, 0, -1))

>>> print ', '.join(m.keys())

10, 9, 8, 7, 6, 5, 4, 3, 2, 1

1.27   缺省字典 (collections.defaultdict)

>>> m = dict()

>>> m['a']

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

KeyError: 'a'

>>>

>>> m = collections.defaultdict(int)

>>> m['a']

0

>>> m['b']

0

>>> m = collections.defaultdict(str)

>>> m['a']

''

>>> m['b'] += 'a'

>>> m['b']

'a'

>>> m = collections.defaultdict(lambda: '[default value]')

>>> m['a']

'[default value]'

>>> m['b']

'[default value]'

1.28   用缺省字典表示简单的树

>>> import json

>>> tree = lambda: collections.defaultdict(tree)

>>> root = tree()

>>> root['menu']['id'] = 'file'

>>> root['menu']['value'] = 'File'

>>> root['menu']['menuitems']['new']['value'] = 'New'

>>> root['menu']['menuitems']['new']['onclick'] = 'new();'

>>> root['menu']['menuitems']['open']['value'] = 'Open'

>>> root['menu']['menuitems']['open']['onclick'] = 'open();'

>>> root['menu']['menuitems']['close']['value'] = 'Close'

>>> root['menu']['menuitems']['close']['onclick'] = 'close();'

>>> print json.dumps(root, sort\_keys=True, indent=4, separators=(',', ': '))

{

    "menu": {

        "id": "file",

        "menuitems": {

            "close": {

                "onclick": "close();",

                "value": "Close"

            },

            "new": {

                "onclick": "new();",

                "value": "New"

            },

            "open": {

                "onclick": "open();",

                "value": "Open"

            }

        },

        "value": "File"

    }

}

(到https://gist.github.com/hrldcpr/2012250查看详情)

1.29   映射对象到唯一的序列数 (collections.defaultdict)

>>> import itertools, collections

>>> value\_to\_numeric\_map = collections.defaultdict(itertools.count().next)

>>> value\_to\_numeric\_map['a']

0

>>> value\_to\_numeric\_map['b']

1

>>> value\_to\_numeric\_map['c']

2

>>> value\_to\_numeric\_map['a']

0

>>> value\_to\_numeric\_map['b']

1

1.30   最大最小元素 (heapq.nlargest和heapq.nsmallest)

>>> a = [random.randint(0, 100) for \_\_ in xrange(100)]

>>> heapq.nsmallest(5, a)

[3, 3, 5, 6, 8]

>>> heapq.nlargest(5, a)

[100, 100, 99, 98, 98]

1.31   笛卡尔乘积 (itertools.product)

>>> for p in itertools.product([1, 2, 3], [4, 5]):

(1, 4)

(1, 5)

(2, 4)

(2, 5)

(3, 4)

(3, 5)

>>> for p in itertools.product([0, 1], repeat=4):

...     print ''.join(str(x) for x in p)

...

0000

0001

0010

0011

0100

0101

0110

0111

1000

1001

1010

1011

1100

1101

1110

1111

1.32   组合的组合和置换 (itertools.combinations 和 itertools.combinations\_with\_replacement)

>>> for c in itertools.combinations([1, 2, 3, 4, 5], 3):

...     print ''.join(str(x) for x in c)

...

123

124

125

134

135

145

234

235

245

345

>>> for c in itertools.combinations\_with\_replacement([1, 2, 3], 2):

...     print ''.join(str(x) for x in c)

...

11

12

13

22

23

33

1.33   排序 (itertools.permutations)

>>> for p in itertools.permutations([1, 2, 3, 4]):

...     print ''.join(str(x) for x in p)

...

1234

1243

1324

1342

1423

1432

2134

2143

2314

2341

2413

2431

3124

3142

3214

3241

3412

3421

4123

4132

4213

4231

4312

4321

1.34   链接的迭代 (itertools.chain)

>>> a = [1, 2, 3, 4]

>>> for p in itertools.chain(itertools.combinations(a, 2), itertools.combinations(a, 3)):

...     print p

...

