一步一腳印
精通Python程式設計

容器及其操作

課程講師: 陳建銘

2018/05/30

課程大綱

- list
  - 串列內存資料型態
  - 串列內建函式
  - 實例說明
- tuple
  - 序對內存資料型態
  - 實例說明
- set
  - 集合內建函式
  - 集合操作運算
  - 實例說明
- dict
  - 字典內建函式
  - 字典操作運算
  - 實例說明

容器

容器是什麼?為什麼需要容器?

ANS: 容器可用來存放資料。好的資料結構有助於加速後續的資料解析與運算。

- **list**
  - 串列內存資料型態
  - 串列內建函式
  - 實例說明
- **tuple**
  - 序對內存資料型態
  - 實例說明
- **set**
  - 集合內建函式
  - 集合操作運算
  - 實例說明
- **dict**
  - 字典內建函式
  - 字典操作運算
  - 實例說明

串列容器

list –

串列是有序且內容可變的容器。串列內容可以是任何型態，例如整數、浮點數、字串、布林值或容器等。

串列容器 Cont.

串列內存資料型態

整數串列	[1, 2, 3, ...]
浮點數串列	[1.1, 2.2, 3.3, ...]
布林串列	[True, False, ...]
字串串列	['value$_1$', 'value$_2$', ...]
複合串列	[6, 'value$_1$', False, 2.8, ...]

串列容器 Cont.

串列內建函式

方法	描述
list.append(x)	將x加入至串列的最後索引位置
list.extend(xList)	將xList中的所有值加入至串列的最後索引位置
list.count(x)	計算串列中x出現的次數
list.index(x)	回傳x在串列中的最小索引值
list.insert(i, x)	將x加入至串列索引為i的位置
list.pop()	取出串列最後一個位置的值
list.remove(x)	移除串列中第一次出現的x
list.reverse()	反轉串列中元素的順序
list.sort(reverse=bool)	排序串列中的元素

串列容器 Cont.

範例1: 串列存取I

```
if __name__ == '__main__':
    myList = [6, 'value1', False, 2.8, 9, 'value2']
    print(myList[0])
    print(myList[2])

    for listValue in myList:
        print(listValue)
```

可透過中括號並指定索引值來取得串列內的值。

可透過迴圈迭代所有串列內的值。

輸出如下:

```
6
False
6
value1
False
2.8
9
value2
```

串列容器 Cont.

範例2: 串列存取II

```
if __name__ == '__main__':
    myList = [6, 'value1', False, 2.8, 9, 'value2']

    newList = myList[2:5]
    print(newList)

    myList[-1] = 'change'

    if myList[5]==myList[-1]:
        print(True)
    else:
        print(False)
```

可透過中括號並指定索引值範圍來取得子串列。

串列索引-1的值代表此串列容器的最後一個值。

輸出如下:

```
[False, 2.8, 9]
True
```

串列容器 Cont.

範例3: 串列操作I

```
if __name__ == '__main__':
    strList = ['value1', 'value2', 'value1']
    digitList = [8, 1, 5, 7, 3, 2, 4, 6, 9]

    digitList.append(0)
    digitList.extend(strList)
    digitList.insert(10, True)

    print(digitList.count('value1'))
    print(digitList.index(0))
    print(digitList)
```

輸出如下:

```
2
9
[8, 1, 5, 7, 3, 2, 4,
 6, 9, 0, True,
 'value1', 'value2',
 'value1']
```

串列容器 Cont.

範例4: 串列操作II

```
if __name__ == '__main__':
    strList = ['value1', 'value2', 'value1']
    digitList = [8, 1, 5, 7, 3, 2, 4, 6, 9]

    digitList.sort(reverse=True)
    digitList.pop()
    digitList.extend(strList)
    digitList.remove('value1')

    print(digitList)
    digitList.reverse()
    print(digitList)
```

輸出如下:

```
[9, 8, 7, 6, 5, 4, 3, 2,
 'value2', 'value1']
['value1', 'value2', 2,
 3, 4, 5, 6, 7, 8, 9]
```

串列容器 Cont.

範例5: 二維串列存取

```
if __name__ == '__main__':
    tableList = [['field1', 'field2', 'field3'],
                 [2, 4, 6],
                 [1, 3, 5],
                 [7, 8, 9]]
    print('串列長度: ' + str(len(tableList)))
    print('串列第一列項目長度: ' + str(len(tableList[0])))
    print('串列第一列第二行: ' + str(tableList[0][1]))
    print('串列第三列第三行: ' + str(tableList[2][2]))
```

輸出如下:
串列長度:4
串列第一列項目長度:3
串列第一列第二行:field2
串列第三列第三行:5

串列容器 Cont.

