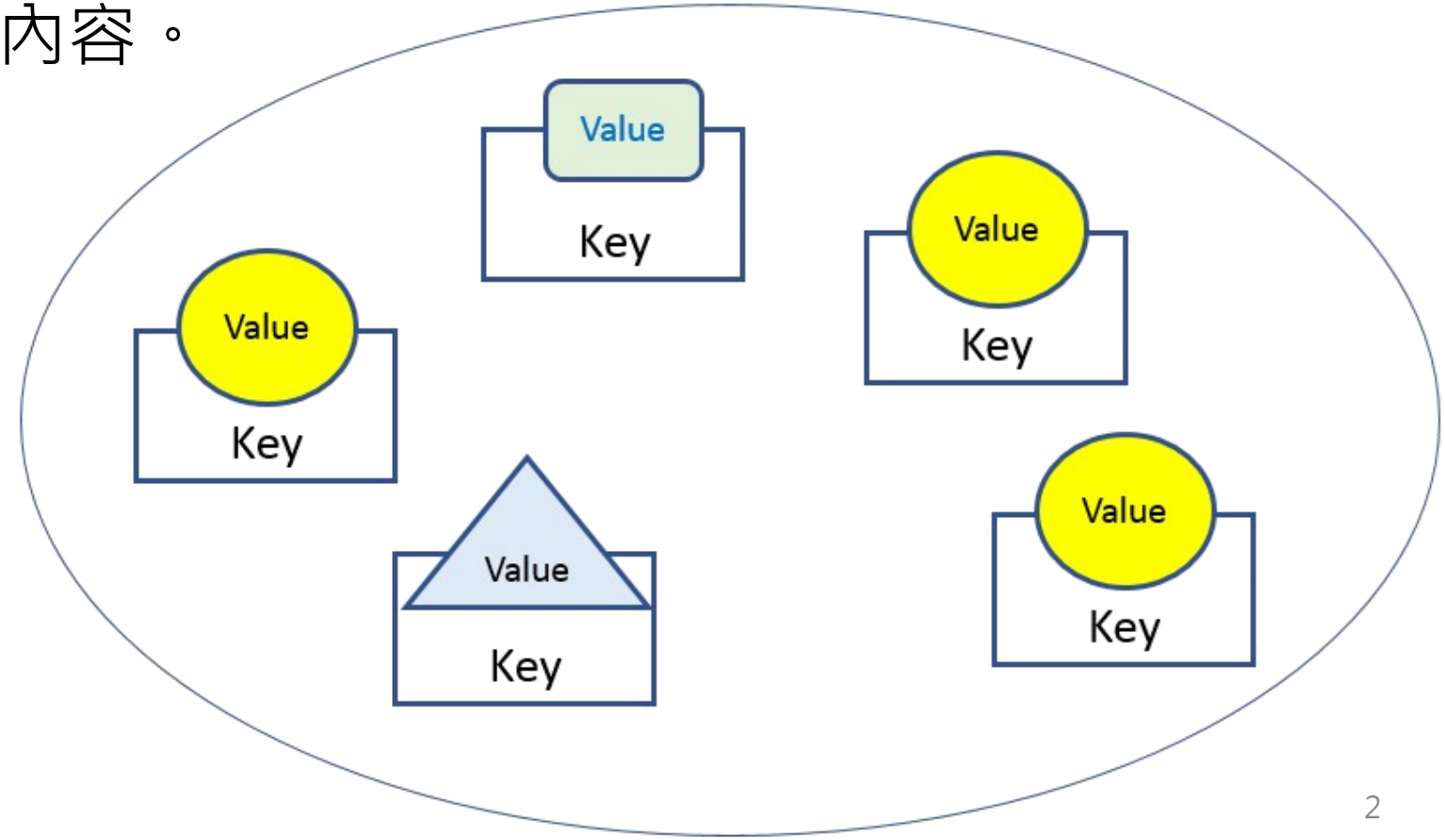
Chapter 9
字典(Dict)

9.1 字典的基本操作

- 串列(list)與元組(tuple)是依序排列，可稱為是序列資料結構。
- 字典(dict)是非序列資料結構，所以無法使用類似串列的索引[0, 1, … n]觀念取得元素內容。



9.1 字典的基本操作

- 定義字典：
  - 字典是一個非序列的資料結構，但是它的元素是用"鍵:值"方式配對儲存，在操作時是用鍵(key)取得值(value)的內容。

myDict = { 鍵1:值1, … , 鍵n:值n, }　　# myDict是字典變數名稱

  - key 一般常用的是字串或數字，在一個字典中不可有重複的 key。
  - value 可以是任何 python 的資料物件，如：數值、字串、串列、元組、字典…等等。
  - 鍵n:值n右邊的","，可有可無。

9.1 字典的基本操作

- 定義字典：
  - Ex 9-1: 以水果行和麵店為例定義一個字典，同時列出字典。

```
dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
dictNodles = {"牛肉麵": 180, "榨菜肉絲麵": 75, "餛飩麵": 55}
print(f"{type(dictFruits) = }, {dictFruits}")
print(f"{type(dictNodles) = }, {dictNodles}")
```

```
type(dictFruits) = <class 'dict'>, {'Watermelon': 299, 'Banana': 29, 'Peach': 25}
type(dictNodles) = <class 'dict'>, {'牛肉麵': 180, '榨菜肉絲麵': 75, '餛飩麵': 55}
```

