勞動部產業新尖兵計畫

人工智慧金融應用與實務培訓班

課程模組： AI 金融科技課程 - Python 程式設計

5. 容器的處理（二）

葉建華 (Yeh, Jian-hua)

tdi.jhyeh@tdi.edu.tw
au4290@gmail.com

講次內容

- 集合型別
- 映對型別
- 集合生成式
- 辭典生成式

集合型別

- 集合型別 set： 語法是以 { } 包住元素，以逗號隔開儲存的元素

```
set1 = set()      # 建立一個空集合
set2 = {'James', 2, 3,'Python'}
print("set1的資料型態: ", type(set1))

print(set2)       # 印出set2的內容
set2.add(10)      # set2加入元素10
print(set2)       # 再印出set2的內容(為什麼10不在尾端)
set2.add(2)       # set2加入元素2
print(set2)       # 再印出set2的內容(為什麼2無法加入?)

set2.remove(2)    # set2移除指定元素2
print(set2)       # 再印出set2的內容
```

集合型別

- 儲存的元素不可以重複，且沒有順序性

```
set1 = {1, 2, 3, 4, 5, 5}
print(set1)                 # 相同物件只存在一個
# {1, 2, 3, 4, 5}

set2 = set(range(10))       # 使用set來建構集合物件
print(set2)
# {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

集合：加入與移除元素

- add、 remove、 discard

```
x = {}              # {}是空辭典，不是空集合!
print(type(x))      # <class 'dict'>

set1 = set()        # 使用建構式建立空集合
set1.add(6)         # 容器可以加入任意的物件
set1.add('hi')      # 容器可以加入任意的物件!
set1.add((1, 2))    # 容器可以加入任意的物件！!
print(set1)         # {(1, 2), 'hi', 6}

set1.remove(6)      # 移除指定的元素
print(set1)         # {(1, 2), 'hi'}
set1.remove(7)      # 移除不存在的元素會出錯，KeyError: 7
set1.discard('hello') #  「丟掉」式的移除，若指定元素不存在會忽視
print(set1)         # {(1, 2), 'hi'}
```

集合之間的操作

- 集合之間可以計算交集、聯集、差集、對稱差集

```
set1 = {0, 1, 2, 3, 4, 5}
print(set1 & set(range(0, 10, 2)))    # 交集，{0, 2, 4}
print(set1 | set(range(0, 10, 2)))    # 聯集，{0, 1, 2, 3, 4, 5, 6, 8}
print(set1 - set(range(0, 10, 2)))    # 差集，{1, 3, 5}
# 0,2,4被減掉了!
print(set1 ^ set(range(0, 10, 2)))    # 對稱差集，{1, 3, 5, 6, 8}
# 對稱差集運作類似xor
```

練習：低於 500 的數字做過濾

- 請問小於 500 且同時是 7 和 9 的倍數，列出並計算這樣的自然數總和，使用集合來解
- 請問小於 500 且是 7 但不是 9 的倍數，列出並計算這樣的自然數總和，使用集合來解
- 請問小於 500 且是 9 但不是 7 的倍數，列出並計算這樣的自然數總和，使用集合來解
- 怎麼做?

練習：低於 500 的數字做過濾

- 請問小於 500
  和，使用集合
- 請問小於 500
  和，使用集合
- 請問小於 500
  和，使用集合
- 怎麼做?

```
set7 = set()
set9 = set()
for num in range(1, 500):
    if num%7==0:
        set7.add(num)
    if num%9==0:
        set9.add(num)

print(set7 & set9)          # 7和9的倍數集合交集
print(sum(set7 & set9))

print(set7 - set9)          # 差集，7的倍數集合減去9的倍數集合
print(sum(set7 - set9))

print(set9 - set7)          # 差集，9的倍數集合減去7的倍數集合
print(sum(set9 - set7))
```

其他特殊容器： frozenset

- frozenset: 不可變的集合

```
fset = frozenset(['a', 'b', 'c'])
print(fset)              # frozenset({'b', 'c', 'a'})

fset.remove('a')         # 不能修改，AttributeError
# frozenset根本沒有remove()可用!
```

講次內容

- 集合型別
- 映對型別
- 集合生成式
- 辭典生成式

映對型別 Mapping

- 主要以辭典型別 dict 作為代表：使用「鍵值 - 資料值」配對存取 (key, value)，描述配對時以冒號：來隔開

