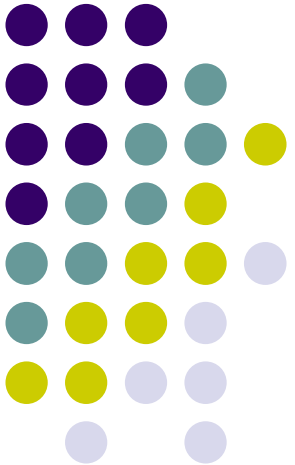
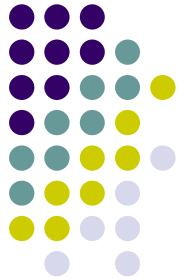
Python程式設計入門
容器-映射

葉難

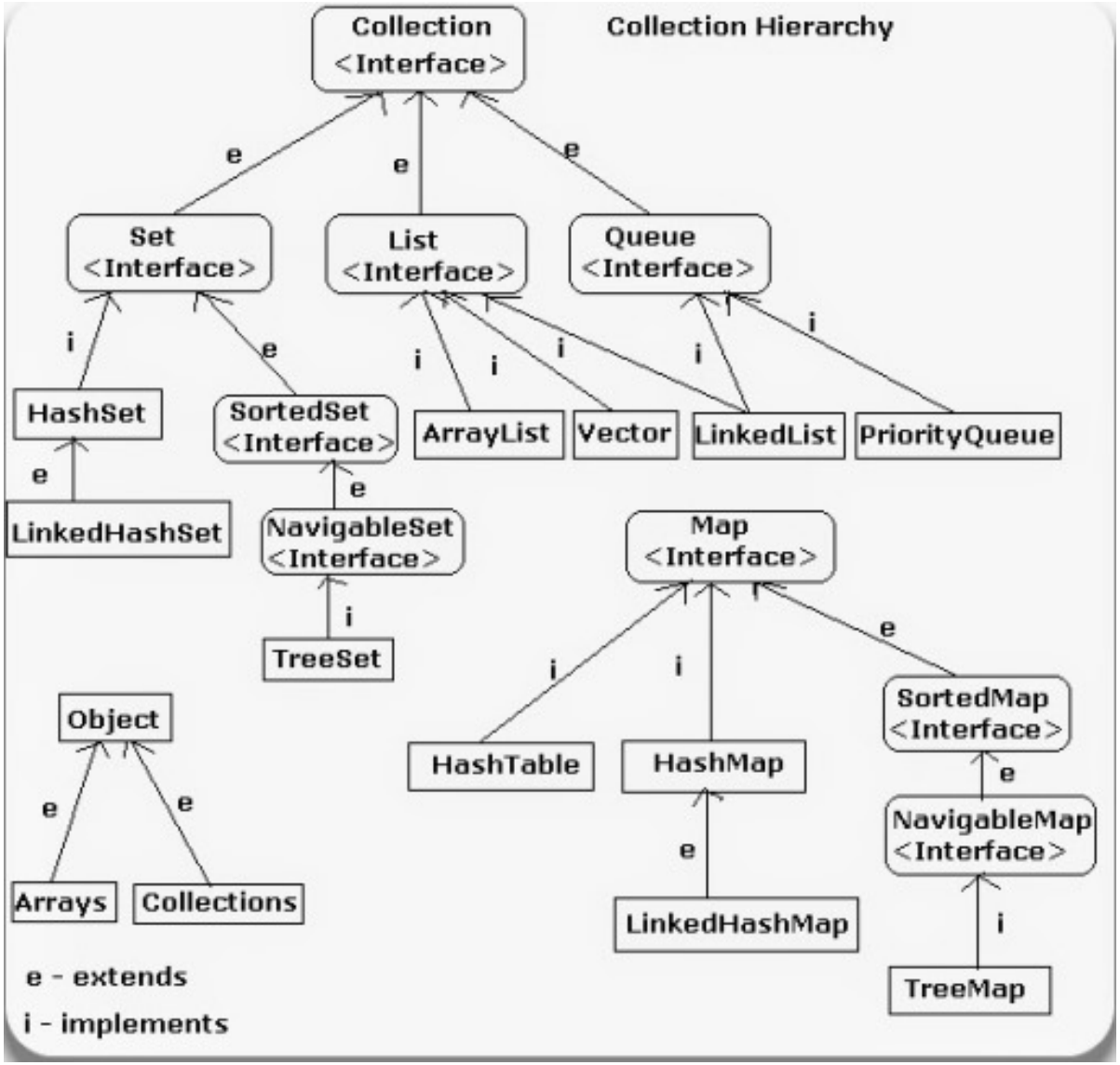


大綱

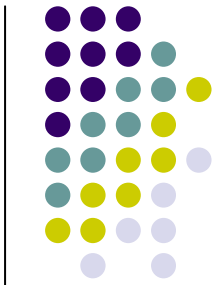


- Mapping（映射）抽象型別
- dict（字典），可雜湊者（hashable）
- set（集合）
- 字典生成式（dict comprehension）
- 集合生成式（set comprehension）

java.util.Collections



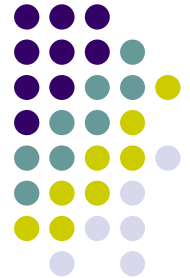
dict基礎介紹



- dict（字典）容器以大括號「{、}」建立，儲存「鍵-值（key-value）」配對，以冒號「:」隔開

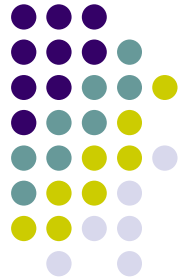
```
>>> d = {'name': 'Amy', 'age': 18}
>>> d
{'age': 18, 'name': 'Amy'}   # 字典不具順序性
>>> d['name']                 # 以 [鍵] 取 值
'Amy'
>>> d['age'] = 29             # 寫入（修改）
>>> d
{'age': 29, 'name': 'Amy'}
>>> d['weight']               # 無此鍵
KeyError: 'weight'
```

dict是映射形式的容器型別



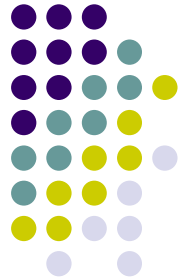
- 從「鍵（key）」映射到「值（value）」
- 值可以是任何型別的物件，int、list、str、dict等
- 鍵必須是不可變物件，正確地說是符合「可雜湊者（hashable）」的物件
- 通常使用str（字串）作爲鍵
- 若使用int作爲鍵，毋須如list般從0開始

範例：鍵與值



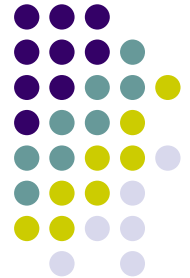
```
>>> d = {44: [0, 1, 2],  (3, 4): {'x': 1}}
>>> d[(3, 4)]      # 鍵是tuple，值是dict
{'x': 1}
>>> d = {8622501: {'name': 'Amy', 'age': 26},