  - Ex 9-2: 遊戲中，敵軍的的資訊可以用字典儲存。用不同顏色標記不同等級的小兵，殺死不同等級的小兵獲得不同的分數。

```
dictSoldierR = {"tag": "red", "score": 3}
print(dictSoldierR)
```

```
{'tag': 'red', 'score': 3}
```

9.1 字典的基本操作

- 列出字典元素的值：
  - 字典是用 key: value 的搭配設定，若想要取得元素的值，可以將 key 當作索引值的方式處理，因此字典裡面不能有相同的 key 值。
  - Ex 9-3: 分別列出Ex 9-1中，peach與牛肉麵的價錢。

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   dictNodles = {"牛肉麵": 180, "榨菜肉絲麵": 75, "餛飩麵": 55}
3   print(f"One peach costs {dictFruits['Peach']}.")
4   print(f"牛肉麵一碗要 {dictNodles['牛肉麵']} 元")
```

```
One peach costs 25.
牛肉麵一碗要 180 元
```

  - Ex 9-4: 分別列出 Ex 9-2 中，小兵的 tag 與score。

```
1   dictSoldierR = {"tag": "red", "score": 3}
2   print(f"打死標記為 {dictSoldierR['tag']} 的小兵")
3   print(f"獲得分數 {dictSoldierR['score']} 分")
```

```
打死標記為 red 的小兵
獲得分數 3 分
```

9.1 字典的基本操作

- 增加字典元素：

  myDict[鍵] = 值　　# mDdict是字典變數

  - Ex 9-5: 為 Ex 9-1 的字典，新增Orange一顆 15 元，以及肉燥麵一碗35元。

```
dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
dictNodles = {"牛肉麵": 180, "榨菜肉絲麵": 75, "餛飩麵": 55}
dictFruits["Orange"] = 15
dictNodles["肉燥麵"] = 35
print(f"{dictFruits = }")
print(f"{dictNodles = }")
```

```
dictFruits = {'Watermelon': 299, 'Banana': 29, 'Peach': 25, 'Orange': 15}
dictNodles = {'牛肉麵': 180, '榨菜肉絲麵': 75, '餛飩麵': 55, '肉燥麵': 35}
```

  - Ex 9-6: 為 Ex 9-2 的字典，增加 x, y 座標與移動速度元素。

```
dictSoldierR = {"tag": "red", "score": 3}
dictSoldierR["xpos"] = 100
dictSoldierR["ypos"] = 30
dictSoldierR["speed"] = "slow"
print(dictSoldierR)
```

```
{'tag': 'red', 'score': 3, 'xpos': 100, 'ypos': 30, 'speed': 'slow'}
```

9.1 字典的基本操作

- 更改字典元素內容：

  myDict[鍵] = 值　　# mDdict是字典變數

  - Ex 9-7: 將 Ex 9-1 的字典，Banana改為 15 元，牛肉麵改為一碗220元。

```
dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
dictNodles = {"牛肉麵": 180, "榨菜肉絲麵": 75, "餛飩麵": 55}
dictFruits["Banana"] = 15
dictFruits["牛肉麵"] = 220
print(f"One Banana costs {dictFruits['Banana']}.")
print(f"牛肉麵一碗要 {dictNodles['牛肉麵']} 元")
```

```
One Banana costs 15.
牛肉麵一碗要 180 元
```

  - Ex 9-8: 將 Ex 9-6 中的小兵，依照速度更改位置。

```
dictSoldierR = {"tag": "red", "score": 3, "xpos": 100,
                "ypos": 30, "speed": "slow"}
print(f"小兵原本的座標為({dictSoldierR['xpos']}, {dictSoldierR['ypos']})")
if dictSoldierR["speed"] == "slow":
    moveX = 1
elif dictSoldierR["speed"] == "medium":
    moveX = 3
else:   #dictSoldierR["speed"] == "fast":
    move = 5
dictSoldierR["xpos"] += moveX
print(f"小兵新的座標為({dictSoldierR['xpos']}, {dictSoldierR['ypos']})")
```

```
小兵原本的座標為(100, 30)
小兵新的座標為(101, 30)
```

9.1 字典的基本操作

- 刪除字典特定元素：

  del myDict[鍵]　　# mDdict是字典變數

  - Ex 9-9: 刪除 Ex 9-1 字典中的，Watermelon與榨菜肉絲麵。

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   dictNodles = {"牛肉麵": 180, "榨菜肉絲麵": 75, "餛飩麵": 55}
3   del dictFruits["Watermelon"]
4   del dictNodles["榨菜肉絲麵"]
5   print(f"{dictFruits = }")
6   print(f"{dictNodles = }")
```

```
dictFruits = {'Banana': 29, 'Peach': 25}
dictNodles = {'牛肉麵': 180, '餛飩麵': 55}
```

9.1 字典的基本操作

- 字典的 pop 方法：
  - 字典的 pop 方法也可以刪除字典內特定的元素，同時傳回所刪除的元素。

retVal = dict.pop(key [, default])

  - 上述 key 是要刪除元素的 key，找到時就從字典內刪除，並將刪除 key 的 value 傳回。
  - 若是要刪除的 key 不在字典內，則傳回 default 的內容。若是 default 沒有設定，則會導致 KeyError，程式異常中止。