(1, 2)

(1, 3)

(1, 4)

(2, 3)

(2, 4)

(3, 4)

(1, 2, 3)

(1, 2, 4)

(1, 3, 4)

(2, 3, 4)

>>> for subset in itertools.chain.from\_iterable(itertools.combinations(a, n) for n in range(len(a) + 1))

...     print subset

...

()

(1,)

(2,)

(3,)

(4,)

(1, 2)

(1, 3)

(1, 4)

(2, 3)

(2, 4)

(3, 4)

(1, 2, 3)

(1, 2, 4)

(1, 3, 4)

(2, 3, 4)

(1, 2, 3, 4)

1.35   按给定值分组行 (itertools.groupby)

>>> from operator import itemgetter

>>> import itertools

>>> with open('contactlenses.csv', 'r') as infile:

...     data = [line.strip().split(',') for line in infile]

...

>>> data = data[1:]

>>> def print\_data(rows):

...     print '\n'.join('\t'.join('{: <16}'.format(s) for s in row) for row in rows)

...

>>> print\_data(data)

young               myope                   no                      reduced                 none

young               myope                   no                      normal                  soft

young               myope                   yes                     reduced                 none

young               myope                   yes                     normal                  hard

young               hypermetrope            no                      reduced                 none

young               hypermetrope            no                      normal                  soft

young               hypermetrope            yes                     reduced                 none

young               hypermetrope            yes                     normal                  hard

pre-presbyopic      myope                   no                      reduced                 none

pre-presbyopic      myope                   no                      normal                  soft

pre-presbyopic      myope                   yes                     reduced                 none

pre-presbyopic      myope                   yes                     normal                  hard

pre-presbyopic      hypermetrope            no                      reduced                 none

pre-presbyopic      hypermetrope            no                      normal                  soft

pre-presbyopic      hypermetrope            yes                     reduced                 none

pre-presbyopic      hypermetrope            yes                     normal                  none

presbyopic          myope                   no                      reduced                 none

presbyopic          myope                   no                      normal                  none

presbyopic          myope                   yes                     reduced                 none

presbyopic          myope                   yes                     normal                  hard

presbyopic          hypermetrope            no                      reduced                 none

presbyopic          hypermetrope            no                      normal                  soft

presbyopic          hypermetrope            yes                     reduced                 none

presbyopic          hypermetrope            yes                     normal                  none

>>> data.sort(key=itemgetter(-1))

>>> for value, group in itertools.groupby(data, lambda r: r[-1]):

...     print '-----------'

...     print 'Group: ' + value

...     print\_data(group)

...

-----------

Group: hard

young               myope                   yes                     normal                  hard

young               hypermetrope            yes                     normal                  hard

pre-presbyopic      myope                   yes                     normal                  hard

presbyopic          myope                   yes                     normal                  hard

-----------

Group: none

young               myope                   no                      reduced                 none

young               myope                   yes                     reduced                 none

young               hypermetrope            no                      reduced                 none

young               hypermetrope            yes                     reduced                 none

pre-presbyopic      myope                   no                      reduced                 none

pre-presbyopic      myope                   yes                     reduced                 none

pre-presbyopic      hypermetrope            no                      reduced                 none

pre-presbyopic      hypermetrope            yes                     reduced                 none

pre-presbyopic      hypermetrope            yes                     normal                  none

presbyopic          myope                   no                      reduced                 none

presbyopic          myope                   no                      normal                  none

presbyopic          myope                   yes                     reduced                 none

presbyopic          hypermetrope            no                      reduced                 none

presbyopic          hypermetrope            yes                     reduced                 none

presbyopic          hypermetrope            yes                     normal                  none

-----------

Group: soft

young               myope                   no                      normal                  soft

young               hypermetrope            no                      normal                  soft

pre-presbyopic      myope                   no                      normal                  soft

pre-presbyopic      hypermetrope            no                      normal                  soft

presbyopic          hypermetrope            no                      normal                  soft