         8622502: {'name': 'Bob', 'age': 33}}
>>> d[8622501]          # 鍵是學號（很大的數字）
{'age': 26, 'name': 'Amy'}  # 儲存學生資料
>>> d[8622501]['name']       # 值是個dict
'Amy'
```

操作動作



```
>>> d = {'age': 33, 'name': 'Amy', 'score':86}
>>> len(d)                  # 長度
3
>>> 'age' in d              # 測試有無此鍵
True
>>> del d['score']          # 刪除鍵值配對
>>> d['grade'] = 86         # 增加（或修改）鍵值配對
>>> for k in d:             # 迭代「鍵」
...     print(k + ' ', end='')
...
grade age name >>>
```

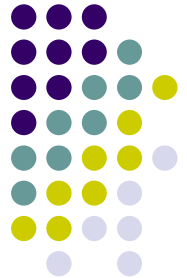
注意：dict不具順序性



- 即便內容相同，但會因Python版本、鍵值配對增減順序而不同

```
>>> keys = ('name', 'age', 'job')
>>> values = ('Amy', 25, 'writer')
>>> d = dict( zip(keys, values) )
>>> d
{'job': 'writer', 'age': 25, 'name': 'Amy'}
>>> del d['job']; del d['age']
>>> d['job'] = 'teacher'; d['age'] = 29
>>> d
{'age': 29, 'job': 'teacher', 'name': 'Amy'}
```

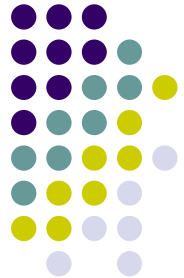
預設



- 方法get、setdefault

```
>>> d = {'name': 'Amy', 'age': 25}
>>> d.get('job', 'freelancer')
'freelancer'
>>> d
{'name': 'Amy', 'age': 25}
>>> d.setdefault('job', 'singer') # 無此鍵時，
'singer'                          # 才會指派
>>> d.setdefault('job', 'dancer') # 已有此鍵
'singer'
```