9.1 字典的基本操作

- 字典的 pop 方法：
  - Ex 9-10: 將 Ex 9-1 字典中Watermelon pop 出來，並印出結果。

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   cost = dictFruits.pop("Watermelon")
3   print(f"Watermelon costs {cost}.")
4   print(f"{dictFruits = }")
```

```
Watermelon costs 299.
dictFruits = {'Banana': 29, 'Peach': 25}
```

  - 刪除的元素不存在將導致異常中止。

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   cost = dictFruits.pop("Litchi")
3   print(f"Litchi costs {cost}.")
4   print(f"{dictFruits = }")
```

```
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 2, in <module>
    cost = dictFruits.pop("Litchi")
KeyError: 'Litchi'
```

  - 刪除的元素不存在，則印出 Item does not exist.。

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   cost = dictFruits.pop("Litchi", "Item doest not exist.")
3   print(f"{cost}")
```

```
Item doest not exist.
```

9.1 字典的基本操作

- 字典的 popitem 方法：
  - 隨機將字典中的元素 pop 一組出來當傳回值，並將此元素從字典中刪除。
  - 傳回值是一個元組 (tuple)，內容為 (key, value)。
  - 若字典是空的，則會有錯誤異常產生。

retTup = dict.popitem()

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   fruit = dictFruits.popitem()
3   print(f"{type(fruit) = }, {fruit = }")
4   print(f"{dictFruits = }")
```

```
type(fruit) = <class 'tuple'>, fruit = ('Peach', 25)
dictFruits = {'Watermelon': 299, 'Banana': 29}
```

9.1 字典的基本操作

- 刪除字典所有元素：
  - 字典提供一個 clear() 的方法，可以將字典所有元素刪除。
  - 字典將變成空的字典。

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   print(f"{dictFruits = }")
3   dictFruits.clear()
4   print(f"{dictFruits = }")
```

```
dictFruits = {'Watermelon': 299, 'Banana': 29, 'Peach': 25}
dictFruits = {}
```

9.1 字典的基本操作

- 刪除字典：

  del myDict

  - 字典將不存在。

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   print(f"{dictFruits = }")
3   del dictFruits
4   print(f"{dictFruits = }")
```

```
dictFruits = {'Watermelon': 299, 'Banana': 29, 'Peach': 25}
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 4, in <module>
    print(f"{dictFruits = }")
NameError: name 'dictFruits' is not defined
```

9.1 字典的基本操作

- 建立空字典：

  myDict = {}

  - Ex 9-11: 建立一個小兵的空字典，並為小兵建立如 Ex 9-2 的元素。

```
1   dictSoldierR = {}
2   print(f"{dictSoldierR = }")
3   dictSoldierR["tag"] = "red"
4   dictSoldierR["score"] = 3
5   print(f"{dictSoldierR = }")
```

```
dictSoldierR = {}
dictSoldierR = {'tag': 'red', 'score': 3}
```

9.1 字典的基本操作

- 字典的拷貝：

  newDict = myDict.copy()

  - Ex 9-12: 複製字典，同時列出新字典所在位址，驗證新字典與舊字典是不同的字典。

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   dictFruitsNew = dictFruits.copy()
3   print(f"{id(dictFruits) = }\t\t{dictFruits = }")
4   print(f"{id(dictFruitsNew) = }\t{dictFruitsNew = }")
```

```
id(dictFruits) = 2118974616960          dictFruits = {'Watermelon': 299, 'Banana': 29, 'Peach': 25}
id(dictFruitsNew) = 2118974617152       dictFruitsNew = {'Watermelon': 299, 'Banana': 29, 'Peach': 25}
```

```
6   print(f"{id(dictFruits['Banana']) = }")
7   print(f"{id(dictFruitsNew['Banana']) = }")
```

```
id(dictFruits['Banana']) = 1798038318256
id(dictFruitsNew['Banana']) = 1798038318256
```

  - copy() 是淺拷貝。

9.1 字典的基本操作

- 字典的拷貝：

```
1   dictA = {"a": [1, 2, 3]}
2   dictB = dictA.copy()
3   print(f"{dictA = }")
4   print(f"{dictB = }")
5   dictB["a"].append(10)
6   print(f"{dictA = }")
7   print(f"{dictB = }")
```

```
dictA = {'a': [1, 2, 3]}
dictB = {'a': [1, 2, 3]}
dictA = {'a': [1, 2, 3, 10]}
dictB = {'a': [1, 2, 3, 10]}
```

  - deepcopy:

```
1   import copy
2   dictA = {"a": [1, 2, 3]}
3   dictB = copy.deepcopy(dictA)
4   print(f"{dictA = }")
5   print(f"{dictB = }")
6   dictB["a"].append(10)
7   print(f"{dictA = }")
8   print(f"{dictB = }")
```