```
dic1 = {1:'Jimmy', 2:'Alex', '3':'john', 4:'mike'}
print("dic1的資料型態: ", type(dic1))

print(dic1)             # 印出dic1的內容 => 「鍵值-資料」值配對
print("鍵值1對應的資料值: "+ dic1[1])     # 使用鍵值1來取出資料值
print("鍵值4對應的資料值: "+ dic1[4])     # 使用鍵值4來取出資料值
print("鍵值3對應的資料值: "+ dic1[3])     # 使用鍵值3來取出資料值...錯!！為什麼?
print("鍵值\'3\'對應的資料值: "+ dic1['3'])     # 使用鍵值'3'來取出資料值

print(dic1.keys())      # 取出所有的鍵值
print(dic1.values())    # 取出所有的資料值
```

辭典型別

- 辭典型態 dict： 所有的鍵值形成集合 (set)，特性如同 set 一樣
  - 也就是說，鍵值不可以重複，且沒有順序性

```
dic1 = {'name': 'Amy', 'age': 18}
print(dic1)           # {'age': 18, 'name': 'Amy'}, 辭典儲存不具順序性
print(dic1['name'])   # 以鍵值來提取資料值: 'Amy'
dic1['age'] = 29        # 以鍵值來寫入資料值(修改)
print(dic1)           # {'age': 29, 'name': 'Amy'}
print(dic1['weight'])   # 錯! 無此鍵值, KeyError: 'weight'
```

辭典型別

- 辭典的基本操作

```
dic1 = {44: [0, 1, 2], (3, 4): {'x': 1}}
print(dic1[(3, 4)])          # 鍵值是tuple，資料值是dict{'x': 1}

dic2 = {8622501: {'name': 'Amy', 'age': 26}, \
        8622502: {'name': 'Bob', 'age': 33}}
print(dic2[8622501])         # 鍵值是學號(很大的數字)
# 資料值是學生資料的dict{'age': 26, 'name': 'Amy'}
print(dic2[8622501]['name'])
# 資料值是dict，所以可以實際上是(dic2[8622501])['name']，結果是'Amy'
```

辭典型別

- 辭典內容存取與增刪

```
dic1 = {'age':33, 'name':'Amy', 'score':86}
print(len(dic1))          # 長度，就是有幾個key-value配對: 3
print('age' in dic1)      # 測試有無此鍵值，True
del dic1['score']         # 刪除鍵值配對
# {'age': 33, 'name': 'Amy'}
dic1['grade'] = 86        # 增加(或修改)鍵值配對
# {'age':33, 'name':'Amy', 'grade':86}

for key in dic1:          # 針對鍵值進行迭代
    print(key + ' ', end='')  # grade age name，不換行
```

辭典型別

- 辭典內容存取與增刪，用 zip 配對

```
keys = ('name', 'age', 'job')
values = ('Amy', 25, 'writer')
dic1 = dict(zip(keys, values))      # zip真好用!
print(dic1)
# {'name': 'Amy', 'age': 25, 'job': 'writer'}

del dic1['job']
# {'name': 'Amy', 'age': 25}
del dic1['age']
# {'name': 'Amy'}

dic1['job'] = 'teacher'
# {'name': 'Amy', 'job': 'teacher'}
dic1['age'] = 29
# {'name': 'Amy', 'job': 'teacher', 'age': 29}
print(dic1)
```

辭典型別

- 設定預設回傳值： get 、 setdefault

```
dic1 = {'name': 'Amy', 'age': 25}
print(dic1.get('job', 'freelancer'))
# 如果沒有鍵值'job'，回傳'freelancer'

dic1.setdefault('job', 'singer')
# 沒有鍵值'job'時，回傳'singer
print(dic1.get('job'))
print(dic1['job'])                  # 同上一行，只是get功能更多
print(dic1)                         # 其實{'job':'singer'}已經偷偷加進去了...