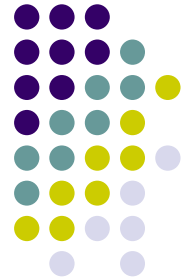
set基礎介紹



- 不具順序性的容器型別，與數學「集合」相同
- 同一個物件只能放進去一次，記錄「有沒有」
- 元素必須是不可變物件，正確地說是符合「可雜湊者（hashable）」的物件
- 語法是以「{、}」包住元素，以冒號「:」隔開

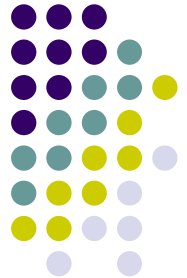
```
>>> x = {1, 2, 3, 4, 5, 5}
>>> x                                  # 相同物件只存在一個
{1, 2, 3, 4, 5}
>>> x = set(range(10))
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

範例：加入、移除



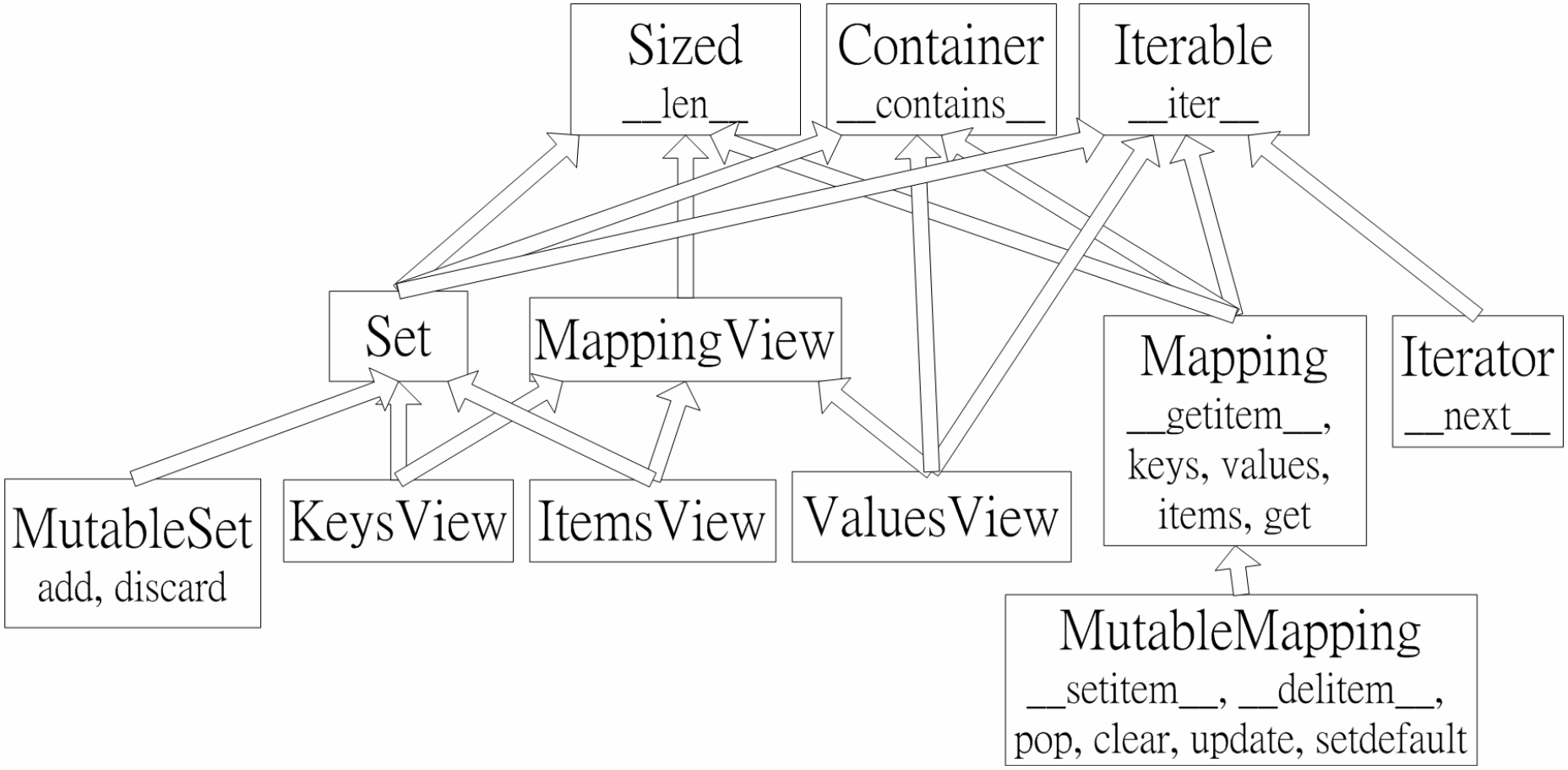
```
>>> x = {}; type(x)    # 注意：{} 是空字典
<class 'dict'>
>>> x = set()          # 使用建構式建立空集合
set()
>>> x.add(6); x.add('hi'); x.add((1, 2))
>>> x
{(1, 2), 'hi', 6}
>>> x.remove(6)
>>> x.remove(7)          # 移除，若不存在會出錯
KeyError: 7
>>> x.discard('hello') # 移除，若不存在會忽視
```

範例：集合基本操作動作

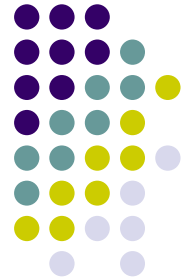