```
dictA = {'a': [1, 2, 3]}
dictB = {'a': [1, 2, 3]}
dictA = {'a': [1, 2, 3]}
dictB = {'a': [1, 2, 3, 10]}
```

9.1 字典的基本操作

- 字典的拷貝：

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   dictFruitsNew = dictFruits.copy()
3
4   print(f"{id(dictFruits['Banana']) = }")
5   print(f"{id(dictFruitsNew['Banana']) = }")
6
7   dictFruits["Banana"] = 20
8
9   print(f"{id(dictFruits) = }\t\t{dictFruits = }")
10  print(f"{id(dictFruitsNew) = }\t{dictFruitsNew = }")
11  print(f"{id(dictFruits['Banana']) = }")
12  print(f"{id(dictFruitsNew['Banana']) = }")
```

```
id(dictFruits['Banana']) = 2116006407344
id(dictFruitsNew['Banana']) = 2116006407344
id(dictFruits) = 2116008129856          dictFruits = {'Watermelon': 299, 'Banana': 20, 'Peach': 25}
id(dictFruitsNew) = 2116008130048       dictFruitsNew = {'Watermelon': 299, 'Banana': 29, 'Peach': 25}
id(dictFruits['Banana']) = 2116006407056
id(dictFruitsNew['Banana']) = 2116006407344
```

9.1 字典的基本操作

- 取得字典元素的數量：
  - 透過 len() 函數。

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25, "Apple": 15}
2   dictNodles = {"牛肉麵": 180, "榨菜肉絲麵": 75, "餛飩麵": 55}
3   dictEmpty = {}
4   print(f"dictFruits 的數量 = {len(dictFruits)}")
5   print(f"dictNodles 的數量 = {len(dictNodles)}")
6   print(f"dictEmpty 的數量  = {len(dictEmpty)}")
```

```
dictFruits 的數量 = 4
dictNodles 的數量 = 3
dictEmpty 的數量  = 0
```

9.1 字典的基本操作

- 檢查元素是否存在：
  - key in myDict / key not in myDict

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   key = input("Please inptut fruit: ")
3   value = int(input("Please inptut the cost: "))
4   if key in dictFruits:
5       print(f"{key} 已經在字典中了")
6   else:
7       dictFruits[key] = value
8       print(f"新增 {key} 到字典中")
9   print(f"{dictFruits}")
```

```
Please inptut fruit: Apple
Please inptut the cost: 20
新增 Apple 到字典中
{'Watermelon': 299, 'Banana': 29, 'Peach': 25, 'Apple': 20}
```

```
Please inptut fruit: Peach
Please inptut the cost: 15
Peach 已經在字典中了
{'Watermelon': 299, 'Banana': 29, 'Peach': 25}
```

9.1 字典的基本操作

- 設計字典時的可讀性：
  - 字典內容複雜時：

```
dictPlayers = {"Stephen Curry": "Golden State Warriors", "Kevin Durant": "Golden State Warriors",
               "LeBron James": "Cleveland Cavaliers", "James Harden": "Huston Recokets", "Paul Gasol": "San Antonio Spurs"}
```

  - 建議一筆元素一行：

```
dictPlayers = {"Stephen Curry": "Golden State Warriors",
               "Kevin Durant": "Golden State Warriors",
               "LeBron James": "Cleveland Cavaliers",
               "James Harden": "Huston Recokets",
               "Paul Gasol": "San Antonio Spurs"}
```

```
dictPlayers = {
    "Stephen Curry": "Golden State Warriors",
    "Kevin Durant": "Golden State Warriors",
    "LeBron James": "Cleveland Cavaliers",
    "James Harden": "Huston Recokets",
    "Paul Gasol": "San Antonio Spurs"
}
```

```
dictPlayers = {
    "Stephen Curry": "Golden State Warriors",
    "LeBron James" : "Cleveland Cavaliers",
    "James Harden" : "Huston Rockets",
    "Kevin Durant" : "Golden State Warriors",
    "Paul Gasol"   : "San Antonio Spurs"
    }
```

9.1 字典的基本操作

- 合併字典：
  - 使用 update() 方法合併字典：

```
dictJPCar = {1: "Nissan", 2: "Toyota", 3: "Honda"}
dictEUCar = {10: "Benz", 11: "BMW", 12: "Porsche"}
dictJPCar.update(dictEUCar)
print(f"{dictJPCar = }")
```

```
dictJPCar = {1: 'Nissan', 2: 'Toyota', 3: 'Honda', 10: 'Benz', 11: 'BMW', 12: 'Porsche'}
```

  - 合併字典時，若是 key 值有重複，則第二個字典的值會取代原先字典的值。

```
dictJPCar = {1: "Nissan", 2: "Toyota", 3: "Honda"}
dictEUCar = {3: "Benz", 4: "BMW", 5: "Porsche"}
dictJPCar.update(dictEUCar)
print(f"{dictJPCar = }")
```

```
dictJPCar = {1: 'Nissan', 2: 'Toyota', 3: 'Benz', 4: 'BMW', 5: 'Porsche'}
```

9.1 字典的基本操作

- 合併字典：
  - Python 3.5 之後提供合併的新方法 "**"：

```
1   dictJPCar = {1: "Nissan", 2: "Toyota", 3: "Honda"}
2   dictEUCar = {3: "Benz", 4: "BMW", 5: "Porsche"}
3   dictAllCar = {**dictJPCar, **dictEUCar}
4   print(f"{dictAllCar = }")
```

```
dictJPCar = {1: 'Nissan', 2: 'Toyota', 3: 'Benz', 4: 'BMW', 5: 'Porsche'}
```

```
1   dictJPCar = {1: "Nissan", 2: "Toyota", 3: "Honda"}
2   dictEUCar = {4: "Benz", 5: "BMW"}
3   dictUSCar = {6: "Ford"}
4   dictAllCar = {**dictJPCar, **dictEUCar, **dictUSCar}
5   print(f"{dictAllCar = }")
```

```
dictAllCar = {1: 'Nissan', 2: 'Toyota', 3: 'Honda', 4: 'Benz', 5: 'BMW', 6: 'Ford'}
```