dic1.setdefault('job', 'dancer')
# 已有鍵值'job'的預設回傳值，所以setdefault無效
print(dic1.get('job'))
```

辭典型別

- 動態反映出 dict 內容 (view)： keys、 values、 items

```
dic1 = {'name': 'Amy', 'age': 23, 'job': 'writer'}
print(dic1.keys())
print(type(dic1.keys()))
# dict_keys(['job', 'name', 'age'])
print(dic1.values())
print(type(dic1.values()))
# dict_values(['writer', 'Amy', 23])
print(dic1.items())
print(type(dic1.items()))
# dict_items ([('job','writer'), ('name','Amy'),('age',23)])
```

其實 keys() 和 items() 可以回傳 set 才對啊!

辭典型別

- 動態反映出 dict 內容 (view)： keys、 values、 items

```
dic1 = {'age':33, 'name':'Amy', 'score':86}
print(len(dic1))          # 長度，就是有幾個key-value配對: 3
print('age' in dic1)      # 測試有無此鍵值，True
del dic1['score']         # 刪除鍵值配對
# {'age': 33, 'name': 'Amy'}
dic1['grade'] = 86        # 增加(或修改)鍵值配對
# {'age':33, 'name':'Amy', 'grade':86}

for key in dic1:          # 針對鍵值進行迭代
    print(key + ' ', end='')  # grade age name，不換行

for k, v in dic1.items():     # 可以用items()取出key-value配對!
    print(k, v, ' ', end=')
```

講次內容

- 集合型別
- 映對型別
- 集合生成式
- 辭典生成式

集合生成式

- 稱為 Set Comprehension
  - 你已經知道 List Comprehension 了哦 ...
- 生成式是一種需要經過運算的表示式
  - 最終產出是一個資料結構
  - 集合生成：{ 運算式 for 名稱 in 可迭代者 if 運算式 }

集合生成式

- 簡單範例

```
list1 = ['a', 'bar', 'candy', 'o', 'car']

# 算出list1中的詞彙有幾種長度
set1 = {len(x) for x in list1}              # {1, 3, 5}
print(set1)

# 找出list1中長度為1的元素
set2 = {x for x in list1 if len(x) == 1} # {'o', 'a'}
print(set2)
```

練習：找質數!

很呆板的方法！!

- 請使用很單純的厄拉托西尼篩法 (sieve of Eratosthenes) 來找質數
  - 列舉 2~n 之間的所有數字，從 2 開始，刪除所有 2 的倍數
  - 每次刪完之後，找出最小數字 k，繼續刪除所有 k 的倍數
  - 直到 n 為止
- 怎麼做?

練習：找質數!

- 請使用很單純的厄拉托西尼篩法 (sieve of Eratosthenes) 來找質數
  - 列舉 2~
  - 每次刪
  - 直到 n
- 怎麼做?

很呆板的方法！！

```
n = 1000
# 最初的集合
set1 = set(range(2, n+1))

for num in range(2, n+1):
    # setK: num2 倍數的集合，由 num2*2 起算
    setK = {num2 for num2 in range(num*2, n+1, num)}
    set1 -= setK

print(set1)
```

講次內容

- 集合型別
- 映對型別
- 集合生成式
- 辭典生成式

辭典生成式

- 稱為 Dict Comprehension
  - 你已經知道 List/Set Comprehension 了哦 ...
- 生成式是一種需要經過運算的表示式
  - 最終產出是一個資料結構
  - 生成辭典:

    { 鍵值運算式 : 資料值運算式 for 名稱 in 可迭代者 if 運算式 }

辭典生成式

- 簡單範例

```
dic1 = {k: k**2 for k in range(5)}    # 資料值是鍵值的平方!
print(dic1)
# {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}

dic2 = {'Amy': 90, 'Joe': 45, 'Kevin': 33}
dic3 = {k: v for k, v in dic2.items() if v < 60}
print(dic3)
# 挑出不及格的, {'Joe': 45, 'Kevin': 33}
```

這個講次中，你應該學到了 ...

- 集合型別如何撰寫與運用
- 辭典型別如何撰寫與運用
- 集合生成式的使用
- 辭典生成式的使用