```
>>> x = {0, 1, 2, 3, 4, 5}
>>> x & set(range(0, 10, 2))  # 交集
{0, 2, 4}
>>> x | set(range(0, 10, 2))  # 聯集
{0, 1, 2, 3, 4, 5, 6, 8}
>>> x - set(range(0, 10, 2))  # 差集
{1, 3, 5}
>>> x ^ set(range(0, 10, 2))  # 對稱差集
{1, 3, 5, 6, 8}
```

映射抽象型別


Sized
__len__
Container
__contains__
Iterable
__iter__
Set
MappingView
Mapping
__getitem__,
keys, values,
items, get
Iterator
__next__
MutableSet
add, discard
KeysView
ItemsView
ValuesView
MutableMapping
__setitem__, __delitem__,
pop, clear, update, setdefault

抽象型別Hashable ：可雜湊者



- 符合此介面的物件，可根據其值（內容）算出獨一無二的雜湊碼
- 物件不同，雜湊碼就會不同
- Python內建的不可變型別，都符合此介面
- Python內建的可變型別，都不符合

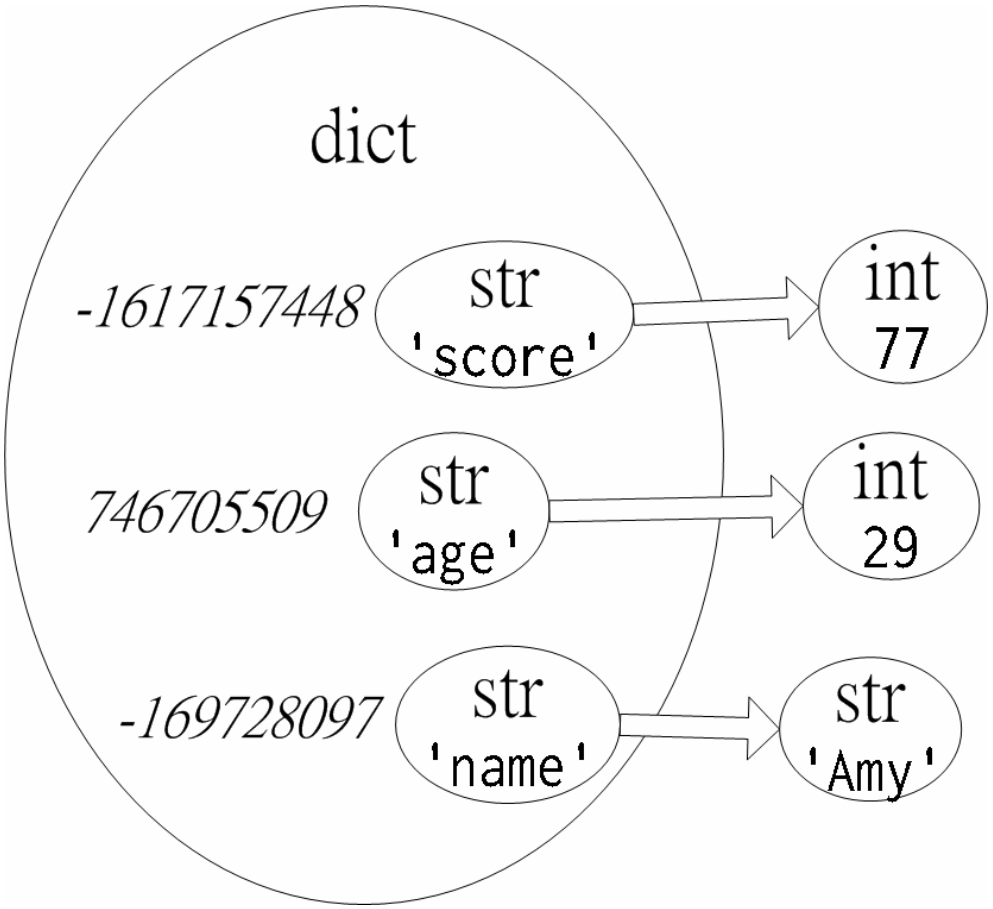
```
>>> hash(99), hash('abc')    # int與str
(99, 1453976822)
>>> hash((1, 2, 3))          # tuple
-378539185
>>> hash([1, 2, 3])          # list是可變的
TypeError: unhashable type: 'list'
```

雜湊演算法（雜湊函數）

- MD5、SHA-1、SHA-256、CRC32、RipeMD等
- 輸入不同內容，就會輸出不同的雜湊值
- 演算法設計良好的話，幾乎不會重複
- 需要傳大檔案時，傳送方先以某雜湊演算法算出檔案的雜湊值，作爲校驗碼（checksum）
- 接收方收到後，也以相同的雜湊演算法算出檔案雜湊值，比對是否相同
- 軟體：HashMyFiles、HashTab；指令：sha1sum、md5sum、openssl sha1等

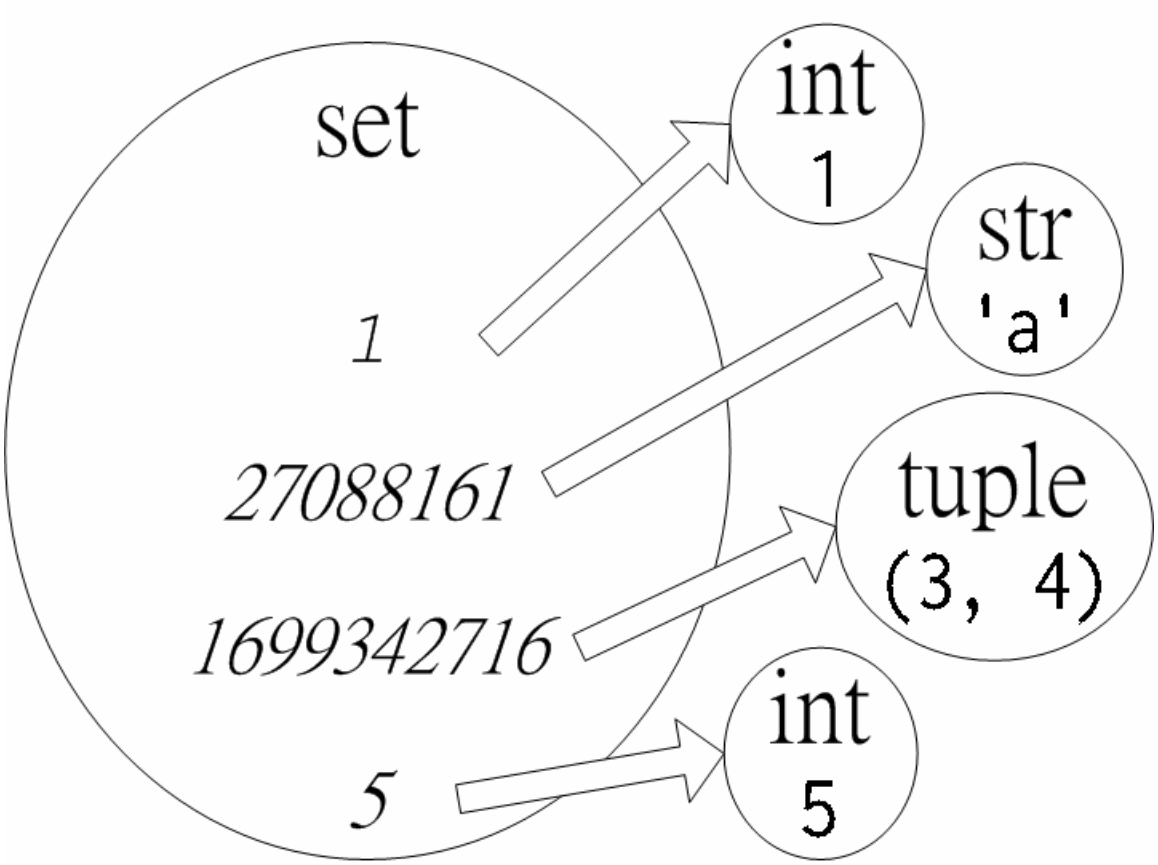
dict示意圖