9.1 字典的基本操作

- dict() 函數：
  - 雙值序列（元素內容長度為二的串列/元組）的資料，如下所示：

    [["日本","東京"], ["泰國", "曼谷"], ["英國", "倫敦"]]

    或是 ("東京", "曼谷", "倫敦")
  - 可以使用dict( )將此序列轉成字典，其中雙值序列的第一個是鍵，第二個是值。

```
1   listNation = [["日本","東京"], ["泰國", "曼谷"], ["英國", "倫敦"]]
2   dictNation =dict(listNation)
3   print(dictNation)
4
5   listNation1 = ("東京", "曼谷", "倫敦")
6   dictNation1 =dict(listNation1)
7   print(dictNation1)
```

```
{'日本': '東京', '泰國': '曼谷', '英國': '倫敦'}
{'東': '京', '曼': '谷', '倫': '敦'}
```

9.1 字典的基本操作

- 使用 zip() 函數產生字典：

```
1   zipData = zip("abcde", range(5))
2   myDict = dict(zipData)
3   print(myDict)
4
5   zipData1 = zip(["a", "b", "c", "d", "e"], range(5))
6   myDict1 = dict(zipData1)
7   print(myDict1)
```

```
{'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4}
{'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4}
```

9.2 遍歷字典

- 遍歷字典的方式：

方法	說明
dict.items()	遍歷字典的 key: value
dict.keys()	遍歷字典的 key
dict.values()	遍歷字典的 value
sorted() 函數	排序內容

9.2 遍歷字典

- 遍歷字典的 key: value：
  - 方法 items() 回傳的資料為元組(tuple)

```
1   dictPlayers = {"Stephen Curry": "Golden State Warriors",
2                  "LeBron James": "Cleveland Cavaliers",
3                  "James Harden": "Huston Rockets",
4                  "Kevin Durant": "Golden State Warriors",
5                  "Paul Gasol": "San Antonio Spurs"}
6   for playerInfo in dictPlayers.items():
7       print(f"{playerInfo = }")
```

```
playerInfo = ('Stephen Curry', 'Golden State Warriors')
playerInfo = ('LeBron James', 'Cleveland Cavaliers')
playerInfo = ('James Harden', 'Huston Rockets')
playerInfo = ('Kevin Durant', 'Golden State Warriors')
playerInfo = ('Paul Gasol', 'San Antonio Spurs')
```

9.2 遍歷字典

- 遍歷字典的 key: value：
  - 方法 items() 可讓我們取得字典的 “key: value“ 配對的元素。

```
dictPlayers = {"Stephen Curry": "Golden State Warriors",
               "Kevin Durant": "Golden State Warriors",
               "LeBron James": "Cleveland Cavaliers",
               "James Harden": "Huston Rockets",
               "Paul Gasol": "San Antonio Spurs"}
for name, team in dictPlayers.items():
    print(f"\n姓名： {name}")
    print(f"隊名： {team}")
```

```
姓名： Stephen Curry
隊名： Golden State Warriors

姓名： Kevin Durant
隊名： Golden State Warriors

姓名： LeBron James
隊名： Cleveland Cavaliers

姓名： James Harden
隊名： Huston Rockets

姓名： Paul Gasol
隊名： San Antonio Spurs
```

  - 雖然輸出順序跟字典順序相同，不過這並非一定。

```
<class 'dict_items'>
dict_items([('Stephen Curry', 'Golden State Warriors'), ('LeBron James', 'Clevelan
d Cavaliers'), ('James Harden', 'Huston Rockets'), ('Kevin Durant', 'Golden State
Warriors'), ('Paul Gasol', 'San Antonio Spurs')])
```

9.2 遍歷字典

- 遍歷字典的 key：
  - 方法 keys()可讓我們取得字典的 "key"。

```
1   dictPlayers = {"Stephen Curry": "Golden State Warriors",
2                  "Kevin Durant": "Golden State Warriors",
3                  "LeBron James": "Cleveland Cavaliers",
4                  "James Harden": "Huston Rockets",
5                  "Paul Gasol": "San Antonio Spurs"}
6   for name in dictPlayers.keys(): #for name in dictPlayers
7       print(f"姓名: {name}")
```

```
姓名: Stephen Curry
姓名: LeBron James
姓名: James Harden
姓名: Kevin Durant
姓名: Paul Gasol
```

```
<class 'dict_keys'>
dict_keys(['Stephen Curry', 'LeBron James', 'James Harden', 'Kevin Durant', 'Paul
Gasol'])
```

