Python的各种推导式（列表推导式、字典推导式、集合推导式）

2018-03-08 21:41:36 Locutus 阅读数 6239更多

分类专栏： Python

推导式comprehensions（又称解析式），是Python的一种独有特性。推导式是可以从一个数据序列构建另一个新的数据序列的结构体。共有三种推导式，在Python2和3中都有支持：

列表(list)推导式

字典(dict)推导式

集合(set)推导式

1. 列表推导式

1.1 使用[]生成list

基本格式：

variable = [out\_exp\_res for out\_exp in input\_list if out\_exp == 2]

1

out\_exp\_res：列表生成元素表达式，可以是有返回值的函数。

for out\_exp in input\_list：迭代input\_list将out\_exp传入out\_exp\_res表达式中。

if out\_exp == 2：根据条件过滤哪些值可以。

实例：

过滤掉长度小于3的字符串列表，并将剩下的转换成大写字母

names = ['Bob','Tom','alice','Jerry','Wendy','Smith']

​

[name.upper() for name in names if len(name)>3]

​

# ['ALICE', 'JERRY', 'WENDY', 'SMITH']

生成间隔5分钟的时间列表序列

["%02d:%02d"%(h,m) for h in range(0, 24) for m in range(0, 60, 5)]

1

求(x, y)，其中x是0-5之间的偶数，y是0-5之间的奇数组成的元祖列表

[(x,y) for x in range(5) if x%2==0 for y in range(5) if y%2==1]

# [(0, 1), (0, 3), (2, 1), (2, 3), (4, 1), (4, 3)]

求M中3, 6, 9组成的列表

M = [[1,2,3], [4,5,6], [7,8,9]]

​

[row[2] for row in M] # [3, 6, 9]

​

# 或者用下面的方式

​

[M[row][2] for row in (0, 1, 2)] # [3, 6, 9]

求M中斜线1, 5, 9组成的列表

M = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

​

[M[i][i] for i in range(len(M))] # 打印M[0][0], M[1][1], M[2][2]

​

# [1, 5, 9]

求M, N中矩阵和元素的乘积

M = [[1,2,3],[4,5,6],[7,8,9]]

N = [[2,2,2],[3,3,3],[4,4,4]]

​

[M[row][col]\*N[row][col] for row in range(3) for col in range(3)]

​

# [2, 4, 6, 12, 15, 18, 28, 32, 36]

[[M[row][col]\*N[row][col] for col in range(3)] for row in range(3)]

# [[2, 4, 6], [12, 15, 18], [28, 32, 36]]

​

[[M[row][col]\*N[row][col] for row in range(3)] for col in range(3)]

​

# [[2, 12, 28], [4, 15, 32], [6, 18, 36]]

把字典中age键，按照条件赋新值

bob = {'pay': 3000, 'job': 'dev', 'age': 42, 'name': 'Bob Smith'}

​

sue = {'pay': 4000, 'job': 'hdw', 'age': 45, 'name': 'Sue Jones'}

​

people = [bob, sue]

​

[rec['age']+100 if rec['age'] >= 45 else rec['age'] for rec in people] # 注意for位置

# [42, 145]

列表推导式

multiples = [i for i in range(30) if i%3 == 0]

print(multiples)

# Output: [0, 3, 6, 9, 12, 15, 18, 21, 24, 27]

def squared(x):

return x\*x

multiples = [squared(i) for i in range(30) if i%3 == 0]

print(multiples)

# Output: [0, 9, 36, 81, 144, 225, 324, 441, 576, 729]

1.2 使用()生成generator

将上述两表推导式的[]改成()，即可得到生成器。

multiples = (i for i in range(30) if i%3 == 0)

print(type(multiples))

# Output: <type 'generator'>

2. 字典推导式

字典推导和列表推导的使用类似，只不过中括号改成大括号。

基本格式：{ key\_expr: value\_expr for value in collection if condition }

举例说明：

用字符串和其长度创建字典

strings = ['import','is','with','if','file','exception']

D = {key: val for val,key in enumerate(strings)}

D

# {'exception': 5, 'file': 4, 'if': 3, 'import': 0, 'is': 1, 'with': 2}

大小写key合并

mcase = {'a': 10, 'b': 34, 'A': 7, 'Z': 3}

mcase\_frequency = {

k.lower(): mcase.get(k.lower(), 0) + mcase.get(k.upper(), 0)

for k in mcase.keys()

if k.lower() in ['a','b']

}

print(mcase\_frequency)

# Output: {'a': 17, 'b': 34}

快速更换key和value

mcase = {'a': 10, 'b': 34}

mcase\_frequency = {v: k for k, v in mcase.items()}

print(mcase\_frequency)

# Output: {10: 'a', 34: 'b'}

3. 集合推导式

集合推导式跟列表推导式也是类似的。 唯一的区别在于它使用大括号{ }。

基本格式：{ expr for value in collection if condition }

举例说明：

集合推导式

squared = {x\*\*2 for x in [1, 1, 2]}

print(squared)

# Output: set([1, 4])

用集合推导式，创建字符串长度的集合

strings = ['a','is','with','if','file','exception']

{len(s) for s in strings} # 有长度相同的会只留一个，这在实际上也非常有用

# {1, 2, 4, 9}

一个由男人列表和女人列表组成的嵌套列表，取出姓名中带有两个以上含字母e的姓名，组成列表

Names=[['Tom','Billy','Jefferson','Andrew','Wesley','Steven','Joe'], ['Alice','Jill','Ana','Wendy','Jennifer','Sherry','Eva']]

tmp = []

# 用for循环实现

for lst in names:

for name in lst:

if name.count('e') >= 2:

tmp.append(name)

print(tmp)

# 用嵌套列表实现

[name for lst in names for name in lst if name.count('e')>=2] # 注意遍历顺序，这是实现的关键

# ['Jefferson', 'Wesley', 'Steven', 'Jennifer']