```
>>> {'name': 'Amy', 'age': 29, 'score': 77}
```

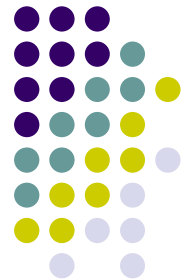


set示意圖

```
>>> {1, 'a', (3, 4), 5}
```

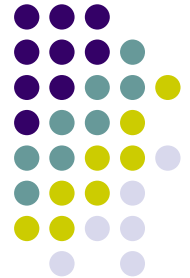


問題：身分證檢查碼
id_checksum.py



- 共10碼：1個英文字母、8個數字、1個數字（驗證碼）
- 英文字母轉成數字（下一頁），個位數乘9再加上十位數，得到一數(A)
- 8個數字，從左到右分別乘上8、7、6、...、1，然後相加，得到一數(B)
- (A)加(B)得到(C)，除以10算出餘數(D)
- (D)若是0，驗證碼就是0；若(D)非0，則驗證碼是「10減(D)」

身分證開頭字母對應數字

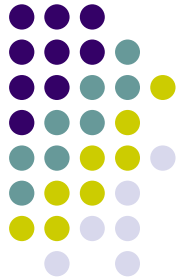


- A=10 台北市
  B=11 台中市
  C=12 基隆市
  D=13 台南市
  E=14 高雄市
  F=15 台北縣
  G=16 宜蘭縣
  H=17 桃園縣
  I=34 嘉義市
  J=18 新竹縣
  K=19 苗栗縣
  L=20 台中縣
  M=21 南投縣
  N=22 彰化縣
  O=35 新竹市
  P=23 雲林縣
  Q=24 嘉義縣
  R=25 台南縣
  S=26 高雄縣
  T=27 屏東縣
  U=28 花蓮縣
  V=29 台東縣
  W=32 金門縣
  X=30 澎湖縣
  Y=31 陽明山
  Z=33 連江縣
- PS：字母後面的第一個數字，1代表男性，2代表女性

舉例：身分證檢查碼

A	1	2	3	4	5	6	7	8	9
	8	7	6	5	4	3	2	1	
1	8	14	18	20	20	18	14	8	121
									1
									9
S	2	1	2	3	0	5	4	3	8
	8	7	6	5	4	3	2	1	
56	16	7	12	15	0	15	8	3	132
									2
									8

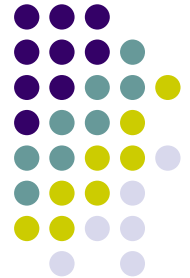
sys.argv：命令列參數



- sys.argv是個list物件，[0]是程式檔名，[1]、[2]是後面跟著的參數

- id_check.py
- 無參數，印出用法
- -g，亂數產生
- -c，後面應跟著id，檢查是否正確，若錯誤，輸出正確的id

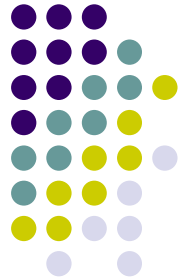
dict的view：動態反映出dict內容



- KeysView， ValuesView，ItemsView

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

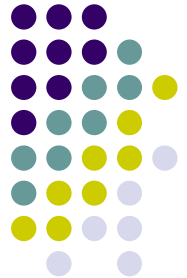
迭代dict



```
>>> for k in d:          # 預設迭代「鍵」
...     print(k + ' ', end='')
...
job name age >>>

>>> for k, v in d.items():    # 迭代鍵與值
...     print(k, v, ' ', end='')
...
job writer  name Amy  age 23  >>>
```

字典生成式（dict comprehension）



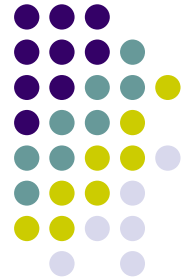
- 語法。「expr」代表運算式

{ 鍵expr:值expr for 名稱 in 可迭代者 if expr }