9.2 遍歷字典

- 遍歷字典的 value：
  - 方法 values()可讓我們取得字典的 "value"。

```
1   dictPlayers = {"Stephen Curry": "Golden State Warriors",
2                  "Kevin Durant": "Golden State Warriors",
3                  "LeBron James": "Cleveland Cavaliers",
4                  "James Harden": "Huston Rockets",
5                  "Paul Gasol": "San Antonio Spurs"}
6   for team in dictPlayers.values():
7       print(f"隊名: {team}")
```

```
隊名: Golden State Warriors
隊名: Cleveland Cavaliers
隊名: Huston Rockets
隊名: Golden State Warriors
隊名: San Antonio Spurs
```

```
<class 'dict_values'>
dict_values(['Golden State Warriors', 'Cleveland Cavaliers', 'Huston Rockets', 'Go
lden State Warriors', 'San Antonio Spurs'])
```

9.2 遍歷字典

- 遍歷字典的 value：
  - 方法 values()可讓我們取得字典的 "value"。

```
1   dictPlayers = {"Stephen Curry": "Golden State Warriors",
2                  "Kevin Durant": "Golden State Warriors",
3                  "LeBron James": "Cleveland Cavaliers",
4                  "James Harden": "Huston Rockets",
5                  "Paul Gasol": "San Antonio Spurs"}
6   for team in dictPlayers.values():
7       print(f"隊名: {team}")
```

```
隊名: Golden State Warriors
隊名: Golden State Warriors
隊名: Cleveland Cavaliers
隊名: Huston Rockets
隊名: San Antonio Spurs
```

  - 字典中， key 值是唯一的，但是 value 並不一定。如上例，就會輸出兩筆 Golden State Warriors。
  - 如果不希望輸出重複的值，到後面可以用集合 (set) 來解決這個問題。

```
6   for team in set(dictPlayers.values()):
7       print(f"隊名: {team}")
```

```
隊名: Huston Rockets
隊名: San Antonio Spurs
隊名: Golden State Warriors
隊名: Cleveland Cavaliers
```

順序？

9.2 遍歷字典

- sorted() 排序函數：
  - 對字典的 key 排序後輸出字典的內容。

```
1  dictPlayers = {"Stephen Curry": "Golden State Warriors",
2                 "Kevin Durant": "Golden State Warriors",
3                 "LeBron James": "Cleveland Cavaliers",
4                 "James Harden": "Huston Rockets",
5                 "Paul Gasol": "San Antonio Spurs"}
6  for name in sorted(dictPlayers.keys()): #sorted(dictPlayers)
7      print(f"姓名: {name}")
```

```
姓名: James Harden
姓名: Kevin Durant
姓名: LeBron James
姓名: Paul Gasol
姓名: Stephen Curry
```

9.2 遍歷字典

- sorted() 排序函數：
  - 對字典的 value 排序後輸出字典的內容。

```
1   dictPlayers = {"Stephen Curry": "Golden State Warriors",
2                  "Kevin Durant": "Golden State Warriors",
3                  "LeBron James": "Cleveland Cavaliers",
4                  "James Harden": "Huston Rockets",
5                  "Paul Gasol": "San Antonio Spurs"}
6   for team in sorted(dictPlayers.values()):
7       print(f"隊名: {team}")
```

```
隊名: Cleveland Cavaliers
隊名: Golden State Warriors
隊名: Golden State Warriors
隊名: Huston Rockets
隊名: San Antonio Spurs
```

9.2 遍歷字典

- sorted() 排序函數：
  - 對字典排序後轉換成串列。

```
1   dictNoodles = {"牛肉麵": 180, "麻醬麵": 35, "餛飩麵": 55, "肉絲麵": 75}
2   print(dictNoodles)
3
4   print(max(dictNoodles.values()))
5   print(min(dictNoodles.values()))
6
7   listNoodles = sorted(dictNoodles.items(), key=lambda item:item[0])
8   print(listNoodles)
```

```
{'牛肉麵': 180, '麻醬麵': 35, '餛飩麵': 55, '肉絲麵': 75}
180
35
[('牛肉麵', 180), ('肉絲麵', 75), ('餛飩麵', 55), ('麻醬麵', 35)]
```

9.3 建立字典串列

- 串列的內容是字典：

```
1   dictSoldierR = {"tag": "red", "score": 3, "speed": "slow"}
2   dictSoldierG = {"tag": "green", "score": 5, "speed": "medium"}
3   dictSoldierB = {"tag": "blue", "score": 10, "speed": "fast"}
4   armys = [dictSoldierR, dictSoldierG, dictSoldierB]
5   for army in armys:
6       print(army)
```

```
{'tag': 'red', 'score': 3, 'speed': 'slow'}
{'tag': 'green', 'score': 5, 'speed': 'medium'}
{'tag': 'blue', 'score': 10, 'speed': 'fast'}
```

  - 這樣設計太沒效率，用迴圈改善。

```
1   dictSoldierR = {"tag": "red", "score": 3, "speed": "slow"}
2   dictSoldierG = {"tag": "green", "score": 5, "speed": "medium"}
3   dictSoldierB = {"tag": "blue", "score": 10, "speed": "fast"}
4
5   armys = []
6   for soldierNum in range(50):
7       armys.append(dictSoldierR)
8
9   for soldier in armys[:3]:
10      print(soldier)
11  print(f"小兵數量: {len(armys)}")
```

```
{'tag': 'red', 'score': 3, 'speed': 'slow'}
{'tag': 'red', 'score': 3, 'speed': 'slow'}
{'tag': 'red', 'score': 3, 'speed': 'slow'}
小兵數量: 50
```