範例6: 二維串列排序

```
from operator import itemgetter

if __name__ == '__main__':
    tableList = [[2, 4, 6],
                 [1, 3, 5],
                 [7, 8, 9]]

    tableList = sorted(tableList, key=itemgetter(2), reverse=True)

    for tableRow in tableList:
        print(tableRow)
```

使用itemgetter，須匯入相關函式庫。

itemgetter可指定排序所欲參考的行數。

若reverse設定為True，則表示降冪排序。反之則為升冪排序。

輸出如下:

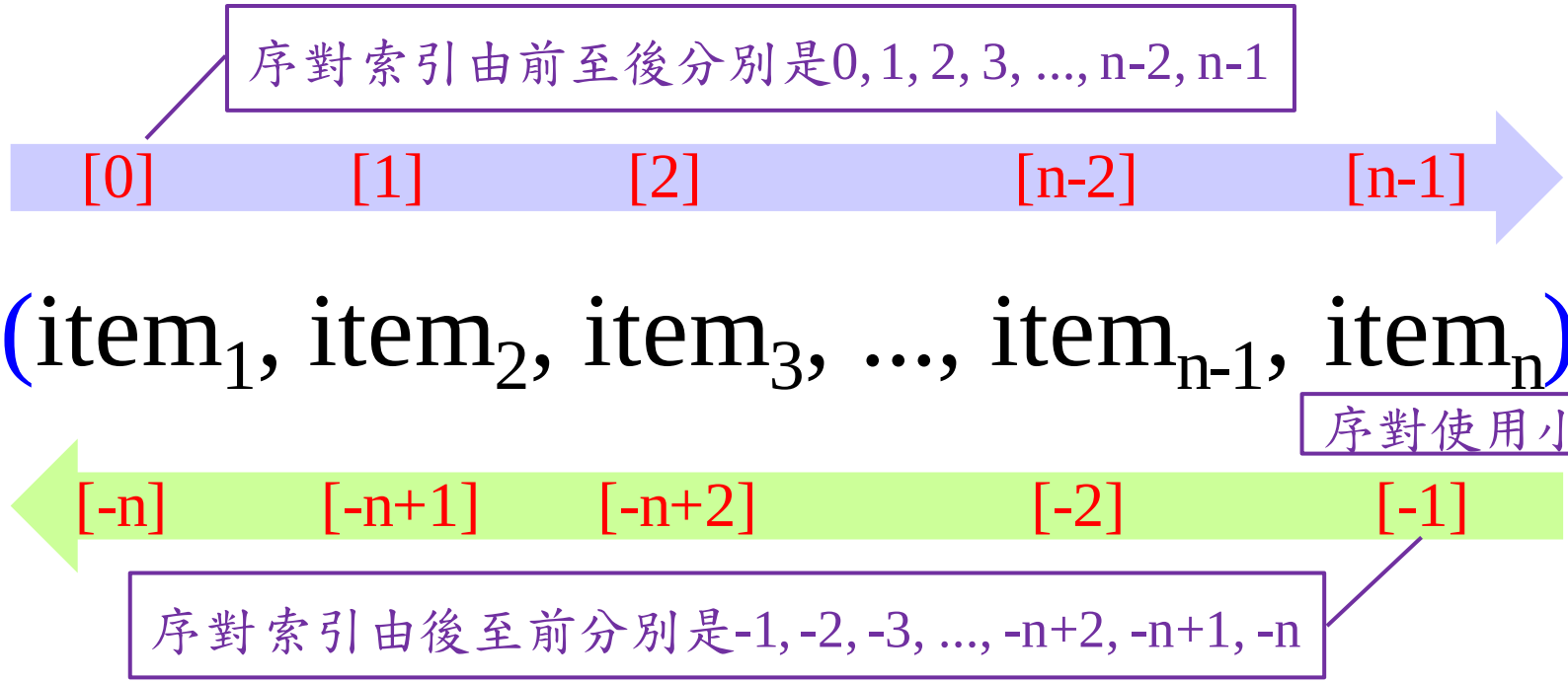
```
[7, 8, 9]
[2, 4, 6]
[1, 3, 5]
```

- list
  - 串列內存資料型態
  - 串列內建函式
  - 實例說明
- **tuple**
  - 序對內存資料型態
  - 實例說明
- set
  - 集合內建函式
  - 集合操作運算
  - 實例說明
- dict
  - 字典內建函式
  - 字典操作運算
  - 實例說明

序對容器

tuple –

序對是有序且內容不可變的容器。可存放的內容與串列相同。但不同於串列，序對只能唯讀。



序對容器 Cont.

序對內存資料型態

整數序對 (1, 2, 3, ...)

浮點數序對 (1.1, 2.2, 3.3, ...)

布林序對 (True, False, ...)