```
>>> {k: k**2 for k in range(5)}
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}

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

範例：有規則的對應關係



```
>>> li = [chr(ord('A')+x) for x in range(5)]
>>> li
['A', 'B', 'C', 'D', 'E']

>>> d = {k:v for v,k in enumerate(li, start=10)}
>>> d
{'A': 10, 'D': 13, 'C': 12, 'E': 14, 'B': 11}
>>> d['A']
10
```

問題：統計成績區間

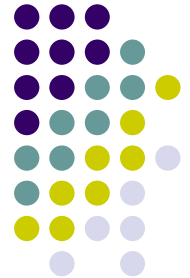
- 學生成績資料如下，以10分爲區間，找出0~9分有哪些人，10~19分有哪些人，依此類推

```
d = {'Amy': 45, 'Bob': 50, 'Cathy': 62,
     ...省略...}
```

- 輸出應類似底下的樣子：

```
{0: ['Kevin'], 4: ['David', 'Amy'], 5:
  ['Bob'], 6: ['Cathy', 'Eason'], ...省略...}
```

集合生成式（set comprehension）



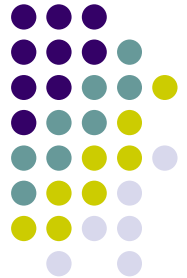
- 語法

```
{ 運算式 for 名稱 in 可迭代者 if 運算式 }
```

- 範例

```
>>> li = ['a', 'bar', 'candy', 'o', 'car']
>>> {len(x) for x in li}
{1, 3, 5}
>>> {x for x in li if len(x) == 1}
{'o', 'a'}
```

問題：丟掉後面重複的元素



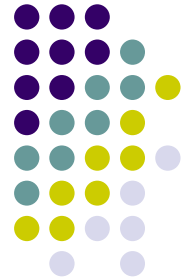
- 找出容器中至少出現過一次的元素

```
li = (3, 1, 'a', 'Amy', 1, 3, 5, 'a')

def unique(iterable):
    result = []
    for x in iterable:
        if x not in result:    # 注意
            result.append(x)
    return result
```

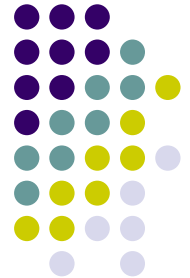
- 請用set改寫

問題：找出2到n之間的質數



- 埃氏篩法
- 首先寫上2到n之間所有的數
- 從最小的質數2開始，刪掉2的所有倍數
- 然後處理下一個質數，也就是下一個比2大、但沒被刪掉的數
- 是3，刪掉3的所有倍數
- 重複上述步驟

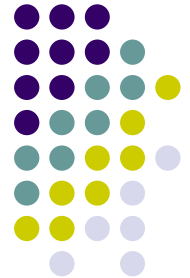
其他特殊容器：**defaultdict**



- defaultdict：有預設值的字典

```
>>> data = [('A', 50), ('B', 70), ('C', 50),
  ('D', 30)]
>>> from collections import defaultdict
>>> d = defaultdict(list)  # list建構式
>>> for k, v in data:
...     d[v].append(k)
...
>>> d
defaultdict(<class 'list'>, {50: ['A', 'C'],
  30: ['D'], 70: ['B']})
```

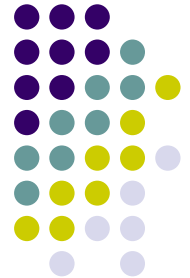
其他特殊容器：OrderedDict



- OrderedDict：具有順序性的字典

```
>>> from collections import OrderedDict
>>> d = OrderedDict()
>>> d['c'] = 33; d['a'] = 4; d['b'] = 99
>>> d
OrderedDict([('c', 33), ('a', 4), ('b', 99)])
```

其他特殊容器：**frozenset**



- frozenset：不可變的集合

```
>>> fs = frozenset(['a', 'b', 'c'])
>>> fs
frozenset({'b', 'c', 'a'})
>>> fs.remove('a')          # 不能修改
AttributeError: 'frozenset' object has no attribute 'remove'

>>> {fs: 1, 'b': 2, 'c': 3}    # 可作爲字典的鍵
{frozenset({'b', 'c', 'a'}): 1, 'b': 2, 'c': 3}
```

Q&A