9.3 建立字典串列

- 串列的內容是字典：
  - 這樣設計太單調，產生不同屬性的小兵。

想想用 random 怎麼做？

```
1   armys = []
2   for soldierNum in range(50):
3       dictSoldierR = {"tag": "red", "score": 3, "speed": "slow"}
4       armys.append(dictSoldierR)
5
6   for soldier in armys[:3]:
7       print(soldier)
8
9   for soldier in armys[35:38]:
10      soldier["tag"] = "green"
11      soldier["score"] = 5
12      soldier["speed"] = "medium"
13      # soldier = dictSoldierG 行不行？
14
15  print(f"列印索引值 35 到 40 的小兵")
16  for soldier in armys[35:41]:
17      print(soldier)
```

這樣行不行？

```
dictSoldierR = {"tag": "red", "score": 3, "speed": "slow"}
for soldierNum in range(50):
    armys.append(dictSoldierR)
```

```
{'tag': 'red', 'score': 3, 'speed': 'slow'}
{'tag': 'red', 'score': 3, 'speed': 'slow'}
{'tag': 'red', 'score': 3, 'speed': 'slow'}
列印索引值 35 到 40 的小兵
{'tag': 'green', 'score': 5, 'speed': 'medium'}
{'tag': 'green', 'score': 5, 'speed': 'medium'}
{'tag': 'green', 'score': 5, 'speed': 'medium'}
{'tag': 'red', 'score': 3, 'speed': 'slow'}
{'tag': 'red', 'score': 3, 'speed': 'slow'}
{'tag': 'red', 'score': 3, 'speed': 'slow'}
```

9.4 字典的 value 是串列

- 字典的 value 內容是串列：

```
1   dictSports = {"Curry": ["籃球", "美式足球"],
2                 "Durant": ["棒球"],
3                 "James": ["美式足球", "籃球", "棒球"]}
4   for name, listSports in dictSports.items():
5       print(f"{name} 喜歡的運動是：")
6       for soprt in listSports:
7           print(f"\t{soprt}")
```

```
Curry 喜歡的運動是：
        籃球
        美式足球
Durant 喜歡的運動是：
        棒球
James 喜歡的運動是：
        美式足球
        籃球
        棒球
```

9.5 字典的 value 是字典

- 字典的 value 內容是字典：

```
dictSports = {"Curry": {"罰球命中率": 0.9, "2分球命中率": 0.28, "3分球命中率": 0.33},
              "Durant": {"罰球命中率": 0.89, "2分球命中率": 0.31, "3分球命中率": 0.29},
              "James": {"罰球命中率": 0.95, "2分球命中率": 0.25, "3分球命中率": 0.28}}
for name, dictStasticInfo in dictSports.items():
    print(f"{name} 的：")
    for shot, info in dictStasticInfo.items():
        print(f"\t{shot} 是 {info}")
```

```
Curry 的：
        罰球命中率 是 0.9
        2分球命中率 是 0.28
        3分球命中率 是 0.33
Durant 的：
        罰球命中率 是 0.89
        2分球命中率 是 0.31
        3分球命中率 是 0.29
James 的：
        罰球命中率 是 0.95
        2分球命中率 是 0.25
        3分球命中率 是 0.28
```

9.6 while迴圈在字典的應用

- Ex 9-13: 利用 while 迴圈，不斷新增字典元素。

```
1   dictSurvey = {}
2   while True:
3       name = input("\n請輸入姓名: ")
4       location = input("請輸入旅遊地點: ")
5       dictSurvey[name] = location
6       repeat = input("是否繼續輸入 (y/n): ")
7       if not(repeat == "y" or repeat == "Y"):
8           break
9
10  print("\n\n大家想去的地點為：")
11  for user, location in dictSurvey.items():
12      print(f"{user} 推薦 {location}")
```

```
請輸入姓名: aaa
請輸入旅遊地點: 奧萬大
是否繼續輸入 (y/n): y

請輸入姓名: bbb
請輸入旅遊地點: 淡水
是否繼續輸入 (y/n): y

請輸入姓名: ccc
請輸入旅遊地點: 墾丁
是否繼續輸入 (y/n): Y

請輸入姓名: ddd
請輸入旅遊地點: 竹山
是否繼續輸入 (y/n): n


大家想去的地點為：
aaa 推薦 奧萬大
bbb 推薦 淡水
ccc 推薦 墾丁
ddd 推薦 竹山
```

9.7 字典常用的函數和方法

- len(): 列出字典的元素個數。

```
1   dictSports = {"Curry": ["籃球", "美式足球"],
2                 "Durant": ["棒球"],
3                 "James": ["美式足球", "籃球", "棒球"]}
4   print(f"dictSports 字典元素個數為 {len(dictSports)}")
5   print(f"dictSports[\"Curry\"] 字典元素個數為 {len(dictSports['Curry'])}")
6   print(f"dictSports[\"Durant\"] 字典元素個數為 {len(dictSports['Durant'])}")
7   print(f"dictSports[\"James\"] 字典元素個數為 {len(dictSports['James'])}")
```

```
dictSports 字典元素個數為 3
dictSports["Curry"] 字典元素個數為 2
dictSports["Durant"] 字典元素個數為 1
dictSports["James"] 字典元素個數為 3
```