字串序對 ('value$_1$', 'value$_2$', ...)

複合序對 (6, 'value$_1$', False, 2.8, ...)

序對容器 Cont.

範例1: 序對存取I

```
if __name__ == '__main__':
    myTuple = (6, 'value1', False, 2.8, 9, 'value2')
    print(myTuple[0])
    print(myTuple[2])

    for tupleValue in myTuple:
        print(tupleValue)
```

輸出如下:

```
6
False
6
value1
False
2.8
9
value2
```

序對容器 Cont.

範例2: 序對存取II

```
if __name__ == '__main__':
    myTuple = (6, 'value1', False, 2.8, 9, 'value2')

    newList = myTuple[2:5]
    print(newList)

    if myTuple[5]==myTuple[-1]:
        print(True)
    else:
        print(False)

    #myTuple[4] = 7
```

輸出如下:

```
[False, 2.8, 9]
True
```

序對只可唯讀，不可更改其值。
若更改其值，則出現程式錯誤。

- list
  - 串列內存資料型態
  - 串列內建函式
  - 實例說明
- tuple
  - 序對內存資料型態
  - 實例說明
- set
  - 集合內建函式
  - 集合操作運算
  - 實例說明
- dict
  - 字典內建函式
  - 字典操作運算
  - 實例說明

集合容器

set –

集合是無序且內容不重複的容器。可存放複合型的資料型態。

$$\{item_1, item_2, item_3, ..., item_{n-1}, item_n\}$$

集合使用大括號

```
if __name__ == '__main__':
    xSet = {2, 5, 8, 4, 6, 2, 4, 3.5, 8, 6, True}
    ySet = {'value', 1, 3, 7, 6, 'value', 3, 9, 6}

    print(xSet)
    print(ySet)
```

輸出如下:

```
{True, 2, 3.5, 4, 5, 6, 8}
{1, 3, 6, 7, 9, 'value'}
```

集合容器 Cont.

集合內建函式

方法	描述
xSet.add(y)	將y加入至集合
xSet.remove(y)	將y從集合中移除
xSet.intersection(ySet)	兩集合交集，xSet∩ySet
xSet.union(ySet)	兩集合聯集，xSet∪ySet
xSet.difference(ySet)	兩集合差集，xSet–ySet
xSet.issubset(ySet)	xSet是否包含於ySet
xSet.issuperset(ySet)	xSet是否包含ySet
xSet.isdisjoint(ySet)	兩集合是否無交集
xSet.copy()	回傳複製的xSet

集合容器 Cont.

集合操作運算

計算	描述
v in xSet	v值是否在xSet中
v not in xSet	v值是否不在xSet中
xSet & ySet	兩集合交集，xSet∩ySet
xSet | ySet	兩集合聯集，xSet∪ySet
xSet - ySet	兩集合差集，xSet－ySet
xSet <= ySet	xSet是否包含於ySet
xSet >= ySet	xSet是否包含ySet
len(xSet)	回傳xSet的內容物個數
min(xSet)	回傳xSet中的最小值，xSet中的內容物必須是相同型態
max(xSet)	回傳xSet中的最大值，xSet中的內容物必須是相同型態

集合容器 Cont.

範例1: 集合操作I

```
if __name__ == '__main__':
    xSet = {'value1', 2, 3.4, True, 2, 0}
    ySet= {3.4, False, 'value1', 3.4, 2}

    xSet.remove(0)
    xSet.add(6)

    newSet1 = xSet.intersection(ySet)
    newSet2 = xSet.union(ySet)
    newSet3 = xSet.difference(ySet)

    print(xSet)
    print(newSet1)
    print(newSet2)
    print(newSet3)
```

輸出如下:

{'value1', 2, True, 6, 3.4}
{'value1', 2, 3.4}
{False, 'value1', 2, True, 6, 3.4}
{True, 6}

集合容器 Cont.

範例2: 集合操作II

```
if __name__ == '__main__':
    xSet = {2, 1, 4, 3}
    ySet = {4, 2, 3, 1, 5, 6}
    zSet = {3, 4, 5}

    print(xSet <= ySet)
    print(xSet >= zSet)
    print('xSet的長度: ' + len(xSet))
    print('zSet的最小值: ' + min(zSet))
    print('ySet的最大值: ' + max(ySet))
```

輸出如下:

```
True
False
xSet的長度: 4
zSet的最小值: 3
ySet的最大值: 6
```

- list
  - 串列內存資料型態
  - 串列內建函式
  - 實例說明
- tuple
  - 序對內存資料型態
  - 實例說明
- set
  - 集合內建函式
  - 集合操作運算
  - 實例說明
- dict
  - 字典內建函式
  - 字典操作運算
  - 實例說明

字典容器

dict –

字典是由多組關鍵字及其對應值所組成的容器。
不同於串列及序對是透過索引來存取其內容物，
字典需透過關鍵字做為索引來存取其內容物。

字典使用大括號

{'key_1':$value_1$, 'key_2':$value_2$, ..., 'key_n':$value_n$}

{6.0:6, 'key_1':'$value_1$', 12:False, True:2.8, ...} NG

{'INT':6, 'STR':'value', 'BOOL':False, 'FLOAT':2.8, ...} GOOD

關鍵字若無其他必要，最好使用字串型態來定義。

字典容器 Cont.