9.7 字典常用的函數和方法

- fromkeys(): 建立字典的一個方法

  myDict = dict.fromkeys(seq[, value])　　# 使用seq序列建立字典

  - seq 可以是串列或是元組。
  - 若沒有設定 value，則會使用 None 當對應的值。

```
1   listSeq = ["city1", "city2"]
2   dictSeq = dict.fromkeys(listSeq)
3   print(f"{dictSeq = }")
4   dictSeq = dict.fromkeys(listSeq, "Taipei")
5   print(f"{dictSeq = }")
6
7   tupleSeq = ("city1", "city2")
8   dictSeq = dict.fromkeys(tupleSeq)
9   print(f"{dictSeq = }")
10  dictSeq = dict.fromkeys(tupleSeq, "Taipei")
11  print(f"{dictSeq = }")
```

```
dictSeq = {'city1': None, 'city2': None}
dictSeq = {'city1': 'Taipei', 'city2': 'Taipei'}
dictSeq = {'city1': None, 'city2': None}
dictSeq = {'city1': 'Taipei', 'city2': 'Taipei'}
```

9.7 字典常用的函數和方法

- get(): 搜尋字典的 key，若存在則傳回該 key 的 value，若不存在則傳回預設值。

  retValue = dict.get(key[, default=none])

  - 若沒有設定 default，則會使用 None 當對應的值。

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   retVal = dictFruits.get("Banana")   #retVal = dictFruits["Banana"]
3   print(f"The cost of Bababa is {retVal}")
4
5   retVal = dictFruits.get("Grape")
6   print(f"The cost of Grape is {retVal}")
7   retVal = dictFruits.get("Grape", -1)
8   print(f"The cost of Grape is {retVal}")
```

```
The cost of Bababa is 29
The cost of Grape is None
The cost of Grape is -1
```

- List.index():

```
1   JapanCars = ["Infinity", "Toyota", "Lexus", "Mazda", "Honda", "Nissan", "Toyota"]
2   print(f"{JapanCars}")
3   idxLexus = JapanCars.index("Lexus")
4   print(f"Index of Lexus in JapanCars is {idxLexus}")
5   idxToyota = JapanCars.index("Toyota")
6   print(f"Index of Toyota in JapanCars is {idxToyota}")
7   idxMazda = JapanCars.index("mazda")
8   print(f"Index of mazda in JapanCars is {idxMazda}")
```

```
['Infinity', 'Toyota', 'Lexus', 'Mazda', 'Honda', 'Nissan', 'Toyota']
Index of Lexus in JapanCars is 2
Index of Toyota in JapanCars is 1
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 7, in <module>
    idxMazda = JapanCars.index("mazda")
ValueError: 'mazda' is not in list
```

9.7 字典常用的函數和方法

- setdefault(): 基本上與 get() 相同，不同處在於 get() 不會改變字典內容。使用 setdefault() 時，若 key 不在，會將 key: value 加入字典中。若有設定預設值，則會加入 key: default value，若沒設定預設值，則會加入 key: None。

  retValue = dict.setdefault(key[, default=none])

```
1   dictFruits = {"Watermelon": 299, "Banana": 29, "Peach": 25}
2   retVal = dictFruits.setdefault("Banana")   #retVal = dictFruits["Banana"]
3   print(f"The cost of Bababa is {retVal}")
4
5   retVal = dictFruits.setdefault("Grape", 99)
6   print(f"The cost of Grape is {retVal}")
7   retVal = dictFruits.setdefault("Grape", -1)
8   print(f"The cost of Grape is {retVal}")
```

```
The cost of Bababa is 29
The cost of Grape is 99
The cost of Grape is 99
```

9.8 字典生成式

- 新字典 = { 鍵運算式 : 值運算式 for 運算式 in 可迭代物件 }

```
1   word = "DeepLearning"
2   dictCount1 = {}
3   for wd in word:
4       if wd in dictCount1:
5           dictCount1[wd] += 1
6       else:
7           dictCount1[wd] = 1
8   print(dictCount1)
9
10  dictCount2 = {alphabet:word.count(alphabet) for alphabet in word}
11  print(dictCount2)
```

```
{'D': 1, 'e': 3, 'p': 1, 'L': 1, 'a': 1, 'r': 1, 'n': 2, 'i': 1, 'g': 1}
{'D': 1, 'e': 3, 'p': 1, 'L': 1, 'a': 1, 'r': 1, 'n': 2, 'i': 1, 'g': 1}
```

e 跟 n 分別會被執行 3 次跟 2 次，可以用集合 (set) 改善。

9.9 動手實做

- 文件加密：
  - 摩斯密碼：
    - 將 DEEPLEARNING 用摩斯密碼表示。

字元	代碼	字元	代碼	字元	代碼	字元	代碼	字元	代碼	字元	代碼	字元	代碼
A	·-	B	-···	C	-·-·	D	-··	E	·	F	··-·	G	--·
H	····	I	··	J	·---	K	-·-	L	·-··	M	--	N	-·
O	---	P	·--·	Q	--·-	R	·-·	S	···	T	-	U	··-
V	···-	W	·--	X	-··-	Y	-·--	Z	--··				

字元	長碼	字元	長碼	字元	長碼	字元	長碼	字元	長碼
1	·----	2	··---	3	···--	4	····-	5	·····
6	-····	7	--···	8	---··	9	----·	0	-----