字典內建函式

方法	描述
dict.clear()	清除dict內的所有項目
dict.copy()	回傳複製的dict
dict.get(key, response)	若dict內有關鍵字key，則回傳其對應值。反之，則回傳response。若不指定response，則預設回傳None
dict.items()	回傳由dict所建立的迭代物件
dict.keys()	回傳由dict關鍵字所建立的迭代物件
dict.values()	回傳由dict對應值所建立的迭代物件
dict.pop(key, response)	若dict內有關鍵字key，則移除此組項目。反之，則回傳response
dict.popitem()	從dict內任意移除一組關鍵字及其對應值
dict.setdefault(key, value)	若dict內有關鍵字key，則回傳其對應值。反之，則新增一組key:value
dict.update({'key':value, ...})	透過新增或修改的方式來更新dict內容

字典容器 Cont.

字典操作運算

計算	描述
dict[key]	取得dict中關鍵字為key的值
dict[key] = value	指定dict中關鍵字為key的值為value
del dict[key]	刪除關鍵字為key的項目
key in dict	dict中是否有關鍵字key
key not in dict	dict中是否沒有關鍵字key
iter(dict)	回傳由dict關鍵字所建立的迭代物件
len(dict)	回傳關鍵字及其對應值的組數

字典容器 Cont.

範例1: 字典操作I

```
if __name__ == '__main__':
    colorCode = {'R':'#FF0000', 'G':'#00FF00', 'B':'#0000FF'}

    for colorType in colorCode.keys():
        print(colorCode[colorType])

    print(colorCode.get('G', '沒有此關鍵字G'))
    print(colorCode.get('P', '沒有此關鍵字P'))

    colorCode.update({'P':'#6600CC'})
    colorCode.pop('G', '沒有此關鍵字G')
    print(colorCode)

    colorCode.clear()
    print(colorCode)
```

輸出如下:

```
#FF0000
#00FF00
#0000FF
#00FF00
沒有此關鍵字P
{'R': '#FF0000', 'B': '#0000FF',
 'P': '#6600CC'}
{}
```

字典容器 Cont.

範例2: 字典操作II

```
if __name__ == '__main__':
    personInfo = {'Name':'Peter', 'Age':25, 'isExpert':False, '??':'xx'}

    del personInfo['??']
    personInfo['Name'] = 'Peter Wang'
    personInfo['isExpert'] = True

    print('共' +str(len(personInfo))+ '項資訊，如下所示 -')

    for info in personInfo.keys():
        print(info +': '+ str(personInfo[info]))
```

修改關鍵字的對應值

輸出如下:

```
共3項資訊，如下所示 -
Name: Peter Wang
Age: 25
isExpert: True
```

字典容器 Cont.

範例3: 字典排序

```
from operator import itemgetter

if __name__ == '__main__':
    myDict = {'key1':2, 'key2':6, 'key3':5, 'key4':1, 'key5':3}

    myDict = dict(sorted(myDict.items(), key=itemgetter(1), reverse=False))

    print(myDict)
```

透過items函式，將字典轉成可迭代串列。
[('key1', 2), ('key2', 6), ('key3', 5), ('key4', 1), ('key5', 3)]

輸出如下:
{'key4': 1, 'key1': 2, 'key5': 3, 'key3': 5, 'key2': 6}

容器

重點整理I–

- 串列的元素是有序的，它可以收集不同資料型態的元素且串列中的元素也允許是串列。串列的特徵是中括號[]，使用索引位置可以存取元素。

- 序對的特徵是小括號()，用法大至上與串列類似，與之最大的不同是，序對是一種唯讀且不可變更的資料結構，亦即不可取代序對中的任意一個元素。

容器 Cont.

重點整理II –

- 集合型態的元素是無序的，其特徵是使用大括號{}，屬於複合資料型態。也就是說單一集合型態物件可以包含多個元素，但沒有重複的元素。

- 字典的元素是無序的，其特徵是使用大括號{}。藉由關鍵字與其對應值做配對來儲存資料。與串列及序對不同，字典不以索引位置來存取元素，而是改以關鍵字做為索引來存取元素。

本週課程終於結束囉^^